Deep-Pollination: A series of ML challenges for multi-class insects classification

Ihsan Ullah, Mohammed Lansari, Isabelle Guyon

Université Paris-Saclay
ihsan.ullah@universite-paris-saclay.fr
https://github.com/ihsaan-ullah/deep-pollination

Abstract. Insects are very important for biodiversity, food chains, and pollination. It is of great importance to recognize insects, their habitats and to secure their natural environment. Machine learning, especially deep learning techniques can be used to recognize and classify various insects. We introduce **Deep-Pollination**, a series of three machine learning challenges organized on Codalab for insects classification. A preprocessed version of the insects dataset is used for these challenges which consists of five classes and more than 200,000 images.

Keywords: insects classification, deep learning, challenge organization

1 Introduction

Insects, especially pollinating insects have a great role in biodiversity, the nutrient cycle, and the functioning of ecosystems [10]. One of the main threats to our ecosystem is the continuous decline in the population and habitats of various pollinating insects due to the uncontrolled use of pesticides, insecticides, and deforestation.
Monitoring the population of pollinating insects has traditionally been done manually. Machine learning and in particular, deep learning techniques [5] offer new possibilities to do this task with more speed, accuracy, and efficiency. The existing state-of-the-art for insect classification uses traditional feature extraction with machine learning classifiers for small image data gathered in lab settings [4]. Researchers have attempted to tackle this problem with deep learning, in the form of a challenge organized on the **RAMP** platform [3] to classify images of insects from the SPIPOLL crowdsourcing project of the Paris Museum of Natural History [8], with the scientific goal of quantitatively studying pollinating insects in France.
We present this problem as a supervised learning multi-class image classification problem in the form of a series of three machine learning challenges on the Codalab platform, intending to solve the problem using the preprocessed extracted features from images and using raw images as the data. A novel boundary for the participants is to gain insight into the model and the data likewise by using Explainable AI (XAI).

2 Material and Methods

The data provided for the challenges consists of five classes: bee, wasp, butterfly, other insects, and others(not insects). The dataset is a combination of two datasets: insects dataset from Kaggle and insects dataset from SPIPOLL science project. The dataset is unbalanced but the validation and the test sets of all challenges are balanced. We use **accuracy** metric to evaluate the performance. More details about the source of datasets, statistics, and examples are in Appendix A.

(a) Bee images with blue background (b) Redundant wasp images Village

Fig. 1: Bee and wasp images to be removed in data cleaning step

Data cleaning is one of the important preprocessing steps applied for data preparation. Some of the images of bees in the dataset have a blue background which introduces bias in the data and can ultimately lead to a classification of bees based on the background and not the actual insect in the image. Some images of wasps in the dataset are taken from video and there are many images that are redundant i.e. captured from the same scene of a video. We have removed all these images to eliminate the bias in the data for a fair evaluation of the participants. Figure 1 shows the sample bees and wasps images which are removed from the dataset. More details about data cleaning and bias in data are given in Appendix B.

Images in both datasets are not in uniform size, resolution, and orientation. Usually, the object of interest is in the middle of the image, therefore the images are first cropped into square images and then resized into 128x128 size using open-cv. The size of the image is chosen after a careful analysis of different resolutions and to keep a good trade-off between image size and dataset size. More details about image size analysis are in Appendix C.

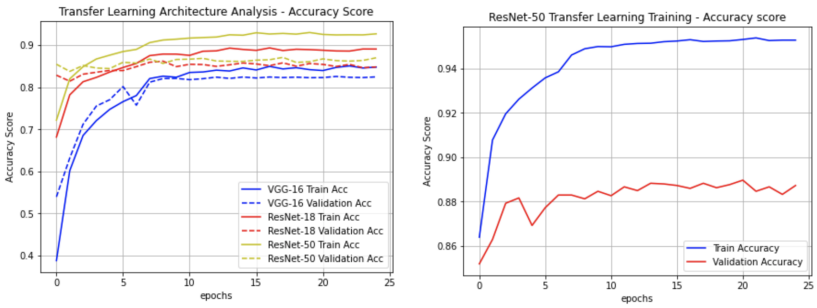
3 Codalab challenges

We have prepared three Codalab insect classification challenges with tabular data (extracted features from images using KAZE descriptor [2]) and preprocessed images data (128x128 resized images). We also provide a module of explainability in one of the challenges i.e. the participants have to provide a mechanism to explain why the classifier has taken a decision(See Appendix D). We also provide code for data reading, visualization, baseline methods, and instructions for results

submission. All details and future updates about this project can be found in the dedicated Github Repository

4 Results

We use some Scikit Learn classification models: Gaussian NB, Ridge Classifier, SVM, MLP Classifier, and Random Forest Classifier, for tabular data(features extracted from images). We use AutoDL self-service [9] to get a baseline performance for both tabular and images data.



(a) multiple models' performance on subset of data (b) best performance on full dataset

Fig. 2: Transfer learning: performance results for raw images dataset

We go further and use transfer learning for insects classification. We use VGG and ResNet architectures (pre-trained on Imagenet [6]) from Keras [1] to get baseline performances on a subset of the dataset(10,000 images). We get the best performance with ResNet-50 architecture. We train it with the full dataset and the best validation accuracy we get is **89%**. Figure 2 shows the performance with transfer learning. More details about AutoDL baseline performance, Scikit Learn performance, and transfer learning experiments are in Appendix E.

