Enhancing the Process of Image Classification using GAN Augmentations and Metaheuristic Optimization Methods

Lenka Kališková, Peter Butka and František Babič

Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovakia
lenka.kaliskova@tuke.sk, peter.butka@tuke.sk, frantisek.babic@tuke.sk

Abstract: *The classification of image-based data can be enhanced and optimized in various ways. The performance of convolutional neural networks (CNNs) is heavily influenced by both the quality of input data, especially their augmentations, and the choice of model network configurations and hyperparameters. This study investigates the combined impact of Generative Adversarial Network (GAN)-based data augmentation and hyperparameter optimization using evolutionary and metaheuristic algorithms. We explored several meta-heuristic optimization algorithms to identify best setup for hyperparameters and networks configurations to achieve a high classification performance, but reduce the time for training. Performance was evaluated using metrics such as accuracy, precision, recall, F1 score and optimization time (training time reduced in comparison to grid search). For testing, we used three image-based data sets from the astronomical domain. Results demonstrate that combining GAN-based augmentation with optimization of hyperparameter and network configuration allows us to provide highly precise classification results with the significant reduction of training time in comparison to grid search. These findings provide strong evidence for the complementary value of advanced augmentation and optimization techniques in automated deep learning training pipelines.*

Keywords: image classification; data augmentation; generative adversarial networks; optimization; metaheuristic algorithms; convolutional neural networks; deep learning

1 Introduction

The usage of deep learning techniques has increased in recent years, especially for processing of image-based astronomical data available in different sky surveys. Due to increasing number of images expected in next years from new instruments, critical aspect of successful training of new models is the availability of a sufficiently representative dataset and optimal strategies for training. Both are

especially important in image-based astronomical datasets. One of the challenges is limited variability in training data, which leads to poor model generalization. A common solution is data augmentation. Geometric transformations, like flipping or rotating, are categorized as classical data augmentation methods. Beyond traditional methods, the use of generative models, specifically Generative Adversarial Networks (GANs) [1], has emerged as an effective alternative for enhancement of variability in input data and improvement of classification results.

In addition to effective data augmentation, another major challenge in training deep learning models is hyperparameter and network configuration optimization, which is a time-consuming task, especially for large astronomical surveys. Therefore, this study also explores the use of advanced optimization algorithms, specifically evolutionary and metaheuristic algorithms, as a more efficient alternative to traditional Grid Search. These methods can often reduce optimization and provide a practical solution for real-world applications.

Our aim is to study the combined impact of these main aspects in parallel, with an emphasis on astronomical image data, to improve the processing of datasets in this domain, which are now streaming from new instruments in large amounts and should be processed automatically and as effectively as possible. The purpose of this study is to investigate the impact of various GAN-based augmentation methods in conjunction with different optimization algorithms to achieve as good as possible classification performance in reduced amount of training time. Experiments are conducted on three astronomical datasets: Galaxy Mergers, FIRST Radio Survey, and Galaxy ZOO – Rings. Classification effectiveness is evaluated using metrics such as accuracy, precision, recall, and F1 score, with optimization runtime also being a key factor considered for their practical applicability for such tasks in future. The combination of these approaches in the context of astronomical image classification has not yet been extensively studied.

2 Related Work

Our analysis of the related work follows two primary aspects: advanced image augmentations using GAN models and advanced optimization techniques for architecture configurations and hyperparameters, primarily employing evolutionary and metaheuristic principles.

Regarding the GAN-based image augmentation, in [2] authors introduced Data Augmentation Generative Adversarial Network (DAGAN), combining UNet and ResNet for the generator and DenseNet for both discriminator and classifier. Applied to Omniglot, EMNIST, and VGG-Face datasets, their approach improved classification accuracy, with Omniglot seeing the highest gain (+13%). The EMNIST dataset saw an improvement of 2.1%, while the VGG-Face dataset

improved by 7.5%. Another study [3] also explored the use of GANs for data augmentation, specifically employing the Balancing Generative Adversarial Network (BAGAN). The authors experimented with four well-known datasets: MNIST, CIFAR-10, Flowers, and GTSRB. Comparative results showed that BAGAN was the most suitable approach for augmenting unbalanced datasets while consistently generating high-quality images, outperforming traditional GANs and ACGANs in their tests. In [4] authors focused on AutoML-based augmentation methods. These approaches consistently outperformed classical techniques across CIFAR-10, CIFAR-100, ImageNet, and MS COCO datasets, achieving gains of up to +3.6% accuracy on ImageNet and +2.2% mean average precision on detection tasks.

