Image Augmentation Using Deep Convolutional Generative Adversarial Network

Muhammad Sabihul Hasan
Bachelor of Science in Computer Science
Habib University
Karachi, Pakistan
mh04387@st.habib.edu.pk

Syed Hammad Ali
Bachelor of Science in Computer Science
Habib University
Karachi, Pakistan
sa04324@st.habib.edu.pk

***Abstract*—The domain of deep learning revolves around solving problems using existing data. However, at most times, the data is not enough in order for the deep learning networks to perform well. At other times, there is enough data for the entire problem set but the division of data is inadequate hindering the deep learning networks to perform to their potential. Such is the problem that arises when carrying out Facial Emotion Classification using FER2013 dataset, which has a disparity of images amongst different classes in the dataset which hinders efficient classification of the dataset. In this paper, we have proposed a solution to counter this deficit in datasets by augmenting the dataset using Generative Adversarial Networks. The GAN architecture proposed in this paper is Deep Convolutional GAN. A CNN classification model has also been proposed in order to evaluate the quality of images generated using the DCGAN. Classes with a lower accuracy on initial classification have undergone image augmentation by generation of 1000 images of those classes using our DCGAN. The classification model is re-trained on augmented data to show the changes in accuracy.**

***Index Terms*—Computer Vision, DCGAN, Deep Learning, Facial Emotion Recognition, Image Augmentation**

I. Introduction

The efficiency of machine learning models depends upon the quality of datasets that they are trained on. To build a reliable model, it is imperative that the dataset is high-quality, sufficiently large, covers all the cases, and is a good representation of reality. This allows the machine learning algorithm to more efficiently model the underlying characteristics of the data, find accurate patterns, generalize them better, and hence give more accurate outputs. Here, data augmentation is very useful especially when we need to oversample minority classes in order to make sure they have a significant presence in the data and hence are better learnt by the model in training [4].

GAN's are an emerging technique in supervised and unsupervised learning to create new synthetic data by training on real values. It can be used to augment numeric data, create new pictures of people who do not exist and also to style one image on to the other.

GANs are characterized by two multi-layer machine learning models competing against each other. One is called the Generator and the other is called the Discriminator. The main job of the generator is to create forgeries of actual data; let us say fake images when fed with actual images; while, the main job of the Discriminator is to tell the real data and synthetic data (from the Generator) apart. The loss functions of both the models are defined as how well one model outperformed the other i.e., how well was the Generator able to fool the Discriminator into accepting its synthetic data as real and how well was the Discriminator able to tell the real and fake data apart. The two models improve in competition with each other; the Generator by creating better forgeries of actual data so that the discriminator cannot tell them apart, and the Discriminator by getting better at distinguishing between actual and synthetic data in order to maximize the Generator loss so that the Generator produces better copies. Eventually, we decide a sweet spot as to how much loss is acceptable for both models [2].

II. Related Work

GANs have been used previously to oversample minority classes. For example, in paper [8] the authors use a GAN to synthetically oversample a class in a numerical dataset with the purpose to train a classifier on it. The classifier showed better results in terms on accuracy and precision when trained on the synthetically balanced data set than when trained on the imbalanced dataset. The GAN also showed better accuracy and precision but worse recall when compared to ADASYN and SMOTE, two other widely used oversampling techniques.

The paper by Karras et al. [6] devises a strategy for GANS to generate synthetic images. The key idea in the paper is to model both the generator and discriminator progressively, first starting off from low resolution and adding further layers to cater for fine details as training progresses. This method helps in speeding up the training and allows generation of extremely high-quality images. The paper also uses techniques such as equalized learning rate and pixelwise feature vector normalization in generator to discourage unhealthy competition between the generator and discriminator. Lastly, the authors use multi-scale statistical similarity to assess the GAN results. The result is good quality fake images produced in high resolution by training on the CELEB dataset.

III. Dataset

Although there are several datasets available consisting of images categorized into different human emotions but the only dataset publicly available is the **FER2013** dataset. The dataset consists of 35,887 images having dimensions 48 x 48. They are gray-scaled and categorized into 7 different facial expressions namely Angry, Fear, Disgust, Happy, Neutral, Sad and Surprised. Some examples from the dataset are displayed in Fig.1. However, the division of data into categories is quite imbalanced as shown in Fig. 2, with the 'disgusted' class having the lowest number of images (547) and the 'happy' class having the highest number of images (8989).

