ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 3, No. 3, XX-YY (2018)
www.astesj.com

ASTES Journal
ISSN: 2415-6698

Automatize landmarks setting on Species morphometry using Deep Neural Networks

Van-Linh Le[*,1,2], Marie Beurton-Aimar[1], Akka Zemmari[1], Nicolas Parisey[3]

[1]*LaBRI-CNRS 5800, University of Bordeaux, 33400, France*

[2]*ITDLU, Dalat University, 670000, Vietnam*

[3]*IGEPP, INRA-1349, 35653, France*

ARTICLEINFO

Article history:
Received:
Accepted:
Online:

Keywords:
Landmarks
Morphometry setting
Deep learning
Convolutional neural networks

ABSTRACT

Morphometry landmarks are known as one of the approaches to analyze the characteristics of organisms. Finding landmarks setting can give to biologists a comprehensive description of the organism. In this study, we propose a convolutional neural network (CNN) to predict the landmarks on biological's species. The network is designed as a combination of the "elementary blocks" including a convolutional layer, a maximum pooling layer, and a dropout layer. After training with a set of manually landmarks dataset, it has been used to predict the morphometric landmarks on biological images automatically. The network has been checked by applying two scenarios: training from scratch and fine-tuning. The predicted landmarks have been evaluated by comparing with the coordinates of manual landmarks which have been provided by the biologists. The network model is implemented by Python on Lasagne framework. This paper is an extension of work originally presented in ***2018 1st International Conference on Multimedia Analysis and Pattern Recognition (MAPR)****.*

1 Introduction

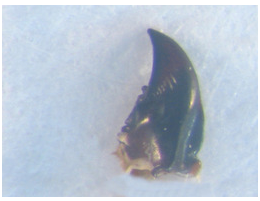
Morphometry analysis refers to measure the topography of an object, for example, its shape and its size. Biologists work with several parameters from organisms such as lengths, widths, masses, angles,... to analyze the interactions between environment and organisms development. Besides the traditional information, landmarks (or points of interest in the image) are known as one of the characteristics to analyze the shape. Instead of collecting all information, the shape is determined by a finite set of points, called landmarks. They store important information about the shape of the object, *for example*, the corners of the human mouth are a kind of landmarks. Mostly, the landmarks are along the outline of the object but in some special cases, it could be defined inside the anatomical part, *i.e.* the landmarks on Drosophila wings are the intersection of veins on fly wings, but the landmarks on pronotum can be located at the shape edge or inside the pronotum. In our study, the morphometric landmarks are specific points defined by biologists. They are used in many biological studyings. Currently, the landmarks are set manually by the entomologist, the operation are time-consuming and difficult to reproduce when the operators change. Therefore, a method that gives automatic location of landmarks could have a lot of interest.

In this study, we work on a dataset including the images of collecting from 293 beetles in Brittany lands. All the images are presented in RGB color with two dimensions. For each beetle, the biologists took images of five parts: *left and right mandibles*, *head*, *body*, *and pronotum* (Figure 1). For each part, a set of manual landmarks has been positioned by an entomologist.

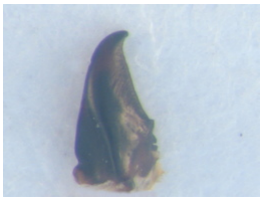
In the concept of automatically landmarks setting, image processing is usually the first choice to apply. This is a process that we apply a set of algorithms (in image processing) to extract and to analyse the object of interest. In which, segmentation is most often the first and the most important step. This task remains a bottleneck to compute features of an image. In some cases, the object of interest is easy to extract and can be analyzed with the help of a lot of very well-known image analysis procedures. Like previous study [13], we

[*]Corresponding Author Name, Address, Contact No & Email

have analyzed two parts beetle mandibles (Figure 1a and Figure1b). These parts are pretty easy to segment (enough good quality for our goals). In that work, we have applied a set of algorithms based on the combination of principal component analysis [24] and SIFT descriptor [18]. Unfortunately, this method is irrelevant with the case of the images that are not precise or difficult to segment, *i.e.* pronotum images. So, the remain question of how to predict the landmarks on the images like the pronotum images? This is the reason why we have turned to a way of analyzing images without need for a segmentation step. So, the next step has been to work with the prontoum images (Figure 1e). For each pronotum image, a set of 8 manual landmarks have been set by the biologists (Figure 2). They are considered as the ground truth to evaluate the predicted landmarks by our method.

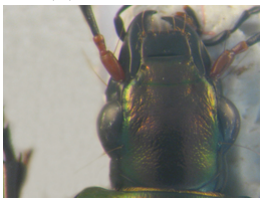


(a) Right mandible

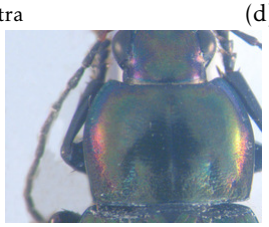


(b) Left mandible

(c) Elytra



(d) Head



(e) Pronotum/thorax

Figure 1: The anatomical parts of beetle

To achieve the landmarks prediction, this work introduces a method for this automatic detection of the landmarks on pronotum images. The main idea consists on design and train of a CNN [15] with a set of manual landmarks. In the first stage, the network has been trained from scratch on the dataset of pronotum images from the first model. In the second step, the training has been modified to improve the quality of prediction by including the fine-tuning[33] step. The network has been implemented by using Python on Lasagne library [7].

