Data Augmented Arthropod Classification with Transfer Learning

FEILLET Eva
CentraleSupélec
eva.feillet@student.ecp.fr

DBIRA Moez
CentraleSupélec
mohamed-moez.dbira@student.ecp.fr

Abstract

Entomology was long a domain reserved to biologists only. Today, with a fast-growing knowledge in entomology and new technological opportunities, artificial intelligence is a powerful tool to help researchers both in monitoring biodiversity and in developing industry-oriented applications such as pest control. Insects are characterized by both a high diversity in their morphological aspects and subtle similarities across species. In this project we focus on a particular set of small animals: 7 groups of species belonging to the arthropods, a taxonomic order gathering animals with an exoskeleton such as ants, beetles or bees. We compare the performances of several types of networks with various training strategies (from scratch, transfer learning, fine-tuning). The high diversity of insect species and the variety of angles they were pictured in make the classification a hard task to achieve in the absence of a large dataset. That is why we also explored data augmentation techniques from classic geometric transformations to GAN-based image generation.

1. Introduction

1.1. Arthropods

In this project we focus on a particular taxonomic order: arthropods. An arthropod is an invertebrate animal characterized by an exoskeleton, a segmented body and jointed appendages. Arthropods correspond to a lineage of life that developed exoskeletons instead of bone-like structures for structural support. They colonized land 100 million years before vertebrates did and represent about 80% of the known insect species[1]. Some well-known insects such as ants, dragonflies and bees belong to the arthropod order. However, this order should not be mistaken with the animals we commonly call insects, it is a much wider taxonomic scale (Figure1). As a matter of fact, scorpions, and crabs belong to arthropods too, although they are not insects (Figure 2 – "other" category).

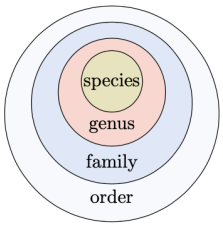

Figure 1: The taxonomic scales relevant to arthropod classification

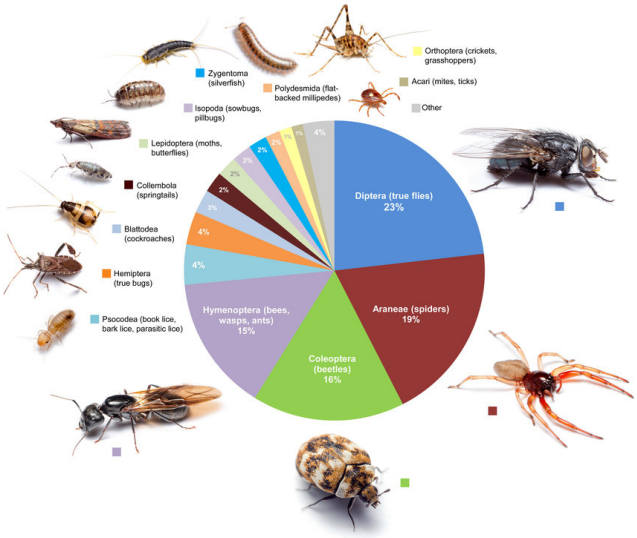

Figure 2: Proportional diversity of arthropod orders across all rooms. Average morphospecies composition calculated across all room types [1].

Our classification problem uses an image dataset containing 7 classes, each one corresponding to a family of arthropods: Araneae (spiders), Coleoptera (beetles), Diptera (mosquitos), Hemiptera (shield bugs), Hymenoptera (wasps), Lepidoptera (butterflies) and Odonata (dragonflies). These families are not an exhaustive

1 https://biologydictionary.net/arthropod/

partition of the arthropod order but account for most of its diversity (Figure 2). Our dataset gathers pictures of animals living in similar environments and therefore sharing common visual features, both from a morphological standpoint (size, colors) and from a background standpoint (leaves, wood etc.) See Figure 3 for a few examples from our dataset.

