Deep Learning for landmarking on morphometry anatomical images

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 Libération, 33405 Talence, France*
[b] *UMR 1349 IGEPP, BP 35327, 35653 Le Rheu, France*
[c] *Dalat University, Dalat, Lamdong, Vietnam*

Abstract

Deep learning has been introduced in the middle of the previous century for artificial intelligence, and in recent years, it has risen strongly because of improvements in the computation performance. It has been applied to solve the problems in different domains such as computer vision, speech recognition, or languages translation. Among different types of deep learning architectures, convolutional neural networks have been most often used in computer vision for image classification, object recognition, or key points detection and they have brought amazing achievements. In this work, we propose a convolutional neural network model to predict the key points (landmarks) on 2D anatomical biological images, specifically beetle's images. Our proposed network is trained and evaluated on a dataset including the images of 293 beetles. During the experiments, the network is tested in two ways: training from scratch and applying fine-tuning process. The quality of predicted landmarks is evaluated by comparing with the manual landmarks which have been provided by the biologists. The obtained results have been considered by the biologists statistically good enough to replace the manual landmarks for the different morphometry

*Corresponding author
Email addresses: van-linh.le@labri.fr (Le Van Linh), beurton@labri.fr (Beurton-Aimar Marie), zemmari@labri.fr (Zemmari Akka), nicolas.parisey@inra.fr (Parisey Nicolas)
[1] both authors contributed equally to this work.

analysis.

Keywords: Deep learning, CNN, fine-tuning, landmarks

1. Introduction

Deep learning is a part of machine learning domain. Computational model of deep learning is composed of multiple layers to learn data representation. Each layer extracts the representation of input data which comes from the previous layers, then it will compute a new presentation for the next layer. In a deep learning model, each layer may contain different number of nodes, called *neurons* which have been inspired from the biological neural system [1]. Currently, deep learning has many kind of variant architectures and each of them has found success in different domains, for example, Deep Neural Network (DNN) can be applied to solve the classification or data analysis problems[2, 3]; Convolutional Neural Network (CNN) is widely applied in computer vision [4, 5, 6]; Recurrent Neural Network (RNN) give the best performance on time sequences analysis [7, 8, 9, 10].

In deep learning architectures, CNN is a specific network for pre-processing data which have grid topology, for examples, time series (1-D), images (2D), or 3D data. From the first architecture [4] until now, many CNN architectures have been proposed and have succeeded in different tasks of computer vision such as image classification [4, 5, 6], object recognition [6, 11, 12], and key points detection [13, 14, 15, 16].

In computer vision, key points detection is an important field. In this field, algorithms try to find the key points (called points of interest (PoI) or landmarks) through images. The landmarks are considered as the points in the image that are invariant when the image changes e.g. by applying some morpholigical transformation. They are most often provided by the biologists. Depending on the object, the number of landmarks may be different, as well as

their position can be defined along the outline of the object or inside the object. These landmarks are used in different objectives of different domains, for example, landmarks are basic characteristics to detect the human face or human pose; in biology, the topology of the objects in an organism can be measured from the location of landmarks.

In this work, we propose a CNN architecture to predict the landmarks on biological anatomical images. The proposed model has been trained on a dataset including the images taken from 293 beetles. We have also conceived a specific procedure to augment our dataset because several hundred images are usually considered as a modest number for applying deep learning methods. After applying our model, the biologists have asserted that the predicted landmarks which have been provided by our model, were enough good to replace the manual landmarks.

This paper is organized as followed: Section 2 discusses the related works about working on deep learning and determining the landmarks on 2D images. Then, a short overview about CNN and its components will be introduced in Section 3. Section 4 presents the approaches to augment the dataset. Section 5 explains the design of new network model. The first experiments of the network on each dataset are presented in Section 6. In the last section, Section 7, we presents a technique to imrpove the results of proposed model: fine-tuning.

2. Related works

In the middle of the previous century, deep learning [9] have been introduced as a method for artificial intelligence applications. However, several problems have appeared when taking into account it into real-world cases because of the limitation of memory size or computing powerfull. Nowadays, improvements of computing capacities, both in memory size and in computing time with GPU

programming, have opened a new perspective for deep learning. In recent years, deep learning architectures have achieved remarkable accomplishments in many tasks of different domains such as **computer vision** [4, 5, 6, 11, 12], **speech recognition** [3, 2], **language translation** [7, 8], **natural language processing** [9, 10, 17], ... In computer vision, deep learning, specifically with CNN, has been used to study difficult tasks in image analysis such as image classification, object or key points detection. LeNet [4] model is considered as the first architecture of CNN. LeCun et al. [4] have used LeNet to classify the handwritten digits in cheques. LeNet exhibits a standard architecture of a CNN which consists of convolutional layers, pooling layers, followed by one or several fully connected layers. Unfortunately, to be applied to realistic problems, this model requires huge computation capacities and a large amount of training data which are not available at the early 2000s. But in the last ten years, capabilities to computing have been drastically improved and in the same time, a huge amount of data are produced without tools to analyze them in a lot of domain. New generate of the neural network appears well adapted to this new environment. One of the first ones is AlexNet [5], which is similar to LeNet [4] but it has a deeper structure: LeNet has 2 convolutional layers and 1 fully connected layer while AlexNet has 5 and 3, respectively. Besides, AlexNet has been modified the parameters such as changing the activation function and dropout layers have been added to prevent the over-fitting. AlexNet won the famous ImageNet Challenge[2] in 2012. From the success of AlexNet, a lot of different models have been proposed to improve the performance of CNN, one can cite ZFNet [18], GoogLeNet [6], VGGNet [19], or ResNet-50 [20]. The main difference between these networks is that their architectures become deeper and deeper by adding more layers, e.g. ResNet-50, which won the champion of ILSVRC 2015, is deeper

[2]This is a challenge where evaluates algorithms for object detection and image classification.

than AlexNet around 20 times.

