Morphometry landmarking using Deep Neural Network

Le Van Linh[a,c,*], Beurton-Aimar Marie[a,1], Zemmari Akka[a], Parisey Nicolas[b,1]

[a] *University of Bordeaux, 351, cours de la Libration, 33405 Talence, France*
[b] *UMR 1349 IGEPP, BP 35327, 35653 Le Rheu, France*
[c] *Dalat University, Dalat, Lamdong, Vietnam*

Abstract

Morphometry landmark is an important characteristic in the morphometric analysis which can use to appreciate the covariances between the ecological factors and the organisms. Mostly, the morphometry landmarks are defined manually or semi-automatically. In this study, we propose a convolutional neural network (CNN) to predict the landmarks on 2D anatomical biological images, specify beetle's images. The network has been trained and evaluated on a dataset includes the images of collecting from 293 beetles. During the experiments, the network has been tested in two ways: training from scratch and applying fine-tuning. The quality of predicted landmarks is evaluated by comparing with the manual landmarks which provided by the biologists. The obtained results are considered to be statistically good enough to replace the manual landmarks for the different morphometry analysis.

Keywords: Morphometry landmarks, deep learning, fine-tuning, morphometry analysis

1. Introduction

Morphometry analysis mentions to measure the topography of an object, *for example*, shape or size of the object. The biologists work with some properties of organisms such as lengths, widths, masses,... to analyze the interaction between

*Corresponding author
Email address: van-linh.le@labri.fr (Le Van Linh)
[1] both authors contributed equally to this work.

environment and organism's development. Besides these properties, *landmarks* are known as another property to analyze the shape of the organism. The shape can be determined if we have enough the landmarks instead of collecting all the information. Landmarks, or *points of interest*, are the points on the image that store important information about the shape of the object, *for example*, the left and right corners of eyes are two important points to detect the human eyes. Depending the object, the number of landmarks may different, as well as their position can be defined along the outline of the object or inside the object, i.e. the landmarks on Drosophila wings [] are stayed on the veins of the wings, but the landmarks on human ears [] can be located on the ear edge or inside the pimas of the ears.

Currently, the landmarks are set manually by the entomologist, one can note that this work is time-consuming and difficult to the procedure when the user change; or they can be set semi-automatically by applying the image processing techniques. However, it is really difficult applying for the complex images. In image processing, segmentation is most often the first and the most important step. This task remains a bottleneck to compute the features of an image. In some cases, the interested object is easy to extract and can be analyzed by applying the well-known image analysis procedures. Nevertheless, we can not apply the same procedure for un-segmentable images. Instead, we have to apply another method without segmentation step to analyze the object. Therefore, a method that can provide the automatic location of landmarks without segmentation could be interested.

This work concerns a method to automatically detect the landmarks on biological images without image processing techniques. The main idea consists of the designing and training a CNN [] on a dataset with manual landmarks. The dataset includes the images which are taken from 293 beetles. For each beetle, the biologists have taken images of five parts: *left and right mandibles, head, elytra, and thorax*. All the images are presented in the RGB color model with two dimensions. Along with each image, a set of landmarks has been marked by experts which can be used as ground truth to evaluate the predicted landmarks.

During the experiments, the proposed network has been trained on the dataset by applying two strategies. In the first strategy, the network is trained from scratch on each dataset; while in the second strategy, the training process has been modified to include a fine-tuning [] stage.

The rest of this paper is organized as followed: Section **??** discusses the related works about determing the landmarks on 2D images. Section **??** shows the procedure of designing the network model. Section **??** gives another approaches to augment the dataset. The first experiment of the network on each dataset is presented in Section **??**. Section **??** presents a modification of training process and its experiment on datasets. Finally, the conclusion is given in Section **??**.

2. Related works

In geometric morphometry, landmarks (or points of interest) are important features to describe the shape. Depending on the complicacy of the objects in the image, setting automatic landmarks can rely on different methods. When the object can be segmented, the image processing techniques may applied to predict the landmarks. Lowe et al. [] have proposed SIFT method to identify the keypoints on 2D images. From the detected keypoints, the method was abled to find the matching points between two images. SURF is another method which has been proposed by Herbert Bay et al. []. The SURF algorithm is the same principles with SIFT but details in each step are different. Palaniswamy et al. [] have applied probabilistic Hough Transform to automatically detect the landmarks on 2D images of Drosophila wings. In a previous work [], we have applied a series of alogrithms to detect the landmarks automatically on beetles mandibles which are considered as the easied objects to segment (with an quality enough good for our need). In that work, the landmarks have been detected by registering two segmentations of images and then, using SIFT descriptor to refine the location of predicted landmarks. After the experiment, we have obtained good enough results on mandibles. Unfortunately, we have observed that the method did not provide good results when the segmentation is not

precise, i.e. on thorax or elytra images. This is explained why we have turned the automatically landmarking into another stage without any segmentation step.

