Pseudonymization of Text and Image Data to Provide Confidentiality

Manthan C S[1], Roopa K[1,✉], Bindu H S[1], B M Apoorva[1], Mala D Madar[1]

[1] Department of Electronics and Communication Engineering,
The National Institute of Engineering,
Mysuru-570008, Karnataka, India

mnthn.ai@gmail.com, ✉roopakodnad@gmail.com,
https://orcid.org/0000-0002-2180-6447
binduhs2016@gmail.com,
bmapoorva.6@gmail.com, malajnv@gmail.com

Abstract. It is important to maintain secrecy of one’s valuable data when it is loaded in the internet. Unintended parties can view the data, resulting in breach of confidentiality. Another serious concern is the possibility of misuse of data to the advantage of an unintended party. To overcome these problems, this paper presents an algorithm developed and implemented to make the data that is uploaded to the servers, pseudo anonymous (pseudonymous). Typically, user sensitive information exists as either texts or images. Hence, such data are used as input to the developed algorithm. From the textual data, the algorithm can detect key words called tags; such as names, places, financials, and replace the original tag with newly created false tags. Detection of the tags is done by the existing Natural Language Processing (NLP) algorithms. For image input, the algorithm detects face and replaces it with a falsified one. This is done by using Generative Adversarial Networks (GANs). Further, the original or true image is hidden in the falsified image using SteganoGAN. This image will be sent over the network. At the receiver, both anonymised text and image data are deanonymised as per the developed algorithm.

Keywords: Confidentiality · Pseudonymous · Text · Image · Natural Language Processing · Generative Adversarial Networks.

1 Introduction

There has been a rise in the number of private data breaches in online systems and services [1]. Actually, there are strict rules on personal data breach notifications [2]. Some of the major data breaches of recent times [3] are either intentional or otherwise. But solution is inevitable. Several users have opted for new methods such as data anonymization to protect their information. Data anonymization is a process by which personal data is irreversibly altered. The anonymised output will denote some meaning but with the sensitives replaced by different data. Even if this modified data is used for training, development, testing and analytics, it will not reveal the identity of the actual person or source. According to Balaji Raghunathan [4], Data anonymization is a technique to de-identify sensitive data while at the same time preserve its format and data type. The masked data may be real or random data. Anonymization algorithms have to transform the data into a form that provides a privacy guarantee with the minimum possible distortion of the original data. Reference [5] describes techniques of data anonymization by generalizing the data and by adding noise to data. A few other techniques are discussed in references [6, 7].

De-anonymization is the reverse process in which anonymous data is cross-referenced with other data sources to re-identify the anonymous data source. The process of obscuring data with the ability to re-identify it later is also called pseudonymization [8, 9].

Recent Anonymization and Pseudonymization techniques use Generative Adversarial Networks (GANs) [10]. GAN is used in image identification, generation and classification. GANs are basically made up of a system of two competing neural network models which compete with each other and are able to analyze, capture and copy the variations within a dataset. In GANs, there is a Generator and a Discriminator. The Generator generates fake samples of data while the Discriminator, on the other hand, tries to distinguish between the real and fake samples.

This paper discusses new algorithms formulated using existing concepts to provide pseudonymization of text and face image data, giving confidentiality for those involved. To achieve this, different forms of GANs are used. Python programming language [11] is used for algorithm implementation. A few GitHub resources are also used with appropriate referencing.

The motivation for the present approaches stems from the following: i) Natural Language Processing (NLP) bridges the gap between human language and machines by text classification using pretrained deep learning models. ii) SpaCy statistical model is used here as it is an open-source library for NLP and can handle large volumes of text. iii) Text pseudonymization algorithm is based on an idea of representing text into an image. iv) Steganography offers a feasible alternative or a supplement to encryption. Use of GAN based steganography helps in powerful construction of generators and discriminators. Moreover, GANs are a relatively new concept in Machine Learning, introduced for the first time in 2014.

2 Related Work

Hakon Hukkelas et al. [12, 13] discussed a model called Deep Privacy making use of a conditional Generative Adversarial Network (GAN). They evaluated their model by anonymizing the validation set of the WIDER-Face dataset [14]; then performed face detection of the anonymized images to measure the impact of anonymization. The conditional GAN they used, generated images based on the surrounding of the face and sparse pose information. They used seven key points to describe the pose of the face; left/right eye, left/right ear, left/right shoulder, and nose.