5 Discussion and Conclusion

To incentivize the problem-solving process, we organize the problem in the form of challenges to allow the participants to contribute a benchmark solution for such a problem. We include a module of explainability in the third challenge to motivate the participants to provide not only the best-performing solutions but also explain the best performance. Future work could be to investigate more the data preprocessing(image cropping, image resizing, and anti-aliasing) and to refine the explainability module.

6 Acknowledgements

We gratefully acknowledge Grégoire Loïs, Colin Fontaine, and Jean-Francois Julien from *National Museum of Natural History Paris* and *SPIPOLL Science project* for donating the insects' dataset for this research project. We are thankful to Team Ecologists: Thibaut Soulard, Vaibhav Arora, Xavier Bou, and Eric Santiago for making this project possible. We received useful input from Phan Anh Vu. We acknowledge the support of Zhengying Liu, Michael Vavvaro, and Adrien Pavao for their help in setting up the Google Cloud and Codalab platform. We also would like to thank the hundreds of volunteers involved in the SPIPOLL citizen science program who pictured and identified insects. This work was partially funded by Labex Digicosme project ANR11LABEX0045DIGICOSME operated by ANR as part of the program Investissement d'Avenir Idex Paris Saclay (ANR11IDEX000302).

References

[1] François Chollet et al. *Keras*. https://keras.io. 2015.

[2] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew Davison. "KAZE Features". In: Oct. 2012. ISBN: 978-3-642-33782-6. DOI: 10.1007/978-3-642-33783-3_16.

[3] Balázs Kégl et al. *The RAMP framework: from reproducibility to transparency in the design and optimization of scientific workflows*. International Conference On Machine Learning. Poster. July 2018. URL: https://hal.archives-ouvertes.fr/hal-02072341.

[4] Maxime Martineau et al. "A survey on image-based insect classification". In: *Pattern Recognition* 65 (May 2017), pp. 273–284. DOI: 10.1016/j.patcog.2016.12.020.

[5] Ivan F. Rodriguez et al. "Recognition of Pollen-Bearing Bees from Video Using Convolutional Neural Network". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 314–322. DOI: 10.1109/WACV.2018.00041.

[6] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[7] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[8] Hortense Serret et al. "Data quality and participant engagement in citizen science: comparing two approaches for monitoring pollinators in France and South Korea". In: *Citizen Science: Theory and Practice* 4.1 (2019), p. 22.

[9] M. Vaccaro, A. Pavao, and I. Guyon. *AutoDL Challenge, Codalab*. URL: https://competitions.codalab.org/competitions/27082.

[10] Wolfgang Weisser and Evan Siemann. "The Various Effects of Insects on Ecosystem Functioning". In: *Ecol. Stud.* 173 (Jan. 2004). DOI: 10.1007/978-3-540-74004-9_1.

A Dataset

The dataset used in this project is a combination of two datasets:

1. Kaggle insects dataset https://www.kaggle.com/jerzydziewierz/bee-vs-wasp
2. Museum insects dataset https://www.spipoll.org/

Each dataset has a missing class, i.e. 'butterfly' in Kaggle dataset and 'other' in Museum dataset, the combination of the two datasets results in a total number of five classes. The 'other' class doesn't have enough images so we add more images to this class from another source. Table 1 shows some statistics about both datasets and Table 2 shows the data split for train, validation and test for all the challenges.

Table 1: Data statistics

Dataset	Bee	Wasp	Butterfly	Insect	Other	Total
Kaggle	3182	4941	-	2439	853	11,415
Museum	74,413	10,796	29,361	96,106	-	210,676

Table 2: Number of Images in each class in train,validation and test

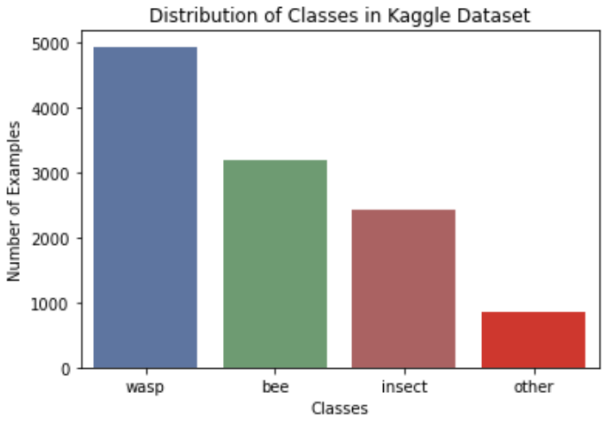
Dataset	Bee	Wasp	Butterfly	Insect	Other	Total
Train	73,773	9,820	26,255	95,442	36,266	241,556
Validation	600	600	600	600	600	3000
Test	2500	2500	2500	2500	2500	12,500
Total	76,873	12,920	29,355	98,542	39,366	257,056

The kaggle insects dataset has four classes: bee, wasp, other insect and other non-insect. Figure 3a shows the distribution of a number of images in each class.

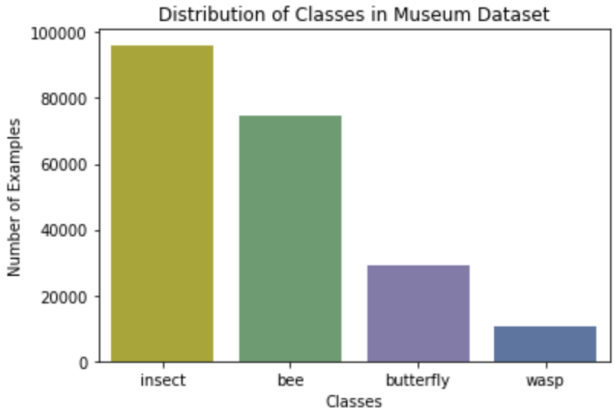
The original Insects dataset [8] is created by National Museum of Natural History, Paris[1]. It has more than 210,000 images in different sizes and orientations. The dataset has hierarchical classes which are listed from top to bottom as: Order, Super-Family, Family and Texa. Each image contains an insect in its natural environment or habitat i.e either on a flower or near to a vegetation. The images are collected by the researchers and hundreds of volunteers of SPIPOLL Science

[1] https://www.mnhn.fr/fr

project[2]. The images uploaded to a centralized server either by using the SPIPOLL website[2], Android[3] or IOS[4] mobile application. Figure 4 shows sample raw and processed images from the dataset.