Besides classification or recognition of objects, CNNs have been also used to detect key points inside images. Liu et al. [13] have presented a method to predict the positions of functional key points on fashion items such as the corners of neckline, hemline and cuff. Yi Sun et al. [14] have proposed a CNNs cascade to predict the facial points on the human face. Their model contains several CNNs which are linked together in a list as a cascade. Three levels of the cascade are set to recognize the human face from the global to local view with the objective to increase the accuracy of predicted key points. In the same topic, Zhanpeng Zhang et al. [15] have proposed a *Tasks-Constrained Deep Convolutional Network* to join facial landmarks detection problem with a set of related tasks, e.g. head pose estimation, gender classification, age prediction, or facial attribute inference. In their method, the input features have been extracted by 4 convolutional layers, 3 pooling layers and 1 fully connected layer which is shared by the multiple tasks in the estimation step. Shaoli Huang et al. [21] have introduced a coarse-fine network to locate the keypoints and to estimate human poses. Their framework consists of the base convolutional layers shared by two streams of keypoint detectors: The first stream, named coarse stream, includes 3 detector branches (3 stacks of Inception modules [6]) which are used to focus on capturing local characteristics and modeling the spatial dependencies between the human parts. The second one, named fine stream, receives features which are concatenated from the coarse stream and provides the accurate localization. Cintas et al. [16] have introduced an architecture which is enable to recognize 45 landmarks on human ears. Their proposed model includes a structure with 2 convolutional layers, 1 pooling layers, and 1 dropout layer to extract the features. This structure is repeated 3 times and is followed by 3 fully connected layers. In the same context of key point detection, we have

developed a CNN to automatize landmarks prediction on beetle's anatomies.

As our work consists of a new architecture proposition for CNN, we have chosen to give in the next section some details about the definition of the different types of layers in CNN. The reader familiar to CNN can jump directly to the section about data augmentation.

3. Overview of Convolutional Neural Network

A CNN is a feedforward network which takes the information following one direction from the inputs to the outputs. Currently, CNNs have many different variations, but in general, it consists of convolutional and pooling layers which are stacked together to convolve and to down-sample the inputs. Then, they are followed by one or more fully connected layers to give the decision as the output of the network.

Fig. 1 shows a classical example of a CNN for classification problem. The network inputs directly an image to several stages of convolutional and pooling layers. Then, the representation is feed into three fully connected layers. A dropout layer is inserted after the second fully connected layer (it is represented by some blue nodes). Finally, the last fully connected layer gives the category label for the input image. This architecture could be seen as the most popular one [4, 5]. Now, we will describe the different types of layers.

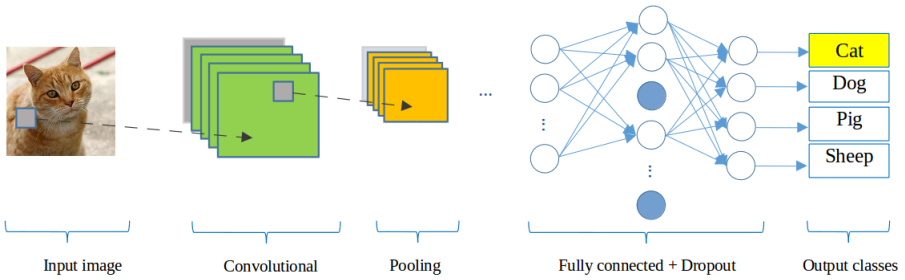


Figure 1: A CNN network for classification problem

Convolutional (CONV) layer: uses as a feature extractor by applying some learnable weights (filters) on the input images. The input image is convolved with the filters in order to compute the new feature maps; then, the convolved results are sent through a nonlinear activation before sending to the next layer. In CONV layer, the neurons are arranged into feature maps. All the neurons within a feature map have the same constraints; however, different features maps within the same CONV layer have different weights so that several features can be extracted at each location of an input image. This step is similar to apply a filter operation in image processing.

Pooling (POOL) layer: is mostly used to down-sampling the size of the input with the purpose to reduce the spatial resolution of the feature map and so to reduce the computation cost. Initially, it was common practice to use average pooling which propagates the average of all the inputs to the next layer. However, in more recent models [5, 22, 12], maximum pooling has been prefered. It propagates the maximum values of the inputs to the next one. Fig. 2 illustrates the differences between maximum and average pooling: Giving an input image of size (4×4), if applying a filter with size of (2×2) and stride of 2, the outputs will have the same size in both of case (2×2) but the values are different. For example, if we apply maximum pooling at yellow region, the output value is 122; but if we use average pooling, the output is 36.25.

Dropout (DROP) [23] is a technique use to prevent the over-fitting in a neural network. The term dropout mentions dropping some out units and their connections (incoming and outgoing) belong to a layer in the network. The units are dropped randomly with a probability p. When applying dropout technique, the network becomes a collection of thinned networks [23] because a number of units are dropped randomly at each presentation of training phase. So, training a neural network with dropout looks like training a collection of

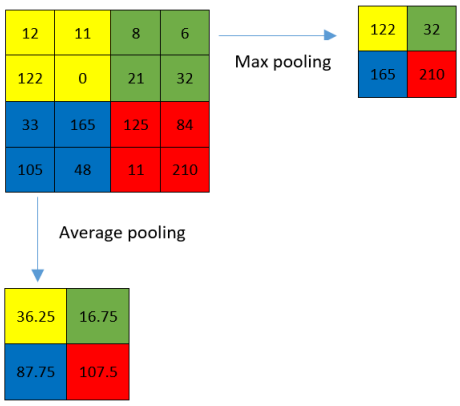


Figure 2: The results of different pooling

thinned networks. The dropouts layers are most often placed after the fully connected layers, but it is possible to use them after the pooling layers to create some kind of images noise augmentation.

Fully connected (FC) layer: usually follows the group of convolutional and pooling layers to extract the abstract feature representations of the image. A CNN may have one or several FC layers. They interpret the feature representations (their inputs) and perform a function of high-level reasoning by applying the activation functions. In practice, the last fully connected layer produces the output of the network and choose the activation function is depended on which kind of problem that network solves.

From AlexNet period to ResNet-50, the obtained success stories [5, 20] have proved that CNN models produce better results on a large dataset. To use this technique, the size of dataset remains as a bottleneck. Before to describe our proposed network architecture, we describe in the next section a way to augment data dedicated to our work.

4. Data augmentation

The fundamentals of deep learning algorithms are to train the models on dataset repeatedly in order to reach the best accuracy. So, providing a large dataset asserts to learn more cases and clearly improves the learnable of the network. Unfortunately, in some application domains as in biology, providing large dataset is costly and could be difficult to obtain. For this reason, one way to solve this problem is to create misshapen data from real data and to add them to the training set. Most often in image processing, dataset augmentation uses the operations like translate, rotate or scale the images, which are known as efficient in image classification problem. However, these kinds of operations are not useful in our case because the analysis of images by CNN (convoluted) is most often invariant with translation or rotation. So, we have designed another method to obtain misshapen images.