In [5] authors proposed a stacked augmentation method combining Style Transfer Data Augmentation (STDA) and DAGAN, optimized using the Pufferfish Optimization Algorithm (POA). Applied to 10 biomedical datasets, the method achieved classification accuracy between 91.83% and 99.11%, surpassing conventional augmentation techniques. Overall, these studies confirm that GAN-based and AutoML-guided augmentation methods effectively improve performance across diverse datasets, particularly in scenarios involving class imbalance or limited training data. In [6], we also analyzed various GAN-based augmentations, with datasets from medicine and astro/geo domain. Both individual and combinations of GAN-augmentations were studied and showed promising results in improvements, if compared to standard augmentation techniques, as was confirmed by other studies on benchmark datasets

The second part of the related work focuses on the selection of optimal settings and configurations of the selected models. Within the framework of optimization, it is possible to use Grid Search, or Random Search (RS), or try evolutionary principles or metaheuristic methods for faster search for settings and configurations of models. Each of the individual works mentioned in this section approaches the given problem innovatively, with the aim of shortening the necessary optimization time, and at the same time finding close to optimal combinations of hyperparameters than grid-based searches, and therefore reduce time for learning optimal models. In the work [7], Population-Based Training (PBT) was used for hyperparameter optimization. The study applied the algorithm to three domains of models: Deep Reinforcement Learning, Machine Translation, and GANs. For Deep Reinforcement Learning, the authors examined three domains, using a population size of 10 to 80 individuals while optimizing hyperparameters such as step length and loss functions. The PBT approach stabilized training across all three domains, enabling the discovery of optimal hyperparameters in a relatively short time and achieving better performance than previous methods. Similarly, in the domain of Machine Translation and GAN models, this method improved performance on both validation and test sets. Although this study did not focus on optimizing CNNs, it effectively demonstrated the strength of the method.

In [8] authors focused on optimizing the hyperparameters of neural networks using the Particle Swarm Optimization (PSO) algorithm. The authors aimed to optimize the number of hidden layers and the number of neurons per layer. The results showed that PSO was able to discover optimal hyperparameters 77% to 85% faster than Grid Search while also achieving high classification accuracy. The work [9] proposed a variant of PSO called Linearly Decreasing Weight PSO (LDWPSO) for optimizing CNN architectures on MNIST and CIFAR-10. For population of 10 individuals, they optimized a wide range of parameters and their model achieved improvement in speed for finding of high-performance models, with superior final accuracy (98.95%) compared to manually configured models.

In [10] authors optimized CNNs using the Variable Length Genetic Algorithm (VLGA), which employs a variable number of individuals rather than a fixed population size. The authors optimized parameters such as the number of feature maps in each convolutional layer, kernel size, activation function, sampling layers, total number of layers, etc. VLGA reached 88.92% accuracy on CIFAR-10 in 25 hours using 20 individuals over five generations, outperforming traditional algorithms, which were not able to achieve more than 80.75% in 30 hours.

In general, evolutionary or metaheuristic approaches showed strong confidence in providing configurations for high-performance models in shorter time to traditional grid or random search methods. Addition of GAN augmentations and combination with optimization techniques might provide better models earlier and help in future automatic processing of astronomical datasets.

3 Methods

3.1 Convolutional Neural Networks (CNNs)

CNNs [11] represent one of the most widely used neural network architectures, particularly for image recognition and classification tasks. A typical CNN, which we used in our experiments, consists of three main layer types: convolutional, pooling, and fully connected layers. The convolutional layer performs the core operation of convolution, where a filter (kernel) slides across the input image, applying element-wise multiplication to extract local features and generate a feature map. The pooling layer focuses on down sampling, which involves reducing the dimensionality of the activation map. This leads to a decrease in the number of parameters and reduction in computational complexity. We can have max or average pooling, if in each region max or average value is selected. The fully connected layer performs the classification based on the features extracted by preceding layers, typically followed by Softmax activation function for target classes.

3.2 Generative Adversarial Networks

GAN provides an approach to data augmentation by synthesizing new images from existing ones, rather than solely modifying pre-existing data. GAN was first introduced in [1] and is a deep learning-based generative model that consists of two primary components: the generator (G) and the discriminator (D). The generator G aims to capture the underlying data distribution and generate realistic samples, while the discriminator D is tasked to distinguish between real samples from the training set and synthetic samples produced by G. The training process is structured as a min-max game, where generative model G seeks to maximize the probability of discriminative model D misclassifying generated samples, while D continuously improves its ability to differentiate between real and synthetic data. This adversarial interaction drives both networks toward improved performance. The overall architecture of a GAN is shown in Figure 1.

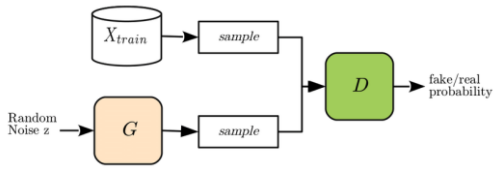

Figure 1
General architecture of GAN

Several GAN variants have been developed to enhance training stability and performance. For instance, Deep Convolutional Generative Adversarial Network (DCGAN) introduces convolutional layers and batch normalization for more stable learning [12]. DAGAN, designed specifically for data augmentation, generates class-consistent images by encoding inputs into a latent space and decoding them back with controlled variability [2]. Wasserstein Generative Adversarial Network (WGAN) replaces the discriminator with a critic and uses Wasserstein distance as the loss function, leading to more stable gradients and higher-quality outputs [13]. BAGAN combines GANs with autoencoders to address class imbalance by initializing the generator with representations learned from all classes. For detailed architectural diagrams and extended discussion of these models, the reader might find more information in our earlier publication [6], which focused solely on GAN-based augmentations.

