Image plagiarism detection using GAN - (Generative Adversarial Network)

Kaustubh Gayadhankar[1], Rishi Patel[2], Hrithik Lodha[3], and Mr. Swapnil Shinde[4,]*

[1]Department of Information Technology, Ramrao Adik Institute Of Technology, Nerul, Navi Mumbai, India
[2]Department of Information Technology, Ramrao Adik Institute Of Technology, Nerul, Navi Mumbai, India
[3]Department of Information Technology, Ramrao Adik Institute Of Technology, Nerul, Navi Mumbai, India
[4]Department of Information Technology, Ramrao Adik Institute Of Technology, Nerul, Navi Mumbai, India

Abstract – In Today's date plagiarism is a very important aspect because content originality is the client's prior requirement. Many people on the internet use others' images and get publicity while the owner of the image or data won't get anything out of it. Many users copy the data or image features from the other users and modify it a little bit or create an artificial replica of it. With sufficient computational power and volume of data, the GAN models are capable enough to produce fake images that look very much similar to the real images. These kinds of images are generally not detected by modern plagiarism systems. GAN stands for generative adversarial network. It has two neural networks working inside. The first one is the generator which generates a random image and the second one is the discriminator which identifies whether the image being generated is a real or a fake image. In this paper, we have proposed a system that has been trained on both fake images (GAN Generated images) and real images and will help us in flagging whether the image is plagiarised or a real image.

Keywords – plagiarism, GAN models, artificial replica, Generator, Discriminator, real image

1. Introduction

There are a lot of images that are uploaded daily by content creators. Numerous people use the internet daily, they share a lot of images and gain plenty of information. Content creators play a very important role to develop unique content and people also like their work that's why people invest more time on the internet. Images used in an e-commerce website for fashion-related stuff or image of products, in such areas plagiarism needs to be avoided.

Content creators work hard daily to develop unique content. But few people on the internet, take the images which are created by content creators, and by making few modifications or creating an artificial replica of it, they use that image for their purpose which jeopardizes the aspect of content originality.

There is no system available to catch such plagiarism. It is very necessary to avoid such plagiarism. In this paper, the main focus is on image plagiarism. There are some methods available such as Fourier Based Image Plagiarism Detection Technique. In the Fourier-based Image Plagiarism Detection Technique, the source image is taken and converted into its discrete form, and also all the images in the database are converted into their discrete form and then, Fourier transform is applied to the source image as well as the database image, so at the output it's going to be a matrix. The source image is searched by the searching method into the database and then comparison is made between both images F1 components by computing distances.

But in this paper, the GAN model is used for detecting Image plagiarism, GAN stands for Generative Adversarial Networks. One part of GAN can generate new images and the other part is used for discrimination. One part creates new images and the other part is used only for detecting whether the generated image is real or fake. This Adversarial technique is shining in the industries and gained the attention of academia as well. GAN has effective generating new image samples technique. GAN has improved a lot of Computer Vision-related tasks and achieved remarkable success in that area.

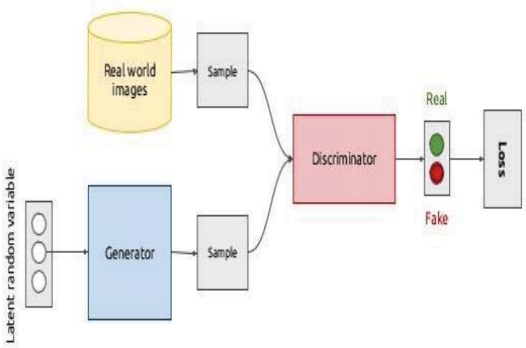

Fig. 1. GAN model architecture

In the proposed system, a CNN model (CNN stands for Convolutional Neural Network) will be used that will be trained on both the real image dataset and the fake image dataset that has been created by the GAN model. This will help in identifying whether the image is plagiarized or not. In the last part of the proposed system, a Flask-based Application is developed. In that application, the user will register and log in to use this system. Then the user will upload an image. The image will undergo feature extraction, and then the generated model results will tell whether the input image is plagiarized or not.