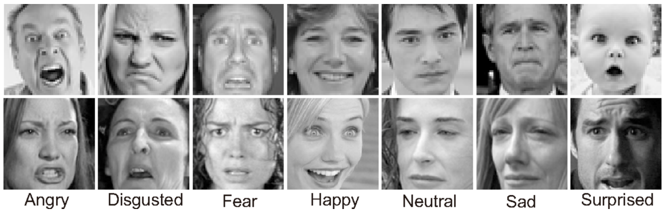

Fig. 1. Examples from FER2013 dataset

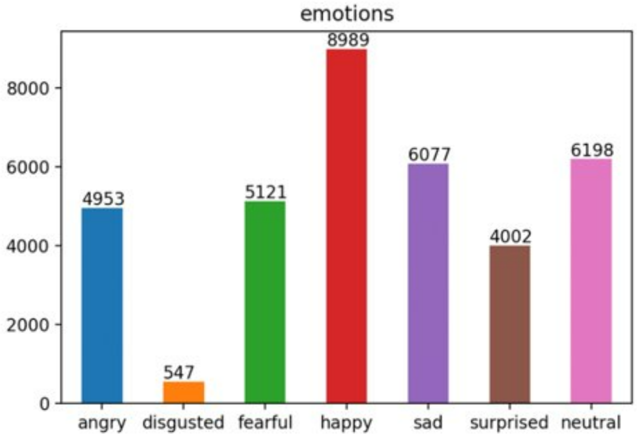

Fig. 2. Number of images in each category in the FER2013 dataset

IV. Classification Model and GAN Architecture

A. GAN Architecture

A DCGAN was used to generate synthetic images using real images from the FER dataset. The Generator consists of two dense layers with 64 neurons each, followed by a sequence of five generator blocks. Each generator block upsamples the input (except the last block in the sequence), applies a 2d convolution, followed by 2D batch normalization and then finally applying the tanh activation function as shown in the architecture in 3. The Generator then returns **3x48x48** synthetic images. Kernel of size 6 is used with 1 stride. The loss of the Generator is defined by the mean of correctly identified fake pictures by the Discriminator. The Discriminator of the GAN consists of a sequential block with 4 discriminator blocks, each applying a @D convolution to the input, followed by a 2D Batch Normalization, and finally applying the tanh activation function as shown in 4. The last block of the sequence only applies the 2D convolution. The loss of the discriminator is defined as the mean of wrongly identified real/ fake pictures × the gradient norm.