3.3 Optimization Algorithms

The goal of optimization is to find a combination of hyperparameters that ensures the best performance of a given model. To achieve this, it is necessary to define the search space, which can be described as a multi-dimensional space where each dimension represents a hyperparameter or configuration setting, and its range

defines the possible values that the hyperparameter can assume. Hyperparameters can be determined not only using optimization algorithms but also manually. In manual optimization, known as hand tuning, hyperparameter combinations are manually created and subsequently tested. However, this approach is not optimal, as the increasing number of hyperparameters make it practically impossible to manually create and evaluate all possible combinations.

Among the fundamental hyperparameters that can be optimized in model development (and are considered as part of configuration) are parameters like number of epochs (number of times the training dataset is processed), batch size (represents the subset of training data used in a single iteration), activation function (type of function which is used in neurons), dropout rate (regulates deactivating of neurons to prevent overfitting), number of neurons in individual layers, optimization function used for model parameters, etc.

Hyperparameter optimization plays a crucial role in enhancing model accuracy and generalization. The definition of an appropriate search space and the utilization of advanced optimization techniques can significantly improve model efficiency while reducing computational costs. The fundamental algorithm used for hyperparameter optimization is Grid Search, which constructs and evaluates a model for every possible combination of hyperparameters, ultimately selecting the configuration that ensures the best model performance. However, when applied to many configurations, it can be become inefficient and computationally demanding. The Random Search algorithm [14] is similar to Grid Search, but instead of evaluating all possible combinations of hyperparameters, it generates and evaluates random combinations, i.e., a predefined number of random combinations are selected for evaluation. As the combinations are random and do not cover the whole search space, it usually provides model which is worse than one from Grid Search. However, it often finds close-to-best model in a shorter time.

Of course, we can use more goal-oriented techniques, like evolutionary and metaheuristic approaches. Evolution Strategy (ES) [15] is an optimization method that belongs to evolutionary algorithms. These are population-based optimization techniques inspired by the principle of natural selection from evolutionary theory. Here, individuals are configurations, which are changing between generations using operations like mutation (changing of parameter) and crossover (swapping of parts of individuals between them), with the force of improvement based on selection of configurations with better accuracy for next generations. Evolutionary algorithms are already standard technique for search of better solutions.

PBT [7] is a relatively recent method, which re-use population-based principle and is designed for hyperparameter optimization, aimed at finding the optimal combination in less time compared to other methods, primarily by simultaneously training multiple models at the same time. The PBT method combines and modifies aspects of Random Search and manual optimization. Initially, several models are trained concurrently with random hyperparameters. These models form a group,

which enables information exchange among them. By leveraging information from the rest of the population, the hyperparameters of the models can be adjusted, focusing on those with better performance. Therefore, PBT is able to find optimal hyperparameter combinations in a shorter time due to focus on the modification of the most promising models.

PSO [16] is a population-based metaheuristic optimization algorithm, where number of particles are initially defined, with each particle representing a potential solution to the problem moving towards subspaces with higher value of goal function (accuracy in our use-case). The primary advantage of using PSO for optimization tasks lies in its ability to efficiently search large search spaces. However, it is important to note that PSO does not guarantee finding the global optimum, but the discovered solution is likely to be close to the global optimum.

Simulated Annealing (SA) is a metaheuristic optimization algorithm designed for efficiently exploring search spaces with many hyperparameters. It is inspired by the annealing process in metallurgy, where a material is gradually cooled to allow its particles to settle into more stable configurations. Similarly, SA probabilistically accepts both better and worse solutions during the search to avoid getting stuck in local optima. The process begins with a randomly selected set of configurations. In each iteration, a neighboring configuration is generated by slightly modifying the current one. If the new configuration performs better, it is accepted. If it performs worse, it may still be accepted based on a probability (modelled by temperature), encouraging early exploration before focusing most promising solutions later.

Parallel Tempering (PT) [17] is an optimization algorithm that can be considered a variation of Simulated Annealing. However, unlike SA, PT does not utilize the process of gradually cooling the temperature. Multiple copies of the system, referred to as replicas, are created, with each replica randomly initialized at different temperatures. The subsequent process of the algorithm follows the same principles as Simulated Annealing, with the key difference that, based on a specific protocol, two replicas of the system can be exchanged between different temperatures. This swapping process helps overcome situations where the search process becomes trapped in a local optimum.