The remaining part of the paper is organized as follows: in section 2, a survey regarding the related work is reviewed. In section 3, a thorough explanation of the proposed methodology along with a system flow diagram and analysis is given. In section 4, results are displayed. In section 5, overall understanding and outcomes are summarized and concluded. In section 6, the future Scope of the system is mentioned. In the last section, references are mentioned.

2. Related Work

This section reviews various research done in the field of GAN models and image plagiarism. Many research people have applied various methods and algorithms in the process to make the system more accurate. For building the proposed system, various works have been studied and gathered information that could make the proposed system more efficient and accurate.

Liang Gonog et al. [1] The authors have introduced the background of the GAN, theoretic models, and extensional variants of GANs, and analyzed whether the variants can further optimize the original GAN or change the basic structures. The authors have also explained the typical applications of GANs. At last, the authors have also summarized the existing problems of GANs and future work of GANs.

Petr Hurtik et al. [2] The authors have proposed a method to search a plagiarised image in a database. The authors have used a tool called FTIP which is based on the F1 transform technique. The algorithm was very fast. It was able to achieve a one hundred percent of success rate for non-cropped images. The authors have claimed that they can achieve the same success rate on cropped images as well but at the expense of high computational time.

Yining Lang et al. [3] The key challenge that the authors have faced is plagiarized clothes are generally modified in a certain region on the real design to escape the supervision by traditional retrieval methods. For this, the authors have proposed a network named Plagiarized-Search-Net (PS-Net) which is based on regional representation, where the authors have utilized the landmarks to assist the learning of regional representations and compare fashion items region by region.

Yang Jie-Cao et al. [4] The authors have analyzed that the adversarial training concept is much more robust in both feature learning and representation. The authors have also mentioned that GAN also has some problems, such as non-convergence, model collapse, and uncontrollability due to a high degree of freedom. Various GAN models like stackGAN, InfoGAN, WGAN, LSGAN, and output are shown in this paper.

Tao Xu et al. [5] The authors have proposed an Attentional Generative Adversarial Network (AttnGAN) that allows attention-driven, multi-stage refinement for a fine-grained text-to-image generation. With the help of this neural net, the model was able to process fine-grained details at separate subregions of the image by paying attention to the relevant words in the natural language description. On top of that, a deep attentional multimodal similarity model is also proposed to calculate a fine-grained image-text matching loss for training the generator.

3. Proposed Work

3.1 Overview of the proposed approach

The dataset that was have used in this paper was created by Oxford University which has 8200 different flower images categorized into 102 different categories. Each category inside this dataset has somewhere between 40 to 258 different flower images. Each image in this dataset has its class label and along with its description. Each set of these flower images inside a particular category will have many overlapping features with some minute differences so that the model can cover up many variations of a specific flower to add variance to the data.

The overall approach of the proposed system is depicted in fig 2. In the proposed system, a flask interface was created for the user to communicate with the system. The user can first log in to the system if the user exists or he/she can sign up if not. After logging into the system, the user can upload the photo for a plagiarism check. After uploading the picture, the pre-processing and feature extraction on that image will be performed wherein all the local and global features of the image are extracted. And then a CNN algorithm is applied on top of it which is

completely trained using the GAN or fake image dataset. This algorithm will then decide whether the uploaded image is a GAN generated i.e. artificially generated or not. The result will be displayed on the website itself. The images in the fake image dataset were of size 8x8 i.e. 64 images in total, this was done so that the user gets more choices to choose his/her image according to their convenience since GAN along with the replicas also creates some noise as a part of output while producing the image. So to make the replicas distinguishable from the noise the images were being generated in the format of 8x8 i.e. 64 images in total.