Our image set are in RGB color map, the first procedure consists of changing the value of one color channel of the three in the original image to generate the new image. A constant values is sampled in a uniform distribution $\in [1, N]$ to obtain a new value caped at 255. For example, Fig. 3 shows an example when we added a constant $c = 10$ to each channel of an original image. Following this way, we can generate three new versions of one image.

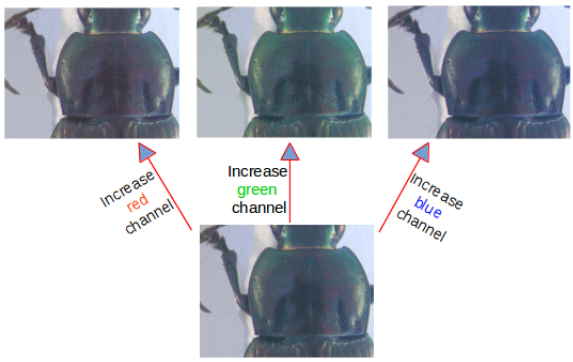


Figure 3: A constant $c = 10$ has been added to each channel of an original image

In the second procedure, we consider each channel separately and we generate one gray image for each channel (Fig. 4). Consequently, we obtain 3 new images (single channel) from an original image. At the end of the processes, we are able to 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).

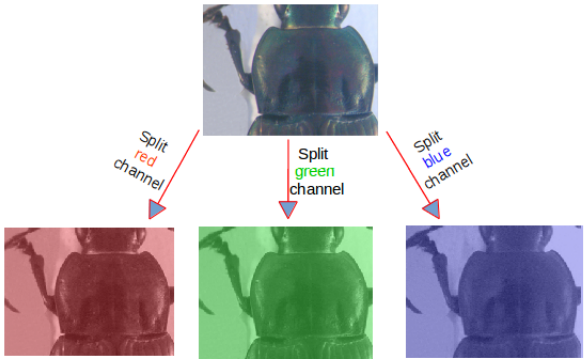


Figure 4: Three channels (red, green, blue) are separated from original image

5. Network architectures designing

From the classical architecture of CNN that we have presented (LeNet), it is possible to use some classical layers to build a CNN model. Then, the model architecture will be improved to increase its efficiency for a specific task by changing the parameters values or by modifying the arrangement of new layers. Several trials have been achieved before to obtain the satisfying model for landmarks estimation. In this section, we present three versions of models that we have designed to solve landmarks estimated task. 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 input image with the size of $(1 \times 192 \times 256)$. Then is three repeated structures of a convolutional

(CONV) layer followed by a maximum pooling (POOL) layer. Most of CNNs, the hyperparameters of CONV 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 CONV layer. So, the depths of CONV layers increase from $32, 64$, and 128 with different size of the kernels: (3×3), (2×2) and (2×2), respectively. Inserting POOL layers after a convolutional layers is a common periodcally. The POOL 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 operation of POOL layers is independent on every depth slice of the input. The most common form is a POOL 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. Therefore, all the kernels of maximum POOL layers have the same size of (2×2) with a stride of 2 as usual. At the end of the model, three full-connected (FC) layers have been added to extract the global relationship between the features and to procedure the outputs. The first of two FC 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 FC layers are $500, 500$ and 16. The output of the last FC layer corresponds to the coordinates (x and y) of 8 landmarks which we would like to predict. Fig. 5 shows details of the first model: The orange rectangles represent for CONV layers while the yellow rectangles represent for maximum POOL layers and three FC layers with their parameters are presented at the end of the model. Nevertheless, the obtained results (Section 6) have proved that this architecture cannot solve the problem, the over-fitting has appeared during the training process. So, we need to modify this architecture to obtain better results.

The second architecture is modified from the first model. The layers are

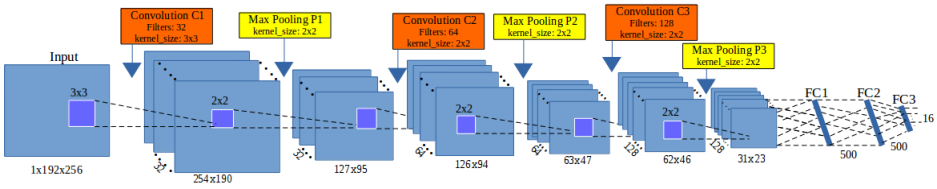


Figure 5: The architecture of the first model

kept the same as the first one but the outputs of the first of two FC layers are changed from 500 (in the first model) to 1000 (Fig. 6). Increasing the value at FC layers is hoping to obtain more features from CONV layer to give the decision without requirements of a lot of new capacity resources. However, the obtained results have not been satisfying, it will be discussed in the last section (Section 6).

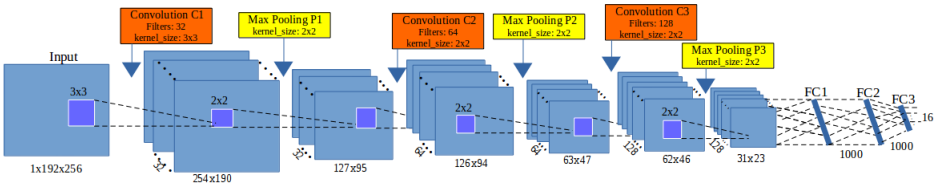


Figure 6: The architecture of the second model

To build the third architecture, we have chosen to define an *elementary block*. An elementary block is defined as a sequence of a CONV (C_i), a maximum POOL (P_i) and a dropout (D_i) layers (Fig. 7). In this case, we use another technique to prevent over-fitting: *dropout*. This significantly reduces overfitting and gives major improvements over other regularization methods [23] by including some variations among different runs. So, we have modified the architecture by combining some *elementary blocks*.