4 Data Sources

4.1 Galaxy Mergers

The first dataset from the astronomical domain, referred to as Galaxy Mergers (available at https://github.com/SpaceML/merger_transfer_learning), comprises of 12001 galaxy images. The dataset is divided into two distinct classes based on the

interaction between galaxies. The Merger class, representing interacting galaxies, contains 4501 images, while the Non-Interacting class includes 7500 images. The dataset was derived from the Sloan Digital Sky Survey Data Release 7 (SDSS DR7). An example of the Galaxy Mergers dataset is shown in Figure 2.

Figure 2
Samples from the Galaxy Mergers: Merger class (left) and Non-Interacting class (right)

4.2 FIRST Data – Morphology of Radio Galaxies

The second dataset in the astronomical domain, referred to as FIRST Data, consists of 528 images of radio galaxies. These images are classified into four classes: Fanaroff-Riley Class I (FRI), Fanaroff-Riley Class II (FRII), Bent-Tailed (BENT), and Compact (COMPT). The FRI, FRII, and BENT classes represent extended radio galaxies, COMPT class corresponds to compact radio galaxies. The distribution of images across these classes is as follows: FRI contains 125 images, FRII includes 216 images, BENT consists of 104 images, and COMPT comprises 83 images. This dataset was derived from FIRST survey (available at https://sundog.stsci.edu/). Samples from FIRST dataset are presented in Figure 3.

Figure 3

Samples from the FIRST dataset: BENT, COMP, FRI, and FRII classes (left to right)

4.3 Galaxy ZOO – Rings

The third dataset, referred to as Demo Rings, is structured as a binary classification task (available at https://github.com/mwalmsley/galaxy-datasets, as a PyTorch dataset ("DemoRings")), is a small dataset designed for model evaluation and experimentation. It comprises a total of 1000 galaxy images, categorized into two classes based on the presence of ring structures. Specifically, the dataset includes

373 images of ring galaxies and 427 images of non-ring galaxies. An example of the Galaxy Zoo Rings Data is shown in Figure 4.

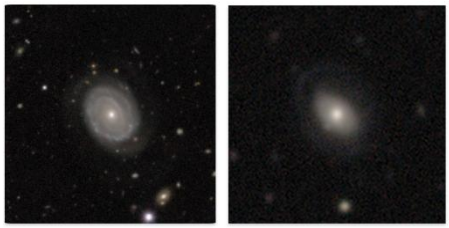

Figure 4
Samples from the Galaxy ZOO - Rings: Ring class (left) and No - ring (right)

5 Experiments

Our experiments focused on the impact of various GAN-based augmentations and optimization methods for reduction of training time while still achieving high-performance classification results for different datasets.

5.1 Setup of CNN Configuration Settings and Hyperparameters

As a preliminary step, the architecture of the baseline CNN was designed. The key components of the architecture include four convolutional layers and four pooling layers. Additionally, the network incorporates two fully connected layers, one of which serves as the output layer. The Softmax activation function is applied in the output layer to perform classification. To prevent overfitting, the dropout regularization technique was implemented. Before optimizing the CNN, it was necessary to define the hyperparameters to be fine-tuned. The possible values considered during the optimization process, along with the manually selected baseline values, are presented in Table 1. An important part of the process was the establishment of a baseline model, which serves as a reference point for comparing the results of various optimizations. The default (baseline) values of parameters were selected manually and consistently across all datasets.

5.2 Generative Adversarial Networks

The architectures of the GAN models used in this study mirror those presented in our previous work [6]. For the summary, the original GAN consists of a generator built with fully connected layers, up-sampling layers, batch normalization, and convolutional layers, using ReLU and Tanh activations. The discriminator uses

convolutional and dropout layers, with LeakyReLU and Sigmoid activations. The DCGAN replaces up-sampling with transposed convolution layers and uses LeakyReLU throughout the generator and discriminator. Its generator architecture follows a reshape–transpose–convolve pipeline for more stable training. For DAGAN, minimal modifications were made to the original GAN. Both the generator and discriminator consist of fully connected layers with LeakyReLU and Tanh/Sigmoid activations in their output layers. The WGAN replaces the discriminator with a critic and maintains a similar architecture to the original GAN, with the primary difference being the loss function and the removal of the Sigmoid activation at the output. The BAGAN model extends the architecture with an autoencoder composed of an encoder and decoder alongside the generator and discriminator. The encoder uses convolutional layers with LeakyReLU, while the decoder incorporates transposed convolution, batch normalization, and LeakyReLU activations. To simplify this study, we refer readers to our previous aforementioned work for a detailed description of the GAN augmentation models and training parameters.