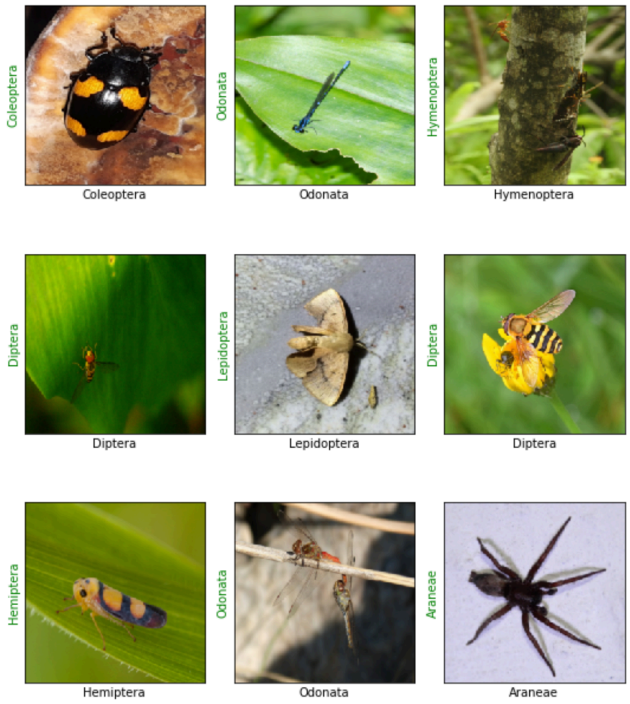

Figure 3: Sample images from the Arthropods dataset.

1.2.Taxonomic studies and their applications

According to the Convention on Biological Diversity[2], "taxonomy is the science of naming, describing and classifying organisms and includes all plants, animals and microorganisms of the world. Using morphological, behavioral, genetic and biochemical observations, taxonomists identify, describe and arrange species into classifications, including those that are new to science." Taxonomic studies are a key tool for biodiversity monitoring and wildlife preservation. As arthropods are a key indicator of an ecosystem's function, their identification also enables researchers to assess the quality and the evolution of a habitat, may it be to measure a current human-made impact or to help paleo historians reconstruct the story of Earth and Life. Deep learning has recently emerged as a helpful way to detect and classify these small animals, a tedious task requiring rare expertise.

Taxonomic studies also have their importance for the agriculture sector. On the one hand crop insect detection is a paramount but tedious task for farmers in order to protect their crops from pest attacks. On the other hand, arthropods such as bees are essential actors of pollination and thus applications to monitor their population are also developed. In a nutshell, applications in agriculture and biodiversity have lately fostered a growing research effort in the field of computer vision in order to develop models for embedded insect detection and classification.

This project aims at building a classifier able to recognize animals among 7 classes of arthropods. We have trained and compared several types of networks and tried to improve the model's performances through classic data augmentation and GAN based data augmentation.

2. Background

2.1. Insect classification

In the last decades, numerous studies focusing on insect classification through classic computer vision methods have been published. They rely on feature extraction through finely designed filters and handcrafted preprocessing to obtain a set of features which is then fed to a machine learning algorithm like Support Vector Machines (SVM) or k-means [2, 3]. Such studies often use pictures which were carefully taken by qualified staff in a controlled environment, with chosen lightning, insect pose and camera angle. The models they propose are thus limited to pictures sharing similar constraints and are not used outside laboratories [4]. This is why the next generation of studies, benefitting from the development of deep learning algorithms and computing power, proposed to tackle a broader type of problems. Living insects pictured in their natural environment with various conditions of lightning, possible blur, truncation or occlusion have made their way to the datasets, sometimes through a community effort. Studies with an agronomic interest - monitoring crop pest - use pictures taken on the fields [5]. However standard benchmarking datasets are not systematically shared among studies, making them hard to compare. A benchmarking dataset for insect pest recognition was proposed at CVPR in 2019 by. Wu et al.[6]. We contacted them to have access to their dataset.

2.2.Light-weight convolutional models

To be able to deploy their models in applications which can be used on the field, researchers must not only focus on the accuracy of their model but also on its weight and computing time. This is why recent studies have chosen to adopt light-weight architectures such as MobileNet to develop embedded insect recognition applications [7].

