

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

In recent years, deep learning, which has been introduced in the middle of the previous century for artificial intelligence, has risen strongly because of improvements in the computation both in memory size and computing time with GPU programming. Deep learning, specially Convolutional Neural Network, has been applied to solve the problems in different domains such as computer vision, speech recognition, or languages translation,... In computer vision, CNN has been used to classify the images into different categories, or to detect the object in images/videos, or to predict the key points of the objects in the images,... and it has brought the amazing achievements. In this work, we propose a CNN model to predict the key points (landmarks) on 2D anatomical biological images, specify beetle's images. Our proposed network is trained and evaluated on a dataset includes the images of collecting from 293 beetles. During the experiments, the network is tested in two ways: training from scratch and applying fine-tuning. The quality of predicted landmarks is evaluated by comparing with the manual landmarks which provided by the biologists. The obtained results are considered to be statistically good enough to replace the manual landmarks for the different morphometry analysis.

Keywords: Landmarks, deep learning, fine-tuning, CNN

*Corresponding author

Email address: van-linh.le@labri.fr (Le Van Linh)

¹both authors contributed equally to this work.

1. Introduction

Deep learning is a part of machine learning. It allows computational model composed of multiple layers to learn the representation of data. Each layer will extract the representation of input data which come from the previous layers, then it will compute a new presentation for the next layer. In a deep learning model, each layer may contain the different number of nodes, called neurons which has been inspired from the biological neural system. Actually, the computation at each layer is made by its nodes. The operation of a node is also the same as a biological neural, it receives different pieces of information and function properties to give the decision. Currently, deep learning has many kind of variant architectures such as Deep Neural Network (DNN), Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN), which have been applied to different fields including computer vision, speech recognition, natural language processing, machine translation, ..., and each variant architecture has found success in a specific domain, i.e. DNN can be applied to solve the classification or data analysis, RNN give the best perform on language modeling, while CNN is widely applied in computer vision.

Like the neural network, a CNN consists of an input, an output layer, as well as one or multiple hidden layers. It has also the score function and the loss function at the end of the model. However, CNN is a specific network for preprocessing data which has grid topology, i.e. time series (1-D) or images (2D). The name of the network is inspired by the convolution operation which is a kind of linear operation. The computing of CNN affects on 3 dimensions of data: width, height, and depth. An input after being put on the network will be passed through a series of hidden layers before giving the outputs at the last layer. In a CNN, the hidden layers typically consist of convolutional layers, pooling layers, fully connected layer. From the first architecture [1] until now, many CNN architectures have been proposed and have succeeded in different tasks of computer vision such as image classification [2], image recognition [3], image restoration [4], or key points detection.

In computer vision, key points detection is also an important field in image analysis. In this field, the algorithms try to find the key points (or called interest points or landmarks) in the image. The landmarks are considered as the points in the image that are invariant when the image change i.e. rotate or translate. Depending 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 such as the landmarks on *Drosophila* wing [] are stayed on the veins of the wings, but the landmarks on human ears [] can be located on the ear edge or inside the Pimas of the ears. The detected key points are then used in different objectives of different domains, for example, they are used to recognize the seabed in geosciences; in computer vision, they are fundamental to detect the human face or human pose; in biology, the topography of the objects of an organism can be measured from the location of landmarks.

Early methods [3, 4, 5] mainly focused on the low level-vision of the image by applying the image processing techniques, and segmentation is most often the first and the most important step in the process. This task remains a bottleneck to compute the features of the complex image. Even if the interested object is easy to extract, the process may take a bit of time to extract the exciting features. The emergence of deep learning has seen success in other fields of computer vision and key points detection is also not an exception. It have used to predict the fashion landmarks [?], human facial key points [29, 30], or ear landmarks [2].