In recent years, deep learning [] is known as a solution for the tasks in computer vision, especially for image analysis. Deep learning has been introduced in the middle of the previous century for artificial intelligence application but it has encountered several problems to take real-world cases. Luckily, the improvement of computing capacities, both in memory size and computing time with GPU programming, has opened a new perspective for deep learning. Many deep learning architectures have been proposed to sovle the problems of classification [], image recognition [], speech recognition [], language translation [], ... Using deep learning, specifically CNN, to predict the landmarks on 2D images has achieved better results even if the images that can not segment. Yi Sun et al. [] have proposed a cascaded CNNs to predict the facial points on the human face. Zhanpeng Zhang et al. [] proposed a *Tasks-Constrained Deep Convolutional Network* to optimize facial landmarks detection. Their model detected the facial landmarks with a set of related tasks such as head pose estimation, gender classification, age estimation, or facial attribute inference. Cintas et al. [] has introduced a network to predict the landmarks on human ears. After training, the network has the ability to detect 45 landmarks on human ears. In the same context of using CNN to predict the keypoints on 2D images, we have applied CNN computing to work on the un-segmentable images of beetles.

3. Network architectures designing

In the process, we have tried three network models before deciding the final architecture for detecting the landmarks on beetle images. Like other CNN models, we have employed the classical layers to construct the models, i.e., convolutional layers, maximum pooling layers, dropout layers and full-connected layers.

The first architecture is very classical one, it receives an image with the size

of $(1 \times 192 \times 256)$ as the input. Then, the network consists on three repeated strucutre of a convolutional layers followed by a maximum pooling layers. Most CNNs, the hyperparameters of convolutional layers have been set to increase the depth of the images from the first layer to the last layer. That is reflected in the setting of the number of filters at each convolutional layer. So, the depths of convolutional layers increase from $32, 64$, and 128 with different size of the kernels: (3×3), (2×2) and (2×2), respectively. Inserting pooling layers after a convolutional layers is a common periodcally. The pooling layer effects to progressively reduce the spatial size of the representation to reduce the number of parameters, computation in the network, and it also controls over-fitting. The operations of pooling layers independent on every depth slice of the input. The most common form is a pooling layer with filters of size (2×2) and a stride of 2. It downsamples every depth by 2 along width and height of the input. Thefore, all the kernels of maximum pooling layers have the same size of (2×2) with a stride of 2 as usual. At the end of the model, three full-connected layers have been added to extract the global relationship between the features and to procedure the outputs. The first of two full-connected layers are set to non-linearity to make sure these nodes interact well and take into account all possible dependencies at the feature level. The outputs of the full-connected layers are $500, 500$ and 16. The output of the last full-connected layer corresponds to the coordinates (x and y) of 8 landmarks which we would like to predict. Fig. 1 shows details of the first model: The orange rectangles represent for convolutional layers while the yellow rectangles represent for maximum pooling layers and three full-connected layers with their parameters are presented at the end of the model.

The second architecture is modified from the first model. The layers are kept the same as the first one but the outputs of the first of two full-connected layers are changed from 500 (in the first model) to 1000 (Fig. 2). Increasing the value at full-connected layers is hoping to obtain more features from convolutional layer and to prevent the over-fitting.

To build the third architecture, we have used the definition of *elementary*

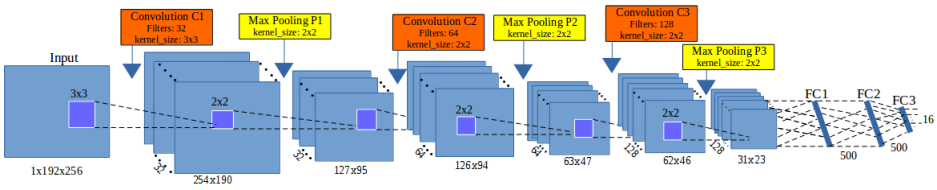


Figure 1: The architecture of the first model

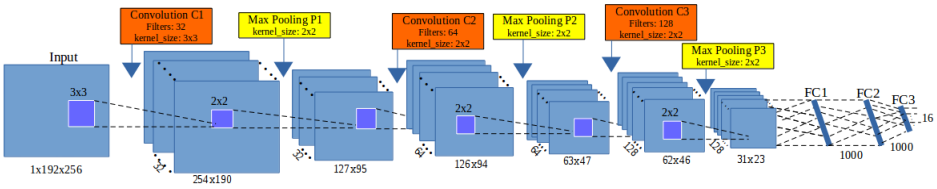


Figure 2: The architecture of the second model

block. An elementary block is defined as a sequence of convolution (C_i), maximum pooling (P_i) and dropout (D_i) layers. This significantly reduces overfitting and gives major improvements over other regularization methods []. The idea of dropout is to include some variations between different runs. During training phase, dropout samples are done from an exponential number of different "thinned network. At test phase, it is easy to approximate the effect of averaging the prediction of all thinned networks by simply using a single unthinned network with smaller weights. So, we have modified the architecture by combining some *elementary blocks*. Fig. 3 illustrates the layers in the third architecture. For our purpose, we have assembled **3 elementary blocks**. The parameters for each layer in each elementary block are as below, the list of values follows the order of elementary blocks ($i = [1..3]$):

- CONV layers:
  - Number of filters: $32, 64$, and 128
  - Kernel filter sizes: $(3 \times 3), (2 \times 2)$, and (2×2)
  - Stride values: $1, 1$, and 1
  - No padding is used for CONV layers

- POOL layers:
  - Kernel filter sizes: $(2 \times 2), (2 \times 2)$, and (2×2)
  - Stride values: $2, 2$, and 2
  - No padding is used for POOL layers
- DROP layers:
  - Probabilites: $0.1, 0.2$, and 0.3

Three full-connected layers (FC) are kept the same as the second architecture: FC1 and FC2 have 1000 outputs, the last full-connected layer (FC3) has 16 outputs. As usual, a dropout layer is inserted between FC1 and FC2 with a probability equal to 0.5.