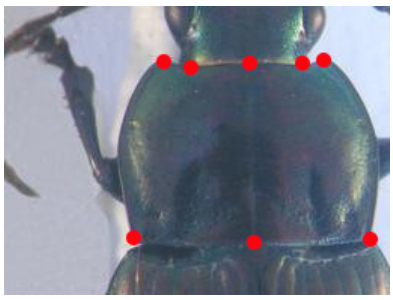


Figure 2: A pronotum image with its eight manual landmarks

The rests of the article is organized as follows. In the next of two sections, we give a briefly overview of the related works on automatically landmarking and a shortly about the CNN. After that, in Section 4, we apply the CNN to predict the landmarks on pronotum images: Firstly, we describe the processes that we have used to augment the dataset (Section 4.1). Secondly, in the Section 4.2, we present the architecture of the proposed network and its parameters also. Finally, we give the first results of the model in Section 4.4. From the first results, we present the steps of fine-tuning procedure to improve the result in Section 5. Finally, we conclude the article with a discussion of future works in Section 6.

2 Related works

A landmark is a specific point that may contain useful information. For example, the tip of the nose or the corners of the mouth are landmarks on human face [27]. Under image processing point of view, when we want to extract the feature from the image, we can consider two kinds of cases: the object of interest can be segmented or not. Setting landmarks can not be achieved in the same way depending on which situation we are. When segmentation can be applied, Lowe et al. [18] have proposed SIFT method to find the corresponding keypoints in the 2D images. From the detected keypoints, the method is able to match two images. Palaniswamy et al. [21] have proposed a method based on probabilistic Hough Transform to automatically locate the landmarks in digital images of Drosophila wings. In previous work [13], we have proposed a method which have been extended from Palaniswamy's method, to determine landmarks on mandibles of beetles. The mandibles of beetle have the simple shape and easy to segment. We have obtained good enough results about determining the landmarks automatically on mandibles. Unfortunately, after several tests, we have had to conclude that this way does not provide good results with the pronotum images because the pronotum segmentation has too many noises.

In recent years, deep learning is known as a solution for many tasks in different topics. In image analysis domain, using deep learning, namely CNN, to determine the landmarks on 2D images has achieved better results even if the images that can not segment. Yi Sun et al. [27] have proposed cascaded CNNs to predict

the facial points of interest on the human face. Zhanpeng Zhang et al. [35] proposed a *Tasks-Constrained Deep Convolutional Network* to optimize facial landmarks detection. Their model determines the facial landmarks with a set of related tasks such as head pose estimation, gender classification, age estimation, face recognition, or facial attribute inference. Cintas et al. [4] has introduced a network to predict the landmarks on human ears. After training, the network has the ability to predict 45 landmarks on human ears. In this way, we have applied CNN computing to work with pronotum landmarks.

3 Convolutional neural networks

Deep learning models are coming from the machine learning theory. They have been introduced in the middle of previous century for artificial intelligence applications but they encounter several problems to take real-world cases. Fortunately, the improvement of computing capacities both in memory size and computing time with GPU programming has opened the new perspective for deep learning.

Deep learning allows computational model composed of multiple processing layers to learn representations of data with multiple levels of abstraction [14]. Each layer extracts the representation of the input data from the previous layer and computes a new representation for the next layer. In the hierarchy of a model, higher layers of representation enlarge aspects of the input that is important for discrimination and suppress irrelevant variations. Each level of representations is corresponding to the different level of abstraction. During training, it uses gradient descent optimization method to update the learnable parameters via backpropagation. The development of deep learning opens promise results for well-known problems artificial intelligence on high dimensional data, therefore applicable to many domains: image recognition and classification [12, 5, 29], speech recognition [20, 9, 23], question answering [2], language translation [28, 10], and recognition [17, 32].

A CNN is created of neurons that have learnable parameters (weights and bias). Each neuron receives some inputs, perform dot product with the parameters and optionally followed by a decision function. In practical, the CNN consists of a number of connected layers. The first layer, called input layer, receives the input data and transform it through a succession of hidden layers. Each hidden layer comprises a set of neurons, where each neuron is connected to all neurons of the previous layer, while the neurons in the same layer are independent and do not share any connections. The last full-connected layer is called the output layer. The layers of a CNN has neurons arranged in three dimensions: *width*, *height*, *and depth* with learnable parameters. Figure 3 shows a classical example of CNN. It includes an input layer, 3 convolutional layers (CONV), 2 pooling layers (POOLING), 2 dropout layers (DROPOUT), and 3 full-connected layers (FC).