Karras T. et al. [15] proposed a generator architecture for GANs leading to an automatically learned and unsupervised separation of high-level attributes.

Anonymization of text in particular has practical applications in maintaining patient records in electronic form which are useful for research. Anonymisation or de-identification allows research uses of clinical data [16, 17, 18, 19]. Thomas Neubauer and Johannes Heurix [20] discuss a solution which provides a guaranteed data privacy allowing primary and secondary use of the data at the same time. Rahul et al. [21] present a review of various Natural Language Processing (NLP) based Machine Learning approaches for summarizing the text data.

In the work reported in this paper, text as well as image data are used as input to the pseudonymizer. It is called so because, at the receiver, de-anonymization is done to retrieve the true information by the intended user.

NLP is used to identify text tag words. Additionally, an algorithm is written for text anonymization. For image anonymization, face images are used. To anonymize face, a fake face has to be generated which would mask the original face. This is done by using GAN [10], StyleGAN [22] or StyleGAN2 [23, 24, 25].

Though the terms anonymization and pseudonymization are used interchangeably to describe the implemented work for ease of use of terminologies, it is actually the pseudonymization that has been carried out since the anonymised data are also de-anonymised to get back their true forms as shown in Fig. 1.

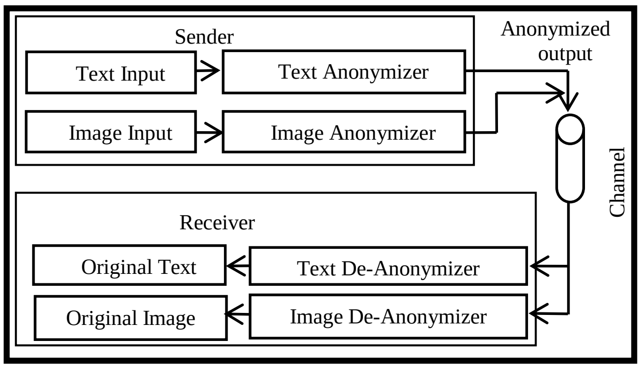

Fig.1. Overall diagram of the anonymizer

3 Methodology

3.1 Implementation of Text Anonymisation

Textual data can contain a lot of user related information such as names, places and contact details. Such user sensitive data can be identified using NLP [26, 27, 28] and categorized according to names, locations and so on. In this work, the main focus for text input is on anonymization of location based tags. A library called SpaCy [29] is available in Python to carry out different NLP tasks.

The steps in text anonymization are:

i) Identify all the tags that are related to location using SpaCy and NLP.
ii) Save all identified tags (original tags) in ‘detected tags’ array.
iii) Substitute the original tags with other fake tags randomly from a table containing all the city names in India [30].
iv) In the text data input, replace all detected tags with fake tags.
v) Perform encryption [31] of all the tags. The result turns out to be an image with concentric circles.

Two fictitious examples are shown in Table 1, with the identified tag, substituted tag and the output text with the replaced tags (keywords).

Table 1. Examples for tag word identification and substitution

Sl. No.	Input	Tags detected by Spacy	Substituted tag for the detected tag	Output with replaced keywords
1	Bob told Alice that he would go to Bombay and Delhi	Bombay, Delhi	Kolkata, Bhopal	Bob told Alice that he would go to Kolkata and Bhopal
2	I am going to Mysuru this weekend	Mysuru	Bhubaneshwar	I am going to Bhubaneshwar this weekend

To encrypt a tag, each of its character is used. Before the actual encryption, a blank space is added between the detected tag (original tag) and the replaced tag and saved in a variable, ‘**joint_tag**’ as,

$$\text{joint\_tag} = \text{original tag} + \text{“ ”} + \text{replaced tag} \tag{1}$$

Inclusion of a space in between the original and replaced tag helps in retrieving the original text during de-anonymization.

An algorithm to encrypt ‘**joint_tag**’ is shown in Fig. 2. A circle is drawn whose radius is equal to the length of the ‘**joint_tag**’. Each character in the ‘**joint_tag**’ is to be plotted as a point on this circle. Algorithm is designed such that the number of points on a circle is equal to the radius of that circle. e.g., if radius of the circle is 3, then there are 3 points on that circle.