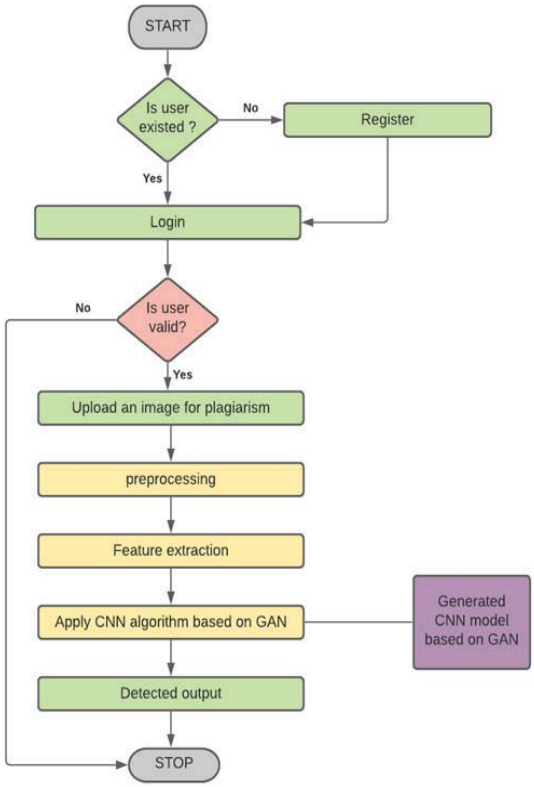

Fig. 2. Flow diagram of the system

3.2 Requirement Analysis

1. Software Requirements

- Operating System: Windows 10, 7, 8.
- Python
- Anaconda Launcher
- Spyder, Jupyter Notebook, Google Colab, Flask
- MySQL

2. Hardware Requirements

- Processor: Intel i5 or above
- RAM: 8 GB or more
- Hard disk: 250 GB or more
- GPU RAM: 4 GB or more

3.3 Design of the system

The architecture of the system is depicted in fig. 3. The architecture is divided into two parts. First is the GAN image generation part. The second is the plagiarism check part. The steps involved are mentioned below:

- In the GAN model, two networks are working inside it. The first is Generator, which produces random images from random values initialized during the first epoch.
- These random images are then sent to the discriminator to identify whether the image generated is fake or real.
- If the discriminator classifies the image as fake, the values inside the latent random variable are adjusted according to the loss calculated in the previous epoch. Based on these new values, the generator now creates a new random image.
- This entire process of creating an image, and then classifying it, is repeated again and again until and unless the generator can fool the discriminator into believing that the image generated is a real image.
- Using this GAN model, a fake image dataset was created which will serve as a dataset for the next phase (plagiarism part).
- In the second phase i.e. plagiarism phase, images from the fake image dataset and real image dataset will be taken one by one, and then pre-processing and feature extraction would be performed.
- A CNN algorithm will be trained and tested on top of it, which help us in classifying the images uploaded.
- On the user end, the user would be requested to upload an image for plagiarism check.
- Then the CNN algorithm which is generated using a fake image dataset classifies whether the image uploaded is a GAN generated or a real image.

In the proposed system, the approach of StackGAN mentioned in [12] was followed with some minor changes for building the GAN model. Fig 4. depicts the architecture of the model mentioned. At first, the text description is converted to text embedding using a text encoder network. This embedding is used for generators and discriminators in both stages as conditioning variables. Before this embedding is sent to the generator network, it's made to pass through a conditional augmentation network. The main for using this CA network is to convert the text embedding into a conditional latent variable. This conditional variable provides some extra information on what to judge and what to create. In this network, the embeddings are broken down and sampled from a vector of means and a vector of variances. A reparameterization trick is also applied to assist in the process of gradient descent (in finding gradients).