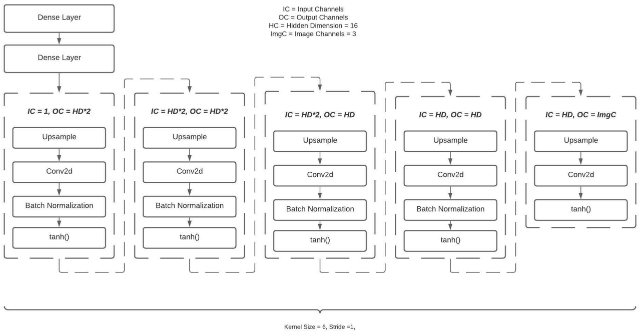

Fig. 3. DCGAN Generator Proposed Architecture

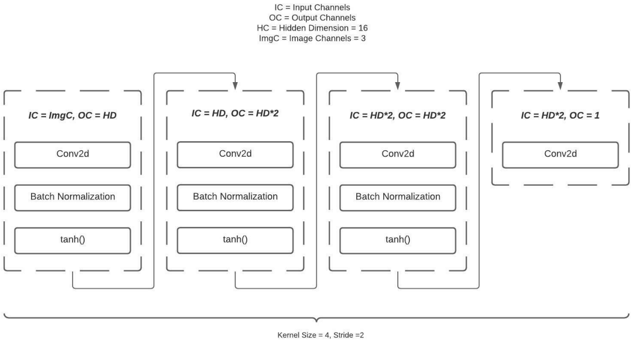

Fig. 4. DCGAN Discriminator Proposed Architecture

B. CNN Model

Convolutional Networks have been widely used for the purpose of image classification especially in applications that lie in the domain of Computer Vision. VGGNet and ALexNet are one of the most commonly used CNN architectures that are used for the purpose of image classification. However for classification of images from the FER2013 dataset there have been various CNN models designed, such as the FER-Net [3] and the model presented in [1]. Similarly, in order to verify the efficacy of the images generated using our GAN architecture, we designed a CNN model to classify images from the FER2013 dataset before and after image augmentation. Our CNN model's architecture is shown in Fig.5. The model consists of several convolutional layers with variable kernel sizes and number of filters, transforming the input image at each layer. Max Pooling Layer determines the maximum value of the window that is applied to the input feature map. The Dropout Layer is added to prevent over-fitting since it randomly assigns zero to a specified ratio of neurons. The two dimensional matrix is then Flattened into a vector which is fed into a dense layer. The ReLU activation function used in the initial layers and the softmax function is used in the output layer comprising of 7 nodes.

$$\mathrm{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

The optimisation algorithm used to update weights across the network is Adam. The loss function used taken into consideration the multi-class classification, was categorical cross-entropy loss.

$$\mathrm{Loss} = -\sum_{i=1}^{\substack{\text{output}\\ \text{size}}} y_i \cdot \log \hat{y}_i$$

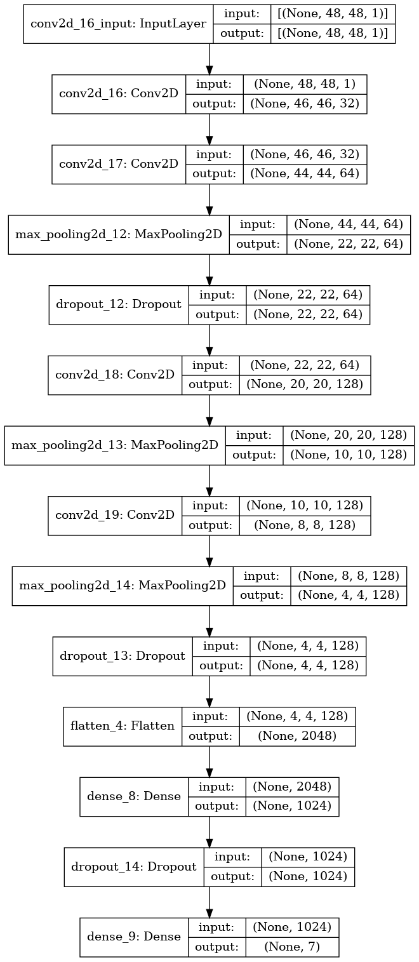
conv2d_16_input: InputLayer | input: [(None, 48, 48, 1)] | output: [(None, 48, 48, 1)]
conv2d_16: Conv2D | input: (None, 48, 48, 1) | output: (None, 46, 46, 32)
conv2d_17: Conv2D | input: (None, 46, 46, 32) | output: (None, 44, 44, 64)
max_pooling2d_12: MaxPooling2D | input: (None, 44, 44, 64) | output: (None, 22, 22, 64)
dropout_12: Dropout | input: (None, 22, 22, 64) | output: (None, 22, 22, 64)
conv2d_18: Conv2D | input: (None, 22, 22, 64) | output: (None, 20, 20, 128)
max_pooling2d_13: MaxPooling2D | input: (None, 20, 20, 128) | output: (None, 10, 10, 128)
conv2d_19: Conv2D | input: (None, 10, 10, 128) | output: (None, 8, 8, 128)
max_pooling2d_14: MaxPooling2D | input: (None, 8, 8, 128) | output: (None, 4, 4, 128)
dropout_13: Dropout | input: (None, 4, 4, 128) | output: (None, 4, 4, 128)
flatten_4: Flatten | input: (None, 4, 4, 128) | output: (None, 2048)
dense_8: Dense | input: (None, 2048) | output: (None, 1024)
dropout_14: Dropout | input: (None, 1024) | output: (None, 1024)
dense_9: Dense | input: (None, 1024) | output: (None, 7)

Fig. 5. Classification Model

V. Experiments

A. Results from Classification Model

1) Initial Results: In order evaluate the fake images, a threshold accuracy is found by training the CNN model over the original dataset. The images were loaded by the ImageDataGenerator API provided by Keras. The API allowed us to randomly split the dataset into test and validation sets. The API also provided image augmentations in the form of horizontal flips, zoom ranges and shear ranges. The threshold accuracy after training the dataset was 64%, which was similar to the accuracy in models [1] and [3]. The model was trained on 70 epochs. The accuracy and loss after every epoch can be seen in Fig.6 and Fig.7. The confusion matrix showed that 'Disgust' (as expected due to its lowest total number of images) and 'Fear' classes had significantly lower accuracy as compared to other classes, as shown in 8.