In this work, we use deep learning, specific CNN, to predict the landmarks on biological anatomical images. The main idea consists of the designing and training a CNN on a dataset with manual landmarks. Like other CNNs, the features are extracted by using a group of different layer types (i.e. convolutional layer, pooling layer, . . .) followed by some dense layers to provide the prediction of the network. After designing, the proposed model will be trained on the dataset includes the images which are taken from 293 beetles. For each beetle, the biologists have taken images of five parts: *left and right mandibles*, *head*, *elytra*, and *pronotum*. All the images are presented in the RGB color model

with two dimensions. Along with each image, a set of landmarks has been marked by experts which can be used as ground truth to evaluate the predicted landmarks. During the experiments, the proposed network has been trained
65 on the dataset by applying two strategies. In the first strategy, the network is trained from scratch on each dataset; while in the second strategy, the training process has been modified to include a fine-tuning [7] stage. Besides, the value of Root Mean Square Error (RMSE) is used to compute the losses during two experiment processes.

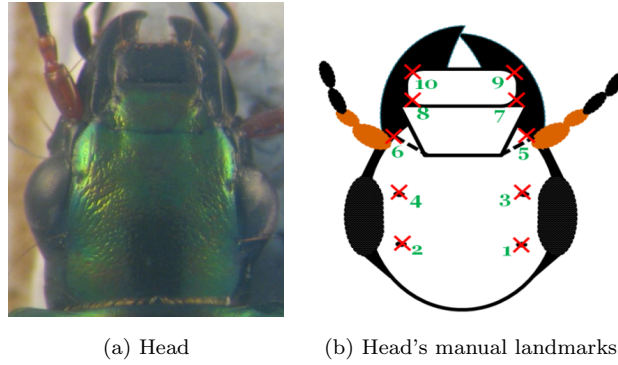


Figure 1: The **head** anatomical of beetle and its manual landmarks

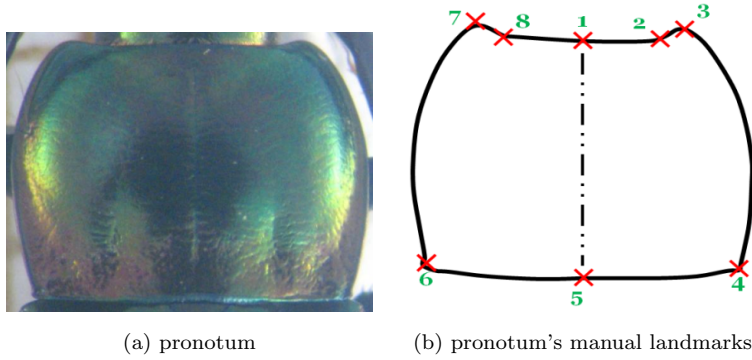
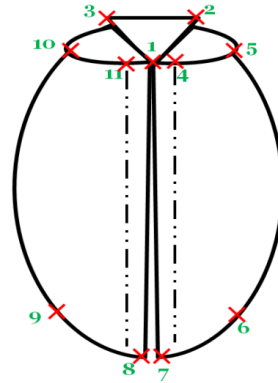


Figure 2: The **pronotum** anatomical of beetle and its manual landmarks

70 The rest of this paper is organized as followed: Section 2 discusses the related works about determining the landmarks on 2D images. Then, a short overview