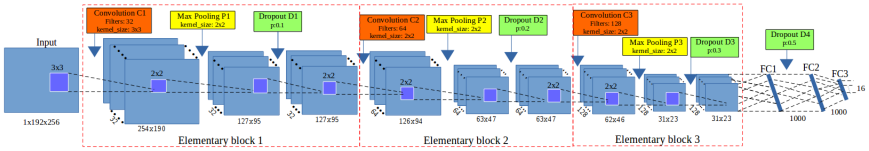


Figure 3: The architecture of the third model

The core of neural network is training over iteration. There are many ways to optimize the learning algorithm, but gradient descent [] is currently a good choice to establish the way of optimizing the loss in neural network. The core idea is following the gradient until we statify with the results will remain the same. So, we have chosen gradient descent in the backward phase to update the values of learnable parameters and to increase the accuracy of the network. The networks are designed with a small sharing learning rate and a momentum. The learning rate is initialized at 0.03 and stopped at 0.00001, while the momentum is updated from 0.9 to 0.9999. Their values are updated over training time to fit with the number of epochs [2]. The implementation of the architectures have been done on Lasagne framework [] by Python.

[2] An epoch is a single pass through the full training set

4. Data augmentation

A characteristic of machine learning and deep learning is using a volume dataset to train the model. Of course, in practice, we are not always have enough data for training. One way to solve this problem is to create the fake data from real data and to add it to the training set. Dataset augmentation has been a particularly effective technique for a specific problem. For example, in images classification problem, the operations like translating, rotating or scaling the images have also effective. The fake images may be generated by translating (rotating or scaling) in each direction. Besides, injecting noise in the input can also see as a form of data augmentation.

Our dataset includes 293 images of beetles (for each anatomical part). All the images are taken with the same camera in the same condition with a resolution of (3264×2448). Each image has a set of manual landmarks provided by biologists, i.e, each thorax has 8 landmarks, each head has 10 landmarks. Applying CNNs to train each part with a small number of images to reach good results is impossible. So, we need to augment the dataset before training the networks. Firstly, we have found that the original solution of the images (3264×2448) are heavy for the neural network. For performance considerations, in most of CNNs [], the size of the input is limited to (256×256) pixels, so we have decided to down-sampling the images to a new resolution (256×192) (to respect the ratio between x and y). Of course, the coordinates of manual landmarks have been also scaled to fit with the new resolution of the images. In usual way, the transformations have been used to augment the dataset (i.e rotation, translation,...) but the analysis of image by CNN is most often translation and rotation invariant. Therefore, two other procedures have been imaged to increase the number of images in the dataset (256×192).

The first procedure is to change the value of a color channel in the original image to generate a new image. According to that, a constant is added to one of the RGB channels each time it is used for training. Each constant is sampled in a uniform distribution $\in [1, N]$ to obtain a new value caped at 255. For

example, Fig. 4 shows an example when we added a constant $c = 10$ to each channel of an original image. Following this way, we can generate three version from an image.

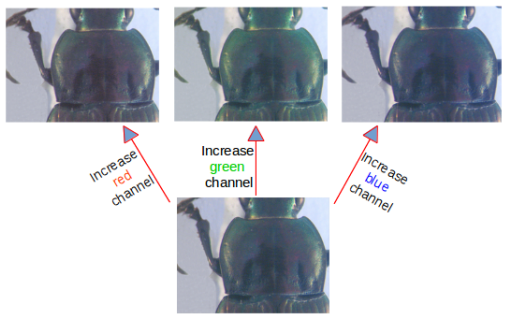


Figure 4: A constant $c = 10$ has been added to each channel of an original image

In the second procedure, we have applied the opposite procedure to the first one. Instead of adding the value, we separate the channels of RGN into three gray-scale images as the network works on single channel images (Fig. 5). At the end of the processes, we are able generate six versions from an original image. In total, we have $293 \times 7 = 2051$ images for each anatomical part of beetle (an original image and six generated images). However, we have not used all images for training and validation. So, we have chosen 260 original images and their generations (1820 images) of each dataset for training and validation processes, the remaining images (33 original images) are used for test process.

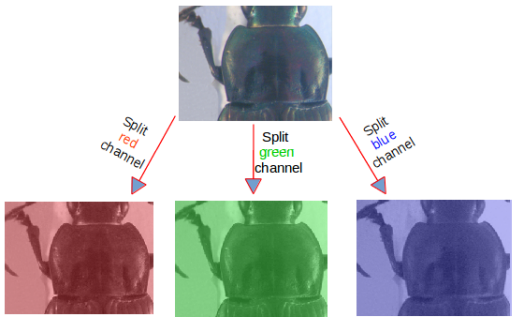


Figure 5: Three channels (red, green, blue) are separated from original image

In practical, to obtain a fast convergence during the computing, it is useful to normalize the brightness of the images to $[0, 1]$ instead of $[0, 255]$ and the coordinates of the landmarks have been also normalized.