Table 1
Hyperparameter search space for optimization and default baseline values (in last column)

Hyperparameter search space	Baseline values	Default
conv2d_1, conv2d_2 (filter size for the first two convolution layers)	[16, 32, 64, 128, 256]	16
conv2d_3, conv2d_4 (filter size for the first two convolution layers)	[16, 32, 64, 128, 256]	32
dropout_1 (dropout rate for regularization)	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]	0.2
layer_1 (no. of neurons in fully connected layer)	[16, 32, 64, 128, 256]	64
activation_1 (activation function used in hidden layers)	[relu, 'tanh']	'relu'
optimizer (type of optimization algorithm)	['adam', 'adagrad', 'sgd', 'rmsprop']	'adam'
batch_size (size of the batch)	[8, 16, 32, 64]	8
epochs (number of training epochs)	[10, 20, 30, 40]	20
data_type (dataset version — original dataset ‘real’ or version extended using GAN augmentations)	[‘real’, ‘BAGAN’, ‘DAGAN’, ‘DCGAN’, ‘GAN’, ‘WGAN’]	‘real’

5.3 Setup of Optimization Methods

Grid Search was primarily used for comparison with the other optimization algorithms in terms of classification success and the time required for optimization.

For the Grid Search (GS) and Random Search (RS) algorithm, it was necessary to define the maximum number of iterations, which was set to 100.

In this study, we use six selected algorithms from the Hyperactive and Sherpa libraries for hyperparameter optimization. Each algorithm requires parameter specification prior to optimization, and the same configuration of optimization algorithms was applied across all datasets. The selected parameter values for all optimization algorithms are summarized in Table 2.

Table 2
Parameter values used for each optimization algorithm: Evolution Strategy (ES), Population Based Training (PBT), Particle Swarm Optimization (PSO), Simulated Annealing (SA), and Parallel Tempering (PT). A dash (–) indicates that the parameter is not applicable for the given algorithm

Parameter	ES	PBT	PSO	SA	PT
Population size	10	10	10	-	10
Mutation rate	0.7	-	-	-	-
Crossover rate	0.3	-	-	-	-
Random position	0	-	0	0	0
Number of generations	-	10	-	-	-
Inertia	-	-	0.4	-	-
Cognitive weight	-	-	0.7	-	-
Social weight	-	-	0.7	-	-
Epsilon	-	-	-	0.05	-
Distribution	-	-	-	‘normal’	-
Number of neighbors	-	-	-	10	-
Annealing rate	-	-	-	0.975	-
Start temperature	-	-	-	1	-
Iterations before swap	-	-	-	-	10

For ES, the required parameters included population size, mutation rate, crossover rate (the probability of mating the best individual), and the probability of jumping to a random position during the iteration. For PBT, population size and the number of generations were specified. The population size determined the number of iterations in each generation. In the first generation, all configurations were randomly initialized. For subsequent generations, 80% of the best-performing configurations were retained, while the remaining 20% were resampled using the top 20% of the previous generation.

PSO algorithm required the specification of population size, inertia, cognitive weight (determines the movement towards an individual's locally best position in the population), social weight (defines the movement towards the globally best position in the population), and the probability of jumping to a random position.

SA involved defining epsilon (the jump distance from the current position), the distribution of the data, number of neighbors evaluated before moving to a new position, the probability of jumping to a random position, the annealing rate (defines the start of the temperature reduction process), and the initial temperature (which determines the probability of jumping to a worse position). For PT, the required parameters included the population size, the number of iterations before a temperature swap, and the probability of jumping to a random position.

6 Evaluation

The performance of the classification model was evaluated using standard metrics, including accuracy, precision, recall, and F1 score. To further analyze the distribution of correctly and incorrectly classified instances, a confusion matrix was employed. All datasets were split into training and testing subsets, all presented results are on test sets. The metrics are defined by following equations:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

6.1 Galaxy Mergers

The baseline model for the Galaxy Mergers dataset, using the hyperparameters listed in Table 2 achieved a classification accuracy of 89.72%. This baseline serves as the starting point for evaluating the impact of optimization algorithms and data augmentation. Detailed performance metrics, including the confusion matrix and classification report (with precision, recall, F1), are shown in Table 3 and Table 4.

Table 3
Confusion matrix of the baseline model for the Galaxy Mergers dataset

		Actual	
		Merger	**Non-interacting**
Predicted	**Merger**	1226	272
	Non-interacting	139	2361

Table 4
Classification report of the baseline model for the Galaxy Mergers dataset

Class	Precision	Recall	F1 score	Support
Merger	0.89817	0.81842	0.85644	1498
Non-interacting	0.89670	0.94440	0.91993	2500

In our optimization experiments, we tested a range of algorithms to find best model, focusing not only on classification accuracy but also on efficiency - how quickly and effectively an algorithm can converge to a high-performing solution. While GS identified the highest overall accuracy at 93.42% due to brute force search, this required over 26 hours of computation (96745 seconds). This result highlights limitations of GS as time-inefficient and computationally expensive for practical usage in automatic workflows. Instead, metaheuristic algorithms reached nearly the same level of performance with dramatically lower computational cost. A summary of the accuracy vs. time trade-offs is shown in Table 5.