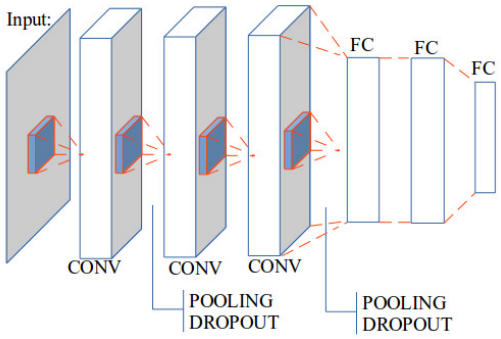


Figure 3: An example of usual convolutional neural network

The layers that are frequently used as hidden layers can be CONV layer, or POOLING layer, or DROPOUT layer, or FC layer,.... A *convolutional* layer computes a dot product between its weights and a small region in the input. At the output, the results of connected local regions are combined. CONV layer uses a set of learnable filters as parameters. Each filter is small spatially but extends the depth of the input. *Pooling* layer is used to down-sampling the input, to reduce the computational cost in remaining layers, and to control overfitting. The operator in POOLING layer is liberated on the deep of the input. *Dropout* layer refers to dropping out units in the network. Dropping a unit out means temporarily removing it from the network, along with all its incoming and outgoing connections. *Full connected* layer refers to the output of the network. The number of outputs of the last full-connected layer are corresponding to the number of predicted values.

From the beginning of deep learning until now, many deep learning frameworks have been developed. These frameworks help the users to design their application by re-using already proposed network architectures. Almost frameworks are open source. According to the written programming languages, the frameworks can be separated into two main groups: C++, *such as Caffe [11], Deeplearning4j [30], Microsoft Cognitive Toolkit [34]* and Python *i.e. Keras [3], Theano [31], PyTorch [22].* Another framework exists using more confidential languages as *Lua*.

Theano [31] is an open source framework developed by the machine learning group at the University of *Montréal*. It is a Python library that allows to define, to optimize and to evaluate mathematical expressions relating multi-dimensional arrays efficiently by using a Numpy package. Theano supports compilation on either CPU or GPU architectures. Lasagne [7] is a lightweight library in Theano. It allows to build and to train the neural networks. In this work, we have used Lasagne to implement the proposed neural network. Recently, Theano has been stopped to develop but its community is still large. The networks which have been designed by Theano are still useful and efficient in deep learning area.

4 Application to landmarks identification

In the previous sections, we have presented an overview of automatic landmarks setting and CNN. In the first of this section, we describe the process to augment the dataset which is considered as a little bit small to apply deep learning. Then, we present the designing the processes of the network architecture that we use to predict the landmarks on pronotum images.

4.1 Data augmentation

In deep learning, data are one of the most important elements besides the network architecture. When we train a network with a lot of parameters, we need to show the network a proportional amount of example to get the best performance. For example, VGGNet [25] have been trained on a dataset including 1.2 million images to classify them into 1000 classes. However, data for deep learning problems are not always available in practice. In some cases, the number of images is very limited. So, we need the techniques to enlarge the data that we have. For instance, to get more data, we just need to make minor alterations to an exsting dataset such as flips or translation or rotation, ... Of course, the network will think these are distinct inputs. More specifically, a CNN can be invariant to translation, viewpoint, or illumination. So, we can apply different techniques than transform procedures to generate new dataset.

Come back with our problem, the main purpose is to predict the landmarks on pronotum images. If we apply the usual augmentation methods (i.e. translation or rotation), we need to re-point the manual landmarks before using. Additional, our model has been designed to work on the input of a channel. So, we have decided to work on the channels of the image to augment the dataset.

Our images come from a collection of 293 beetles from Brittany lands. All the images are taken with the same camera under same conditions with a 3264×2448 resolutions. For each specific part, a set of manual landmarks has been determined by biologists. The provided dataset contains 293 pronotum images, each image with 8 landmarks provided by biologists (Figure 2). The dataset was split into a training set with 260 images (training and validation) and a testing set of 33 images. During the training, the network learned the information through a pair of *image and manual landmarks* in the training set. At the testing stage, the image without landmarks is given to the trained network and the predicted landmarks coordinates will be given as output. Figure 2 shows an example of pronotum image with its manual landmarks.

In some published networks [12, 27, 4], the maximum size of the inputs is not over 256 pixels. In our case, the resolution of the image is large, it becomes a difficulty for the computing. Before training and testing, the images are down-sampling to a new resolution of 256×192. Obviously, the landmark coordinates of the image are also scaled to suit their new resolution. Then, we have applied two another procedures to increase the number of images in the dataset.

The first procedure has been applied to change the value of each channel in the original image. According to this, a constant is added to a channel of RGB image and for each time, we just change the value of one of three channels. For example, from an original RGB image, if we add a constant $c = 10$ to the red channel, we will obtain a new image with a different profile histogram. By this way, we can generate three new RGB images from one RGB image. Figure 4 presents 3 new versions of an original image by adding a value $c = 10$ to each channel for each time.