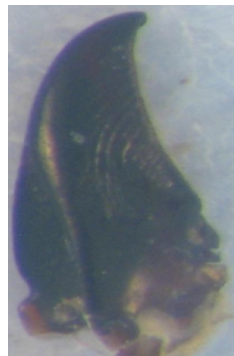


(a) Elytra

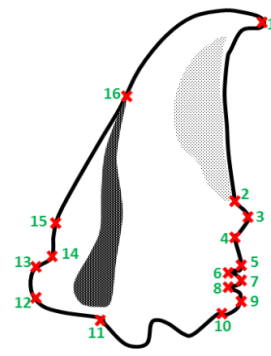


(b) Elytra's manual landmarks

Figure 3: The **elytra** anatomical of beetle and its manual landmarks



(a) Left mandible



(b) Left mandible's manual landmarks

Figure 4: The **left mandible** anatomical of beetle and its manual landmarks

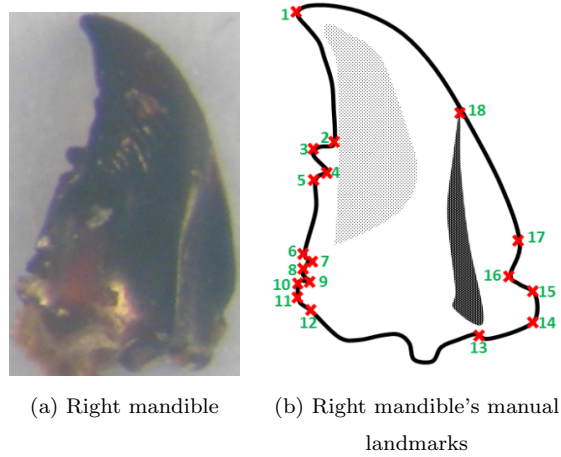


Figure 5: The **right mandible** anatomical of beetle and its manual landmarks

about CNN and its components will be introduced in Section 3. Section 4 gives another approaches to augment the dataset. Section 5 shows the procedure of designing the network model. The first experiment of the network on each dataset is presented in Section 6. Section 7 presents a modification of training process and its experiment on datasets. Finally, the conclusion is given in Section 8.

2. Related works

Deep learning [8] has been introduced in the middle of the previous century for artificial intelligence application but it has encountered several problems to take real-world cases such as memory or computing time. Luckily, the improvements of computing capacities, both in memory size and in computing time with GPU programming, have opened new perspective for deep learning. In recent years, deep learning architectures are known as the solutions for the tasks in computer vision. Especially in image analysis, CNNs, which consist of convolutional layers, pooling layers, full connected layers, are used to solve the classification or recognition tasks. For example, LeNet [9] has been known as

the first architecture of CNN to read the zip codes and digits. It had a standard structure stacked convolutional layers (normalization and max-pooling) are followed by one or more full-connected layers. From this architecture, many models have been proposed to improve the accuracy as well as computing times in recent years. That must be mentioned to AlexNet [10], ZFNet [11], GoogLeNet [12], VGGNet [13], or ResNet [14]. Besides, deep learning has been obtained the achievement in other domains such as **image recognition** [12, 15, 16], **speech recognition** [17, 18], **language translation** [19, 20], **natural language processing** [8, 21, 22],

Besides object classification or recognition, **key points detection** is another task in image analysis. Early methods of machine learning have been achieved good results. These methods are basically divided into two groups: the regression-based methods and template fitting methods. A regression-based method predicts landmark locations by regression using image features [23, 24, 25]. While a template fitting method builds a template to fit with the input image, then the landmarks are setting [26, 27, 28]. With the comeback of deep learning, CNN has been used to predict the landmarks on 2D and it has gained better results. Yi Sun et al. [29] have proposed a cascaded CNNs to predict the facial points on the human face. Their model includes the networks which have separated into three levels of the cascade. The networks recognize the human face from the global to the local view to increasing the accuracy of predicted key points. Zhanpeng Zhang et al. [30] proposed a *Tasks-Constrained Deep Convolutional Network* to optimize facial landmarks detection. Their model detected the facial landmarks with a set of related tasks such as head pose estimation, gender classification, age estimation, or facial attribute inference. Shaoli Huang et al. [31] introduced a coarse-fine network, which composed of several coarse detector branches, to locate the keypoints and to estimate the human pose. Cintas et al. [2] has introduced a network to predict the landmarks on human ears (Fig. 6). After training, the network has the ability to detect 45 landmarks on human ears.