Fig. 8 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 described as below, the list of values follows the order of

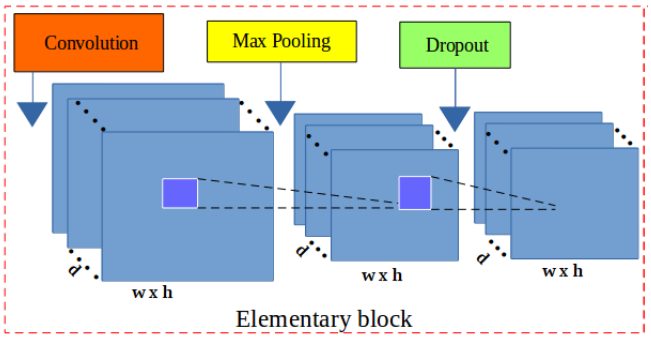


Figure 7: The layers in an elementary block. It includes a CONV layer (red), a maximum POOL layer (yellow) and a DROP layer (green).

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 FC layer (FC3) has 16 outputs. As usual, a dropout layer is inserted between FC1 and FC2 with a probability equal to 0.5.

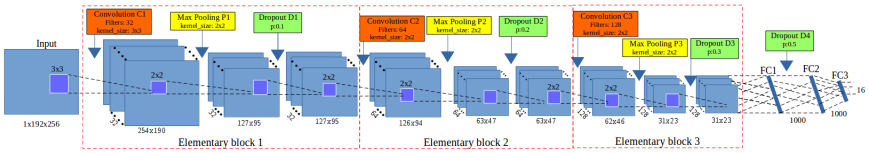


Figure 8: The architecture of the third model

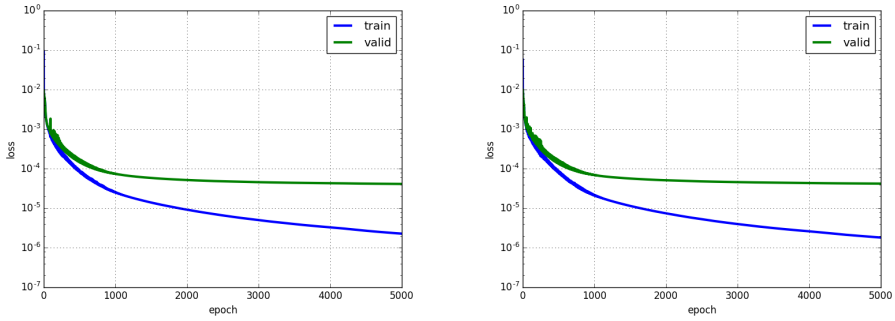
The core of CNN is training over iteration. There are many ways to optimize the learning algorithm, but gradient descent [24] is currently a good choice to establish the way of optimizing the loss in neural network. The core idea is following the gradient until we statify with the results will remain 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 [3]. The implementation of the architectures have been done on Lasagne framework [25] by Python code. More information about the model can be obtained from the repository on GitHub: `https://github.com/linhlevandlu/CNN_Beetles_Landmarks`

6. Experiments and results

Before widely applying to all anatomical parts, we have firstly tried with pronotum part to evaluate the performance. The networks have been trained in $5,000$ epochs on Ubuntu machine by using NVIDIA TITAN X cards. 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. The

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

manual landmarks coordinates had been provided by the biologists and they had been considered as ground truth for evaluation. In deep learning, many kinds of loss expressions have been provided. Using which kind of loss is depending on the class of problem that the network solves, e.g. cross-entropy loss is usually used in a classification problem while Root Mean Square Error (RMSE) is used in regression problems. In the context of deep learning, landmark prediction can be seen as a regression problem. Therefore, RMSE has been used to compute the losses of architectures during the training process.



(a) The first model (b) The second model

Figure 9: The losses (training and validation) of the models

Fig. 9a shows the training errors and the validation errors during training phase of the first architecture. The blue curve presents the RMSE errors of training process while green curve is the validation errors. Clearly, over-fitting has appeared in the first model. One can note that the training losses are able to decrease but the validation losses are stable. In the second model (Section 5), we have modified the parameters of full-connected layers to prevent the over-fitting but it seems that this solution is still not satisfying and the results are also the same with the first architecture (over-fitting is still appears).

Fig. 10 illustrates the losses during the training of the third model. As the same meaning in Fig. 9, the blue line is training loss, the green line is validation

loss. In the opposite with two previous models, the losses are different (far) from the beginning but after several epochs, the loss 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.*

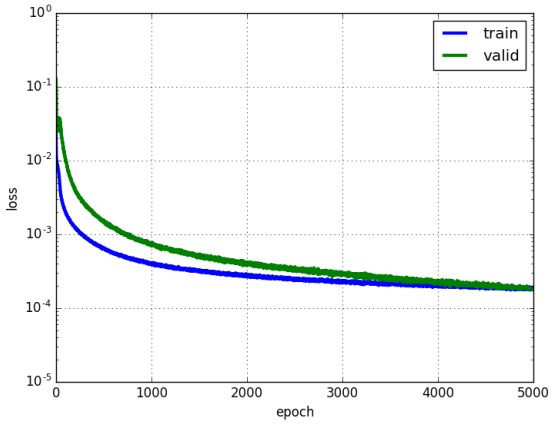


Figure 10: The losses (training and validation) of the third model

In order to have the predicted landmarks for all pronotum images (instead of only 33 images), we have applied *cross-validation* to choose the test images, called *round*. For each round, we have chosen a different fold of 33 images as testing images ($293/33 \approx 9$ rounds), the remaining images are used as training and validation images. Of course, the training and validation images have been augmented before providing to the model for training. Following that, the network has been trained with different datasets, then the trained model have been 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 pronotum images. Clearly, the training/validation loss among rounds are tiny (stable) and the RMSE values look pretty good ($\approx 1.7 - 2.1$ pixels).

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 pronotum images

To evaluate the coordinates of predicted landmarks, the correlation metrics between the manual landmarks and corresponding predicted one have been computed. Table. 2 shows the correlation scores of 3 metrics (using *scikit-learn* [26]), i.e. coefficient of determination (r^2), explained variance (EV), and Pearson correlation. These three metrics are both appropriate for our dataset type. The results closed to 1 show that the predicted coordinates are very close with the ground truth. It proves that our prediction is enough good to provide to the biologist in some cases of statistical analysis. However, standing on the side of image processing, we are looking forward to seeing the real coordinates than the statistical results. So, the distances (in pixels) between manual coordinates and predicted coordinates have been calculated on all images. Then, the averages of distances are taken into account by landmarks.

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

Table 2: Correlation scores between manual landmarks and predicted landmarks

Table. 3 shows the average distances by landmarks on all images of pronotum 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.