While anonymizing multiple tags, there might be a chance that a new ‘**joint_tag**’ will have the same length as that of a previously generated ‘**joint_tag**’. This will lead to an overlap of data on the circle as the radii become the same. Whenever such cases arise, ‘**joint_tag**’ is padded with another blank space at the end, and then radius value is incremented. These extra spaces are removed while decrypting [30] to get the original data back.

The color of the point depends on the Unicode value (‘**unicode_value**’, in Fig. 2) returned by the ‘**ord**’ method in Python for a character in the ‘**joint_ tag**’. This value is added to a colour component in one of the 3 colour spaces. The colour spaces are declared in the array as shown in (2). The assigned colour space to a character depends on radius value as shown in (3),

$$\text{color\_spaces} = [\text{‘RGB’}, \text{‘HSV’}, \text{‘CMYK’}] \tag{2}$$

$$\text{assigned\_colorspace} = \text{color\_spaces}[\text{radius} \,\%\, 3] \tag{3}$$

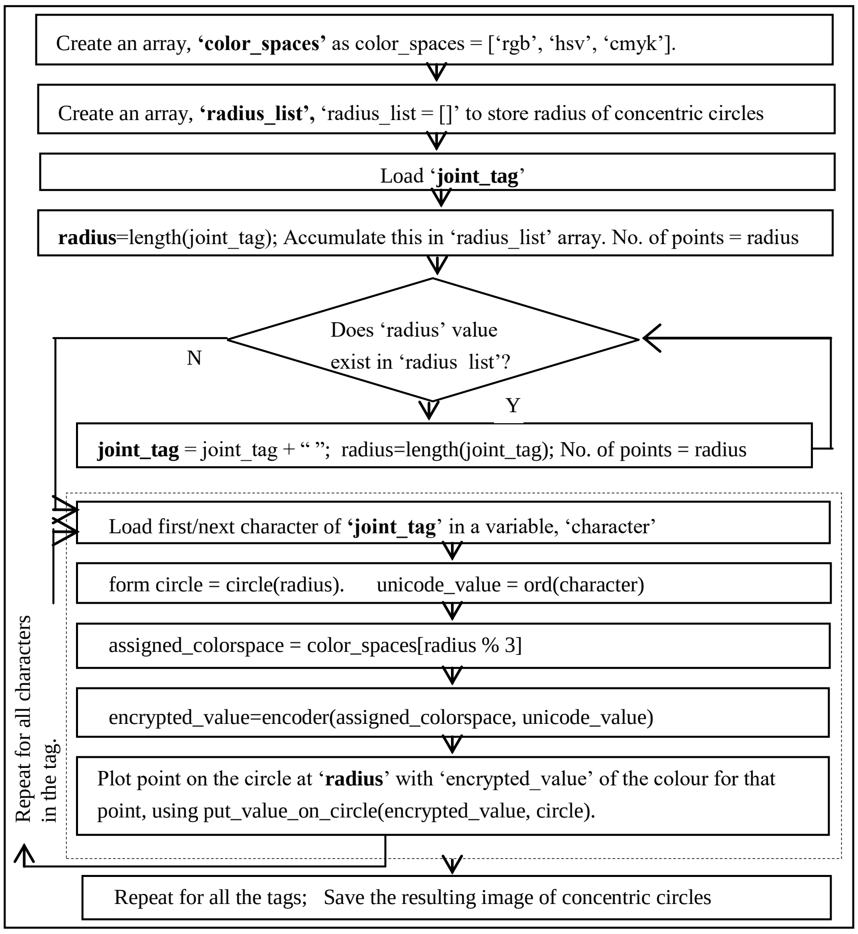
Create an array, ‘color_spaces’ as color_spaces = [‘rgb’, ‘hsv’, ‘cmyk’].
Create an array, ‘radius_list’, ‘radius_list = []’ to store radius of concentric circles
Load ‘joint_tag’
radius=length(joint_tag); Accumulate this in ‘radius_list’ array. No. of points = radius
Does ‘radius’ value exist in ‘radius list’?
N
Y
joint_tag = joint_tag + “ ”; radius=length(joint_tag); No. of points = radius
Load first/next character of ‘joint_tag’ in a variable, ‘character’
form circle = circle(radius). unicode_value = ord(character)
assigned_colorspace = color_spaces[radius % 3]
encrypted_value=encoder(assigned_colorspace, unicode_value)
Plot point on the circle at ‘radius’ with ‘encrypted_value’ of the colour for that point, using put_value_on_circle(encrypted_value, circle).
Repeat for all characters in the tag.
Repeat for all the tags; Save the resulting image of concentric circles