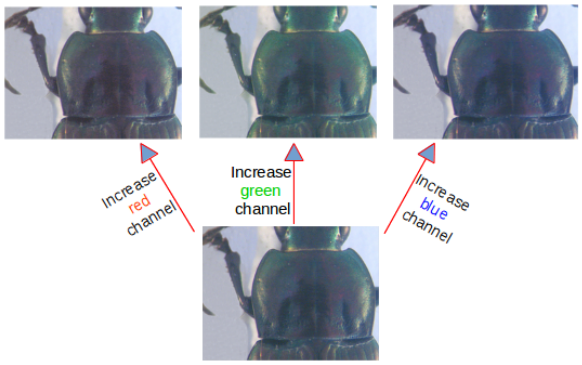


Figure 4: A constant (c=10) has been added to each channel of an original image for each time

The second procedure splits the channels of RGB images. It means that we separate the channels of RGB into three gray-scale images. Figure 5 shows the values at three individual channels of an image. At the end, we can generate six versions of original image, the total number of images used to train and validate is $260 \times 7 = 1820$ images (six versions and original image). The dataset that has been used for training and validation is split randomly by a ratio (training: 60%, validation: 40%) that has been set during the network setup.

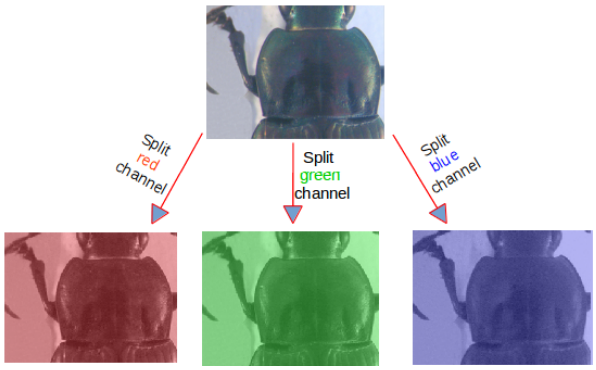


Figure 5: The individual channels have been extracted from an original image

In practical, when we work with CNN, convergence is usually faster if the average of each input variable over the training set is close to zero. Moreover, when

the input is set closed with zero, it will be more suitable with the sigmoid activation function [16]. According to [16], the brightness of the image is normalized to $[0,1]$, instead of $[0,255]$ and the coordinates of the landmarks are normalized to $[-1,1]$, instead of $[0,256]$ and $[0,192]$ before to be giving to the network.

4.2 Network architecture

Three different networks models have been proposed and trained to perform the best architecture for automatically landmarking predictions. They receive the same input of $1 \times 256 \times 192$ to train but they have different number of layers. In this section, we introduce the architectures of the networks and the process to improve the architecture from the beginning of designing.

4.2.1 The first networks

The designing begins with some basically layers in CNN. The network consists on three repeated-structure of a convolutional layer followed by a maximum pooling layer. Figure 6 shows the architecture of the first model which is a very classical one. The depth of convolutional layers increases from $32, 64$, and 128 with different sizes of the filter kernels: 3×3, 2×2, and 2×2. All the kernels of pooling layers have the same size of 2×2. The kernel sizes are classical as the literature. At the end, three full connected layers have been added to the network. The outputs of the full connected layers are $500, 500$, and 16, respectively. The output of the last full-connected layer corresponds to 8 landmarks (x and y coordinates) which we would like to predict. The training result shows that the architecture of this model provides overfitting and so is not good enough to solve the problem.

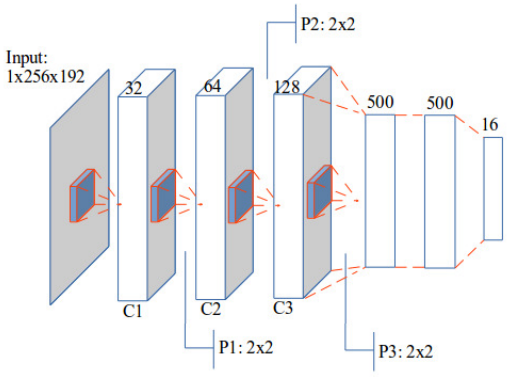


Figure 6: The architecture of the first network

The second network has been improved from the first architecture by modifying the output of the full-connected layers from 500 to 1000. The goal of this change is to have more connections to all activations in the previous layer and to prevent the overfitting but the result did not lead to better performances.

In the next step to lead to the performance of the network, instead of changing the parameters, we have decided to add some layers into the model. Srivastava et al. [26] suggest to use dropout sequence to correct overfitting artifacts. This is considered as the good solution to prevent the overfitting. The idea of dropout is to randomly drop units from the neural network during training. It prevents units from co-adapting too much. During training, dropout samples are done from an exponential number of different "thinned" network. At test times, 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. This significantly reduces overfitting and gives major improvements over other regularization methods [26]. After several testing, we have decided to keep the concept of *"elementary blocks"* to compose and to create the network.

4.2.2 Elementary block and selected architecture

An *elementary block* is defined as a sequence of convolution (C_i), pooling (P_i) and dropout(D_i) layers that can be repeated several times before to achieve the computation with the full-connected layers. For our purpose, we have assembled 3 *elementary blocks* in our model (see Fig.7). The parameters for each layer are as below, the list of values follows the order of *elementary blocks*:

- CONV layers:
  - Number of filters: $32, 64$, and 128,
  - Kernel filters size: $(3 \times 3), (2 \times 2)$, and (2×2),
  - Stride values: $1, 1, 1$,
  - No padding is used for CONV layers.
- POOL layers:
  - Kernel filters size: $(2 \times 2), (2 \times 2)$, and (2×2),
  - Stride values: $2, 2, 2$.
  - No padding is used for POOL layers.
- DROP layers:
  - Probabilities: $0.1, 0.2$, and 0.3.