5. Experiments and results

Before widely applying to all anatomical parts, we have firstly tried with thorax part to evaluate the performance. The networks have been trained in $5,000$ epochs on Ubuntu machine by using NVIDIA TITAN X cards. The set of images that used for training and validation are merged together.

During the training, the images are chosen randomly from the dataset with a ratio of 60% for training and 40% for validation. The training step takes into account a pair of information (*images, manual landmarks coordinates*) as training data. In the context of deep learning, landmark prediction can be seen as a regression problem. So, we have used Root Mean Square Error (RMSE) to compute the loss of implemented architectures.

At the test phase, images without landmarks are given to the trained network to produce output coordinates of the predicted landmarks. The results then evaluated by comparing with the manual landmarks coordinates provided by biologists which have been seen as ground truth. Fig. 6 shows the training errors and the validation errors during traning phase of the first architecture. The blue curve presents the RMSE errors of training process, the green curve presents the validation errors. Clearly, over-fitting has appeared in the first model. The training losses are able to decrease but the validation losses are stable. In the second model (section **??**), we have modified the parameters of full-connected layers to prevent the over-fitting but it seems that this solution is still not suitable. The over-fitting is also appeared as the first model.

Then, we have continued to train the third model on the same dataset of thorax images. Fig. 7 illustrates the losses during the training of the third model. Like the previous figure (Fig. 6), the blue line is training losses, the green line is validation losses. In the opposite with two previous models, the

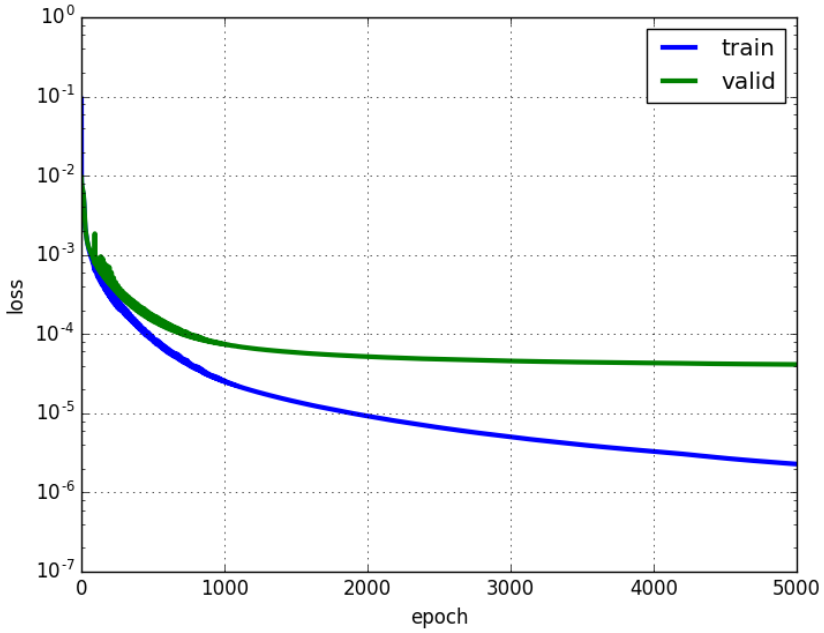


Figure 6: The training and validation losses of the first model

losses are different (far) from the beginning but after several epochs, the values become more proximate and the over-fitting problem has been solved. This proves that adding dropout layers to build the elementary blocks have been effects to prevent over-fitting and contributory improve the accuracy of the model. *So, we have decided to keep the architecture of the third model for our landmarking problem.*

In order to have the predicted landmarks for all thorax images (instead of only 33 images), we have applied *cross-validation* to choose the test images, called *round*. For each time, we have chosen a different fold of 33 images as testing images, the remaining images are used as training and validation images ($293/33 \approx 9$ rounds). Following that, the network will be trained with many different datasets, then the trained model will be used to predicted the lanmarks on the images in the corresponding test set. Table. 1 resumes the losses of 9 rounds when we trained the third model on thorax images.