Fig. 2. Flow diagram to encrypt ‘**joint_tag**’

If ‘radius % 3’ in RHS of (3) evaluates to 0, ‘RGB’ colour space is chosen, if it evaluates to 1, ‘HSV’ colour space is chosen, else, if it evaluates to 2, ‘CMYK’ colour space is chosen. Which particular colour component in the ‘**assigned_colorspace**’ is added with the ‘**unicode_value**’ depends on which character in ‘joint_tag’ is being encrypted. e.g., if first character in ‘joint_tag’ is to be encrypted, first colour component in the ‘**assigned_colorspace**’ is added with the ‘**unicode_value**’. As another example, if 3rd character is being encrypted and the selected colour space (calculated as per (3)) is say, ’CMYK’ , the ‘**unicode_value’** is added to the 3rd colour component of CMYK which is Y’.

If desired, the algorithm may be modified to select different colour spaces and colour components. Use of multiple colour spaces adds another level of security as all the values of the string cannot be found properly in a single colour space. Plotting is done in Python code with the help of a package known as **matplotlib** [32].

The steps involved in encryption are explained below through an example:

i) Original tag: Mysuru, Substituted tag: Bhubaneshwar.
ii) **joint_tag** = Mysuru Bhubaneshwar (Note the space in between the city names).
iii) length(Mysuru Bhubaneshwar) = 19, including the middle space.
iv) Hence, a circle of radius 19 is to be plotted with 19 points on it.
v) The index for ‘**color_spaces**’ is 19 modulo 3 which is 1. Therefore, **assigned_colorspace** = color_spaces[1] which is ‘HSV’ since the array indexing starts from zero.
vi) The ‘**unicode_value**’ of ‘M’ is added to the Hue (H) part of HSV.
vii) This colour value is plotted on the circle. Similar steps are carried out for subsequent characters in ‘**joint_tag**’.
viii) The points are plotted on the circle in a clockwise direction, with the initial point being plotted on the positive Y-axis.
ix) The process is repeated for every character and for all tags. Thus concentric circles of tags are generated with a common center at the origin (0, 0). The output of text anonymization is shown in the ‘RESULTS’ section.

3.2 Implementation of Text De-Anonymisation at the Receiver

The input to the decrypter is a substituted message and an image consisting of concentric dotted circles having dots of different colours. For text deanonymization, the following steps are used:

i) Detect the origin or center of the circles.
ii) Group all points that are equidistant from the center. Number of groups formed will be the number of tags that were replaced by fake data in the string.
iii) The point that lies on the Y-axis is the starting point and continues clockwise. The initial color space is found by finding the modulo of the total number of points or equivalently, the length at which the points were found.
iv) The ‘**unicode_value’** of the character that was encoded in the point is obtained by decrypting the point in the color space.
v) The ‘**unicode_value’** is converted back to the letter.

vi) The process is repeated for all the points and the decrypted letters are pushed into an array. The elements of the array are joined to get a decrypted string which has the original tag and the replaced tag with a space in between.
vii) The decrypted string is split at the space to get the original tag and the replaced tag. The replaced tag is found in the received message and replaced by the original tag to get the original message back.

3.3 Working of Anonymizer for Pictorial Input Data

In this work, the focus is on anonymising images that contain faces. Original face (True image, in Fig.3) is anonymised using a fake face. With the help of a StyleGAN, highly realistic photos of anonymised faces can be generated.