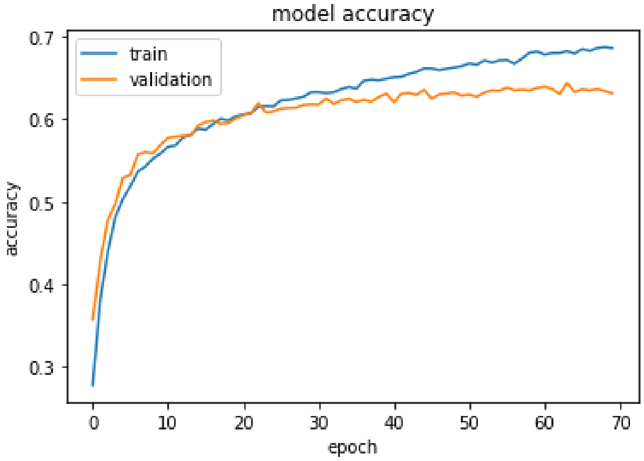

Fig. 6. Accuracy vs. Epochs: Before Image Augmentation using GAN

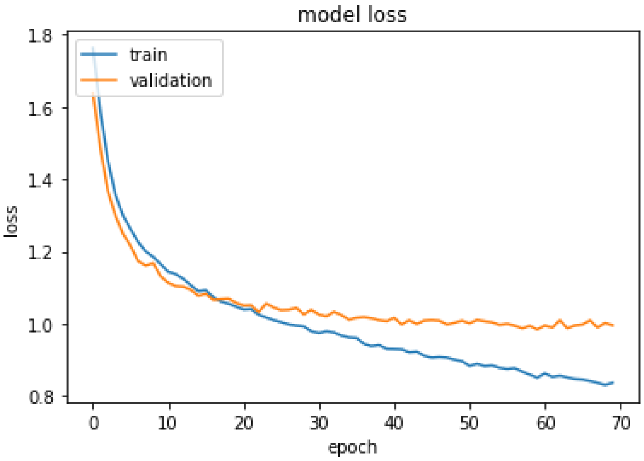

Fig. 7. Loss vs. Epoch: Before Image Augmentation using GAN

2) Final Results: The results shown by our confusion matrix were consistent with the research in [9] such that 'Disgust' and 'Fear' were responsible for a lower overall accuracy. Hence, after 1000 images were generated for each category, the model was re-trained on the same number of epochs to evaluate the new changes. The new accuracy and loss after every epoch can be seen in Fig.9 and Fig.10. The new confusion matrix can be seen in Fig.11.

B. Fake Images Generated using DCGAN

As shown in Fig.2, the number of images in the 'disgust' class is significantly lower than the other classes. This may