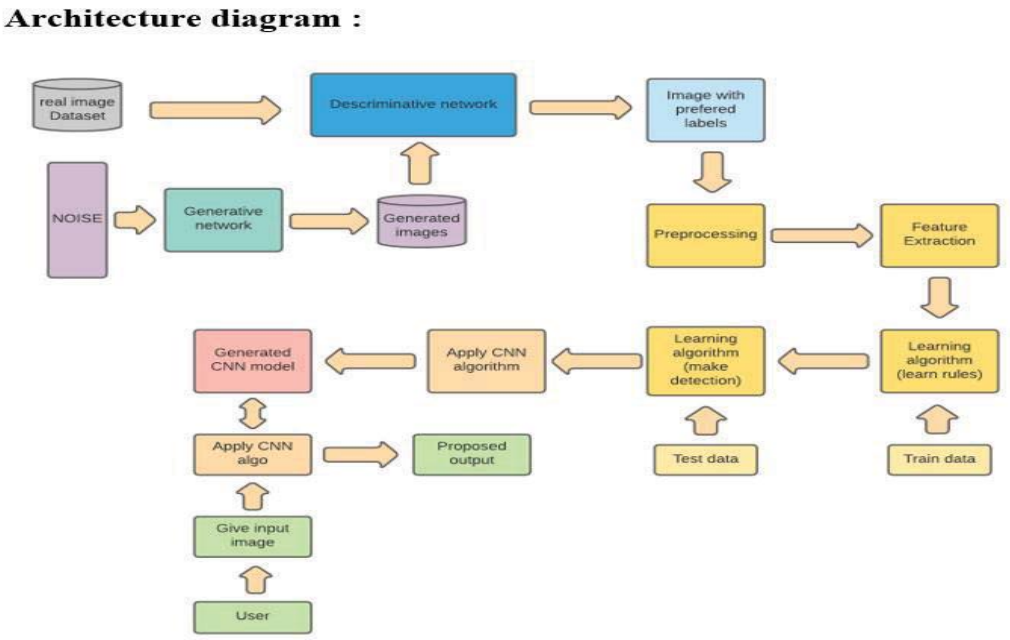

Fig. 3. The architecture of the system

A conditional variable from the previous step is sent to the generator model along with some random noise which is normally distributed with a mean of 0 and variance of 1. This is upsampled in the generator to create stage 1 images or low-resolution images of the format 64x64. These images along with the real images are compared and downsampled and then sent to the discriminator. In the discriminator, the output of the downsampling block is concatenated with the spatially compressed embeddings and sent to the classifier to classify whether the image is a real or fake image.

Low-resolution images created in stage one are sent to the stage two generator along with the text embedding. The second layer has its conditioning augmentation layer. This layer creates its own conditional variable by passing the text embedding through it. The generator in the second layer uses both i.e. stage one images and the conditional variable to encode and then decode the final high-resolution image. At last, stage two generated images, and the real images are compared and downsampled and then passed to the discriminator. The output of the downsampling block is concatenated with the spatially compressed embeddings and sent to the classifier to classify whether the image is a real or fake image.

The loss function that was mentioned in the original paper [6] was the minimax GAN loss function. There are two loss functions in this process. First is the Discriminator loss function which maximizes the average of the log of probability for real images *D(x)* and the log of the inverse probability for fake images *G(z)*.

$$maximize \log D(x) + \log(1 - D(G(z))) \quad (1)$$

The second is the Generator loss function which minimizes the log of the inverse probability predicted by the discriminator for fake images *G(z)*.

$$minimize \log(1 - D(G(z))) \quad (2)$$

The generator loss had a problem, the loss function for the generator saturated very quickly i.e. the generator cannot learn as quickly as the discriminator and if the discriminator job was very easy, the discriminator would win the game within no time and the model cannot be trained efficiently because of it. So to remove this problem, there was a small change that was made in the generator loss function. Now the generator would maximize the log of the discriminator probabilities for generated images *G(z)* instead of minimizing it. So now the Eq 2. would be reformulated as:

$$maximize \log(D(G(z))) \quad (3)$$

With the help of this, better gradient information could be supplied while updating weights, and the training phase would become more stable.