[2] https://www.cbd.int/convention/

MobileNets is a class of deep learning models proposed in 2017 for mobile vision applications [8]. Thanks to its streamlined architecture that uses depth-wise separable convolutions, it provides an efficient way of computing convolutions. Other types of architectures such as ShuffleNet or SqueezeNet have also been considered [7].

2.3.Generative Adversarial Networks (GANs)

GANs were originally introduced in a publication by Goodfellow et al. in 2014 [9]. Until that date, the most successful deep learning models had involved discriminative models. These models are especially used in supervised learning and statistical classification. Whereas generative models have had less of an impact because they are often more difficult to evaluate and less readily applicable to business problems than discriminative models[10]. So, the adversarial nets framework proposed in the publication boosted the rise of generative modeling in deep learning. The architecture of this framework consists of introducing an adversary network, also called discriminator, opposing the generator network. More precisely, the generator is a generative model in charge of producing fake data and the adversary is a discriminative model that learns to determine whether a sample is fake or real [9]. During training, the generator is constantly trying to outsmart the discriminator by generating better and better fakes, while the discriminator is working to become a better fake detector.

GANs can be used to deal with different problems such as increasing resolution, generating new images, image inpainting or speech synthesis [11]. In our project, we are using DCGAN which can be seen as an extension of GAN except that it uses convolutional layers in the discriminator and convolutional-transpose layers in the generator. This framework was first presented by Radford et al.[12] and is used to generate synthetic images from a natural image dataset.

2.4.GAN as a data augmentation technique

In order to build a robust deep neural network classifier model, sufficient number and quality of image data is necessary [13,14]. Collection of large amounts of image data is time-consuming and sometimes requires expensive high-quality expertise for labeling, which is the case with arthropods classification. In order to reduce the effort and time needed, GAN-based augmentation is used in order to synthesize artificial but realistic images. This technique can improve the performance of CNN models by oversampling minority classes and providing larger training data sets. It can also be useful to avoid using original data for privacy reasons [13,14].

3. Proposal and Experimental design

3.1 Dataset

The dataset we used in this project is available on Kaggle[3] website. Images are collected from various sources, including inaturalist.org[4] and similar tools to build crowdsourced datasets. Every image is originally published under a Creative Commons license. The original dataset has a size of 12.1GB and is relatively balanced across its 7 classes (Table 1), with a total amount of 15376 images of high resolution (Megapixels), each containing at least one arthropod. All images do not present the same level of difficulty for a classifier: the size and pose of the arthropod, the lightning, possible truncation, blur or occlusion are the major sources of variation (Figure 4), let alone the specie. Although most images are centered on one single arthropod, some of them picture a group of arthropods, for example a couple of dragonflies, an ant colony or a beehive. This dataset was initially proposed for a detection task, hence the high resolution, but we will focus on a classification task and benefit from the high resolution of pictures to feed GANs.

Class	# Entities	%Entities
Araneae	2418	15,7%
Coleoptera	2110	13,7%
Diptera	2030	13,2%
Hemiptera	2387	15,5%
Hymenoptera	2048	13,3%
Lepidoptera	2106	13,7%
Odonata	2277	14,8%

Table 1: Entity repartition in the arthropod dataset

The dataset was split with the following proportions: training set: 74% i.e., 11393 images, validation set: 17%, i.e., 2562 images, test set: 9% i.e., 1421 images. The training set has a slight imbalance which we will later be addressed through data augmentation.

3.2.Classification task

We use the accuracy to assess the performance of the classifier and we have made sure to have the same number

[3] https://www.kaggle.com/mistag/arthropod-taxonomy-orders-object-detection-dataset

[4] https://www.inaturalist.org/

of examples from each class in the test set. We also use the full metrics table featuring precision, recall and f1-score for each class to compare performances from one class to another.

Figure 4: Examples (left) Butterfly which colors are similar to the background color, (right) 2 ants, one of them is blurred and truncated in the background.