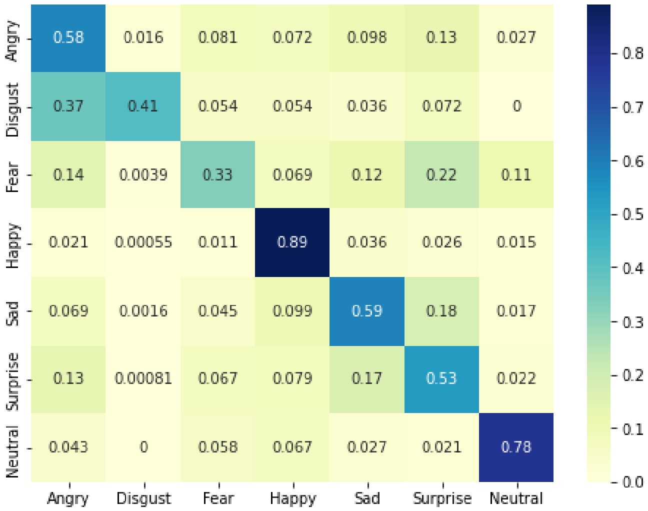

Fig. 8. Confusion Matrix: Before Image Augmentation using GAN

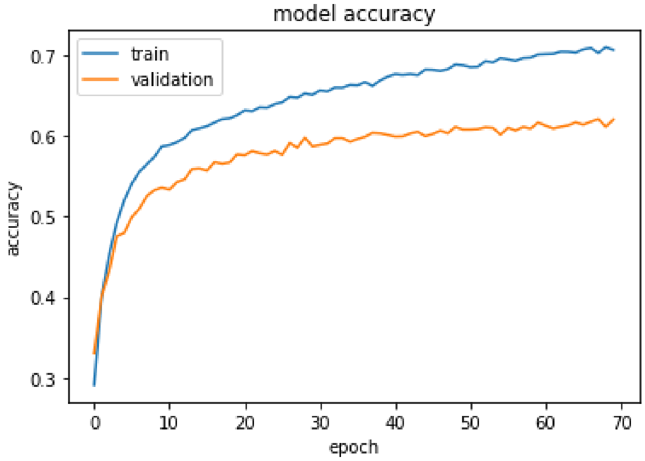

Fig. 9. Accuracy vs. Epoch: After Image Augmentation using GAN

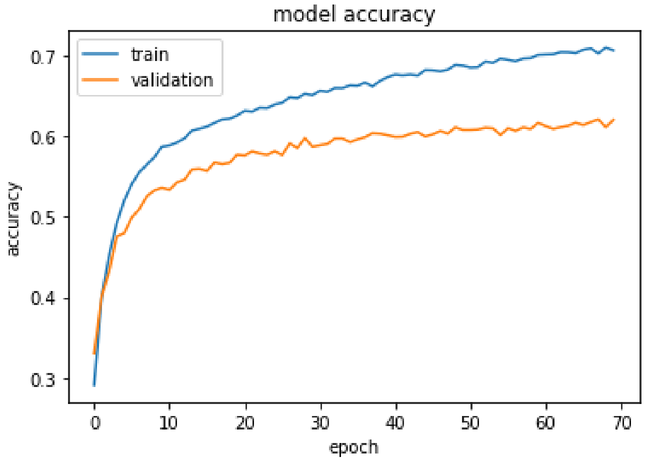

Fig. 10. Loss vs. Epoch: After Image Augmentation using GAN

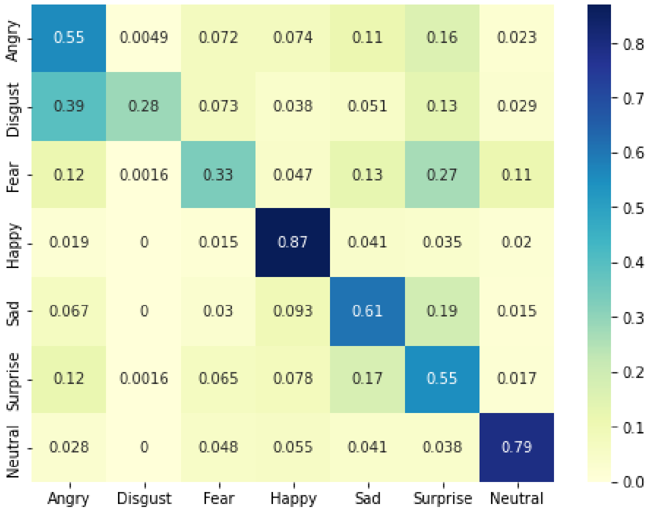

Fig. 11. Confusion Matrix: After Image Augmentation using GAN

result in poor classification of the class hence in order to counter that problem, 1000 images for the ’disgust’ class are generated using our DCGAN model. Some of the images are shown in Fig.12. The GAN was executed for 500 epochs due to lower number of overall images of the class in the dataset. The discriminator loss and the generator loss was monitored for every epoch.

Additionally, as per the results of the initial classification model in Fig.8 and research [9], the ’fear’ class also had a significantly lower accuracy, hence it was deemed appropriate to create 1000 images for the ’fear’ class as well. Some of the images are shown in Fig.13. The GAN was executed for 100 epochs. The discriminator loss and the generator loss was monitored for every epoch.