As an association with the biologist, we study to develop an automatic

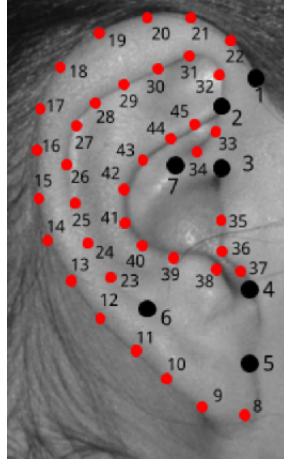


Figure 6: Landmarks on human ear in study of Cintas et al. [2]

method for predicting the landmarks on beetle's anatomical (Figs.1a, 2a, 3a,
 120 4a, 5a). In the past, locating landmarks on biological images is mainly using
 image processing techniques where the object in the image is easily to segment.
 For example, in a previous work [32], we have applied a series of algorithms to
 detect the landmarks automatically on beetles mandibles which are considered
 as the eased objects to segment (with an quality enough good for our need). In
 125 that work, the landmarks have been detected by registering two segmentations
 of images and then, using SIFT descriptor to refine the location of predicted
 landmarks. After the experiment, we have obtained good enough results on
 mandibles. Unfortunately, we have observed that the method did not provide
 good results when the segmentation is not precise, i.e. on pronotum or elytra
 130 images. This is explained why we have turned the automatically landmarking
 into another stage without any segmentation step. Using CNN for landmarking
 seems that a good choice for un-segmentable images.

3. Overview of Convolutional Neural Network

CNN is a feedforward network which takes the information following one
 135 direction from the inputs to the outputs. Currently, CNNs have many different

variations, i.e. AlexNet [10], GoogLeNet [33], ResNet [14], but in general, it consists of convolutional and pooling layers which are stacked together to convolve and to down-sample the input. Then, they are followed by one or more fully connected layers to give the decision as the output of the network.

Fig. 7 shows an 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 a popular one that we can find from the literature [9, 10]. However, several architectures have been proposed recurrently to improve the accuracy or to decrease the computation costs. In this section, we will mention to the most popular layers in a CNN: convolutional layers, pooling layers, and fully connected layers.

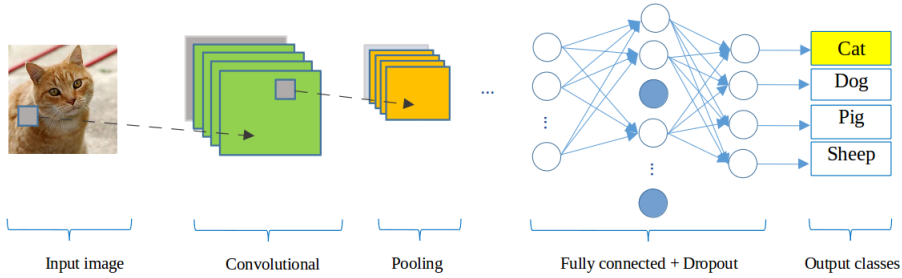


Figure 7: A CNN network for classification problem

Convolutional (CONV) layers: use as a feature extractor by applying some learnable weights on the input images. The input image is convolved with the learnable weights in order to compute the new feature maps; then, the convolved results are sent through a nonlinear activation. In convolutional layer, the neurons are arranged into feature maps. All the neurons within a feature map have constrained to be equal; however, different features maps within the same convolutional layer have different weights so that several features can be extracted at each location of an input image.

Pooling (POOL) layers : are mostly used to down-sampling the size of the input with the purpose to reduce the spatial resolution of the feature map to
 160 reduce the computation cost. Initially, it was common practice to use average
 pooling which propagates the average of all the input to the next layer. However,
 in more recent models [10, 34, 16], max pooling which propagates the maximum
 value (selecting the largest value) with a receptive field to the next layer. Fig.
 8 illustrates the differences between max and average pooling. Give an input
 165 image of size 4×4 , if applying a 2×2 filter and stride of two is applied, the output
 will be had the size of 2×2 for both cases. However, the value in each element
 of the output is different because the max pooling outputs the maximum values
 of the filter region, while average pooling outputs the average value of the same
 region.