(a) Kaggle: class-wise image distribution



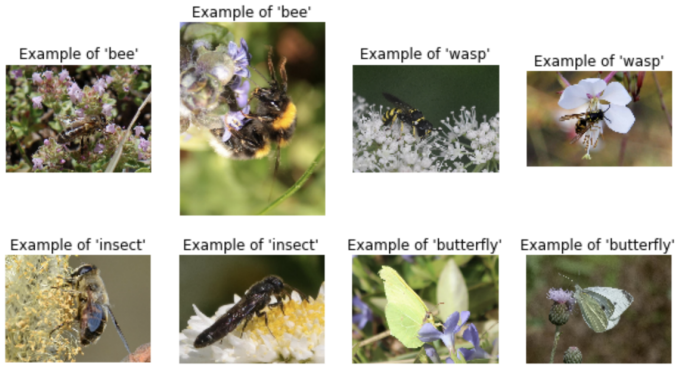
(b) Museum: class-wise image distribution

Fig. 3: Kaggle and museum datasets: class-wise image distribution

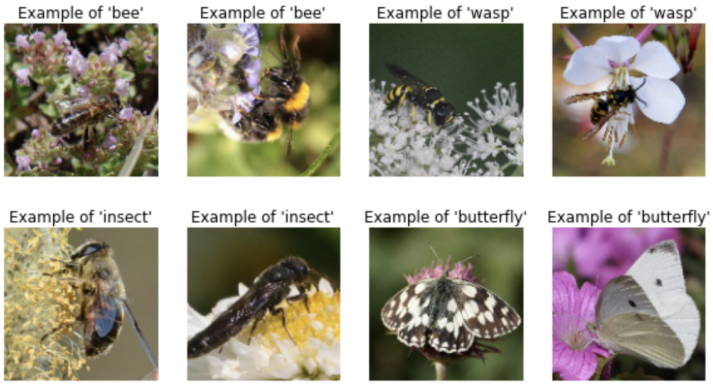
[2] https://www.spipoll.org/
[3] https://play.google.com/store/apps/details?id=fr.eneo.spipoll&hl=fr
[4] https://apps.apple.com/fr/app/spipoll/id1495843067

Table 3: Meta-Data with Museum Data

URL	picture file URL
YMD	date in YYYY-MM-DD format
Lat	Latitude in decimal degrees
Long	Longitude in decimal degrees
Flower	flower taxa as from proposed Spipoll flower taxa list
collection	set of 520 mn pictures of arthropods on specific flower specie
user-id	User id
B&W-taxonomy	Taxonomy i.e. Bees, wasps, butterflies and Other insects
Nom-taxon	spipoll taxa name
ORDRE	hierarchical taxonomical level "order" (no ranking)
IFOR	hierarchical taxonomical level "infra-order"
SPFM	hierarchical taxonomical level "super-family"
FM	hierarchical taxonomical level "family"



(a) Sample raw images



(b) Sample preprocessed images

Fig. 4: Sample images from insects dataset

B Data cleaning and bias

We have made some important observations during the exploration of the Kaggle dataset and we have identified some possible sources of bias:
Some samples were originally created with the intention to detect whether bees were carrying pollen or not. This makes that set of samples very similar because of the similar blue backgrounds. Therefore, these sets of samples are not completely independent from each other as shown in Figure 5. These kinds of images are problematic for the model because the model can make correlations between the background and the label. On the other hand, some of the samples were created from videos shot at the entrance of a wasp colony. This also introduces the problem discussed for bee images (See Figure 6) . We have removed these images from the datasets for a fair evaluation of the participants.

Fig. 5: Images of bees with blue background

Fig. 6: Redundant wasp images extracted from video

We further analyse the images of the two datasets to identify any bias due to the source of the dataset. The images in both datasets are probably not be taken in the same condition i.e. camera lens, brightness, time of the day, camera angle etc. To observe a fairly general difference between the two datasets, we compute the mean of each red, green and blue component for each class in each dataset. The result obtained is shown in Figure 7.
As observed, bees images have the same average color but the brightness is not the same. Museum images are more luminous than the Kaggle images for the same color. The could be due to the camera used or the conditions in which the pictures are taken, which may differ. For wasps, the average color is clearly not the same for both datasets.