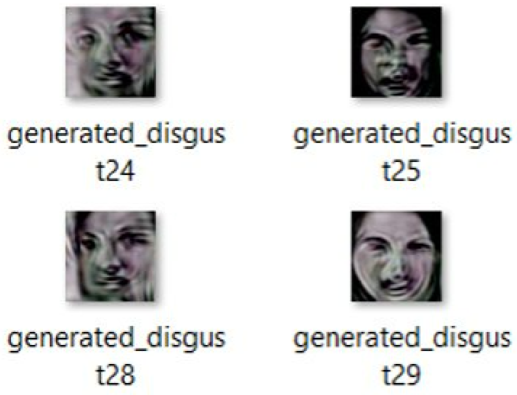

Fig. 12. Sample of images generated for the ’disgust’ class after 500 epochs

VI. Conclusion & Future Work

The overall accuracy of the classification model decreased from 64% to 61% after increasing the dataset by 2000 images. The accuracy of ’fear’ class remained the same while the

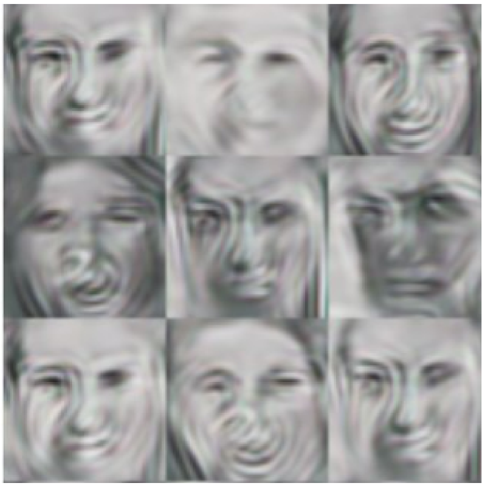

Fig. 13. Sample of images generated for the 'fear' class after 100 epochs

accuracy of 'disgust' class decreased from 41% to 28%. Even though the classification results were not as per our expectations, we believe there may be several reasons behind it. As stated in [9], the FER2013 dataset contains many wrongly classified images, such that in the initial classification of 'disgust' images, the model predicted 37% of 'disgust' images as 'angry', and this value increased to 39% after the introduction of 1000 more disgust images. Another reason for the miss-classification of images might be due to the quality of images generated by the DCGAN model.

DCGAN is one of the many architectures of Generative Adversarial Networks [5]. It is also one of the earliest architectures of GANs. In order to ensure better results/images, Conditional GANS [7] might be a better approach which makes use of class labels in order to get better results from a GAN. Other than using a different iteration of GAN to generate better images for classification, we can redesign our existing classification models to ensure a better classification on the existing datasets. One example may involve using spatial transformer networks that learns different key areas of the images. These transformers can be embedded with expert knowledge in order to facilitate classification decisions, leading to better classification.

Acknowledgment

We would like to express our gratitude to the Computer Science Program Director at Habib University, Dr. Abdul Samad, for sharing his knowledge and wisdom with us during the course CS 316 - Introduction to Deep Learning which enabled this research.

References

[1] Agrawal, Abhinav & Mittal, Namita. (2020). Using CNN for facial expression recognition: a study of the effects of kernel size and number of filters on accuracy. The Visual Computer. 36.

[2] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative Adversarial Networks: An overview," IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 53–65, 2018.

[3] Celik, Turgay & Ma, Hui. (2019). FER-Net: Facial Expression Recognition using Densely Connected Convolutional Network. Electronics Letters. 55.

[4] G. Goel, L. Maguire, Y. Li, and S. McLoone, "Evaluation of sampling methods for learning from imbalanced data," Intelligent Computing Theories, pp. 392–401, 2013.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14). MIT Press, Cambridge, MA, USA, 2672–2680.

[6] Karras, Tero et al. "Progressive Growing of GANs for Improved Quality, Stability, and Variation." ArXiv abs/1710.10196 (2018): n. pag.

[7] Mirza, Mehdi & Osindero, Simon. (2014). Conditional Generative Adversarial Nets.

[8] Tanaka, Fabio Aranha, Claus. (2019). Data Augmentation Using GANs.

[9] T. U. Ahmed, S. Hossain, M. S. Hossain, R. ul Islam and K. Andersson, "Facial Expression Recognition using Convolutional Neural Network with Data Augmentation," 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), 2019, pp. 336-341,

[10] Z. Cai, Q. Fan, R. S. Feris, en N. Vasconcelos, "A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection", in Computer Vision – ECCV 2016, 2016, bll 354–370.