Greek paGAN – Dealing with Synthetic Data Generation with GANs

Carolina Macedo, Fengyi Chen, Emilie Guido, Nana Maglakelidze, and Gaëtan Drouet

École nationale des chartes, Université PSL

June 2024

Κρείττον οψιμαθή είναι ή αμαθή.

— Σωκράτης

1 Introduction

During the 2022-23 academic year, the Encyclop team[1] from our university participated in the ICDAR competition[2] on Detection and Recognition of Greek Letters on Papyri, securing a prestigious 3rd place. Our observations from this competition indicated that increasing the volume of data improves results. Therefore, we aim to explore whether generating a synthetic complementary dataset can further enhance our results in the absence of additional real datasets.

All implementation details, including data and notebooks, are available in our GitHub repository. For comprehensive information on each step of the process, please visit our GitHub repository.

2 Background

The development and refinement of image classification techniques alongside Generative Adversarial Networks (GANs) have significantly advanced artificial intelligence's capabilities in processing and generating visual content. In a GAN architecture, a generator network produces images intended to look genuine, while a discriminator network assesses their authenticity. This competitive relationship drives the improvement of both networks.

Deep learning models have revolutionized image classification, enabling complex tasks such as scene understanding and object interactions, which are crucial in fields like autonomous driving [7] and medical diagnostics. These advancements are reflected in the discussion by Fogel et al.[1], who emphasize the significance of deep learning in improving optical character recognition (OCR) systems, particularly for handwritten text recognition (HTR), which faces unique challenges compared to printed text due to style variability.

GANs employ a generative technique alongside a supervised training approach, as explained by Mahmood Mohammadi [3]. This contrasts with the unsupervised training methods typically

[1]Chahan Vidal-Gorène, Malamatenia Vlachou Efstathiou, Violette Saıag, Noé Leroy and Carolina Macedo.

[2]See [6] for more details on the teams' approaches.

associated with generative models. GANs are pivotal in generating synthetic data, especially useful in domains where data privacy is a concern[5] or real data is insufficient[8]. They enhance the volume and diversity of training datasets, which is critical for improving the robustness of classifiers.

According to Vidal-Gorène et al. (2023), GANs achieve "very convincing results in different scenarii". The authors highlight that the latest advancements utilize a style-transfer approach, adapting generated data to targeted datasets, thereby enhancing the realism and utility of synthetic data. By generating synthetic images to augment training datasets, GANs provide a robust solution for enhancing model performance.

3 Image Classification Using YOLO

3.1 Task

At this stage, our task is to classify images of Greek characters using a specialized letter classifier based on the YOLO (You Only Look Once) framework. The goal is to accurately categorize images into designated classes representing different Greek letters. Specifically, we focuses on identifying Greek letters, demonstrating the classifier's effectiveness in recognizing and interpreting complex character sets.

3.2 Dataset and Methodologies

We used the character crops that we had produced in our previous work[3] as our initial dataset. The first step involved performing classification tasks to organize all the images in a directory, categorized by their class and the name of the letter.

Our dataset was methodically segmented into three subsets—training, validation, and testing—utilizing the `train_test_split` function to ensure a balanced distribution of data across different stages of model evaluation.

3.3 Model

We employed the YOLO Ultralytics framework to develop a specialized letter classifier. This advanced model leverages the principles of Convolutional Neural Networks (CNNs)[4] to accurately categorize images into designated classes.

3.4 Training

The classifier underwent rigorous training with the training dataset, which not only helped in assigning precise labels to each image but also calculated a confidence score reflecting the prediction's reliability. The model was trained using different parameters. Let's look at an example:

- **Model**: YOLOv8s-cls
- **Image Size**: 64x64
- **Epochs**: 20
- **Dropout**: 0.3
- **Top-1 Accuracy**: 0.773
- **Top-5 Accuracy**: 0.939

[3]See the GitHub repository for more detailed information.
[4]More detailed information can be found here [4]

3.5 Evaluation

To evaluate the classifier's performance on the test data, we employed an occlusion technique commonly used in computer vision. This method involves blocking portions of an image during training to challenge the network. By doing so, the network is forced to learn from non-canonical features, which helps prevent overfitting and enhances its ability to generalize from the training data to unseen images.