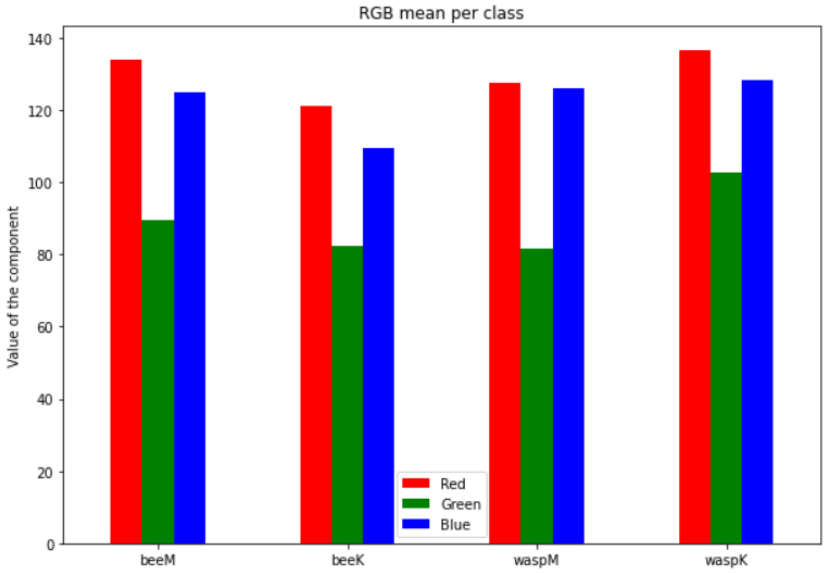


Fig. 7: Mean of each RGB coponent for each class of each dataset. The letter K means Kaggle and M means Museum

C Image size analysis

For this project, we have chosen the image size of 128x128 after experiment and analysis of different resolutions: 64x64, 128x128 and 256x256. This experiment helps us to do a compromise between dataset size and machine learning performance. We use ResNetV2 architecture from Keras for this experiment. We observe that 256x256 gives the highest scores among the observed sizes but the dataset size also increases with a factor of two. As compared to 64x64, 128x128 size has a reasonable score and less variance when compared to 256x256. We keep 128x128 for the challenge to have a good trade-off between image size and dataset size. In addition, not like 64x64, 128x128 images are clearly recognizable to human eyes. See Figure 8 for results of this experiment.

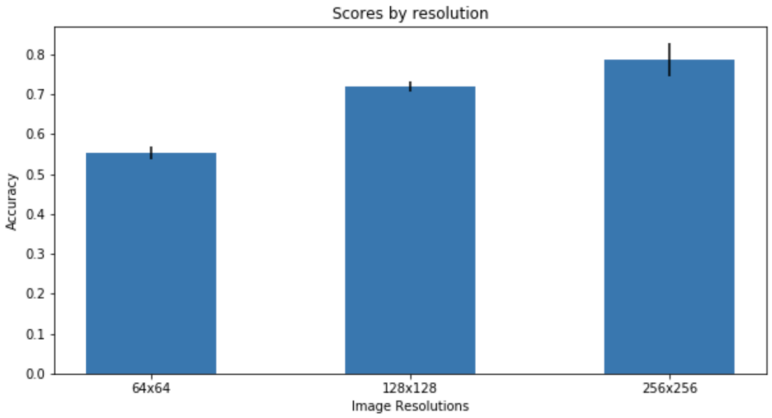


Fig. 8: Comparison of image size: 64x64 vs 128x128 vs 256x256

D Explainability

Explainable AI (XAI) helps to understand the model's outputs. This helps in analysing:

- the model is behaving as expected
- recognize bias in the model
- get ideas for ways to improve the model and also the data

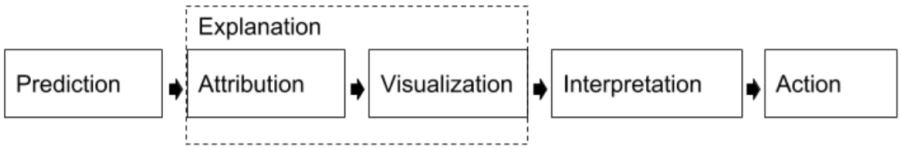


Fig. 9: Chosen XAI workflow for the challenge

For the purpose of the challenge creation, we particularly propose the use of XAI for the third point, i.e. to get a better glimpse into the data. There are multiple methods for explainability which are dependent on the type of the dataset. We have focused on the post-hoc explanation method: feature attribution, which is best suited for image data. Feature attributions can indicate how much each feature in the model contributed to the predictions for each given example. We use a gradient based method (applicable only to differentiable models, such as neural networks, used especially for models with large feature spaces).

In essence, gradient-based methods for feature attribution propagate the prediction score, layer by layer of the deep network, from the output of the network back to its input, in the process assigning each pixel a score proportional to its importance. This gives us an opportunity to visualize the predicted class, along with an overlay of another image (a heatmap) showing which pixels in the image contributed most strongly to the resulting prediction.

We have implemented GRAD-CAM[7], a member of the family of gradient based methods known for its simplicity and effectiveness. It is simple because the entire method can be summarized as global-average pooling of the gradients of the predicted class with respect to the inputs (or rather feature maps of a specific CNN layer). It is effective because we don't need to re-train the model for just explainability and we can use the existing model as it is. We used VGG-16, pre-trained on ImageNet, and trained its last convolution block (block5_conv) along with the final layers on our complete dataset, freezing the rest of the layers (the training time was 6 hrs on average for 10 epochs). The choice lies in the fact that VGG-16 is amongst the simplest in architecture which still is known to deliver good performance, thus easier to fine-tune. Figure 10 shows the flowchart

of how Grad-CAM works with VGG. Figure 11 shows the performance of VGG on the insects dataset.