Concerning classification, we have made 3 major rounds of experiments. The first round gave us baseline performances without data augmentation. Then we measured the improvement of our classifiers made possible by augmenting the dataset with classic transformations. The third round featured new training images created by GANs.

To train the classifiers we used Pytorch version 1.7.1 and hyperparameter tuning framework ray.tune version 1.2.0 on an AWS SageMaker instance with 4 GPUs and 16 GB of RAM.

Each time we train and tune the hyperparameters of several types of models on the same training and validation sets. First, a basic 6-layer CNN: this small, rudimentary CNN is used as baseline and as a means of testing our code.

Then, MobileNet v2 (pretrained on ImageNet): a more elaborated network architecture as available in Pytorch. We chose MobileNet v2 because it offers a good compromise between expressivity and computational efficiency. We use it first as a feature extractor by simply adapting the last layer of the pretrained network from 1000 outputs as in ImageNet to 7 outputs as in our use case. All parameters are frozen except those of the last fully connected layer. Next step is fine tuning : all parameters are unfrozen and learnt through backpropagation. Since ImageNet contains a few classes intersecting with our arthropod dataset, such as tarantulas, beetles or bees[5], we expect to be able to reuse not only low-level features but also some higher-level features for our classification.

Resnet50 (pretrained on IP102 by Wu et al.) : we make the same experiments with this bigger network which was pretrained on a dataset containing 102 classes of pest insects [6]. First, we computed the mean and variance of the IP102 dataset, graciously shared by Mr. Wu. Then we transformed our own images according to these values when feeding them to the pretrained network for further learning.

3.3.Data augmentation : classic methods

Deep learning-based models have many parameters to learn during training. This implies the need for a large training data set to help the model catch the phenomenon. CNNs are powerful when it comes to image processing and although convolution is a translation invariant operation, it is sensitive to other geometrical transforms such as zooming or rotation. Thus, in order to help build a more robust deep learning models, we can apply these transformations on the training dataset. This helps enlarge the data set and we expect to also help avoid overfitting. In fact, in real world application, the image could be taken in different orientations, meteorological conditions, zooming etc. Which is the case of our project where we imagined the use of the arthropod classification by farmers or researchers in crop fields. So, we thought about testing this data augmentation classic technique on our dataset and compare the models' performances with and without the geometrical transforms.

3.4.GAN-based data augmentation

After we applied classical methods which helped multiply the dataset volume by a factor of 7, we thought that using the same image a lot of times although with different transformations, had its limits. That is why we looked for another way to obtain new images. We chose to train a DCGAN in order to generate realistic synthetic images of the different classes of arthropods. This helped us have a larger and more balanced amount of image data for training. Finally, we compared the performance of the above neural networks on the testing set with and without the data augmentation. This was done only on 3 classes for practical reasons, the GANs taking about 24 hours to train on a Google VM.

3.5.Experiment settings for GANs

In order to train the DCGAN image generator we used Pytorch version 1.7.1 on a Google Cloud Deep Learning VM 4vCPUs and 16 GB of RAM. We replicated the same architecture as described in the DCGAN paper [12] and we re-trained the network with data coming from each class in order to synthesize images for each type of arthropod.

4. Experiments

4.1. Hyperparameters

[5] See class list at https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a

Optimization consisted in a Stochastic Gradient Descent with a momentum of 0.9. Each time we tuned the initial value of the learning rate, chosen from a distribution from 1e-4 to 1e-1, and the batch size, chosen from the set {4, 8, 16}. We also experimented with our small CNN's architecture : in addition to learning rate and batch size, we tuned the number of filters for the first convolutional layer and the number of units for the first fully connected layer. Rather than a classic grid search, we randomly picked a given number of hyperparameter combinations for each model type and learning type. Training models without a GPU on costly platforms limited us in the number of combinations we could try[6].

Due to these limitations, we usually trained the models for around 6 to 12 epochs. For convenience we implemented checkpointing to enable "warm start" training i.e., resuming training from a relatively good previous state of the network. Looking at the small number of training epochs, early stopping was used more to prevent a waste of computing resources than to prevent overfitting.