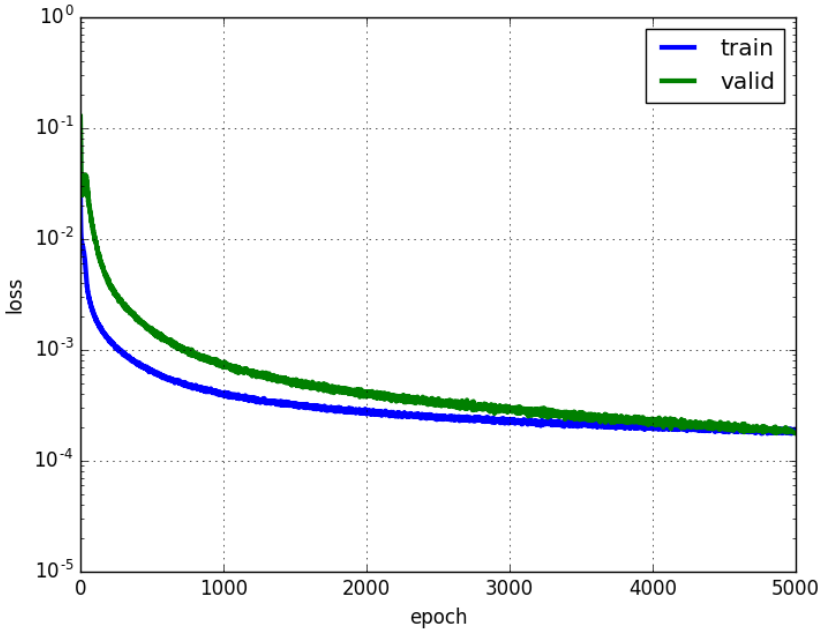


Figure 7: The training and validation losses of the third model

Round	Training loss	Validation loss
1	0.00018	0.00019
2	0.00019	0.00021
3	0.00019	0.00026
4	0.00021	0.00029
5	0.00021	0.00029
6	0.00019	0.00018
7	0.00018	0.00018
8	0.00018	0.00021
9	0.00020	0.00027

Table 1: The losses during training the third model on thorax images

To evaluate the coordinates of predicted landmarks, the correlation metrics have been computed the correlation between the manual landmarks and their corresponding predicted one. Table. 2 shows the correlation scores of 3 metrics (using *scikit-learn* []), i.e, coefficient of determination (r^2), explained variance (EV), and Pearson correlation. All of three metrics have the same possibility. The best score is 1.0 if the correlation data is good, lower values are worse. It means that our predicted coordinates are very close with the ground truth. However, the measure is not enough good to provide a useful result to biologists. Moreover standing on the side of image processing, we are looking forward to seeing the predicted coordinates than the statistical results.

Metric	$\mathbf{r^2}$	**EV**	**Pearson**
Score	**0.9952**	**0.9951**	**0.9974**

Table 2: Correlation scores between manual landmarks and predicted landmarks

The main goal of computing is to predict the coordinates of landmarks, so the distances (in pixels) between the coordinates of manual landmarks and corresponding predicted landmarks have been taken into account on all images. Then, the average of distances are computed by landmarks. Table. **??** shows the average distances by landmarks on all images of thorax dataset. With images of resolution 256×192, we can consider that an error of 1% corresponds to 2 pixels that could be an acceptable error. Unhappily, our results exhibit average distance of 4 pixels in the best case, landmark 1 and more than 5 pixels, landmark 6. Other error distances are more than 2% pixels.

Fig. 8 shows the distribution of the distances on the first landmark of all images. The accuracy based on the distance in each image can be separated into three spaces: the images have the distance less than average value (4 pixels): 56.66%; the images have the distance from average value to 10 pixels (average distance plus standard deviation): 40.27%; and the images have the distance greater than 10 pixels: 3.07%.

To illustrate this purpose, Fig. 9 shows the predicted landmarks on two test

Landmark	**Distance** (in pixels)
1	4.002
2	4.4831
3	4.2959
4	4.3865
5	4.2925
6	5.3631
7	4.636
8	4.9363

Table 3: The average distances on all images per landmark.

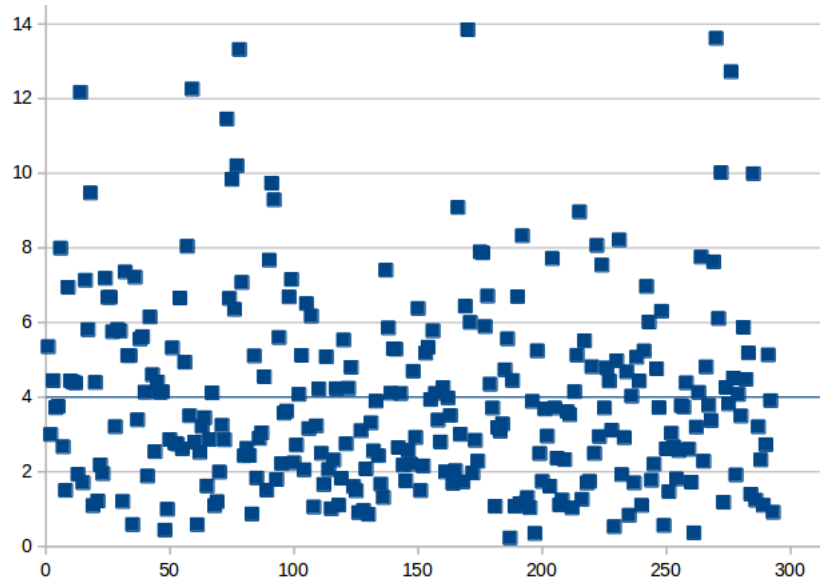
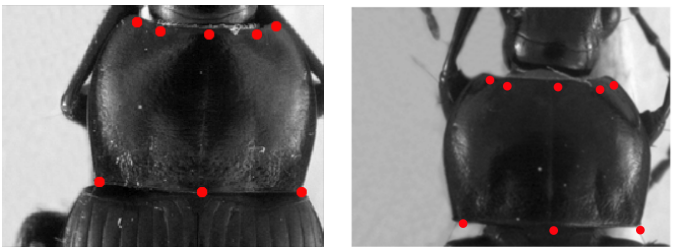


Figure 8: The distribution of the distances on the first landmark. The blue line is the average value of all distances.

images. One can note that even some predicted landmarks (Fig. 11a) are closed to the manual ones, in some case (Fig. 11c) the predicted ones are far from the expect results. The next step has been dedicated to the improvement of these results.