In the last full-connected layers (FC), the parameters are: FC1 output: 1000, FC2 output: 1000, FC3 output: 16. As usual, a dropout layer is inserted between FC1 and FC2 with a probability equal to 0.5. Actually, we keep the same value for the parameters of the convolutional ($32, 64$, and 128), pooling (3×3, 2×2, and 2×2) and full-connected layers ($1000, 1000$, and 16) as the second one.

4.3 Training

The proposed model has been trained with $1,820$ image in 5000 epochs[1]. To obtain the predicted landmarks on all images, we have applied cross-validation technique to select the training data in the training process. During the training, the network is designed with a small sharing learning rate and a momentum. As usual, these parameters have been used to perform gradient descent during backward phase to update the parameters of the layers. The value of learning rate and momentum are updated over training time to fit with

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

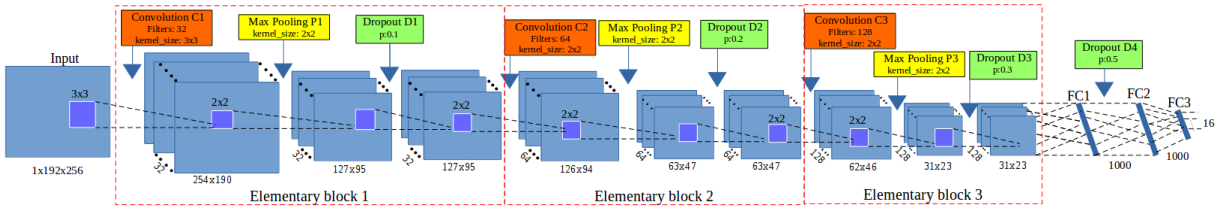


Figure 7: The architecture of the last network

the remaining number of iterations: the momentum value has been adjusted in a range of $0.9 \rightarrow 0.9999$ and the learning rate value has been adjusted from 0.03 to 0.00001.

The implementation of this architecture used Python on Lasagne framework [7] which allows to train the network on GPU. The training process took around 3 hours using NVIDIA TITAN X cards. The design of the network is available on GitHub[2].

4.4 First results

Usually, the first performance of a CNN is appreciated from the loss values in training and validation steps. In the context of deep learning, landmark prediction can be seen as a regression problem. Therefore, to evaluate the results, we have used root mean square error (RMSE) to compute the accuracy of the implemented architecture.

Figure 8 and 9 show the training errors and the validation errors of a training time on the first and the third model, respectively. The blue curves present RMSE on training dataset, the green curves present the validation errors. Clearly, the overfitting has appeared in the first model. In Figure 8, we can see that if the training is able to decrease with the number of epochs, it is not the case of validation loss. At the opposite in the third model, we can see some different values for the two losses at the beginning but after several epochs, these values become more proximate and the overfitting problem has been solved.

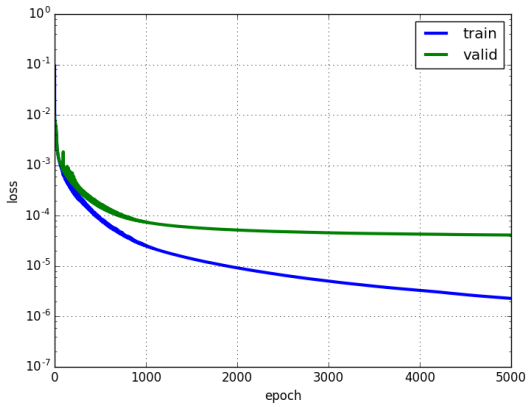


Figure 8: Learning curves of the first model.

[2]It is freely obtained by request the authors.

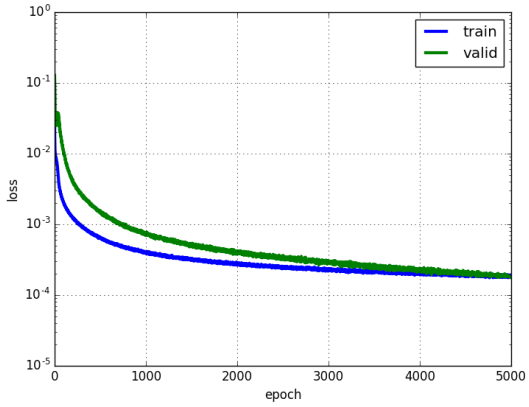


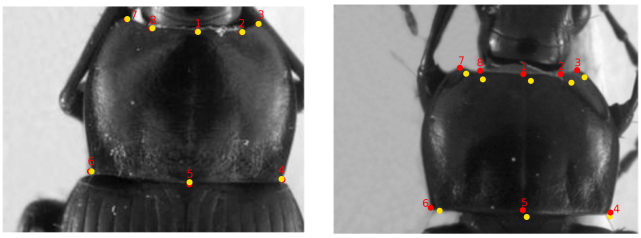
Figure 9: Learning curves of the last model.