In the case of MobileNet and Resnet we have used learning rate decay to improve accuracy. We tested a cyclic learning rate decay with a triangular profile taking 700 to 2000 batches for up and down between a learning rate of 5e-3 and 5e-5. The results, although promising, were also quite bumpy. So, without data augmentation, most of hyperparameter tuning was performed using a multiplicative profile, because it produced more reliable performances. Learning rate was multiplied by 0.9991 at each batch, meaning for around 750 batches and 10 epochs the learning rate would decrease by a total factor of 1e-3 through training. In the second round of experiments, our dataset was more than six times bigger and cyclic decay proved to be a more efficient way of reaching better minima of the cross-entropy loss function.

4.2. Baseline classification

No data augmentation : baseline performances. Our models should have an accuracy above 14.29%, corresponding to a random guess out of 7 classes. See Table 2 for results.

Chosen CNN here has 16 filters in the first convolutional layer and 512 units in the first fully connected layer. A very simple architecture but already 1.6 million parameters to learn, more than half the number of parameters of the Mobilenet network (27 layers), but its modeling potential is really lower. In fact, Mobilenet has a relatively small number of parameters to learn because of its particular computing structure.

We had the highest expectations for the pretrained Resnet50 model because the original focus of this model is similar to our interest dataset. However, this model is also the biggest so it could have needed more data than the others to reach its expressivity potential. It was also by far the longest to train. The accuracy of the pretrained Resnet50 network was disappointing. We expected it to be at least as good as Mobilenet's accuracy because the original dataset focuses on pest insects, so the domain is really close to our arthropod dataset. However, by taking a closer look at the IP102 pictures, we understood that images looked quite different from ours. Some have been cropped so that the insect is centered and occupies most of the picture, whereas background is a major information in our dataset and many pictures feature flowers, leaves or wood. Images also differ from ours as so far as they may contain multiple captions, zooms and legends in Chinese, or may show a field affected by pest but without any visible insect on the picture.

Figure 5: Sample images from the IP102 dataset by Wu et al.

MobileNet gets the best accuracy by far. This can be explained by the diversity of low and higher-level features it has already learnt on ImageNet, as well as by the fact that 36 of the original 1000 classe intersect with our arthropod dataset.

4.3.Classic data augmentation

Performing simple geometric transforms on the available data may seem inefficient, but it has proven to be very helpful in order to increase the CNNs performance. In fact, although CNNs are translation invariant, they are sensitive to zooming and rotations for example. That is why we performed multiple image augmentation techniques provided by torchvision transforms[7]. The transformations

[6] Free GPU on Google Colab is limited or not always available. We also tried to access GPUs through AWS and Google Cloud but either there was no available resource, or our account type did not match the requirements.

[7] https://androidkt.com/pytorch-image-augmentation-using-transforms/

we used are image rotation where we change the orientation of images with a random angle value. We also applied a random crop that consists of zooming in or out images and then choose a segment of the image. In addition, we flipped randomly vertically and horizontally the images. We also applied random color transforms by changing brightness, contrast and saturation of an image. We have added random Gaussian noise too. See Figure 7 for some examples.

Model	Initial lr	Batch size	LR decay	Training	Parameters	Test Acc
basic CNN	0.01104	4	mult.	from scratch	1.682.183	29.3
MobileNet v2	0.00856	8	cyclic	transfer learning	8.967	63.5
MobileNet v2	0.01208	4	cyclic	fine tuning	2.232.839	65.8
Resnet50 (Wu et al.)	0.00935	16	mult.	transfer learning	14.343	55.1
Resnet50 (Wu et al.)	0.01502	8	mult.	fine tuning	23.522.375	58.3