(a) Image with well-predicted landmarks

(b) Image with inaccuracy landmarks

Figure 9: The predicted landmarks, in red, on the images in test set.

From the success of the third architecture on thorax dataset, we apply the same procedures (data augmentation, training, ...) on other parts of beetle: *elytra and head*. However, we have modified the number of the last full-connected layer to adapt with each dataset before training. Arcoding, the values at the last full-connected layer are set to 22 and 20 outputs corresponds to 11 and 10 landmarks on elytra and head, respectively. Of course, we have also applied *cross-validation* to select testing data to get all predicted landmarks for all images in each dataset. Then, the quality of predicted landmarks are evaluated by comparing with the corresponding manual landmarks (distance computation). Table. 4 and 5 show the average distances on each landmark of elytra and head anatomical, respectively. The losses during training of both two parts are presented in Appendix **??**. Comparing with the average distances on the thorax part, it seems that the proposed architecture provides more accurate predictions on the elytra dataset, but the results are opposite on head dataset.

Landmark	**Distance** (in pixels)
1	3.8669
2	3.9730
3	3.9166
4	3.8673
5	4.0151
6	4.8426
7	5.2125
8	5.4685
9	5.2692
10	4.0709
11	3.9896

Table 4: The average distance on all images per landmark on **elytra** images

Landmark	**Distance** (in pixels)
1	5.5280
2	5.1609
3	5.3827
4	5.0345
5	4.8393
6	4.4516
7	4.7937
8	4.5322
9	5.1412
10	5.0564

Table 5: The average distance on all images per landmark on **head** images

6. Resulting improvement by fine-tuning

The proposed network (third architecture) presented in section **??** have been trained from scratch on three datasets (thorax, elytra, and head). At the first step, the network was able to predict the landmarks on the images. But as we have discussed, even if the strength of the correlation seems to validate the results, when we display the predicted landmarks on the images, the quality of the predicted coordinates are also not enough precise, and the average error are also still high (of course, we have the distances are higher than the average distances).

In order to reach more acceptable results for biologists, we have broadened model with another step of deep learning: **transfer learning**. That is a method enables to re-uses the model developed for a specific task/dataset to lead another task (called *target task*) with another dataset. This process allows rapid process and improves the performance of the model on the target task []. The most

popular example has been given with the project ImageNet of Google [] which has labeled several millions of images. The obtained parameter values which can be used in another context to classify another dataset, eventually very different dataset []. The name of this procedure to re-use parameters to pretrain a model is currently called **fine-tuning**.

Fine-tuning does not only replace and retrain the model on the new dataset but also fine-tunes the weights of a trained model by continuing the backpropagation. Unfortunately, *some rapid tests have shown that re-using ImageNet features has not been relevant for our application.* We have designed a way to reproduce the method with our own data. It is worth noting that of course the size of data to pre-train has drastically decreased. For our pre-training step, the network has been trained on the whole dataset including the images of three parts of beetle *i.e thorax, elytra and head.* Then, the trained model has been used to fine-tune and test on each dataset.

6.1. Data preparation and training

The images training dataset is combined from the images of three sets: *thorax, elytra, and head* (after augmentation). When applying the training from the scratch, we have used cross-validation to select the data (9 folds). It means that for each dataset, we have some different training data and corresponding testing data. So, the images that use to train the model are just select from one of the folds in each dataset. Specifically, we have taken $1,820$ images of each part. In total, it includes $5,460$ images ($260 \times 7 \times 3$).

However, another problem has been appeared when we combined the images from different dataset. That is the different number of landmarks on each part: 8 *landmarks on thorax part,* 10 *landmarks on head part, and* 11 *landmarks on elytra part.* Fig. 10 shows the possition of the landmarks on each part. Because of the meaning of landmarks on each anatomical part for biologists, we cannot insert the landmarks arbitrary. So, we have decided to keep the landmarks on thorax as reference and to remove the landmarks on elytra and head parts instead of adding. We kept 8 (landmarks) as a reference number, then we

