Image Processing: Impact of Train and Test Sizes on Custom Image Recognition Algorithms

Luis Marques, Luca Lopes, Miguel Ferreira, Cristina Wanzeller, Pedro Martins, and Maryam Abbasi

Abstract This paper intends to demonstrate results on applying machine learning algorithms to process image recognition to identify professions. This kind of project points us to a relation between humans and machines, so in a way, we might say that the human brain and vision process are being passed to a machine in order to bring us many benefits in our daily life. In this paper, we decided to compare how different parameters influence the performance and accuracy of the following neural networks: EfficientNetB0, NASNetMobile, MobileNetV2, ResNet50, InceptionV3, and DenseNet121.

Keywords Machine learning · Image recognition · ResNet · InceptionV3 · MobileNetV2 · DenseNet · NASNetMobile · EfficientNet

L. Marques · L. Lopes · M. Ferreira
Polytechnic of Viseu, Viseu, Portugal
e-mail: estgv5989@alunos.estgv.ipv.pt

L. Lopes
e-mail: stgv13082@alunos.estgv.ipv.pt

M. Ferreira
e-mail: estgv7447@alunos.estgv.ipv.pt

C. Wanzeller · P. Martins (✉)
CISeD—Research Centre in Digital Services, Polytechnic of Viseu, Viseu, Portugal
e-mail: pedromom@estgv.ipv.pt

C. Wanzeller
e-mail: cwanzeller@estgv.ipv.pt

M. Abbasi
CISUC—Centre for Informatics and Systems of the University of Coimbra, Coimbra, Portugal
e-mail: maryam@dei.uc.pt

J. L. Reis et al. (eds.), *Marketing and Smart Technologies*, Smart Innovation, Systems and Technologies 279, https://doi.org/10.1007/978-981-16-9268-0_30

1 Introduction

Image classification is used in various applications, such as security, educational, and promotional systems. In recent years, much research has been done to design automated systems to extract fundamental features from images. Convolutional neural network (CNN) is an effective method for image classification which uses convolutional, pooling, and fully connected layers for the learning process [11]. CNN has redefined the state of the art in many real-world applications, such as facial recognition, image classification, human pose estimation, and semantic segmentation [19]

The training of machine learning models for image processing is a process that consumes many resources [10]. Despite the computational power we currently have at our disposal in personal computers, especially in graphics processing units (GPU) [5], this power is still limited for usage in computer vision, mainly in CNNs. This class of application usually relies on heavy computations on massive datasets. Therefore, parallel computing is traditionally considered to run the training process in a feasible time using the GPU [2]. Acquiring these hardware implies risks under and over-utilization, depreciation of the hardware, and failures. There are also costs related to maintenance, energy, and human resources [2]. However, there are solutions in the cloud in all significant providers, Amazon AWS [16], Microsoft Azure [1], and Google Cloud [3], that can aid in this process, making the machine learning process more accessible. The researcher/developer does not take the risk of acquiring hardware that can quickly become obsolete and only pay for the resources that he will use [10]. However, some free alternatives in the cloud, namely Google Colaboratory (commonly referred to as "Google Colab" or just "Colab") [6] and Paperspace Gradient [14], are the ones used in this work. Computational resources are not the only requirement for training machine learning models; in-depth knowledge of applied mathematics and deep learning libraries is also required [10]. While this might pose a problem, there are, however, libraries that simplify this challenge, namely ImageAI. ImageAI is a Python library built to empower developers, researchers, and students to build applications and systems with self-contained deep learning and computer vision capabilities using simple and few lines of code [13].

This study aims to compare some of the ImageAI provided algorithms ResNet50, MobileNetV2, InceptionV3, DenseNet121, and two more EfficientNetB0, NASNet-Mobile, which were made available through some custom code, on the IdenProf [12] dataset and two customizations of it, with different train and test sizes in order to recognize professions in images, using Paperspace Gradient and Google Colab as research environments. This document is organized into five sections. In Sect. 2, we discuss related work in the area. Section 3 presents the method applied in study. Section 4 shows the obtained results. Finally, in Sect. 5, conclusions are drawn, followed by future work guidelines.