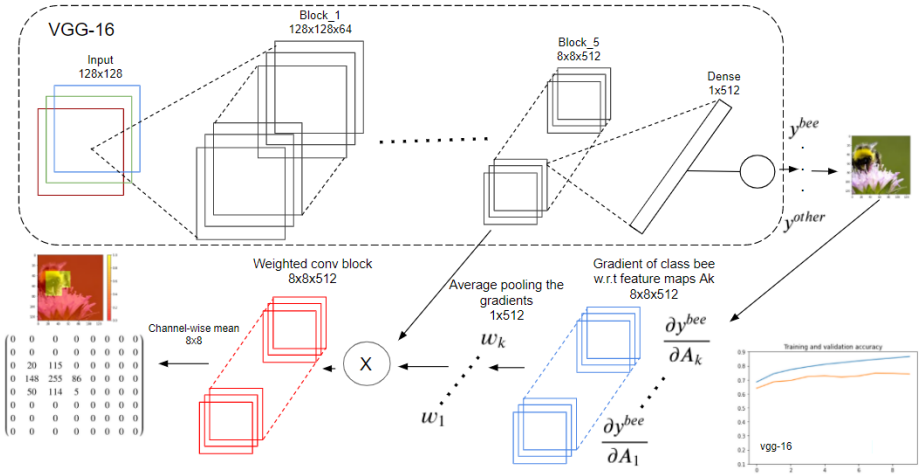


Fig. 10: Working flow of Grad-CAM

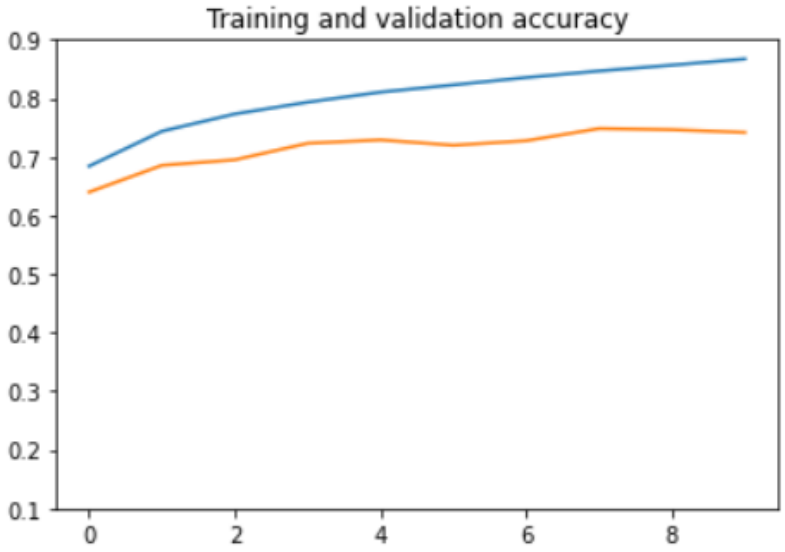


Fig. 11: Performance of VGG-16 based on transfer learning used for Explainability

In Figure 12, we visualize the overlayed images of the feature attributions and the original image, which gives us much insight into the model and the data, and we see some examples where the class attribution is incorrect and the model detects counterfactual objects in the background.

Quantifying explainability for the challenge

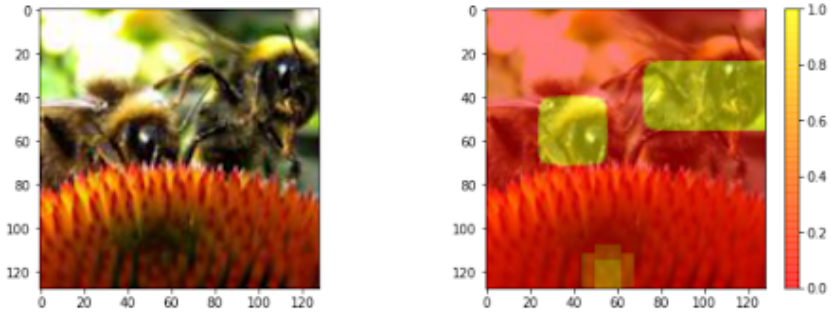
For the challenge in which we have proposed *Explanability* as one of the tasks, a non-trivial issue is then to quantify how well the challengers perform in this task. The existing methods in the literature for image data XAI are instance-based and qualitative in nature.

Inspecting the feature attributions for an individual prediction may provide good insight, but the insight may not be generalizable to the entire class for that individual instance, or the entire model. To get more generalizable insight, there then seems to be a need to aggregate attributions over subsets over the entire dataset. But this is in effect contrasting to the nature of the method deployed, which offers instance-level feature attribution. Therefore, we encourage the participants to improve their models via getting a score for correctly annotating objects in the image using Grad-CAM or other feature attribution methods for images.

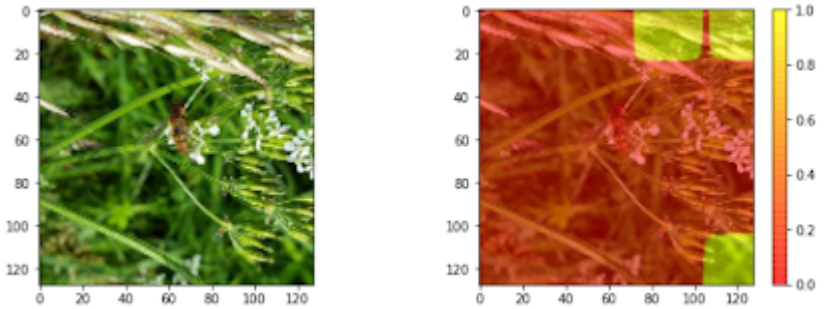
Ideally, a model should give importance to features where the class is present and not the objects around it, else its predictions are not trustworthy. Many authors have previously used humans to assess the trustworthiness. Given the nature of the challenge, it is not feasible to have human evaluators in the loop. Yet the trustworthiness of the model can be assessed if the model makes its decisions based on features relevant to the object, and not other factors in the background. Therefore, we challenge the participants to find the center of their model's feature attributions, which should be close to the center of the hand-annotated bounding boxes, quantifying good feature attribution. We hand-annotated a small subset of the train dataset (10 images per class) to form bounding boxes around the class instance for all the subsets of the dataset. We carefully picked instances where the object is not in the center but rather around the edges of the image, to avoid a heuristic prediction of the center.