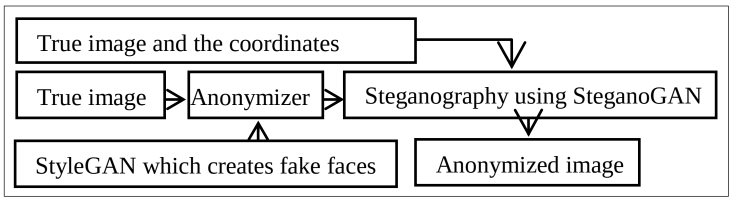
True image and the coordinates
True image
Anonymizer
Steganography using SteganoGAN
StyleGAN which creates fake faces
Anonymized image

Fig. 3. Block diagram of the image anonymizer.

Fig. 3 shows the block diagram of the image anonymizer which involves the following steps:
i) The StyleGAN generated fake image is given to the anonymizer.
ii) The anonymizer identifies the convex hull of the faces using the '**dlib**' package provided by Python, in both the original image and StyleGAN generated image by triangulating over the faces by a method called Delaunay triangulation [33].
iii) The coordinates where the convex hull of only the face is found in the entire true input image, are saved. Various other details (other than face) which may be present in the true input image are not considered for processing. They remain as it is. This step is shown in Fig. 4(a). In the figure, the convex hull is indicated with a rectangle bounding box. The image shown is representative and not of original size for the actual coordinates of (X,Y,W,H) = [143,601,130,150]. It is also possible to take any other information in the image as input, instead of face. In this work, face of the input image is considered for anonymization as it is a prime feature.
iv) The true face image alone as shown by the bounding box in Fig. 4(a) and its convex hull coordinates [143,601,130,150] are input to steganography using SteganoGAN. The other input to SteganoGAN is the Anonymizer output as shown in Fig. 3 and its image is shown in Fig. 4(b).
v) In the convex hull, the triangles of one face are imposed on another by matching the vertices. This produces a hard output of the face. This makes sure that the expression of the original face is also preserved.

vi) The hard imposition after detecting the convex hull of both the faces is smoothened for brightness and contrast using OpenCV's seamless clone algorithm [34] which gives a realistic fake face.

vii) The original convex hull [35] is hidden inside the smoothed image using steganographic technique, SteganoGAN [36, 37]. Image steganography is used to hide messages inside pictures. In this work, true image (face) is hidden in the anonymysed image. SteganoGAN is a tool for creating steganographic images using adversarial training. Optionally the convex hull can be encrypted with other cryptographic algorithms such as AES (Advanced Encryption Standard) and then sent for steganography at the cost of additional computations. This process generates the final anonymised image. The output of image anonymization is shown in the 'RESULTS' section.

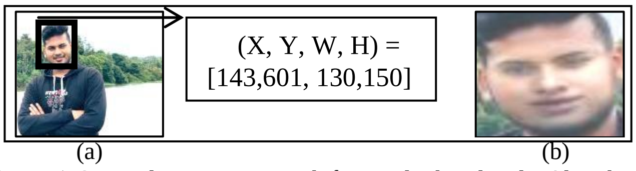

Fig. 4. a) Original input image with face and other details. Also shown is the convex hull identified for face at (X,Y,W,H) = [143,601,130,150].
b) Original face Anonymised by overlapping with a StyleGAN generated image.

3.4 Retrieval of Original Face from the Pseudonymized Image at the Receiver

The image received contains the original face and the coordinates where it was hidden (steganographed) in the anonymized face. Thus, it is sent to the SteganoGAN which retrieves the original face and coordinates.

The coordinates retrieved is an array containing the X, Y points and the width and height of the face found. (X, Y) specify the starting point of the original face in the anonymized face. This is simply used to retrieve the original face from the anonymized image according to the width and height it was originally steganographed in. This will output the original image as it was, before anonymization and the retrieval is complete.

4 RESULTS

The algorithm was tested for a total of 50 text input data with single and multiple real city names. It was also tested for a total of 50 images dataset generated by the authors. The system worked well for both data sets in terms of anonymization and deanonymization. The programs were executed in Apple MacBook Air system with 1.8 GHz Dual-Core Intel i5, 8 GB, 1600 MHz DDR3. The average execution times were 656 ms and 1453 ms for anonymization of text and image respectively. The average execution times were 224 ms and 183 ms for de-anonymization of text and image respectively. The execution times are within the acceptable limits for general applications.