2 Related Work(s)

This paper focuses on the topic of computer vision and its technology. It is easy for humans to describe and understand the objects that we see from the world. Our visual system can perceive a three-dimensional structure with enough information, such as the objects' shape, appearance, and color. However, this is not easy for a computer [20]. Researchers in this field try to mimic the capacity of how human vision works using computers. However, it is not an easy task, and literature on artificial visual processing is usually categorized into visual processing algorithms, which consist in the recreations of the human vision, and classifiers, which are remodeling of the human decision techniques [4].

Computer vision is a vast research field where mathematics, geometry, and physics are applied [20]. However, some tasks are commonly accomplished with computer vision, object detection, recognition, and classification. This paper is focused on the classification part. Image classification was once a task that required domain expertise and the use of problem-specific models. Much of this has changed with the emergence of deep learning as a general-purpose modeling technique for predictive tasks in computer vision. Both the machine learning literature and image classification contests are now dominated by deep learning models that often do not require domain expertise since such models identify and extract features automatically, eliminating the need for feature engineering [9].

Usage of libraries like ImageAI allows us to train and generate image classification models with CNN without extensive knowledge of the inner network workings. However, it allows us to use its potential as this kind of algorithm is broadly used in computer vision. However, the requirements of computation for training models with these algorithms are high. With our experimentation, we used ResNet50 [7], DenseNet121 [8], and InceptionV3 [17] as these are also used by the majority of studies related to image object recognition, and we also used MobileNetv2 [15], NASNetMobile [21], and EfficientNetB0 [18]. CNNs provide high accuracy. The main reason for this is because the number of features increases dramatically. The research done about computer vision relies on the accuracy of the validation and improving that accuracy. Also, studies are showing us that if we keep training our model with thousands of pictures, we can reach into overfitting issues. There is a balance that needs to be respected when training models [10] Last but not least, as shown by some studies, the quality of the datasets impacts creating and training a model.

Setting up an on-premises solution to research and build models from machine learning algorithms can be expensive and not the only available option. All major cloud providers offer services in the cloud for the same purpose with access to a huge computing capacity. Some of these offers are Amazon Sagemaker [16], Microsoft Azure Machine Learning [1], Google Cloud AI infrastructure [3], and also free options like Google Colab [6] or Paperspace Gradient [14]. These last two platforms used in the experiment.

3 Experimental Setup

This project's architecture consists of using Paperspace Gradient provided Docker containers, which provided the necessary infrastructure for the code developed in Jupyter Notebooks and the base storage. Google Colab was also used (and integrated with Google Drive) in the final work for collecting graphics and metrics of TensorBoard logs.

The Paperspace Gradient free supplied containers include dedicated NVIDIA Maxwell GPU with 8 GB of GPU memory, 30 GB of memory, 8vCPU, and 5 GB of storage space. These resources are all free but with a limit of the run session of 6 h maximum. For Google Colab, the type of GPUs available in Colab varies over time, often including Nvidia K80s, T4s, P4s, and P100s. The standard available RAM in Colab is 12 GB. It should be noted that these resources are shared between users of the platform, so the available capacity of the resources varies over time.

3.1 Datasets

The author of ImageAI [13] Python library also created a dataset IdenProf [12], which was used as the base for this study, but we also created two custom datasets from it. The IdenProf dataset contains 11,000 images that span over ten categories of professions. Each profession category consists of 1100 images, 900 of which are used for training and the remaining 200 for testing. Our custom datasets consist of the same 1100 as the base dataset. However, the training and test sizes are different, 800 and 300 for one dataset and 1000 and 100. These datasets will be referenced as DS100, DS200, and DS300, matching the respective test sizes from now on.

The images in the dataset have a resolution of 224 * 224 pixels and represent subjects dressed in uniforms of their respective professions. The dataset distribution as of represented subject is as follows, 19.4% female an 80.6% male, 91.1% white, and 8.9% dark skins.

The process of acquisitions of the dataset images, as the dataset author describes it "*The images in the dataset were obtained from Google Image search. The images were searched and collected based on the 15 most populated countries in the world. The dataset does not comply with EU GDPR has the individuals whose images were contained were not explicitly contacted for consent*" [12].