We use a soft bounding-box metric between the ground truth and the predicted center to assess the model's quality towards explainability. The participants should be able to achieve this, for example in the case of Grad-CAM via extracting the highest pixel corresponding to the highest intensity of the gradient of the predicted class with respect to the output of the last convolution block. We use the following formula for calculating the explainability score.

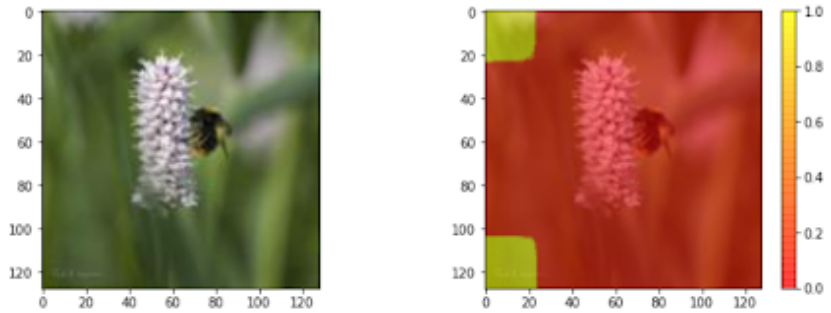
$$\exp\left(-0.5772 * \left[\frac{(x - x0)^2}{a^2} + \frac{(y - y0)^2}{b^2} \right] \right)$$



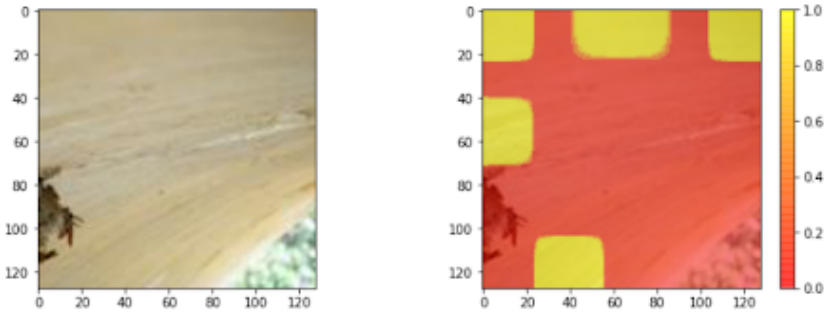
(a) Correct prediction based on object



(b) Correct prediction based on background



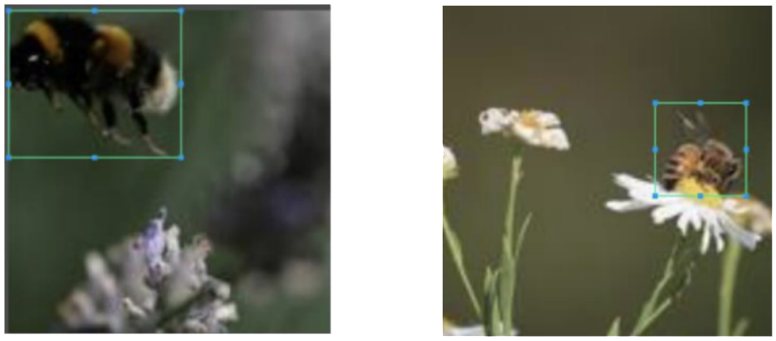
(c) Wrong prediction based on background



(d) Wrong prediction based on background

Fig. 12: Explainability sample images visualization

where (x_0, y_0) are the coordinates of the center of the given bounding box, (x, y) the coordinates predicted by the participant, and a and b are the length and height of the bounding-box. This represents a score whose sensitivity or gradient is a factor of the size of the object in the image, lower for larger images, thus the idea of a soft bounding box. The model we used for the purpose of XAI is VGG-16 trained on the entire dataset with a test accuracy of 78.3%. We provide it as a baseline model along with the relevant code of Grad-CAM for the explainability task. The participants would be expected to create a csv in a pre-defined format with the predicted center of objects which would be scored using the defined methodology. See Figure 13a for visualization of soft bounding boxes provided with the dataset and Figure 13b for visualization of the predicted and given centers of object.



(a) Soft bounding boxes of insects

(b) Predicted and given centres of object

Fig. 13: Explainability sample images visualization

E Baseline results

E.1 Scikit Learn baselines

We use some Scikit Learn classification models: Gaussian NB, Ridge Classifier, SVM, MLP Classifier and Random Forest Classifier, as baseline models for the Challenge 1. This challenge has the features extracted from images using KAZE descriptor [2] as the tabular data.

The KAZE descriptor is an algorithm in Open-CV to detect and describe multi-scale 2D features of images. These features, which can be specific points, edges, objects or shapes, can be used to identify specific elements or objects in an image. Consequently, the KAZE descriptor can be used to extract key-points, feature matching between different images or object recognition. Thus can be used for image classification. What makes this descriptor different from the others like SIFT, ORB, BRIEF etc is the scale space on which it is based. Most of the methods rely on Gaussian Scale Spaces, whose main drawback is the fact that they blur edges in images, smoothen noise and details at the same level, reducing localization accuracy and distinctiveness. In case of KAZE descriptor, the scale space is a nonlinear one obtained using diffusion filtering, a task with higher computing costs but leading to better results in terms of description and detection.