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 on pronotum images.

Fig. 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 10 pixels (5% acceptable errors): 40.27%; and the images have the distance greater than 10 pixels: 3.07%.

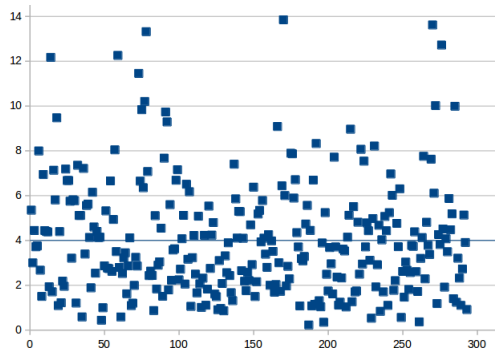
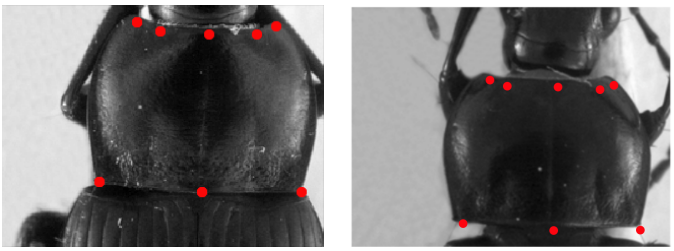


Figure 11: The distribution of the distances on the first landmark. The blue line is the average value of all distances.

To illustrate this purpose, Fig. 12 shows the predicted landmarks on two test images. One can note that even some predicted landmarks (Fig. 12a) are closed to the manual ones, in some case (Fig. 12b) the predicted ones are far from the expect results. This result explains why the average distance by landmarks are

enough good while some predicted landmarks are so far from the manual one. So, the next step has been dedicated to the improvement of these results.



(a) Image with well-predicted landmarks

(b) Image with inaccuracy landmarks

Figure 12: The predicted landmarks, in red, on the images in test set.

From the success of the third architecture on pronotum dataset, we apply the same procedures (data augmentation, training,...) on other parts of beetle: *left mandible, right mandible, elytra, and head*. However, we have modified the number of output of the last full-connected layer to adapt with each dataset before training. According, the values at the last full-connected layer are set to 32, 36, 22 and 20 outputs corresponds to 16, 18, 11 and 10 landmarks on left mandible, right mandible, 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 shows the average distances on each landmark of elytra, head, left and right mandibles anatomical, respectively. Comparing with the average distances on the pronotum part, the average distances on elytra and head parts are very close, but a little bit far on the mandible parts.

Landmark	**Distance (in pixels)**			
	Right mandible	Left mandible	Elytra	Head
1	**9.4981**	**9.1267**	**3.8669**	**5.528**
2	7.1657	**6.7198**	3.973	5.1609
3	7.242	6.8704	3.9166	5.3827
4	**7.0436**	6.7719	3.8673	5.0345
5	7.1599	7.125	4.0151	4.8393
6	7.5699	6.9441	4.8426	**4.4516**
7	7.4251	7.3158	5.2125	4.7937
8	7.6636	7.4142	**5.4685**	4.5322
9	7.7906	7.5846	5.2692	5.1412
10	8.0197	7.6349	4.0709	5.0564
11	8.314	7.6873	3.9896	-
12	8.1564	8.4248	-	-
13	8.8879	7.9983	-	-
14	9.1842	7.4919	-	-
15	8.7875	7.7903	-	-
16	8.3141	8.5198	-	-
17	8.2866	-	-	-
18	8.8928	-	-	-

Table 4: The average distances on all images per landmark on right mandible, left mandible, elytra and head images.

7. Resulting improvement by fine-tuning

The proposed network (third architecture) presented in Section 5 have been trained from scratch on five datasets of beetles (left mandible, right mandible, pronotum, 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 not enough precise, and the average errors are a little bit high.

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 parameters values obtained from a model 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 [27]. The most popular example has been given with the project ImageNet of Google [28] 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 [29]. 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 a set of images which including the training images (in one round) of three parts, *i.e pronotum, elytra, and head*. Then, the trained model has been used to fine-tune and test on each dataset.

7.1. Data preparation and training

The images are combined from the training images of three sets: *pronotum, elytra, and head* (after augmentation). Remember that we have used cross-validation to select the data during training from scratch (9 folds). It means the model has been trained over different training datasets. So, to create the dataset for pre-training, we just select the images from one fold at each dataset (after augmentation). Specifically, we have taken $1,820$ images of each part. In total, it includes $5,460$ images ($260 \times 7 \times 3$).

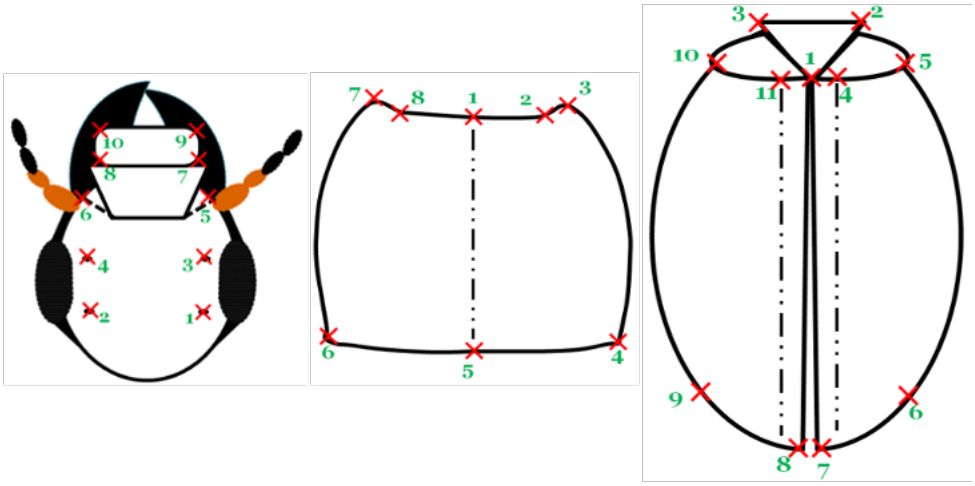


Figure 13: A presentation of head, pronotum and elytra part with corresponding manual landmarks

However, another problem has been appeared when we combined the images from different parts. That is the different number of landmarks on each part: 8 *landmarks on pronotum part,* 10 *landmarks on head part, and* 11 *landmarks on elytra part.* Fig. 13 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 pronotum 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

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. Additional, we have shuffled the training data 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.