3.2 Parameters

ImageAI library provides several algorithms that can be used for image classification, namely ResNet50, DenseNet121, MobileNetV2, and InceptionV2. Other two

Table 1 Setup parameters

Training cycle						
Epochs 25						
Per Epoch iterations						
	DS100		DS200		DS300	
Batch size	Train	Validation	Train	Validation	Train	Validation
16	625	62	562	125	500	187
32	312	31	281	62	250	93
64	156	15	140	31	125	46

algorithms not present in ImageAI were also used, namely NASNetMobile and EfficientNetB0. All algorithms were used to train models from the datasets during 25 epochs in three different image batch sizes 16, 32, and 64 for the three datasets.

A preliminary test using a batch size of 128 was additionally thought, but due to resources limitations and time constraints was not carried over. In Table 1, we can observe a resume of setup parameters.

4 Results and Analysis

From the facts gathered, accordingly to Table 2 InceptionV3 was the algorithm that lost less trainable parameters followed by ResNet50, all other algorithms had losses over 1%. Both algorithms also detect more parameters, being, in this case, ResNet50 has more parameters detected, which translates into more significant model sizes. MobileNetV2 algorithm was the one that lost most parameters that could be trained; also, this was the one that detected fewer parameters, which translates to smaller size models. A final fact we can observe from analyzing all algorithms, as the number of parameters detected grows, so does the model sizes grow proportionally.

As we can observe in Table 3, related to temporal data, the faster algorithm in dataset DS100 was MobileNetV2 for batch sizes 16 and 64; however, for a batch size of 32, it was EfficientNetB0. For dataset DS200, MobileNetV2 was faster for batch size 16, in batch size 32, it was EfficientNetB0, and in batch size 64, it was NASNetMobile. For dataset, DS300 EfficientNetB0 was faster in batch size 16 but slowest in batch size 32, where InceptionV3 was faster with batch size 64.

In other facts gathered, for dataset DS100, there was a reduction in the train time from a batch size of 16 to a batch size of 64. As of dataset DS200, that only happened with NASNetMobile and EfficientNetB0, from batch size 16 to batch size 32, there was an overall reduction excluding MobileNetV2. In dataset DS300, only NASNetMobile had time reduction from batch size 16 to batch size 64, and all others

Table 2 Algorithms facts collected

	ResNet50	DenseNet121	InceptionV3	MobileNetV2	NASNetMobile	EfficientNetB0
Total pa rams	23,608,202	7,047,754	21,823,274	2,270,794	4,280,286	4,062,381
Trainable pa rams	23,555,082	6,964,106	21,788,842	2,236,682	4,243,548	4,020,358
Non trainable pa rams	53,120	83,648	34,432	34,112	36,738	42,023
Of params model (%)	0.23	1.19	0.16	1.50	0.86	1.03
Size (MB)	90.5	27.8	83.9	9	17.9	16

Table 3 Training times facts

Dataset	Batch size	ResNet50	DenseNet121	InceptionV3Mobile	MobileNetV2	NASNetMobile	EfficientNetB0
DS1000	16	52m46s	48m07s	47m50s	46m59s	58m38s	47m22s
	32	55m46s	50m21s	50m27s	56m00s	51m24s	50m11s
	64	43m09s	42m49s	43m06s	41m46s	47m45s	42m49s
DS2000	16	40m5s	41m23s	42m11s	38m4s	40m5s	51m11s
	32	36m43s	36m38s	35m53s	39m45s	40m32s	36m17s
	64	54m02s	47m23s	46m45s	50m43s	46m22s	46m20s
DS200	16	50m01s	42m56s	45m37s	53m13s	52m54s	40m13s
	32	48m55s	46m15s	45m58s	46m46ss	47m11s	52m48s
	64	53m26s	51m11s	47m01s	53m32s	48m04s	47m48s

had time increased. For DenseNet121 and InceptionV3, as the batch size increased, so did the train time. We can extract from the facts that the best training time obtained across all datasets and all batch sizes was for InceptionV3 in dataset DS200 and batch size of 32.

According to Tables 4 and 5, we can observe that for all datasets and all batch sizes, the training accuracy is slightly better than validation accuracy; however, for NASNetMobile algorithm only for batch sizes of 16, this difference is slight for other batch sizes, validation accuracy is much lower, and the same behavior occurs with MobileNetV2 for dataset DS100 on a batch size of 32 and remaining datasets in batch size 64. Also for an expected behavior with all algorithms across all datasets and batch sizes, accuracy tends to increase with batch size increase, as for validation accuracy that tendency is inverse, meaning with batch size increase validation accuracy tends to decrease, but for some batch sizes and algorithms that is not always the case. Analyzing Table 4, another result was gathered, for all algorithms without exception, in dataset DS100 higher accuracies were achieved than those of DS200, and these were greater than DS300.