The loss function that was used in this work is Wasserstein Loss Function. It's a modified version of the GAN loss function in which the discriminator does not classify instances i.e. real or fake, rather it outputs a number. It tries to make the number bigger for real instances than for fake instances and there's no constraint on the range of this number. It also includes two loss functions. First is the discriminator loss function which maximizes the difference between its output on real instances *D(x)* and its output on fake instances *G(z)*.

$$maximize (D(x) - D(G(z))) \quad (4)$$

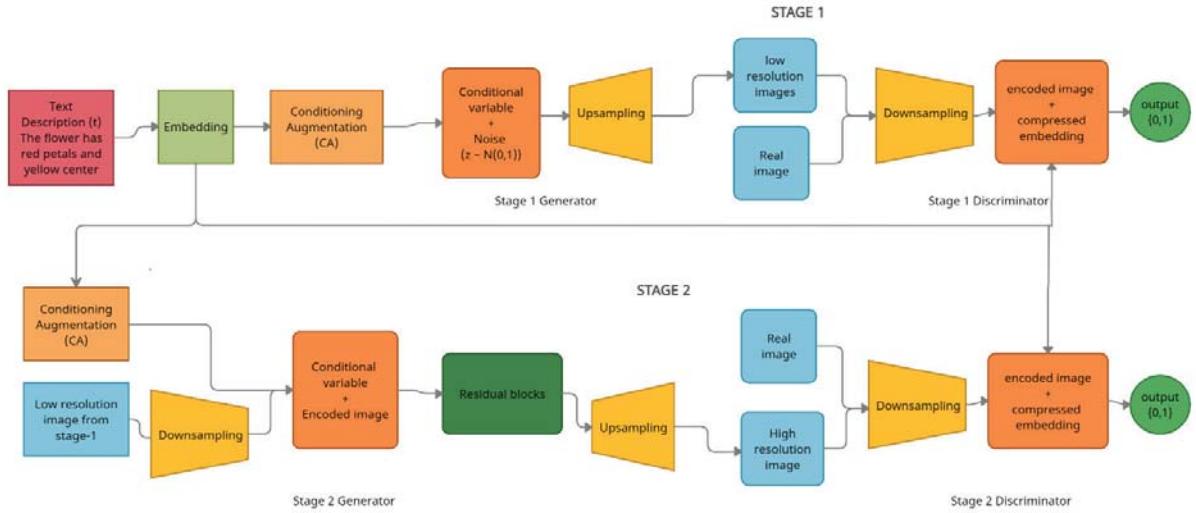

Fig. 4. Stacked GAN model architecture

For the generator loss function, it maximizes the discriminator's output for its fake instances *G(z)*.

$$maximize\ D(G(z)) \qquad (5)$$

This loss function is less vulnerable to getting stuck than the minimax-based GAN loss and it also avoids problems with vanishing gradients. One more significant advantage this has is rather than using cross-entropy as a metric it uses earth mover distance as a metric which is considered as a true metric for measuring the distance in a space of probability distribution. The generator is trying to maximize the value because it would only then help the generator to convert its fake image into a real image since the higher is the number, the higher is the probability of the fake image being classified as a real image.

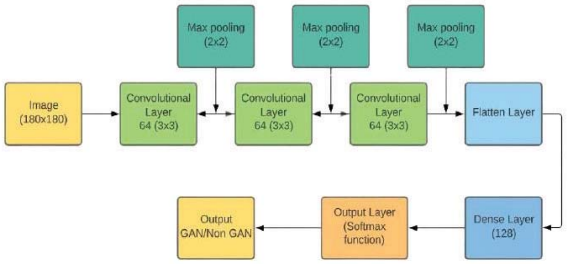

Fig. 5. CNN model used in the second part of the system i.e. plagiarism system

Fig 5. depicts the CNN model used in the plagiarism system. The images given from the first part of the system i.e. GAN part will serve as a training dataset for the second part of the system. The dataset for this part includes both GAN-generated images as well as the real images (grouped in 8x8 format). These images are then pre-processed. In the pre-processing step, the images are read from their respective file paths and then rescaled to the size of 180x180 format. After loading the images, the data is divided into two sets – images and their class labels i.e. GAN or Non-GAN. The images are then scaled down by a factor of 255 to reduce the complexity of the training process (which helps in easing out the calculations).