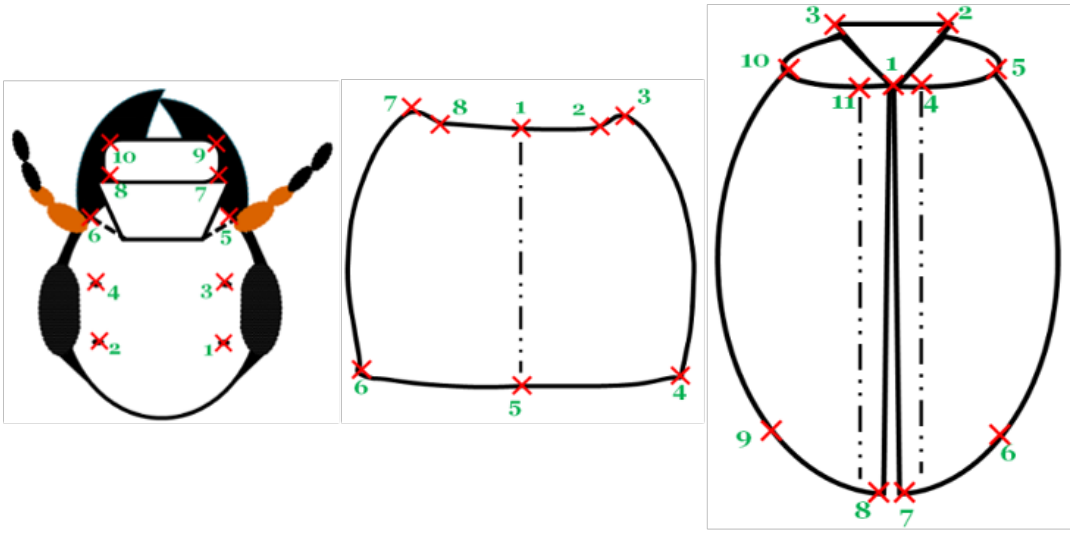


Figure 10: A presentation of head, thorax and elytra part with corresponding manual landmarks

have removed the supernumerary when it is unnecessary. Specifically, we have removed three landmarks on the elytra part $(1^{st}, 6^{th}, 9^{th})$, and two landmarks on the head part $(5^{th}, 6^{th})$.

During training the proposed architecture on the combined dataset, the parameters of the network (learning rate, momentum, ...) are kept the same as training from scratch but the number of epochs are increased to $10,000$ instead of $5,000$ to achieve better learning on the parameters (weights). Additional, we have shuffled the training set. Because the neural network learns the faster from the most unexpected sample. It is advisable to choose a sample at each iteration that is the most unfamiliar to the system. Shuffling the examples will be helped the model works with different anatomical parts rather than the same anatomical samples in each training time.

6.2. Fine-tuning on each dataset

The combined dataset then used to train the third architecture (with 8 outputs). Then, the trained model is used to fine-tuning on each dataset. To compare the result with the previous one, we have also fine-tuned the trained

model with different dataset (applying cross-validation). Firstly, we consider on the losses during fine-tuning. *For example*, Table. 6, 8, 10 show the losses during fine-tuning on thorax, elytra, and head dataset, respectively. Comparing with the losses when we trained the model from scratch, *i.e.* on thorax, the validation losses of this scenario have been significantly improved (around 40%).

On each part, the landmarks are predicted on the test images. Then, the average error based on the distances between predicted and corresponding manual landmarks have been also computed. Table. 7, 9, and 11 show the average distances per landmark on thorax, elytra, and head dataset, respectively. **From scratch** columns remind the previously average distances. **Fine-tune** columns present the new average distances after applying fine-tuning on each part. It is clearly shown that the result of predicted landmarks with the help of fine-tuning is more precise than training from scratch. For example, when we compare the average distances between two processes, the worse case of fine-tuning process is still better than the best case of training from scratch.

Round	Training loss	Validation loss
1	0.00019	0.00009
2	0.00018	0.00010
3	0.00018	0.00010
4	0.00019	0.00008
5	0.00019	0.00009
6	0.00018	0.00008
7	0.00019	0.00008
8	0.00018	0.00006
9	0.00018	0.00009

Table 6: The losses during fine-tuning model on thorax dataset

#Landmark	**From scratch**	**Fine-tune**
1	**4.00**	**2.49**
2	4.48	2.72
3	4.30	2.65
4	4.39	2.77
5	4.29	2.49
6	**5.36**	**3.05**
7	4.64	2.68
8	4.94	2.87

Table 7: The average error distance per landmark of two processes on thorax images

In other view, Fig. 12 shows the comparation of the average distance distribution on each dataset in two procedures (from scratch and fine-tuning). In

Round	Training loss	Validation loss
1	0.00020	0.00006
2	0.00020	0.00006
3	0.00021	0.00006
4	0.00021	0.00006
5	0.00019	0.00006
6	0.00019	0.00006
7	0.00018	0.00005
8	0.00020	0.00006
9	0.00019	0.00006

Table 8: The losses during fine-tuning model on elytra dataset

#Landmark	From scratch	Fine-tune
1	**3.87**	2.34
2	3.97	2.27
3	3.92	2.27
4	**3.87**	**2.25**
5	4.02	2.27
6	4.84	3.14
7	5.21	3.14
8	**5.47**	3.29
9	5.27	**3.42**
10	4.07	2.49
11	3.99	2.30

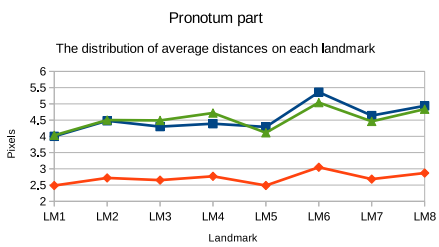
Table 9: The average error distance per landmark of two processes on elytra images

which:

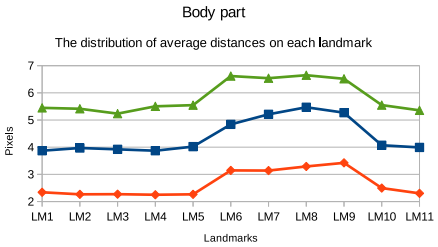
- **Blue** curves: present for the average distances on each landmarks when we train the model from scratch.
- **Orange** curves: describe for the average distance on each landmark when we fine-tune the trained model.
- **Green** curves: illuslate for the average distances **plus its standard deviation** on each landmark in fine-tuning case.