Table 6 is relative to the training loss. This is a metric worth analyzing, as this can indicate how good the predictive model is, as lower the loss, the better the predictions are. Observing these values, we see some similarities between some algorithms. In datasets, DS200 and DS300, all algorithms excluding EfficientNetB0 decreased loss as the batch size increased, as for EfficientNetB0 decreased from batch size 16–32 and increased slightly from batch size 32–64, still lower than batch size 16. EfficientNetB0 and MobilNetV2 for dataset DS100 decreased loss as the batch size increased, while all other algorithms had the same behavior, decreasing from batch size 16–32 and increasing from 32 to 64. NASNetMobile had the lower loss in all datasets, in DS100 was in batch size 32, as for the others was with 64.

Observing the values in Table 7, relative to the validation loss, we saw some similarities between some algorithms, and MobileNetV2 and NASNetMobile stand out as having much higher losses than the rest in all datasets for the majority of batch sizes. For EfficientNetB0, the loss behavior was the same per batch size across the datasets, increasing loss as the batch size increased. ResNet50 for dataset DS100 and DS200 performed the same with loss decrease from batch size 16–32, but with an increase in 64, while for dataset DS300, as the batch size increased, the loss decreased. DenseNet121 and InceptionV3 had the same behavior as ResNet50 in dataset DS100, while in dataset DS200, InceptionV3 maintained the same behavior, and DenseNet121 increased loss as batch size increased for DS200 and DS300 datasets. The lower loss was obtained with DenseNet121 for dataset DS100 and batch size of 64, while for the remaining datasets, InceptionV3 had a lower loss with batch size 32.

Table 8 represents the difference between the losses from training and from validation. This is a fundamental metric to pay attention to, as we can check if the models trained might be overfitting or underfitting. Ideally, this difference should be zero, or as close as we could get, but usually, some overfitting occurs. At first glance, NASNetMobile stands out as having a more significant difference than the

Table 4 Train accuracy on datasets across batch sizes

Dataset	Batch Size	ResNet50	DenseNet121	InceptionV3	MobileNetV2	NASNetMobile	EfficientNetB0
DS100	16	0.8293	0.8877	0.867	0.8321	0.918	0.8481
	32	0.898	0.9111	0.9258	0.8471	0.9444	0.8586
	64	0.8392	0.9044	0.9045	0.857	0.9297	0.8669
DS200	16	0.8251	0.8659	0.8575	0.808	0.9154	0.8231
	32	0.8695	0.8884	0.898	0.8388	0.9296	0.8553
	64	0.8704	0.9041	0.926	0.8499	0.9335	0.8494
DS300	16	0.7979	0.8524	0.8356	0.8026	0.9145	0.8155
	32	0.8129	0.871	0.8845	0.8285	0.9125	0.8395
	64	0.8696	0.8835	0.924	0.8436	0.9381	0.8444

Table 5 Validation accuracy across batch sizes and datasets

Dataset	Batch Size	ResNet50	DenseNet121	InceptionV3	MobileNetV2	NASNetMobile	EfficientNetB0
DS100	16	0,8034	0,8367	0.8286	0.8105	0.8024	0.8054
	32	0.8115	0.8313	0.8406	0.1	0.1917	0.7927
	64	0.7853	0.8327	0.8468	0.8075	0.2601	0.7923
DS200	16	0.7845	0.821	0.814	0.7835	0.7055	0.784
	32	0.7974	0.8196	0.8311	0.7263	0.2041	0.7823
	64	0.7918	0.814	0.8191	0.1003	0.126	0.7777
DS300	16	0.7784	0.8142	0.8031	0.7878	0.6397	0.7841
	32	0.7779	0.8095	0.8222	0.4694	0.2023	0.7769
	64	0.7952	0.8077	0.8173	0.1005	0.1345	0.7748