7.2. Fine-tuning on each dataset

The combined dataset then used to train the third architecture with 16 outputs (8 landmarks). 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 by applying cross-validation. Firstly, we consider on the losses during fine-tuning. *For example*, Table. 5, 7, 9 show the losses during fine-tuning on pronotum, elytra, and head dataset, respectively. Comparing with the losses when we trained the model from scratch, *i.e.* on pronotum, the validation losses of all round in this scenario have been significantly decreased (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. Tables. 6, 8, 10, 11, and 12 show the average distances per landmark on pronotum, elytra, head, left and right mandibles 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, the average distance at each landmark has decreased. Additional, when comparing 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 5: The losses during fine-tuning model on pronotum dataset

#LM	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 6: The average error distances per landmark of two deep learning processes on pronotum images

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 7: The losses during fine-tuning model on elytra dataset

#LM	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 8: The average error distances per landmark of two deep learning processes on elytra images

In another view, Fig. 14 shows the comparison of the average distance distributions on each dataset in two procedures (from scratch and fine-tuning).

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 9: The losses during fine-tuning model on head dataset

#LM	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 10: The average error distances per landmark of two deep learning processes on head images

#LM	From scratch	Fine-tune
1	**9.1267**	**6.7655**
2	**6.7198**	**5.2952**
3	6.8704	5.3468
4	6.7719	5.332
5	7.125	5.4391
6	6.9441	5.3004
7	7.3158	5.5314
8	7.4142	5.6486
9	7.5846	5.8864
10	7.6349	5.9245
11	7.6873	5.972
12	8.4248	6.5755
13	7.9983	6.1067
14	7.4919	5.6307
15	7.7903	5.8522
16	8.5198	7.174

Table 11: The average error distances per landmark of two deep learning processes on left mandible images

#LM	From scratch	Fine-tune
1	**9.4981**	6.3236
2	7.1657	5.1347
3	7.242	5.1613
4	**7.0436**	**5.0537**
5	7.1599	5.1372
6	7.5699	5.301
7	7.4251	5.2064
8	7.6636	5.5168
9	7.7906	5.6858
10	8.0197	5.7495
11	8.314	6.1975
12	8.1564	6.1898
13	8.8879	6.7612
14	9.1842	**7.0694**
15	8.7875	6.5293
16	8.3141	6.1147
17	8.2866	6.2881
18	8.8928	6.8367

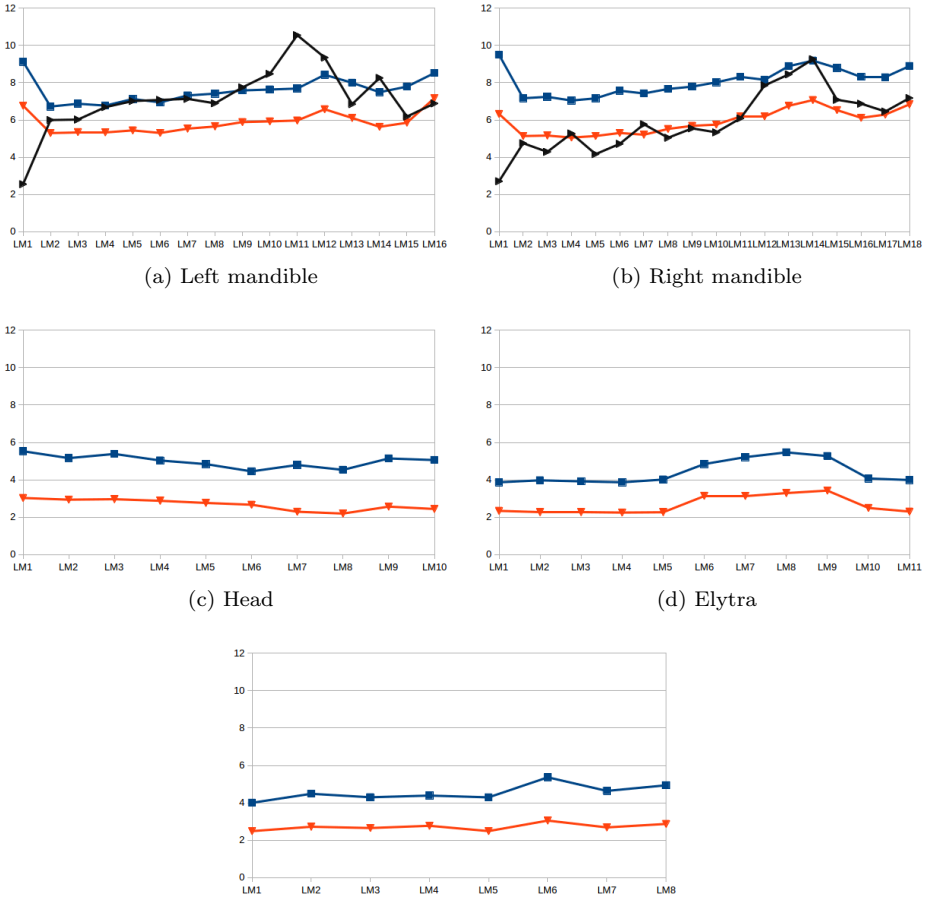
Table 12: The average error distances per landmark of two deep learning processes on right mandible images

In 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.
- **Black** curves (in the case of left and right mandibles): illuslate for the average distances when we applied the image processing procedures to predict the landmarks on segmentable images.

The fine-tuning process has improved the results of the proposed architecture on both 5 datasets: left, right mandible, pronotum, elytra and head. All the average distances are significantly decreased. Specially, the results have been improved $\approx$ 26.9% on left mandible, $\approx$ 22.8% on right mandible, $\approx$ 40.3% on pronotum, $\approx$ 39.8% on elytra, and $\approx$ 46.4% on head part based on considering the average distances per landmark. Addition, in the cases of pronotum and head part, even if we plus the average distance and its standard deviation, the results are also less than the result when we trained the model from scratch. For segmentable images, we have a comparison between the results of deep learning and early method where we have applied image processing techniques to predict the landmarks [30]. Clearly, the result with fine-tuning has improved the location of estimated landmarks. Even the average distances which obtained from scratch training are still high but they are more stable than the results from the early method: most of the average distance(or landmarks) of left mandibles are less than the results of the early method, while the average distances are very closed in the case of right mandibles.