An example of an Anonymized text output in the form of an image with two concentric circles with radii 19 units and 11 units is shown in Fig. 5. Fig. 6 shows the results of face anonymization. Fig. 6(a) shows the original face image input to the anonymizer. Its triangulated image is shown in Fig. 6 (b). Figs. 6(c), (e), (g) and (i) are taken from StyleGAN generated images which are input to the anonymizer. Figs. 6(d), (f), (h) and (j) are the respective anonymized output images from SteganoGAN. De-anonymization will result in the true image of Fig. 6(a).

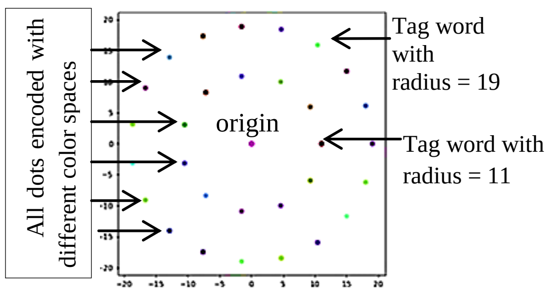

Fig. 5. Results of text anonymization

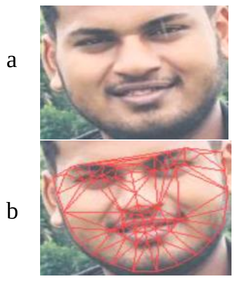

Fig. 6: Results of face anonymization. a) Input face image b) Face triangulated

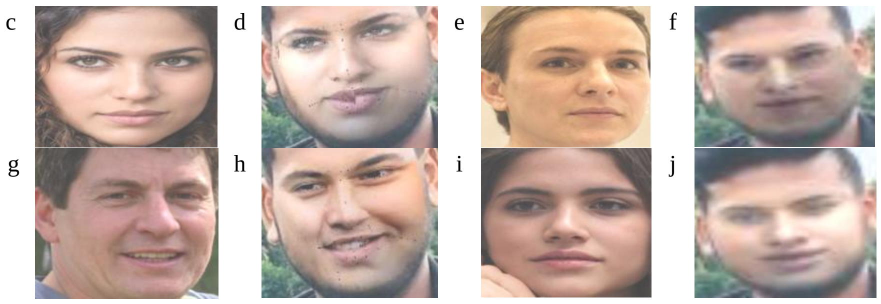

Fig. 6: Results of face anonymization contd. c), e), g), i) are taken from StyleGAN generated images which are input to the anonymizer; d), f), h), j) are the respective anonymized output images from SteganoGAN.

5 CONCLUSION

This paper discussed a pseudonymizer for text and image input data. This software system will be useful in applications like sending medical data of a patient over public network or sharing confidential data over social networking sites. The system worked well for a total of 100 representative inputs randomly taken inclusive of text and image. However, the limitations are that, as the textual algorithm uses an NLP model based on probability to detect tags, there is a possibility that the model might miss out some tags when a different, test dataset is used and in the case of image input data, the posture of the face is not specifically considered here for validation. Hence, for posture variant image data, the algorithmic performance evaluation has to be carried out separately.

As future scope, the authors suggest the following:

i) Provide option to select multiple types of tags, e.g., city, personal names and numbers.
ii) Provide option to choose entire face, only eyes, only hair style or background etc.
iii) Provide flexibility in the extent to which anonymization is to be performed. A suggestive measure for this would be, in triangulation, the precision must be controlled by providing a variable parameter so that the user can preview and select the % anonymization which is satisfactory for a particular application.
iv) Evaluate the quantitative performance parameters showing the difference between the original and the anonymized image.