To have the correct view about the predicted landmarks, we would like to evaluate the results in other views by using the manual landmarks coordinates (which have provided by the biologists) as ground truth. Firstly, we have considered the results in a statical point of view. We have computed the quality metric to measure the linear cofficient between the coordinates of predicted and manual landmarks. The scores have been calculated based on 3 quality metrics: coefficient of determination (r^2), explained varicance (EV), and Pearson correlation (using *scikit-learn* [8]) like Cintas et al. [4] have done. Table. 1 shows the comparison between our and their scores. From the scores, we can see that the quality of predicted coordinates are very precise.

Metric	$\mathbf{r^2}$	**EV**	**Pearson**
Cintas et al	0.884	0.951	0.976
Our score	**0.9952**	**0.9951**	**0.9974**

Table 1: The cofficient on the quality metrics

Then, we have considered the results on the point view of image processing. Figure 10 display the landmarks on the images. The red points are manual landmarks and the yellow points are predicted landmarks. The landmarks in Figure 10a are a well-prediction, while they are less accurated in the case of Figure 10b.



(a) Image with well-predicted landmarks

(b) Image with fewer accuracy landmarks

Figure 10: The predicted landmarks on an image in test set (yellow points)

To have a correct assessment of predicted coordinates, we have calculated the distances between the predicted and corresponding manual landmarks in all images. Then, we have computed the average distance per landmark as presented in Table. 2. The minimum distance is around 4 pixels in the 1^{st} landmark, while the worst distance is more than 5 pixels in the 6^{th} landmark. The other distances are approximate 4.5 pixels.

#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 2: The average distance per landmark

In other view, we have computed the standard deviation [1] which is used to quantify the dispersion of a set of distances. For example, Figure 11 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 6.5 pixels (average distance plus standard deviation): 28.33%; and the images have the distance greater than 6.5 pixels: 15.02%.

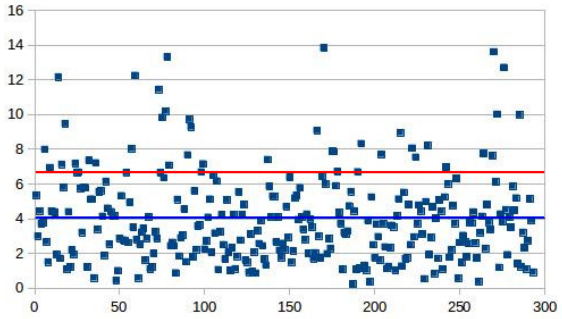


Figure 11: The distribution of the distances on the first landmark. The blue line is the average value, the red line is standard deviation value.

Figure 12 shows the proportion of acceptable landmarks, i.e. a predicted landmark is acceptable if the distance between it and the corresponding manual one is less than the average distance plus the standard deviation value. Most of the landmarks have been detected with an accuracy greater than 75%.

In the image processing point of view, the distance in 4 pixels (1^{st} landmark) is still high, it could be considered as not enough precise. Morever, when we consider the distribution of the distance in the 1^{st} landmark, the percent rate of the distance less than average value is not enough good. It means that we have less the predicted landmarks closed to the manual ones as we expect (like in the case of Figure 10b). So, the next step has been decided to improve these results.

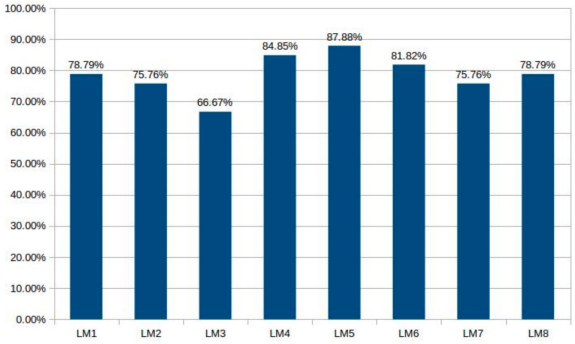


Figure 12: The proportion of acceptable predicted landmarks

5 Result improvement by using fine-tuning

As we have discussed in previous sections, the proposed network has been trained from scratch on the pronotum dataset. Even if the strengh of the correlation seems to validate the results, when the predicted landmarks are displayed on the image, the results are not enough precise since the average error is still high (≥ 4 pixels).

In order to reach more acceptable results for biologists, we have broadened the model with a step of transfer learning. That is a method that re-uses model developed for a specific task/dataset to lead another task (called *target task*) with another dataset. This allows rapid progress and improves the performance of the model on the target task [33]. The most popular example has been given with the project ImageNet of Google [6] 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 [19]. The name of this procedure to re-use parameters to pre-train a model is currently called *fine-tuning*.

In practical, pre-trained a network in thousand epochs is very time-consuming. Fortunately, we can easily get the pre-trained model from the sharing of the community of deep learning frameworks, *for example*, **Model Zoo** in Caffe framework, or *Keras Application* in Keras framework, ... The implementation and pre-

trained weight that are provided, are popular on some well-known network, i.e. VGG, Inception, or RestNet, ... on the common dataset like the ImageNet or CIFAR.