Table 6 Train loss across batch sizes and datasets

Dataset	Batch Size	ResNet50	DenseNet121	InceptionV3	MobileNetV2	NASNetMobile	EfficientNetB0
DS100	16	0.4973	0.3427	0.3834	0.4956	0.2385	0.4351
	32	0.3066	0.2774	0.224	0.4427	0.1705	0.4128
	64	0.4583	0.2882	0.2749	0.4227	0.2075	0.3772
DS200	16	0.5881	0.3981	0.415	0.5556	0.2527	0.5101
	32	0.373	0.3325	0.2943	0.4764	0.2156	0.4117
	64	0.3666	0.2879	0.2195	0.4316	0.199	0.432
DS300	16	0.5814	0.438	0.4736	0.5734	0.261	0.5373
	32	0.5338	0.3879	0.3352	0.5006	0.2491	0.4654
	64	0.3784	0.3397	0.2248	0.4619	0.18	0.4504

Table 7 Validation loss across batch sizes and datasets

Dataset	Batch Size	ResNet50	DenseNet121	InceptionV3	MobileNetV2	NASNetMobile	EfficientNetB0
DS100	16	0.6164	0.5086	0.5151	0.5727	0.6552	0.6203
	32	0.5803	0.5332	0.589	2.924	4.907	0.6632
	64	0.6683	0.4919	0.5327	0.6025	3.721	0.6832
DS200	16	0.6401	0.5507	0.5491	0.6252	0.9749	0.6499
	32	0.6386	0.5514	0.5429	0.7716	3.91	0.6525
	64	0.672	0.544	0.6154	3.227	4.247	0.6919
DS300	16	0.6501	0.5431	0.5533	0.62	1.179	0.636
	32	0.6469	0.5497	0.5378	1.632	3.329	0.6439
	64	0.6239	0.5538	0.5835	3.524	4.561	0.6943

Table 8 Difference between validation loss and training loss across batch sizes and datasets

Dataset	Batch Size	ResNet50	DenseNet121	InceptionV3	MobileNetV2	NASNetMobile	EfficientNetB0
DS100	16	0.1191	0.1659	0.1317	0.0771	0.4167	0.1852
	32	0.2737	0.2558	0.365	2.4813	4.7365	0.2504
	64	0.21	0.2037	0.2578	0.1798	3.5135	0.306
DS200	16	0.052	0.1526	0.1341	0.0696	0.7222	0.1398
	32	0.2656	0.2189	0.2486	0.2952	3.6944	0.2408
	64	0.3054	0.2561	0.3959	2.7954	4.048	0.2599
DS300	16	0.0687	0.1051	0.0797	0.0466	0.918	0.0987
	32	0.1131	0.1618	0.2026	1.1314	3.0799	0.1785
	64	0.2455	0.2141	0.3587	3.0621	4.381	0.2439

rest for all datasets, which indicates a case of overfitting. MobileNetV2 also generated models overfitting in all datasets, for batch size 32 in DS100 and batch sizes 32 and 64 for DS300; as for batch size 16 in all datasets, this algorithm trained some of the best-adjusted models. Overhaul as the batch size increases, so do the generated models overfitting. The less overfitting models for datasets DS100 and DS300 were MobileNetV2 with a batch size of 16, as for DS200, it was ResNet50 also with a batch size of 16.

5 Conclusions and Future Work

After testing the six algorithms and observing the results, we can conclude that all algorithms tend to achieve higher accuracy when the dataset train size increased and test size increased. Another conclusion we extract is that accuracy is achieved between algorithms and batch sizes are similar, with slight differences. The best algorithm does not exist; they all tend to adapt to the datasets.

For the three datasets, the algorithm which achieved higher accuracy under 25 epochs was InceptionV3. Excluding NASNetMobile and MobileNetV2, which presented overfitting, all the other algorithms had validation/training loss differences lower, meaning slightly less overfitting than InceptionV3. Considering accuracy and validation/training loss difference, DenseNet121 appears to be a better algorithm for the datasets in the study.

EfficientNetB0 in all datasets showed that increasing batch size, the accuracy decreased slightly; the same happened to ResNet50 and DenseNet121. This result makes us believe that increasing batch sizes are not beneficial for these algorithms. MobileNetV2, contrary to previous ones, seems to increase accuracy slightly, which seems that increasing batch size might benefit this algorithm. Given its generated smaller model size, this algorithm might be worth exploring in situations where limited resources are available.