Initially, this approach involves measuring the predictive accuracy of the YOLO model on a test image. As the process unfolds, a mask systematically moves across the image, and the impact of each movement on the prediction is assessed. A heatmap is then generated to visually represent how these mask movements affect the predictions. For practical execution, all test images were resized to a uniform dimension of 64x64 pixels. Considering that the original sizes of these images range from 20x20 to 150x150 pixels, we chose a median size. Additionally, we employed an 8x8 pixel occlusion that shifts by 8 pixels after each movement, ensuring optimal precision while preventing excessive computational load on our systems.

After forming grayscale `numpy` arrays of heatmaps for all the test images, these heatmaps are stored in lists according to their class number, each class representing a letter. Then, they are overlaid on top of each other through the calculation of an unweighted average. For example, to identify the discriminative areas for the letter 'alpha', we overlaid all the heatmaps from the different images of 'alpha'[A]. The goal is to obtain a 'general' heatmap for each letter, even though the information obtained might be biased.

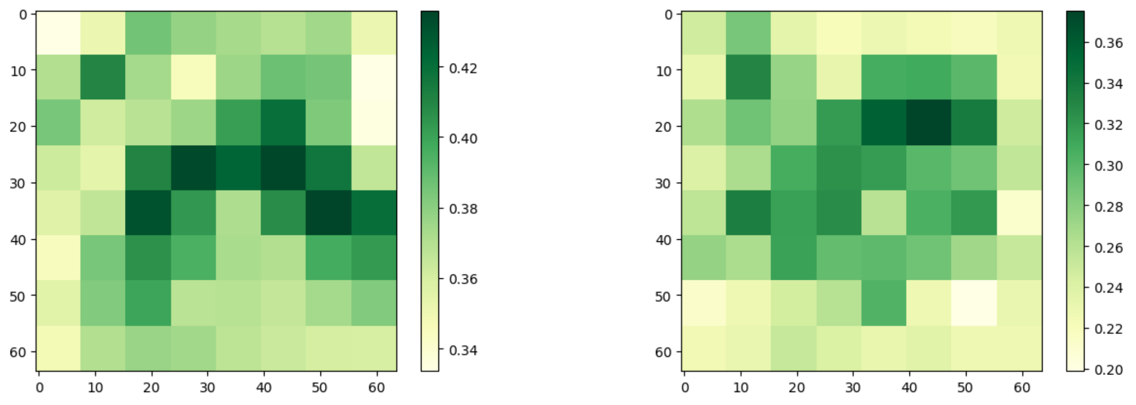

Figure 1: Overlaid heatmaps for characters C and X

4 Data augmentation with synthetic data

4.1 Task

The primary task of this project is to generate and classify Greek characters using a Conditional Generative Adversarial Network (cGAN) (see [2]). The goal is to produce realistic images of specific characters grouped into predefined sets, and subsequently classify these characters accurately.

4.2 Dataset

The dataset consists of images of Greek characters, organized into subfolders by character. Each subfolder contains multiple images of a specific character. The dataset is divided into five groups based on the character sets, with the number of images per group as follows:

- Group 1: A, H, R, Z (6887 images)
- Group 2: E, Θ, U, Ψ (6068 images)
- Group 3: I, K, L, Φ, X (6324 images)
- Group 4: N, O, T (8372 images)
- Group 5: Γ, Δ, M, Π, Σ, Ω (7620 images)

The images are loaded using TensorFlow's `image_dataset_from_directory` method, which creates a dataset where each item is a tuple consisting of the image and its corresponding label.

4.3 Model

The model used in this project is a Conditional Generative Adversarial Network (cGAN), which includes a generator and a discriminator. The generator aims to produce realistic images of Greek characters based on input noise and character labels, while the discriminator evaluates the authenticity of the generated images and distinguishes them from real images.

4.3.1 Generator

The generator is designed to take random noise and a character label as input, and output an image of the specified character. The architecture includes:

- An embedding layer for the character labels.
- Dense and reshaping layers to process the noise and label embeddings.
- Transposed convolutional layers to generate the final image.