Table 5
Optimization results for the Galaxy Mergers dataset

Algorithm	GS	RS	SA	PT	PSO	PBT	ES
Accuracy (%)	93.42	93.30	93.20	93.15	92.42	92.22	92.17
Δ Accuracy (%)	-	- 0.12	- 0.22	- 0.27	- 1.00	- 1.22	- 1.25
Time (s)	96745	28354	26409	2006	1700	6004	5303
Time Saved (%)	-	70.7	72.7	97.9	98.2	93.8	94.5

From this comparison, several clear insights emerge:

- Random Search and Simulated Annealing achieved nearly identical accuracy to Grid Search—within 0.22%—but required only ~30% of the time, making them far more practical for real-world applications.
- Parallel Tempering achieved an accuracy just 0.27% lower than GS, yet finished in only 2006 seconds (just over 33 minutes) - a time saving of nearly 98%, therefore providing very cost-effective method while preserving high classification performance.
- Algorithms like Particle Swarm Optimization and Population-Based Training, despite losing 1.00–1.22% in accuracy, executed in under 10% of the time compared to GS. These are also compelling choices when rapid iteration is prioritized over absolute accuracy.

These results show that Parallel Tempering is the most practical optimization algorithm for this task when balancing accuracy and computation time. The combination of PT and DCGAN resulted in a highly efficient model. This configuration achieved an accuracy of 93.15%, while reducing training time by nearly 98% compared to Grid Search. Its confusion matrix and report on other metrics are presented in Table 6 and Table 7.

Table 6
Confusion matrix of the model found by Parallel Tempering for the Galaxy Mergers dataset

		Actual	
		Merger	**Non-interacting**
Predicted	**Merger**	1388	164
	Non-interacting	110	2336

Table 7
Classification report of the model found by Parallel Tempering for the Galaxy Mergers dataset

Class	**Precision**	**Recall**	**F1 score**	**Support**
Merger	0. 8944	0. 9266	0. 9103	1498
Non-interacting	0. 9550	0. 9344	0. 9446	2500

As we can see, metaheuristic approaches, like PT, achieve near-optimal accuracy with significantly reduced resources. DCGAN-based augmentation further improved performance, increasing accuracy by over 3.5 % for the best model.

6.2 FIRST Data – Morphology of Radio Galaxies

The baseline model for the FIRST Data dataset, using the hyperparameters listed in Table 2, achieved a classification accuracy of 85.85%. The confusion matrix and other metrics for the baseline model are presented in Table 8 and Table 9. Several optimization algorithms were applied to FIRST dataset to improve model performance. Each method was evaluated based on two key criteria: classification accuracy and optimization time. Comparative results are shown in Table 10.

Table 8
Confusion matrix of the baseline model for the FIRST dataset

		Actual			
		BENT	**COMP**	**FRI**	**FRII**
Predicted	**BENT**	17	0	1	3
	COMP	0	17	0	0
	FRI	0	1	21	3
	FRII	4	0	3	36

Table 9
Classification report of the baseline model for the FIRST dataset

Class	Precision	Recall	F1 score	Support
BENT	0.80952	0.80952	0.80952	21
COMP	0.94444	1.00000	0.97143	17
FRI	0.84000	0.84000	0.84000	25
FRII	0.85714	0.83721	0.84706	43

Table 10
Optimization results for the FIRST dataset

Algorithm	GS	RS	SA	PT	PSO	PBT	ES
Accuracy (%)	94.34	90.56	88.68	93.40	91.51	87.73	87.73
Δ Accuracy (%)	-	- 3.78	-5.66	-0.94	-2.83	-6.61	-6.61
Time (s)	24366	993	1098	213	692	500	480
Time Saved (%)	-	95.9	95.5	99.1	97.2	97.9	98.0

The following insights emerge from this evaluation:

- Grid Search attained the highest accuracy 94.34% but required nearly 6.8 hours of optimization time, highlighting its inefficiency for practical scenarios despite its superior performance.
- Parallel Tempering closely followed with an accuracy of 93.40%, just 0.94 percentage points below GS, but required only 213 seconds (~3.5 minutes). This yields a 99.1% time saving, making it the most efficient algorithm in this comparison.
- Particle Swarm Optimization also delivered strong results, achieving 91.51% accuracy in 692 seconds (~11.5 minutes), offering a well-balanced compromise between performance and speed.
- Random Search, Simulated Annealing, and Evolution Strategy demonstrated varying levels of accuracy, but took longer or matched less-effective trade-offs between time and performance.

Results show that PT is the most practical optimization algorithm for the FIRST dataset when balancing accuracy and computation time. The combination of PT and BAGAN resulted in the best-performing model. This configuration achieved an accuracy of 93.40%, improving upon the baseline by over 7.5%. The confusion matrix and classification report for PT model are shown in Table 11 and Table 12.