As cloud platforms allow developers, scientists, and researchers to use free resources for machine learning, despite its limitations, these solutions prove worth the time exploring, as it would be in-viable carrying out this study on typical household computers. Paperspace Gradient proved us to be a powerful and versatile platform for carrying out these tests.

5.1 Future Work

Our tests showed a slight tendency for more accurate models trained on datasets with more extensive train sets. Another tendency was that bigger batch sizes reduce both train time and accuracy. Similar tests could be done on different datasets and adaptations from the same dataset to validate and improve this study, trying to test bigger batch sizes and more extensive datasets to see if these findings still hold valid.

Acknowledgements "National Funds fund this work through the FCT—Foundation for Science and Technology, IP, within the scope of the project Ref UIDB/05583/2020. Furthermore, we would like to thank the Research Centre in Digital Services (CISeD), the Polytechnic of Viseu, for their support."

References

1. Azure, M.: Azure machine learning—ml as a service—microsoft azure, June 2021. URL: https://azure.microsoft.com/en-us/services/machine-learning/. Retrieved: 2021–06–29
2. Carneiro, T., Medeiros Da No´Brega, R. V., Nepomuceno, T., Bian, G.-B., De Albuquerque, V. H. C., Filho, P.P.R.: Performance analysis of google colaboratory as a tool for accelerating deep learning applications. IEEE Access, **6**, 61677–61685 (2018)
3. Cloud, G.: Ai infrastructure ml and dl model training—google cloud, June 2021. URL: https://cloud.google.com/ai-infrastructure. Retrieved: 2021–06–29
4. Czimmermann, T., Ciuti, G., Milazzo, M., Chiurazzi, M., Roccella, S., Oddo, C.M., Dario, P.: Visual-based defect detection and classification approaches for industrial applications—a survey. Sensors **20**(5) (2020)
5. Farber, R.: Chapter 2—cuda for machine learning and optimization. In: CUDA Application Design and Development (pages 33–61). Morgan Kaufmann, Boston (2011)
6. Google.: Colaboratory: Frequently asked questions, June 2021. URL: https://research.google.com/colaboratory/faq.html. Retrieved: 2021–06–26
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)
8. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks (2018)
9. Hull, I.: Machine Learning for Economics and Finance in TensorFlow 2: Deep Learning Models for Research and Industry, 1st edn. Apress (2021)
10. Martins, M., Mota, D., Morgado, F., Wanzeller, C., Martins, P., Abbasi, M.: Imageai: comparison study on different custom image recognition algorithms. In: Rocha, A., Adeli, H., Dzemyda, G., Moreira, F., Ramalho Correia, A.M. (eds.), Trends and Applications in Information Systems and Technologies (pp. 602–610). Springer International Publishing, Cham (2021)
11. Momeny, M., Latif, A.M., Sarram, M.A., Sheikhpour, R., Zhang, Y.D.: A noise robust convolutional neural network for image classification. Results Eng. **10**, 100225 (2021)
12. Olafenwa, M.: Idenprof, June 2021. URL: https://github.com/OlafenwaMoses/IdenProf. Retrieved: 2021–06–26
13. Olafenwa, M.: Imageai, June 2021. URL: https://github.com/OlafenwaMoses/ImageAI. Retrieved: 2021–06–26
14. Paperspace.: Gradient—effortless deep learning at scale, June 2021. URL: https://docs.paperspace.com/gradient/ Retrieved: 2021–06–26
15. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: Mobilenetv2: Inverted residuals and linear bottlenecks (2019)
16. Services, A.W.: Amazon sagemaker—machine learning—amazon web services, June 2021. URL: https://aws.amazon.com/sagemaker/. Retrieved: 2021–06–29
17. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision (2015)
18. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks (2020)
19. Tran, H.-D., Bak, S., Xiang, W., Johnson, T.T.: Verification of deep convolutional neural networks using imagestars. In: Lahiri, S.K., Wang, C. (eds.) Computer Aided Verification, pp. 18–42. Springer International Publishing, Cham (2020)

20. Zhang, Y.B.B.X.M.: Genetic Programming for Image Classification: An Automated Approach to Feature Learning, Volume 24 of Adaptation, Learning, and Optimization, 1 edn. Springer International Publishing (2021)
21. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition (2018)