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 back-propagation. Unfortunately, we have tried to fine-tune our data on some well-known pre-trained model such as VGG-16, VGG-19, or RestNet50 that re-using ImageNet features but the result has not been relevant for our application. Therefore, we have decided to design 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. pronotum, body and head. Then, the trained model will be used to fine-tune and test on pronotum set.

5.1 Training data preparation

As we have mentioned, the training dataset includes a combination of the images from three sets: pronotum, body, and head (Figure 13). For each set, 260 original images have been chosen randomly for training and validation. By applying the same procedure in section 4.1, the training dataset was enlarged to 5,460 images ($260 \times 7 \times 3$). However, another problem has appeared that is the number of manual landmarks on each part is different: 8 *landmarks on pronotum part*, 11 *landmarks on body part*, *and* 10 *landmarks on head part* (Figure 13). Because of the manual landmarks have a specific meaning for the biologists. So, we cannot insert the landmarks arbitrary. Instead of to do that, we keep the smallest number of landmarks among the three parts as a reference and remove the supernumerary when it is needed. Specifically, we have removed three landmarks on the body part ($1^{st}; 6^th; 9^{th}$) and two landmarks on the head part ($5th; 6^{th}$) (Figure 13).

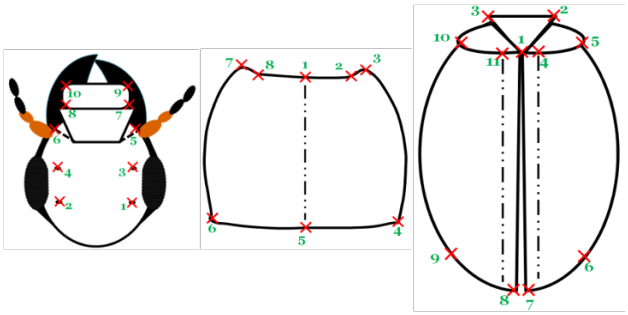


Figure 13: A presentation of head, pronotum and body part with corresponding manual landmarks

5.2 Using fine-tuning for pronotum dataset

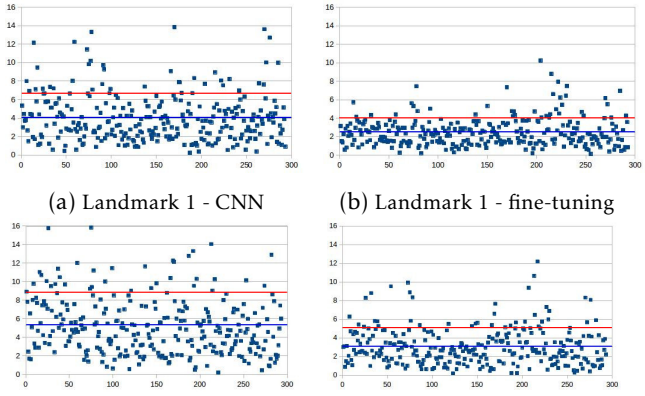
At the first step, the network is trained with 5,460 images following the same way as previously. After that, this pre-trained model is used to fine-tune the pronotum dataset. To compare the result with the previous one (training from scratch), the trained model has been fine-tuned on pronotum images with different cross-validation.

At the end of test stage, landmarks are predicted on all images in pronotum dataset. The average distances have been computed again. Table 3 shows a comparison of average distance values and standard deviation values from two studying processes: with and without fine-tuning. The **Average** *from scratch* column reminds the average distance obtained previously. The **Average** *with fine-tuning* column presents the new average distance after fine-tuning the pronotum from the trained model. Besides, the standard errors of both cases have been presented (SD columns). It is clearly shown that the result of predicted landmarks with the help of fine-tuning is more precise than the first way to do it. For example, if we consider the 1^{st} landmarks, the average have been decreased from 4.002 to 2.486 (app. 40%) and the SD value have been decreased also. Even the worst average distance with fine-tuning (3.0492 pixels at the 6^{th} landmark) is still smaller than the best case with training from scratch (4.002 pixels at the 1^{st} landmark).

Landmark	From scratch		With fine-tuning	
	Average	SD	Average	SD
LM1	**4.002**	**2.5732**	**2.486**	**1.5448**
LM2	4.4831	2.7583	2.7198	1.7822
LM3	4.2959	2.7067	2.6523	1.8386
LM4	4.3865	3.0563	2.7709	1.9483
LM5	4.2925	2.9086	2.4872	1.6235
LM6	**5.3631**	**3.4234**	**3.0492**	**1.991**
LM7	4.636	2.8426	2.6836	1.7781
LM8	4.9363	3.0801	2.8709	1.9662

Table 3: A comparing between the average error distances, the standard deviation values per landmark of two steps.