4.3.2 Discriminator

The discriminator is tasked with distinguishing between real and generated images. It takes an image and a character label as input and outputs a probability score indicating the likelihood of the image being real. The architecture includes:

- An embedding layer for the character labels.
- Convolutional layers to process the image and label embeddings.
- Dense layers to produce the final probability score.

4.4 Training

The training process involves alternating between updating the generator and the discriminator. For each epoch, the following steps are performed:

1. Real images and their labels are fed into the discriminator to compute the real loss.
2. Fake images are generated using the generator and fed into the discriminator to compute the fake loss.
3. The generator is updated based on the feedback from the discriminator to reduce the fake loss.

The model was trained for 30 epochs with a batch size of 128. Despite experimenting with various parameters, the generated images were not discernible.

4.5 Evaluation

The evaluation of the model's performance involved assessing the quality of the generated images and the discriminator's accuracy. Unfortunately, the generated images lacked clarity and were not easily recognizable as the intended characters.

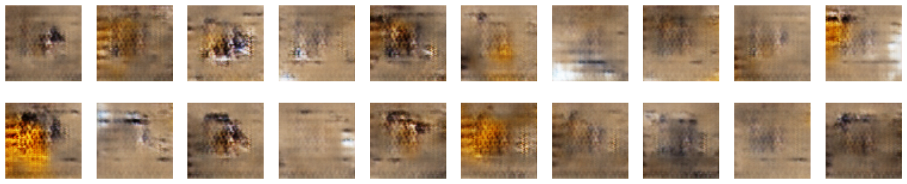

Figure 2: Generated images for Group 3 using cGAN.

5 Conclusion

This project utilized Conditional Generative Adversarial Networks (cGANs) to generate and classify Greek characters. Despite training the model with different parameters, the generated images lacked clarity. Using the YOLO framework, we achieved reasonable classification accuracy but faced challenges due to the complex nature of ancient script characters. Future work should focus on refining GAN architectures and exploring larger datasets to improve image quality and classification performance. This study demonstrates the challenges and opportunities in using GANs for generating synthetic data in specialized applications.

References

[1] Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, and Roee Litman. Scrabblegan: Semi-supervised varying length handwritten text generation. `https://arxiv.org/abs/2003.10557`, 2020.

[2] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv*, November 2014.

[3] Mahmood Mohammadi. Synthetic data generation using generative adversarial networks (gans) part 2. `https://medium.com/data-science-at-microsoft/synthetic-data-generation-using-generative-adversarial-networks-gans-part-2-9a078741d3ce`, 2023.

[4] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv*, December 2015.

[5] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, 2018. Publisher Copyright: © 2018 VLDB Endowment 21508097/18/4.; 44th International Conference on Very Large Data Bases, VLDB 2018 ; Conference date: 27-08-2018 Through 31-08-2018.

[6] Mathias Seuret, Isabelle Marthot-Santaniello, Stephen A. White, Olga Serbaeva Saraogi, Selaudin Agolli, Guillaume Carrière, Dalia Rodriguez-Salas, and Vincent Christlein. Icdar 2023 competition on detection and recognition of greek letters on papyri. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 14188 LNCS, pages 498–507. Springer, 2023.

[7] Zhihang Song, Zimin He, Xingyu Li, Qiming Ma, Ruibo Ming, Zhiqi Mao, Huaxin Pei, Li-hui Peng, Jianming Hu, Danya Yao, and Yi Zhang. Synthetic data for training models in autonomous driving. *Journal of Autonomous Systems*, 5(3):200–220, 2024.

[8] Yisheng Yue, Palaiahnakote Shivakumara, Yirui Wu, Liping Zhu, Tong Lu, and Umapada Pal. An automatic system for generating artificial fake character images. In Ioannis Kompatsiaris, Benoit Huet, Vasileios Mezaris, Cathal Gurrin, Wen-Huang Cheng, and Stefanos Vrochidis, editors, *MultiMedia Modeling*, pages 291–301, Cham, 2019. Springer International Publishing.