Then a model with three convolutional layers and three max-pooling layers, a dense layer as hidden layers, and ReLU as the activation function was created. For the output layer, a softmax activation function was used which will help us in classifying whether the image is a GAN image or a Non-GAN image. The training data is then fitted on top of this model. In the proposed system, “Adam” as the optimizer function was used. The loss function used is “sparse categorical cross-entropy”. The validation split was chosen 0.2 and the model was trained for seven epochs.

3.4 Preprocessing

At first, all of the class ids are supposed to be loaded which are stored in a pickle file. Then, load the filenames, which are also stored in a pickle file. Then, the text embeddings that have been created using a text encoder network are supposed to be loaded which are also present in a pickle file. Next, the bounding boxes are created to extract objects from the crude images. This function will return a dictionary of file names and their corresponding bounding boxes. After this, a function is created to load the image and crop it around the bounding box according to the image size passed as an argument to that function. In this work, an image size of 64 has been used as the argument for this function. At last, all of the above methods are combined i.e. all the images, their class labels, and corresponding embeddings to get the dataset which will then be used for the training process.

3.5 Training phase

At first, the values of the hyperparameters required for this training process are initialized. The learning rate for generator and discriminator was initialized to 0.0002. Adam optimizer is defined for the training. The learning decay rate was set to 600. Then, two tensors with labels real and fake were created. These are required for the training of generator and discriminator. In this work, label smoothing was also used in this step. Next, the dataset from pre-processing step is used for training purposes. For the next step, a "for" loop was created which should execute for the number of times given in the number of epochs variable. Inside this loop, another "for" loop will execute based on the number of batches specified. Next, a mini-batch of data is sampled for the current iteration. In this iteration, a noise vector is initialized, a batch of images and batch of embeddings is selected and images are also normalized.

After this, the generator network is used to generate fake images. For this step, first, the Stage-I generator is used to generate low-resolution fake images, then the Stage-II generator network is used to generate high-resolution images. Then a compressor model is used to compress the embedding. Next, train the discriminator model on the fake, the real images. Following this, training of the adversarial model will be done, which is a combination of generator and discriminator models. And then, after every epoch, images are generated and saved and even the losses are computed and saved for evaluation purposes. At last, the model (weight values) is saved as well.

4. Results

4.1 Phase – I (Text to Image)

```
the bright red flower features a pistil that
is partially obscured.
the petals of this flower are red with a
short stigma
this flower is red in color, with petals
that are connected to each other.
the flower has petals that are red with a
white center.
the flower is made of a dark red bloom with
a small yellow center.
this flower has bright red petals, a slight
trumpet shape, and a yellow center.
the red petals are in the shape of a flat
circle on the edges and form a vase around
yellow stamen.
this flower has petals that are red and has
yellow style
this flower has red petals as well as a
white stamen.
this flower has a large funnel-shaped bright
red petal with generally rounded edges and
small white stamen.
```

Fig. 6. Text description

Fig. 7. Corresponding flower image i.e. for the above text description

Fig. 8. GAN generated image for the above text description

Fig 6. depicts text descriptions that will be given as input to the GAN model. Fig 7. depicts the corresponding image for the text description. Fig 8. Depicts 8x8 GAN generated image that will be produced by the GAN model as the output.

Some of these images are very similar to the original image i.e. we can see that some flowers have the same dark red petals and yellow center as the original one, and even the shape and the distribution of these petals around the center is very similar to the real one. Some flowers are not exactly similar to the original flower but have some features in common like the yellow center and the vase-like distribution of petals around the center. But there are some flowers in this plot that are not at all similar to the real flower, these are noises that the GAN model has generated. These noises can be reduced to a significant extent by increasing the number of epochs in the training phase.

4.2 Phase – II (Plagiarism check)

Fig. 9. 8x8 Real image

Fig. 10. The result on the web app (Real Image)

Fig 9. depicts 8x8 real image i.e 64 real images clustered together in 8x8 format. Fig 10. depicts the result of the web app for the real image. We can observe from the above result that the image has been classified as a non-gan or not artificially produced image.