To compare the results between image processing procedures and deep learning, we have run the test on an image of all parts in both methods. Then, we



(a) Left mandible

(b) Right mandible

(c) Head

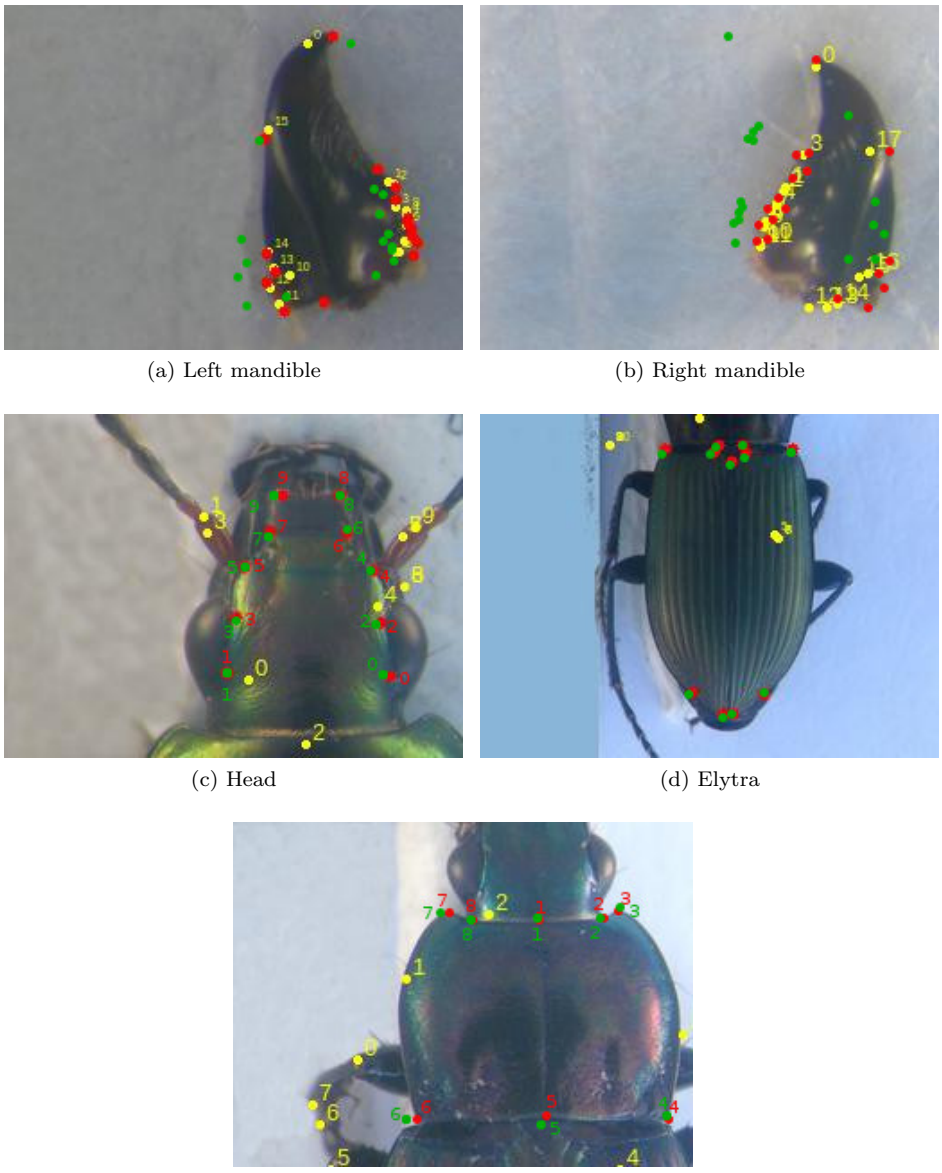
(d) Elytra

(e) pronotum

Figure 14: The distribution of average distances on each landmark of each beetle's anatomical. Black, blue, and orange lines present for the results of image processing procedures, deep learning (from scratch and fine-tuning), respectively.

have calculated the distances between manual and predicted landmarks (in both cases). Fig. 15 shows the locations of manual and predicted landmarks on each beetle's part from both two methods. In these images, the **red points** present the manual landmarks, the **yellow points** are estimated landmarks from image processing procedures and the **green points** are predicted landmarks which have been provided by deep learning.

Fig. 16 shows the distances by landmarks when we did a test on one image of each part. In these charts, the **black lines** present the distances when we apply the calculation on image processing procedures; the **blue and orange lines** show the distances with deep learning: training from scratch and fine tuning, respectively. As described in [30], we have combined some image processing procedures to output the predicted landmarks and this has become also a disadvantage of this method. If one procedure of them provides a bad result, it will affect the final result, i.e. if the result of segmentation step is bad, it seems that can not provide the output. In all beetle's anatomical, the mandibles are considered as the easy case to segment because the images are clear (they just contains the mandibles); while other parts are much noise, besides the main objects they have also the subcomponents of beetle, i.e. leg, antennae, ... That explains why we have obtained good results on mandibles but bad results on head, elytra, and pronotum part when applying the image processing procedures. In the opposite side, the results with deep learning, either training from scratch or fine-tuning, are very stable (Fig. 16). In the case of mandibles, which have good results from image processing techniques, then the results from deep learning are not much difference.