Table 2: classification results without data augmentation

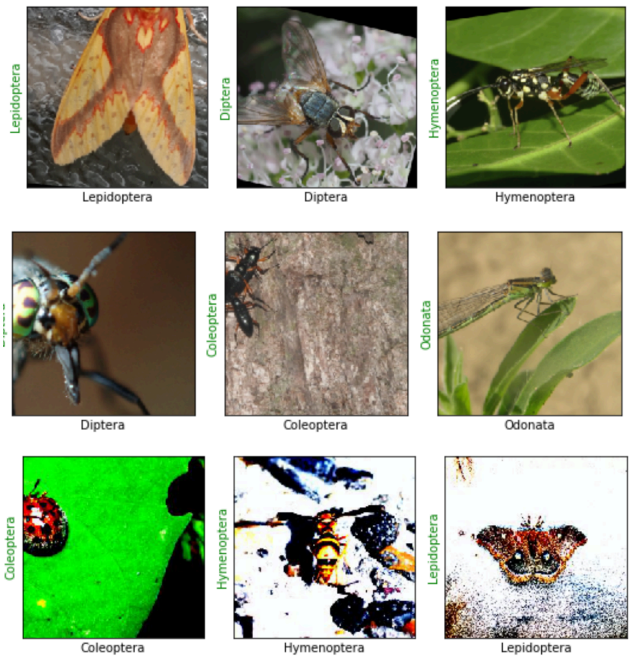

Figure 7: Example of augmented images obtained by making large crops and rotations (first row), small crops (second row), shifting colors and adding noise (third row).

By applying these transformations, we obtain a balanced dataset with a total of around 72,000 training images and 15,000 validation images. Test set remains unchanged and already had the exact same number of examples from each class. We are trying to get a more robust model that should be able to grasp visual characteristics with a wider range of scales, orientation and lightning. Cropping could also help getting better predictions when the insect is occluded.

Classification results Since the dataset was bigger, training was longer – about one day for 8 epochs on Resnet50 with our set up. Test accuracy was dramatically improved in the case of our small CNN : 58.4% could be achieved. However, the network was prone to overfitting : after 14 epochs, training accuracy reached 99% but validation accuracy, after scores above 62%, was dropping. So, we kept limiting the trainings to 6 to 10 epochs to avoid overfitting. MobileNet and Resnet50 could be improved too, although less spectacularly. Fine tuning was much too long so we focused on tuning the last layer only. We regret we could not try a lot of hyperparameter combinations due to practical limitations, but we were able to obtain better scores, respectively 66.7% for MobileNet and 59.2% for ResNet50 in transfer learning. We think we could have obtained much better scores provided we had more computing resources.

4.4. Data augmentation with DCGAN

DCGAN design and tuning for image generation Radford et al. [12] gave detailed guidelines for a stable architecture of DCGAN. The generator generates a 64x64 pixel image from a latent space vector that is drawn from a standard uniform distribution. It is comprised of a sequence of 4 strided transpose convolutional layers stacked with batch norm layers and ReLU activations. Then another transpose convolutional layer as an output. The discriminator is a binary classification network. It has the reverse architecture of the generator except for the activation we use a LeakyReLU as mentioned by the DCGAN paper.

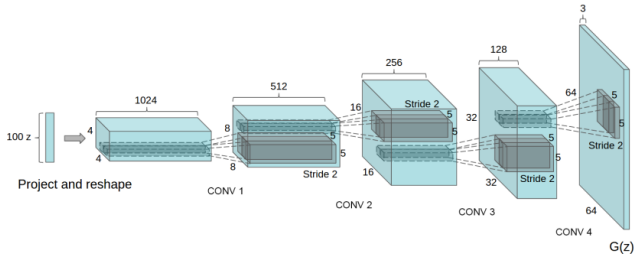

Figure 8 [12] : Example of DCGAN generator used for LSUN scene modeling

The authors also gave tips about model weights initialization that should be initialized from a zero-centred Normal distribution. The optimizer that should be used is the Adam optimizer with tuned hyperparameters. And finally, the recommended learning rate is 0.0002[12].