Fig. 11. 8x8 GAN generated image

Fig. 12. The result on the web app (GAN generated image)

Fig 11. depicts 8x8 GAN generated image i.e 64 images clustered together in 8x8 format. Fig 12. depicts result on web app for the GAN generated image. We can observe from the above result that the image has been classified as GAN or artificially produced image.

The CNN (Convolutional neural network) algorithm was trained for a plagiarism check system with about 120 images from both the fake image dataset (created by the GAN model) and the real image dataset. The validation split was set to 0.2 i.e. 20 percent of these images were taken in the validation set. The validation accuracy at the end of the training was observed to be **97** percent. Accuracy here is defined as the number of correctly predicted points out of all the data points. The data points here belong to two classes i.e GAN and non-GAN.

5. Conclusion

In this paper, the major focus was on the image plagiarism problem, i.e. checking whether the image is artificially produced (GAN produced) or not. We proposed plagiarism detection system based on GAN using the CNN technique which was mainly used in the pre-processing step. In this project, we first apply the GAN to the dataset and this output will be sent as an input to the CNN. Then this model will predict whether the image is a GAN generated or not. This will help a lot in identifying plagiarized images since many modern plagiarism systems are not able to detect such kinds of artificially created images.

6. Future Scope

We have trained the GAN model on the Flowers dataset for this project but the same architecture can be extended to generate and detect any category of the image by training the model on that category of data. The proposed system can be used on any platform where there is a high chance of using plagiarized images i.e. artificially generated images, specifically on blog websites and in the fashion industry. We were able to achieve good results by training the GAN model on our machine but the accuracy of the system can also be further extended by a significant amount by training the model on machines that have very high computational power.

References

[1] L. Gonog and Y. Zhou, "A Review: Generative Adversarial Networks," 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 505-510, (2019)

[2] P. Hurtik and P. Hodakova, "FTIP: A tool for an image plagiarism detection," 2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR), Fukuoka, Japan, , pp. 42-47, (2015)

[3] Y. Lang, Y. He, F. Yang, J. Dong, H. Xue, "Which is Plagiarism: Fashion Image Retrieval based on Regional Representation for Design Protection," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2595-2604, (2020)

[4] Y. J. Cao, L. L. Jia, Y. X. Chen, N. Lin, C. Yang, B. Zhang, Z. Liu, X. X. Li, H. H. Dai, "Recent Advances of Generative Adversarial Networks in Computer Vision," in IEEE Access, vol. 7, pp. 14985-15006, (2019)

[5] T. Xu , P. Zhang , Q. Huang , H. Zhang , Z. Gan, X. Huang, X. He, "AttnGAN: Fine- Grained Text to Image Generation with Attentional Generative Adversarial Networks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1316-1324, (2018)

[6] I. J. Goodfellow, J. P. Abadie , M. Mirza, B. Xu, D. W. Farley, S. Ozair, A. Courville, Y. Bengio, "Generative Adversarial Nets," arXiv preprint arXiv:1406.2661, vol: 1, (2014)

[7] K. Ak, A. Kassim, J. Lim and J. Y. Tham, "Attribute Manipulation Generative Adversarial Networks for Fashion Images," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 10540-10549, (2019)

[8] I. Amerini, L. Galteri, R. Caldelli and A. Del Bimbo, "Deepfake Video Detection through Optical Flow Based CNN," 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 1205-1207, (2019)

[9] M. Chen, Y. Qin, L. Qi and Y. Sun, "Improving Fashion Landmark Detection by Dual Attention Feature Enhancement," 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 3101-3104, (2019)

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol: 1, pp. 886-893, (2005)

[11] K. E. Ak, J. H. Lim, J. Y. Tham and A. A. Kassim, "Efficient Multi-attribute Similarity Learning Towards Attribute-Based Fashion Search," 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1671-1679, (2018)

[12] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang and D. Metaxas, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks," 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 5908-5916, (2017)