Table 11
Confusion matrix of the model found by Parallel Tempering for the FIRST dataset

		Actual			
		BENT	**COMP**	**FRI**	**FRII**
Predicted	**BENT**	21	0	0	0
	COMP	0	17	0	0
	FRI	0	0	23	2
	FRII	4	0	1	38

Table 12
Classification report of the model found by Parallel Tempering model for the FIRST dataset

Class	**Precision**	**Recall**	**F1 score**	**Support**
BENT	0. 84000	1. 00000	0. 91304	21
COMP	1.00000	1.00000	1.00000	17
FRI	0. 95833	0. 92000	0. 93878	25
FRII	0. 95000	0. 88372	0. 91566	43

Consistent improvements across all methods, especially those using BAGAN-generated data, demonstrate the value of synthetic augmentation for enhancing generalization in datasets like FIRST with underrepresented classes. In combination with PT optimization, it provides a powerful and efficient approach.

6.3 Galaxy ZOO – Rings

The baseline model for the Galaxy ZOO - Rings dataset, using the hyperparameters listed in Table 2, achieved a classification accuracy of 65.0%. This serves as the reference point for evaluating subsequent improvements through optimization and data augmentation. Detailed performance metrics, including the confusion matrix and other metrics, are shown in Table 13 and Table 14. Optimization algorithms were then applied to improve accuracy, focusing both on near-peak performance and time, with results summarized in Table 15.

Table 13
Confusion matrix of the baseline model for the Galaxy ZOO - Rings dataset

		Actual	
		Ring	**No-ring**
Predicted	**Ring**	52	28
	No-ring	42	78

Table 14
Classification report of the baseline model for the Galaxy ZOO - Rings dataset

Class	Precision	Recall	F1 score	Support
Ring	0.55319	0.65000	0.59770	80
No-ring	0.73585	0.65000	0.69027	120

Table 15
Optimization results for the Galaxy ZOO - Rings dataset

Algorithm	GS	RS	SA	PT	PSO	PBT	ES
Accuracy (%)	79.50	75.99	75.00	75.00	77.40	74.00	76.49
Δ Accuracy (%)	-	- 3.51	- 4.50	- 4.50	- 2.10	- 5.50	- 3.01
Time (s)	13196	2053	553	182	911	721	614
Time Saved (%)	-	84.4	95.8	98.6	93.1	94.5	95.3

From this comparison, several important observations can be made:

- PSO delivered a competitive accuracy of 77.40% - just 2.1% below GS - but completed its optimization in just 911 seconds (~15 minutes). This represents a 93% time saving, making it best choice in this case.
- Evolution Strategy also performed well, reaching 76.49% in only 614 seconds (~10 minutes). Although it lags slightly behind PSO in accuracy, it also provides an excellent trade-off between speed and performance.
- Parallel Tempering or Simulated Annealing completed in under 10 minutes, offering rapid convergence with a slightly lower peak accuracy.

Results show that when time and efficiency are both prioritized, as they typically are in real-world scenarios (like automated processing workflows), metaheuristic algorithms like PSO and ES offer a superior balance of speed and performance.

The best-performing model was obtained using PSO combined with DCGAN data augmentation, further boosting the model's ability to detect ring structures in galaxies. This configuration reached an accuracy of 77.40%, outperforming the baseline by more than 12%. The confusion matrix and classification report are provided in Table 16 and Table 17.

In datasets with limited or imbalanced classes like Galaxy Zoo–Rings, DCGAN-based augmentation enhanced model generalization. For this dataset, combining efficient optimization algorithms with targeted data augmentation yielded substantial performance gains. Among these, PSO with DCGAN-augmented data achieved the best performance without the computational cost of exhaustive search methods.

Table 16
Confusion matrix of the model found by Particle Swarm Optimization for the Galaxy ZOO – Rings

		Actual	
		Ring	**No-ring**
Predicted	**Ring**	68	12
	No-ring	33	87

Table 17
Classification report of the model found by Particle Swarm Optimization for the Galaxy ZOO – Rings

Class	**Precision**	**Recall**	**F1 score**	**Support**
Ring	0.67327	0.85000	0.75138	80
No-ring	0.87879	0.72500	0.79452	120

At the end, we provide the overview of the best setup of CNNs for each dataset found by the best optimization technique in every use case. The final optimized hyperparameter settings of best models for all evaluated datasets are summarized in Table 18.

Table 18
Optimized hyperparameters of the CNN model for individual datasets

DATASET	**BASELINE**	**Galaxy Mergers**	**FIRST Data**	**Galaxy ZOO – Rings**
conv2d_1, conv2d_2	16	256	64	64
conv2d_3, conv2d_4	32	16	64	32
dropout_1	0.2	0.1	0.2	0.4
layer_1	64	128	16	32
activation_1	'relu'	'relu'	'tanh'	'tanh'
optimizer	'adam'	'adam'	'adam'	'rmsprop'
batch_size	8	32	16	8
epochs	20	10	30	10
data_type	‘real’	DCGAN	BAGAN	DCGAN

7 Conclusions

This study explored the potential for combination of parameter optimization and GAN-based data augmentation in enhancing CNN-based classification across three astronomical datasets: Galaxy Mergers, FIRST, and Galaxy Zoo – Rings. Our findings clearly demonstrate that both optimization strategy and synthetic data augmentation, substantially influence model accuracy, efficiency and overall practical viability.