Result interpretation : DCGAN images. It is always difficult to evaluate an unsupervised model. By visualizing real and generated images side by side, we can perceive that the DCGAN performed better on some classes than others. For example, on the class Araneae (figure 9-b), we can distinguish the shape of a spiders and its legs. On the other hand, if we see the class of Diptera (figure 9-a) we can see that the DCGAN figures a round dark shape in the middle of the image but the wings are often melted with the background. These results can be explained by the size of the training data set (only ~2k per class). We tried to improve the model by increasing the number of epochs, but we saw that after 250 epochs the loss functions of the generator and the discriminator stagnated. Although the results are not satisfying, we tried to train the classifiers with the generated images.

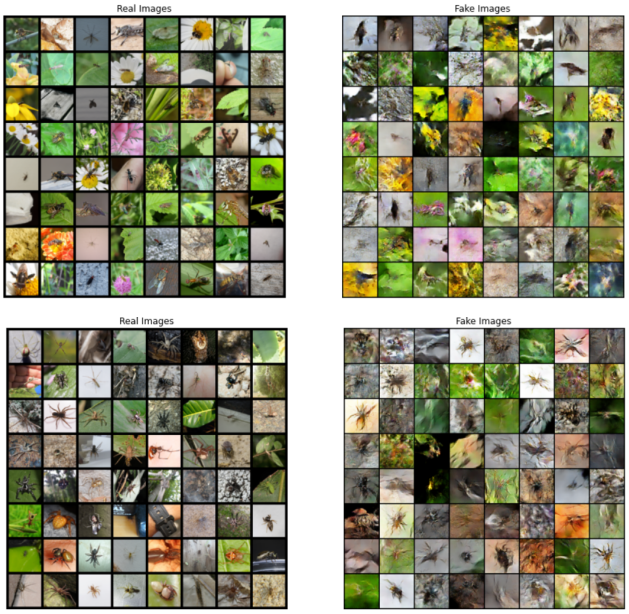

Figure 9 : Comparison between the real images and the DCGAN-generated images, (a) of the class Diptera (b) of the class Araneae. See more examples on our GitHub repository[8].

Classification results. To evaluate the data augmentation with DCGAN and for time limit constraints, we calculated the performance of different classification models on only 3 classes (Araneae, Coleoptera, Diptera). The models were first trained with the original dataset and then we added the DCGAN-generated images to the training set. We compared the performances on the same testing set. As shown in the next table, we increased the size of the training set by 21% and we tried to balance the size of samples from every class.

Training set	Original dataset		Augmented dataset	
Class	# Entities	%Entities	# Entities	% Entities
Araneae	1930	36,8%	2122	33,2%
Coleoptera	1688	32,2%	2136	33,4%
Diptera	1624	31%	2136	33,4%

Table 3: Training data set composition before (yellow)and after (blue) data augmentation with DCGAN

Model	lr	Training	Test acc (basic)	Test acc (DCGAN)
basic CNN	0.01104	from scratch	43,1	43,92
MobileNet v2	0.00856	transfer learning	78,0	78,6
MobileNet v2	0.01208	fine tuning	79,28	80,14
Resnet50 (Wu)	0.00935	transfer learning	63,55	64,13
Resnet50 (Wu)	0.01502	fine tuning	64,3	65,27

Table 4: classification results with and without DCGAN augmented training dataset. Note that the baseline has changed here : with only 3 classes, we expect at least 33.3% of correct predictions.

Although being very promising, it is true that the DCGAN technique improved the accuracy of our trained models, but the improvement was not as big as expected. In fact, it did not exceed the threshold of 1%. This limited improvement can be explained by the size of our training set that was very small. Also, the diversity of the images inside the dataset can be a factor limiting the performance of DCGAN. Indeed, apart from having a few images, these images are very different from each other: having different backgrounds, different perspectives… Actually, our images are not very realistic and quite small (64 by 64 pixels, much less than the 224 by 224 resized high-resolution images) so it is already good that they did not decrease the accuracy of the models. Finally, we can conclude that the idea had a positive effect on the performance of every model we

[8] https://github.com/EvaJF/arthropod_clf/

tested, but on the scale of our dataset it did not provide the expected improvements.