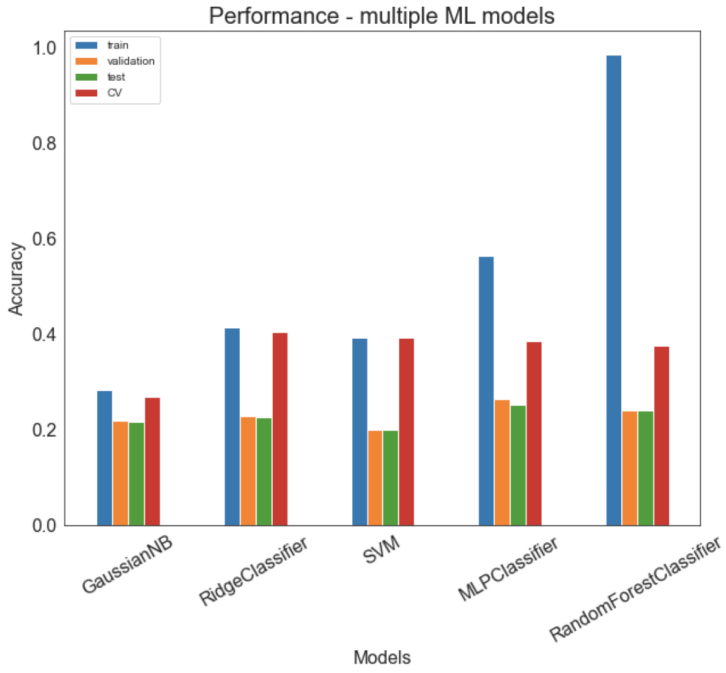
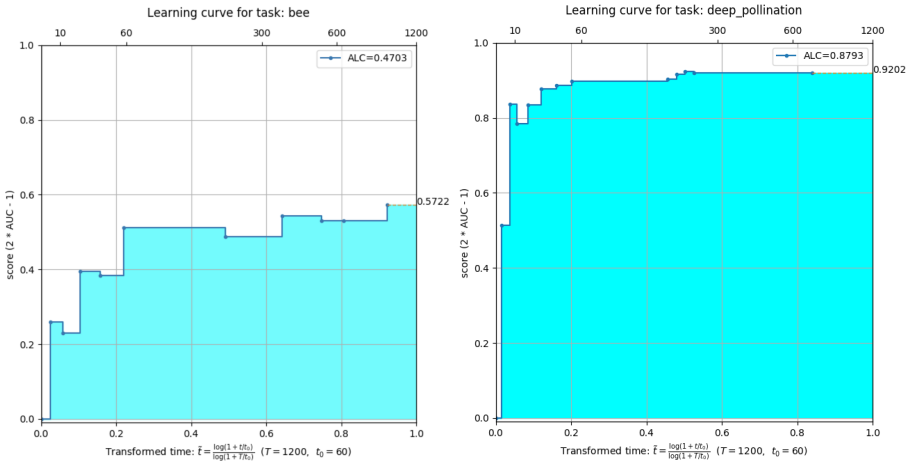


Fig. 14: Scikit-learn models performance comparison

We have evaluated the Scikit Learn models with the Train set, Validation set, Test set and a Cross Validation method (see Figure 14).
The performance results are not very good for the validation and test sets. The difference between the train and the validation/test score is significant and huge overfitting can be observed in Random Forest Classifier. The results give us insight into the data itself as the use of extracted features as input data to the classifiers does not produce satisfactory results. For this reason, we have two more challenges that uses deep neural networks for the classification and not classical machine learning classifiers. Another difference is the input data i.e. we use preprocessed images and not extracted features as the input.

E.2 AutoDL baselines

We have used a small subset of our training data and structured it according to the defined Data Format[5] for AutoDL challenge [9] and submitted it to the AutoDL challenge [9]. This challenge is an inverted challenge that takes a formatted dataset as input and runs the model of the winners of AutoDL solution on the data.
We have prepared two datasets: one with features extracted from images in tabular form and the second with images in pre-processed form with 128x128 size. From Figure 15, we observe that the model performs better for images data rather than tabular data and we conclude that deep neural networks work better for images than extracted features.



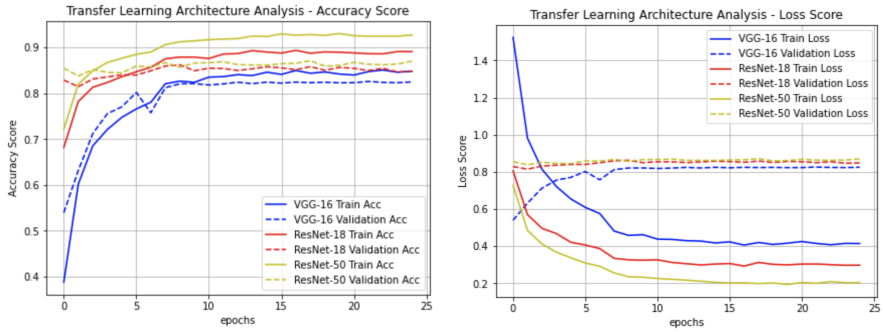
(a) Tabular data performance (b) Images data performance

Fig. 15: AutoDL baseline performances

[5] https://github.com/zhengying-liu/autodl-contrib

E.3 Transfer learning

We use transfer learning for this problem. In transfer learning, a model is trained on a source dataset (typically ImageNet [6]) and is then used for a target dataset which is normally similar to the source. The pre-trained model can be fine-tuned to adjust the weights according to the target dataset. As a preliminary exercise, we took a subset of the data consisting of 10,000 samples and trained the CNN architectures: VGG, ResNet-18, and ResNet-50. These architectures are pre-trained on Imagenet [6]. Figure 16 shows the accuracy and loss for each architecture.