(a) Left mandible

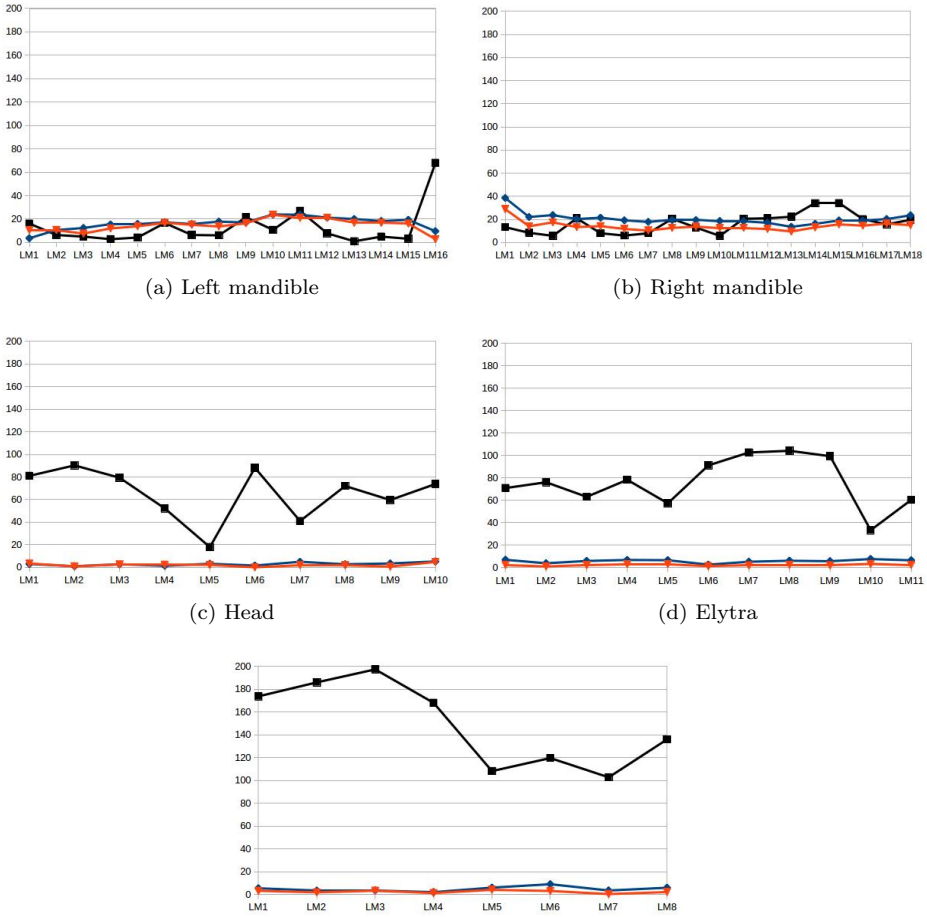
(b) Right mandible

(c) Head

(d) Elytra

(e) Pronotum

Figure 15: The presentation of manual and predicted landmarks on each beetle's part by applying two methods: image processing procedures and deep learning. The red points, yellow points, and green points present for manual landmarks, estimated landmarks by applying image processing procedures and using deep learning, respectively.



(a) Left mandible

(b) Right mandible

(c) Head

(d) Elytra

(e) Pronotum

Figure 16: The distances between manual and predicted landmarks of a test image when applying different techniques of each beetle's anatomical. Black, blue, and orange lines present for the results of image processing procedures, deep learning (from scratch and fine-tuning), respectively.

8. Conclusion

In this work, we have presented how to apply convolutional neural network to predict the landmark on 2D anatomical images of beetles. After going through many trial models, we have presented a convolutional neural network for automatic detection landmarks on anatomical images of beetles which includes the repeated of some elementary blocks (an elementary block consists of a convolutional layer, a max pooling layer, and a dropout layer) followed by fully connected layers. Then, the proposed model have been trained and tested 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.

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

References

[1] M. A. Arbib, Brains, machines, and mathematics, Springer Science & Business Media, 2012.

[2] G. Hinton, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal Processing Magazine 29 (6) (2012) 82–97.

[3] T. Mikolov, et al., Strategies for training large scale neural network language models, in: Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on, IEEE, 2011, pp. 196–201.

[4] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[5] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[6] C. Szegedy, et al., Going deeper with convolutions, Cvpr, 2015.

[7] S. Jean, K. Cho, R. Memisevic, Y. Bengio, On using very large target vocabulary for neural machine translation, arXiv preprint arXiv:1412.2007.

[8] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.

[9] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, Journal of Machine Learning Research 12 (Aug) (2011) 2493–2537.

[11] C. Farabet, C. Couprie, L. Najman, Y. LeCun, Learning hierarchical features for scene labeling, IEEE transactions on pattern analysis and machine intelligence 35 (8) (2013) 1915–1929.

[12] H. Li, Z. Lin, X. Shen, J. Brandt, G. Hua, A convolutional neural network cascade for face detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5325–5334.

[13] Z. Liu, S. Yan, P. Luo, X. Wang, X. Tang, Fashion landmark detection in the wild, in: European Conference on Computer Vision, Springer, 2016, pp. 229–245.

[14] Y. Sun, X. Wang, X. Tang, Deep convolutional network cascade for facial point detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, pp. 3476–3483.

[15] Z. Zhang, et al., Facial landmark detection by deep multi-task learning, in: European Conference on Computer Vision, Springer, 2014, pp. 94–108.

[16] C. Cintas, et al., Automatic ear detection and feature extraction using geometric morphometrics and convolutional neural networks, IET Biometrics 6 (3) (2016) 211–223.

[17] R. Collobert, J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning, in: Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 160–167.

[18] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European conference on computer vision, Springer, 2014, pp. 818–833.

[19] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

[20] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[21] S. Huang, M. Gong, D. Tao, A coarse-fine network for keypoint localization, in: The IEEE International Conference on Computer Vision (ICCV), Vol. 2, 2017.

[22] D. Ciregan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 3642–3649.

[23] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting., Journal of machine learning research 15 (1) (2014) 1929–1958.

[24] Y. A. LeCun, et al., Efficient backprop, in: Neural networks: Tricks of the trade, Springer, 2012, pp. 9–48.

[25] S. Dieleman, et al., Lasagne: First release. (Aug. 2015). doi:10.5281/zenodo.27878.
URL http://dx.doi.org/10.5281/zenodo.27878

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

[27] L. Torrey, J. Shavlik, Transfer learning, Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques 1 (2009) 242.

[28] J. Deng, et al., ImageNet: A Large-Scale Hierarchical Image Database, in: CVPR09, 2009.

[29] J. Margeta, et al., Fine-tuned convolutional neural nets for cardiac mri acquisition plane recognition, Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization 5 (5) (2017) 339–349. arXiv:https://doi.org/10.1080/21681163.2015.1061448, doi:10.1080/21681163.2015.1061448.
URL https://doi.org/10.1080/21681163.2015.1061448

[30] V. L. Le, M. Beurton-Aimar, A. Krahenbuhl, N. Parisey, MAELab: a framework to automatize landmark estimation, in: WSCG 2017, Plzen, Czech Republic, 2017.
URL https://hal.archives-ouvertes.fr/hal-01571440