To illustrate the final results, we display the distribution of the distance of both the best and the worst results (resp. landmark 1 and 6). The Figure 14 shows in (a) and (b) diagrams how much the average distances (blue lines) and standard errors (red lines) have been improved for the landmark 1, the (c) and (d) diagrams for the landmark 6. The x and y axes are the number of images and distance in pixels, respectively. Each point in diagram presents to the distance between predicted and corresponding manual landmark. In the diagrams, the range of the distances, when we train the model from scratch, is wider than the range of the distances when we fine-tune the pre-trained model, i.e. in the case of training from scratch, the range of the 1^{st} landmark is $[0:14]$ and $[0:16]$ for the 6^{th} landmark, while these values in fine-tuning process are $[0:10]$ and $[0:12]$, respectively. Because of the points in diagram represent for the distances, so if they are presented in the range from 0 to average value, are considered as a good result. We can see both in the best and the worst cases, the distances have been decreased and the errors have been reduced with fine-tuning.



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

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

Figure 14: The distribution of distance error on 1^{st} and 6^{th} landmarks of all images in two testing steps (CNN and fine-tuning). The blue and red lines present the average distances and standard deviation values, respectively.

As result files, the program outputs the predicted landmarks of the images as TPS files (a standard file format for morphometric analysis softwares). With the outputs as TPS files, the user can use MAELab framework[3] to display the landmarks on the images.

6 Conclusion

In the context of collaboration with biologists, the project towards replacing the manual landmarks task by automatic ones on beetle's pronotum images. The pronotum images are difficult to segment, so methods which are not supposed to be based on segmentation are necessary. In this paper, after testing several models, we have presented a convolutional neural network for automatic detection landmarks on this case. It includes three times repeated structure of *"elementary block"* which consists of a convolutional layer, a max pooling layer, and a dropout layer, followed by the connected layers. During the training stage, suitable techniques are used to prevent overfitting, a common issue of the neural networks. The network was trained several times in different selections of training data. After training with the manual landmarks given by the biologist, the network is able to predict the landmarks on the set of unseen images.

In our case, the training dataset is limited. So, we have applied some techniques to augment the dataset. The results from the test set have been evaluated by calculating the distance between manual landmarks and corresponding predicted-landmarks. The average of distance errors on each landmark has been also considered. Additional, a statistic on acceptable predicted landmarks has been computed with an accuracy greater than 75%. In order to improve the results, the model has been trained on a dataset including the images of all three parts of beetles. Then, the trained model has been used to fine-tune and to test on pronotum images.

These results have 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 manual landmarks.

Currently, we are applying the network on other parts of beetle, i.e. head and elytra part and the results seem to be good also. Besides, after finding that the well-known pretrained model do not accord to our problem, the next step is to study more deeply how to characterize the learning problem to design the right pretrained model depending the class of the problem.

Acknowledgment The research has been supported by DevMAP project[4]. We would like to thank our colleague, ALEXIA Marie, who have provided manual landmarks on beetle images.

References

[1] J Martin Bland and Douglas G Altman. Statistics notes: measurement error. *Bmj*, 313(7059):744, 1996.

[2] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*, 2014.

[3] François Chollet et al. Keras, 2015.

[4] Celia Cintas, Mirsha Quinto-Sánchez, Victor Acuña, Carolina Paschetta, Soledad de Azevedo, Caio Cesar Silva de Cerqueira, Virginia Ramallo, Carla Gallo, Giovanni Poletti, Maria Catira Bortolini, et al. Automatic ear detection and feature extraction using geometric morphometrics and convolutional neural networks. *IET Biometrics*, 6(3):211–223, 2016.

[5] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] Sander Dieleman, Jan Schlter, Colin Raffel, Eben Olson, Sren Kaae Snderby, Daniel Nouri, et al. Lasagne: First release., August 2015.

[8] Pedregosa et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[9] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[10] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[3] MAELab is a free software written in C++. It can be directly and freely obtained by request at the authors.

[4] https://www6.rennes.inra.fr/igepp_eng/Researchteams/Demecology/Projects/INRASPEDevMAP

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] Van Linh LE, Marie BEURTON-AIMAR, Adrien KRÄHENBÜHL, and Nicolas PARISEY. MAELab: a framework to automatize landmark estimation. In *WSCG 2017*, Plzen, Czech Republic, May 2017.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[15] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE, 2010.

[16] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[17] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015.

[18] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[19] Jan Margeta, Antonio Criminisi, R Cabrera Lozoya, Daniel C Lee, and Nicholas Ayache. Fine-tuned convolutional neural nets for cardiac mri acquisition plane recognition. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 5(5):339–349, 2017.

[20] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 196–201. IEEE, 2011.

[21] Sasirekha Palaniswamy, Neil A Thacker, and Christian Peter Klingenberg. Automatic identification of landmarks in digital images. *IET Computer Vision*, 4(4):247–260, 2010.

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[23] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8614–8618. IEEE, 2013.

[24] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[27] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3476–3483, 2013.

[28] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[30] DJD Team. Deeplearning4j: Open-source distributed deep learning for the jvm. *Apache Software Foundation License*, 2, 2016.

[31] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[32] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014.

[33] Lisa Torrey and Jude Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009.

[34] Dong Yu, Adam Eversole, Mike Seltzer, Kaisheng Yao, Zhiheng Huang, Brian Guenter, Oleksii Kuchaiev, Yu Zhang, Frank Seide, Huaming Wang, et al. An introduction to computational networks and the computational network toolkit. *Microsoft Technical Report MSR-TR-2014–112*, 2014.

[35] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer, 2014.