5. Conclusions and future work

As a conclusion, we recognize our numerical performances could be improved with more data augmentation and more computing resources. However, we could observe trends similar to those described in the papers we read. First, we have compared the performances of various models and explored the possibilities of transfer learning. MobileNet v2 offers the best accuracy and a relatively small training time, comforting its choice for embedded applications. Fine-tuning is helpful to improve accuracy but should be done with enough data to avoid overfitting. We have demonstrated the need for data augmentation by obtaining an accuracy improvement using classic data augmentation methods. Last but not least, we have tried to further improve our models using images generated by GANs. We think the results are encouraging and would have been much better provided we had more computing resources to train the GANs, enabling us to obtain more realistic insect images.

We would like to continue this work by improving both the GANs and the classifiers. It could be interesting to visualize their feature maps. We would also like to work on the detection of morphological characteristics to bridge the gap between symbolic and neural approaches : associating to an input image a list of attributes such as number of legs, body segments, antennas, roundness, etc. which could be used to feed either a rule engine or another supervised classifier.

References

[1] Bertone, Matthew & Leong, Misha & Bayless, Keith & Malow, Tara & Dunn, Robert & Trautwein, Michelle. (2016). Arthropods of the great indoors: Characterizing diversity inside urban and suburban homes. PeerJ. 4. e1582. 10.7717/peerj.1582.

[2] Fina Faithpraise, Philip Birch, Rupert Young, J Obu, Bassey Faithpraise, and Chris Chatwin. Automatic plant pest detection and recognition using k-means clustering algorithm and correspondence filters. *International Journal of Advanced Biotechnology and Research*, 4(2):189–199, 2013.

[3] R Uma Rani and P Amsini. Pest identification in leaf images using SVM classifier. *International Journal of Compu- tational Intelligence and Informatics*, 6(1):30–41, 2016.

[4] Maxime Martineau, Donatello Conte, Romain Raveaux, Ingrid Arnault, Damien Munier, et al. A survey on image-based insect classification. Pattern Recognition, Elsevier, 2017, 65, pp.273 - 284.10.1016/j.patcog.2016.12.020 . hal-01441203

[5] Thenmozhi Kasinathan, Dakshayani Singaraju, Srinivasulu Reddy Uyyala, Insect classification and detection in field crops using modern machine learning techniques, Information Processing in Agriculture, 2020 ,ISSN 2214-3173, https://doi.org/10.1016/j.inpa.2020.09.006. (https://www.sciencedirect.com/science/article/pii/S2214317320302067)

[6] Wu, Xiaoping et al. “IP102: A Large-Scale Benchmark Dataset for Insect Pest Recognition.” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019): 8779-8788.

[7] Visalli F., Bonacci T., Borghese N.A. (2021) Insects Image Classification Through Deep Convolutional Neural Networks. In: Esposito A., Faundez-Zanuy M., Morabito F., Pasero E. (eds) Progresses in Artificial Intelligence and Neural Systems. Smart Innovation, Systems and Technologies, vol 184. Springer, Singapore. https://doi.org/10.1007/978-981-15-5093-5_21

[8] Howard, Andrew & Zhu, Menglong & Chen, Bo & Kalenichenko, Dmitry & Wang, Weijun & Weyand, Tobias & Andreetto, Marco & Adam, Hartwig. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

[9] Goodfellow, Ian and Pouget-Abadie, Jean and Mirza, Mehdi and Xu, Bing and Warde-Farley, David and Ozair, Sherjil and Courville, Aaron and Bengio, Yoshua. 2014, Advances in Neural Information Processing Systems, vol. 27. *Curran Associates, Inc., Generative Adversarial Nets*

[10] David Foster. (2019). Generative Deep Learning. ed. *O’Reilly Media*, chapter 1

[11] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018

[12] Alec Radford & Luke Met, Soumith Chintala. 2016. 'Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks' arXiv:1511.06434v2

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

[14] Lu, Chen-Yi & Rustia, Dan Jeric & Lin, Ta-Te. (2019). Generative Adversarial Network Based Image Augmentation for Insect Pest Classification Enhancement. 52. 1-5. 10.1016/j.ifacol.2019.12.406.