The fine-tuning process has improved the results of the proposed architecture on both 3 datasets: thorax, elytra and head. All the average distances are significantly decreased. Specially, the results have been improved $\approx 40.3\%$ on thorax, $\approx 39.8\%$ on elytra, and $\approx 46.4\%$ on head part based on considering the average distances per landmark.

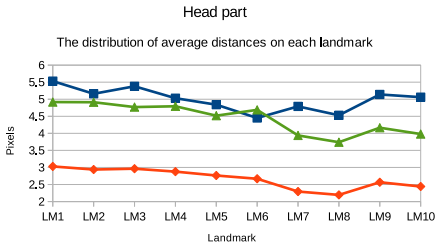
To illustrate the final results, we display the distribution of the distances of both the best and the worst results (resp. landmark 1 and 6) on thorax dataset.



(a) Thorax



(b) Elytra



(c) Head

Figure 11: The distribution of average distances on each landmark of each part.

Round	Training loss	Validation loss
1	0.00022	0.00007
2	0.00022	0.00007
3	0.00023	0.00008
4	0.00023	0.00008
5	0.00022	0.00008
6	0.00023	0.00007
7	0.00022	0.00008
8	0.00023	0.00007
9	0.00024	0.00008

Table 10: The losses during fine-tuning model on head dataset

#Landmark	From scratch	Fine-tune
1	5.53	**3.03**
2	5.16	2.94
3	**5.38**	2.96
4	5.03	2.88
5	4.84	2.76
6	**4.45**	2.67
7	4.79	2.29
8	4.53	**2.20**
9	5.14	2.57
10	5.06	2.44

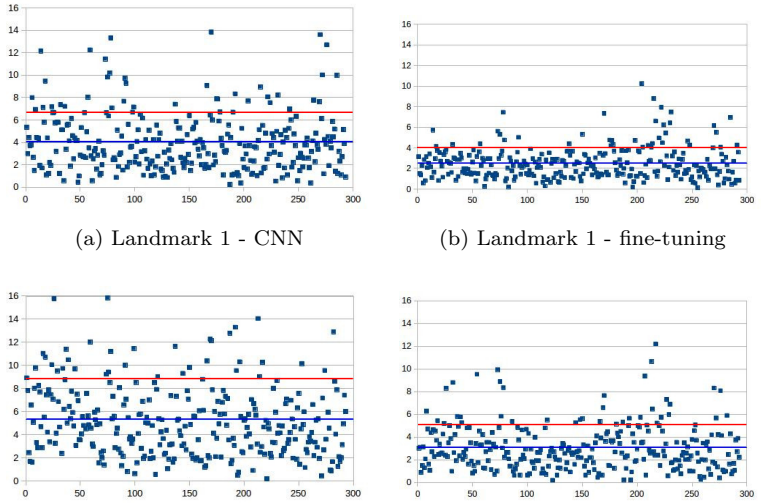
Table 11: The average error distance per landmark of two processes on head images

The Fig. 12 shows in (12a) and (12b) diagrams how much the average distances (blue lines) and standard errors (red lines) have been improved for the landmark 1, the (12c) and (12d) diagrams for the landmark 6.

7. Conclusion

In this work, we have presented how to apply convolutional neural network to predict the landmark on 2D anatomical images of beetles. After testing several models, we have presented a convolutional neural network for automatic detection landmarks on anatomical images of beetles. The training and testing processes have been finished by using two strategies: *train from scratch* and *fine-tuning*.

In our case, the size of dataset is limited. Therefore, we have applied the image processing techniques to augment dataset. The predicted landmarks have been evaluated by calculating the distance between manual landmarks and corresponding predicted landmarks. Then, the average of distance errors on each landmarks has been considered.



(a) Landmark 1 - CNN (b) Landmark 1 - fine-tuning

(c) Landmark 6 - CNN (d) Landmark 6 - fine-tuning

Figure 12: The distribution of distances on 1^{st} and 6^{th} landmarks of all images in two testing steps (CNN and fine-tuning) (on thorax dataset). The blue and red lines present the average distances and standard deviation values, respectively.

The results have been shown that using the convolutional network to predict the landmarks on biological images leads to satisfying results without need for segmentation step on the object of interest. The best set of estimated landmarks has been obtained after a step of fine-tuning using the whole set of images that we have for the project, i.e. about all beetle parts. The quality of prediction allows using automatic landmarking to replace the manual ones.

References