For the Galaxy Mergers dataset, Parallel Tempering achieved 93.15% accuracy with a runtime of just over 33 minutes, offering the best trade-off between performance and efficiency. While Grid Search reached a slightly higher accuracy (93.42%), its runtime exceeded 26 hours. Random Search also performed well (93.30%) with reduced time, but Parallel Tempering delivered similar accuracy with even greater computational savings, making it the most practical choice.

For the FIRST dataset, Parallel Tempering emerged as the optimal algorithm, offering a near-peak accuracy of 93.40% with a runtime of only 213 seconds - representing a 99.1% time reduction compared to Grid Search. When paired with BAGAN-based augmentation, it delivered the most effective model, underscoring the strength of GANs in mitigating class imbalance and data scarcity.

For Galaxy Zoo - Rings, where baseline accuracy was relatively low (65.0%), the application of Particle Swarm Optimization and DCGAN augmentation produced the best practical results, improving accuracy to 77.40%. Despite Grid Search reaching a slightly higher peak, PSO's significantly lower runtime made it the most feasible solution for deployment.

Across all datasets, GAN-based augmentation - especially DCGAN and BAGAN - consistently enhanced model generalization, particularly in cases of imbalanced or limited data. Similarly, metaheuristic optimization methods like Parallel Tempering and PSO proved to be highly effective in achieving competitive accuracies with dramatically reduced computational costs.

Our results highlight that optimal model performance is not solely about accuracy, it is about balancing performance with efficiency. The integration of efficient optimization algorithms and synthetic augmentation techniques offers a scalable and resource-aware approach for astronomical image classification, especially for the future needs for automated processing workflows with the new and larger astronomical datasets.

Acknowledgement

This work was supported by the Slovak VEGA research grant no. 1/0685/21.

References

[1] Goodfellow, I. et al.: Generative adversarial networks, Communications of the ACM, 63(11), 2020, 139-144

[2] Antoniou, A., Storkey, A. and Edwards, H.: Data augmentation generative adversarial networks, 2018

[3] Mariani, G. et al.: BAGAN: Data augmentation with balancing GAN, arXiv preprint, arXiv:1803.09655, 2018

[4] Mumuni, A. and Mumuni, F.: Data augmentation with automated machine learning: approaches and performance comparison with classical data augmentation methods, arXiv e-prints, arXiv:2403.08352, 2024

[5] Veedhi, B.K. et al.: Balancing data imbalance in biomedical datasets using a stacked augmentation approach with STDA, DAGAN, and pufferfish optimization to reveal AI's transformative impact, International Journal of Information Technology, 17, 2025, 455-480

[6] Kališková, L. and Butka, P.: Improvement of Classification Results of Convolutional Neural Networks Using Various GAN-Based Augmentation Techniques, Acta Electrotechnica et Informatica, 24(4), 2024, 11-18

[7] Jaderberg, M., Dalibard, V., Osindero, S., et al.: Population based training of neural networks, arXiv preprint arXiv:1711.09846, 2017

[8] Qolomany, B. et al.: Parameters optimization of deep learning models using particle swarm optimization, Proc. of 13th Int. Wireless Communications and Mobile Computing Conf. (IWCMC), IEEE, 2017, 1285-1290

[9] Serizawa, T. and Fujita, H.: Optimization of convolutional neural network using the linearly decreasing weight particle swarm optimization, arXiv preprint arXiv:2001.05670, 2020

[10] Xiao, X. et al.: Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm, arXiv preprint arXiv:2006.12703, 2020

[11] LeCun, Y., Bengio, Y., and Hinton, G.: Deep learning, Nature, 521(7553), 2015, 436-444

[12] Radford, A., Metz, L., and Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv preprint arXiv:1511.06434, 2015

[13] Arjovsky, M., Chintala, S., and Bottou, L.: Wasserstein generative adversarial networks, Proc. of Int. Conf. on Machine Learning, PMLR, 2017, 214-223

[14] Bergstra, J. and Bengio, Y.: Random search for hyper-parameter optimization, Journal of Machine Learning Research, 13(2), 2012, 281-305

[15] Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y. and Schmidhuber, J.: Natural evolution strategies, arXiv preprint arXiv:1106.4487, 2011

[16] Lorenzo, P. et al.: Particle swarm optimization for hyper-parameter selection in deep neural networks, Proc. of the Genetic and Evolutionary Computation Conference, 2017, 481-488

[17] Pushkarov, V., Efroni, J., Maksymenko, M. and Koch-Janusz, M.: Training deep neural networks by optimizing over nonlocal paths in hyperparameter space, arXiv preprint arXiv:1909.04013, 2019