(a) Multi-models accuracy (b) Multi-models loss

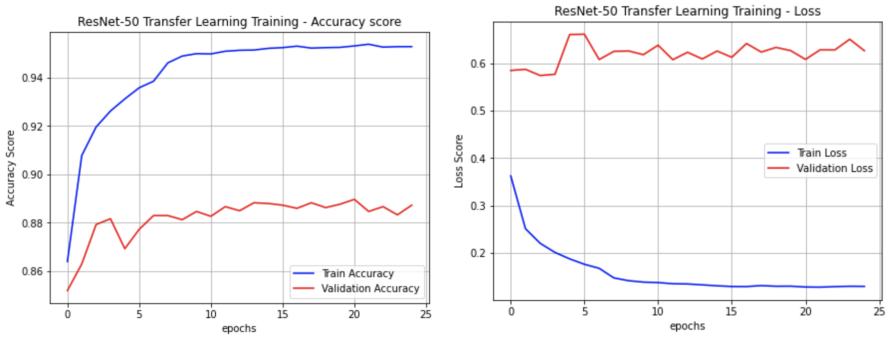
Fig. 16: Transfer learning performance results on subset of data

We observe that there is a stronger accuracy learning curve for VGG-16, while for ResNet-18 and ResNet-50 it is not as prominent. Although we can see some overfitting in the three architectures, but it is fairly slight. However, when it comes to the loss, we can observe that the train loss decreases as expected over epochs but validation loss does not. For the two ResNet architectures it stays constant and, for VGG-16, it even increases for the first epochs before staying constant. This could be due to overfitting and the fact that, in ImageNet (on which ResNet and VGG is trained on), there is no 'wasp' class.

The best results achieved for the subset of data is with ResNet-50 with a 87.13% validation accuracy score. It was trained on Cross Entropy Loss and Stochastic Gradient Descent over 25 epochs, with a batch size of 64 and a learning rate of 0.001. A learning rate decay by a factor of 0.1 every 7 epochs was used as well. This was developed using Pytorch and trained on Google Colab. Overall, it took about 9 hours to train the model (45 minutes per epoch).

We train the ResNet-50 architecture with the same experimental settings and hyper-parameters on the entire dataset. The best validation accuracy score we

have obtained is 89%. Although reasonable, we can see there is slight overfitting. We then proceeded to evaluate the model with the test set, composed of 12500 samples (2500 samples per class). This yielded an accuracy of 87.8%, fairly close to the validation accuracy score. Figure 17 shows the accuracy and loss for this experiment. To evaluate how the model takes decisions, we computed a confusion matrix for the model, which can be seen in Figure 18.



(a) ResNet-50 accuracy (b) ResNet-50 loss

Fig. 17: ResNet-50 performance on entire dataseet

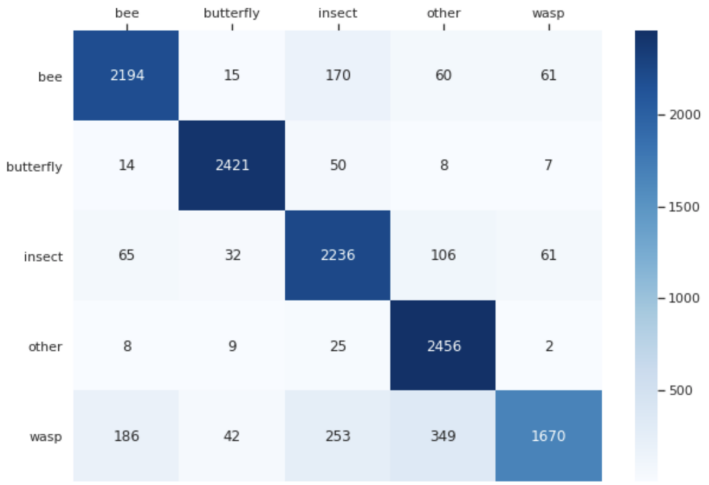


Fig. 18: Confusion matrix for the ResNet-50 model on the test set

We observe from the confusion matrix that the *wasp* class is considerably miss-classified with a higher degree than the rest, and more easily classified as *others* or *insect* than *bee* or *butterfly*. This makes sense since we have notoriously fewer *wasp* samples (12,920) as compared to the rest of the classes, and we have almost ten times more samples of *others* (98,542) than *wasps*. Something interesting is that, although being the second shortest class in samples (29,355), the *butterfly* class is the second best classified by our model. This can be due to the characteristic physical differences that butterflies present compared to bees, wasps and other pollinating insects. In the future, we could try to even out the number of wasps samples to the rest of the classes and observe if this miss-classification decreases.

F Important links

Github repository :
https://github.com/ihsaan-ullah/deep-pollination

Challenge 1 :
https://competitions.codalab.org/competitions/28635

Challenge 2 :
https://competitions.codalab.org/competitions/28996

Challenge 3 :
https://competitions.codalab.org/competitions/29425

Contact

For any query about the Deep-Pollination challenges, reach us out by email ihsan.ullah@universite-paris-saclay.fr or ihsan2131@gmail.com