References

1. https://www.enisa.europa.eu/topics/data-protection/personal-data-breaches, accessed June 2020.
2. https://www.i-scoop.eu/gdpr/personal-data-breach-notification/, accessed June 2020.
3. https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html, accessed June 2020.
4. Balaji Raghunathan.: The Complete Book of Data Anonymization: From Planning to Implementation. Infosys Press and CRC Press, 2013.
5. https://policies.google.com/technologies/anonymization?hl=en-US, accessed June 2020.
6. https://www.imperva.com/learn/data-security/anonymization/, accessed June 2020.
7. https://iapp.org/media/pdf/resource_center/Guide_to_Anonymisation.pdf, accessed June 2020.
8. Schwartmann, Weiß (eds.).: White Paper on Pseudonymization. Drafted by the Data Protection Focus Group for the Safety, Protection, and Trust Platform for Society and Businesses in Connection with the 2017 Digital Summit, 2017.
9. Pseudonymisation Techniques and Best Practices, November 2019, Available on https://www.enisa.europa.eu/publications/pseudonymisation-techniques-and-best-practices, accessed June 2020.
10. https://developers.google.com/machine-learning/gan, accessed June 2020.
11. https://www.python.org/, accessed June 2020.
12. Hakon Hukkelas, Rudolf Mester, Frank Lindseth.: DeepPrivacy: A Generative Adversarial Network for Face Anonymization. Advances in Visual Computing: 14th International Symposium on Visual Computing, ISVC 2019, Springer International Publishing, pp. 565—578, 2019. Available on https://arxiv.org/pdf/1909.04538.pdf, accessed June 2020.
13. www.github.com/hukkelas/DeepPrivacy, accessed June 2020.
14. Yang S, Luo P, Loy C. C., Tang X.: WIDER FACE: A Face Detection Benchmark. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5525-5533, Las Vegas, NV, 2016. doi: 10.1109/CVPR.2016.596.

15. Karras T, Laine S, Aila T.: A Style-Based Generator Architecture for Generative Adversarial Networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4396-4405, Long Beach, CA, USA, 2019. doi: 10.1109/CVPR.2019.00453.
16. Cardinal R. N.: Clinical records anonymisation and text extraction (CRATE): an open-source software system. BMC medical informatics and decision making, 17(1), 50, 2017. doi: 10.1186/s12911-017-0437-1, available on https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5405523/, accessed June 2020.
17. https://crateanon.readthedocs.io/en/latest/nlp/index.html, accessed June 2020.
18. Mamede N, Baptista J, Dias F.: Automated anonymization of text documents. 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 1287-1294, Vancouver, BC, 2016. doi: 10.1109/CEC.2016.7743936.
19. Ribeiro S. L., Nakamura E. T.: Privacy Protection with Pseudonymization and Anonymization in a Health IoT System: Results from OCARIoT. In: 2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE), pp. 904-908, Athens, Greece, 2019. doi: 10.1109/BIBE.2019.00169.
20. Thomas Neubauer, Johannes Heurix.: A methodology for the pseudonymization of medical data. International Journal of Medical Informatics. 80(3), 190-204 (2011).
21. Rahul, Adhikari S, Monika. NLP based Machine Learning Approaches for Text Summarization. In: 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), pp. 535-538, Erode, India, 2020. doi: 10.1109/ICCMC48092.2020.ICCMC-00099.
22. https://machinelearningmastery.com/introduction-to-style-generative-adversarial-network-stylegan/, accessed June 2020.
23. https://medium.com/analytics-vidhya/from-gan-basic-to-stylegan2-680add7abe82, accessed June 2020.
24. https://towardsdatascience.com/stylegan2-ace6d3da405d, accessed June 2020.
25. https://neurohive.io/en/news/stylegan2-new-improved-stylegan-is-the-new-state-of-the-art-model/, accessed June 2020.
26. https://machinelearningmastery.com/natural-language-processing/, accessed June 2020.
27. https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1, accessed June 2020.
28. https://www.wonderflow.co/blog/natural-language-processing-examples, accessed June 2020.
29. https://spacy.io/, accessed June 2020.
30. https://simplemaps.com/data/in-cities, accessed July 2020.
31. William Stallings.: Cryptography and Network Security Principles and Practice. 6edn. Pearson, 2014.
32. https://matplotlib.org/, accessed July 2020.
33. https://in.mathworks.com/help/matlab/math/delaunay-triangulation.html, accessed June 2020.
34. https://www.learnopencv.com/seamless-cloning-using-opencv-python-cpp/, accessed June 2020.
35. https://www.cs.auckland.ac.nz/software/AlgAnim/convex_hull.html, accessed June 2020.
36. https://arxiv.org/abs/1901.03892, accessed July 2020.
37. https://github.com/DAI-Lab/SteganoGAN, accessed July 2020.