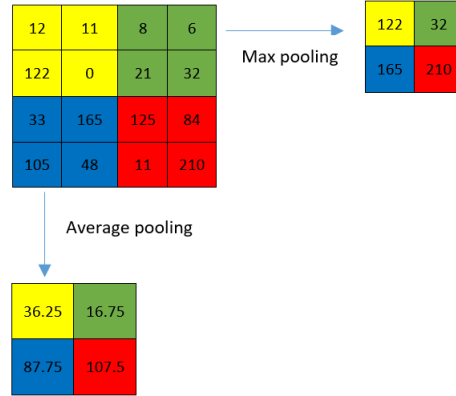


Figure 8: The results of different pooling

170 Dropout [35] is a specific layer, it is used to prevent the over-fit of a neu-
 ral network. The term dropout mentions dropping the out units and their
 connections (incoming and outgoing) of a layer in the network. The units are
 dropped randomly with a probability p independent with other units. The idea
 of dropout is not limited to feed-forward neural nets. Applying dropout tech-
 175 nique will make the network become a collection of thinned networks 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 the thinned networks. Normally, the dropouts layers are placed after the fully connected layers, but it is possible to use dropout after the pooling layers with
180 creating some kind of images noise augmentation.

Fully connected (FC) layers: are usually followed the convolutional and pooling layers which used to extract the abstract feature representations. A CNN may have one or more fully connected layers. It interprets the feature representations and performs a function of high-level reasoning (i.e. giving classification
185 score) by applying the activation functions. In practice, the last fully connected layer refers to the output of the network and choosing the activation function is depended on which kind of problem that network solve, i.e. we usually softmax activation function for a classification problem but it is not useful in a support vector machine.

To solve a problem by using deep learning, besides designing the network
190 architecture then training dataset is also a very important thing. The nature of deep learning algorithm is training the model on dataset repeatedly to reach the best accuracy. So, providing a large dataset will provide more learning cases for the model and it will clearly improve the learnable of the network. Unfortunately, data in practice has usually not enough. So, some methods have
195 been applied to generate the fake data with the purpose to augment the dataset (i.e. rotate, translate, flip the images). Our case is also not an exception, we work on a small dataset of beetles with 293 images for each part. This number is really modest to apply deep learning. In the next section, we describe the
200 procedure to augment the dataset before coming to design the architectures.

4. Data augmentation

A characteristic of machine learning and deep learning is using a volume dataset to train the model. Of course, we do not always have enough data for training in practice. One way to solve this problem is to create fake data from
205 real data and to add it to the training set. Dataset augmentation has been a

particularly effective technique for a specific problem. For example, in images classification problem, the operations like translating, rotating or scaling the images have also effective. The fake images may be generated by translating (rotating or scaling) in each direction. Besides, injecting noise in the input can
210 also see as a form of data augmentation.

Our dataset includes 293 images of beetles (for each anatomical part). All the images are taken with the same camera in the same condition with a resolution of (3264×2448) . Each image has a set of manual landmarks provided by biologists, i.e, each pronotum has 8 landmarks, each head has 10 landmarks.
215 Applying CNNs to train each part with a small number of images to reach good results is impossible. So, we need to augment the dataset before training the networks. Firstly, we have found that the original solution of the images (3264×2448) are heavy for the neural network. For performance considerations, in most of CNNs [2, 6, 29], the size of the input is limited to (256×256) pixels,
220 so we have decided to down-sampling the images to a new resolution (256×192) (to respect the ratio between x and y). Of course, the coordinates of manual landmarks have been also scaled to fit with the new resolution of the images. In the usual way, the transformations have been used to augment the dataset (i.e rotation, translation,...) but the analysis of image by CNN is most often
225 translation and rotation invariant. Therefore, two other procedures have been imaged to increase the number of images in the dataset (256×192) .

The first procedure is to change the value of a color channel in the original image to generate a new image. According to that, a constant is added to one of the RGB channels each time it is used for training. Each constant is sampled
230 in a uniform distribution $\in [1, N]$ to obtain a new value caped at 255. For example, Fig. 9 shows an example when we added a constant $c = 10$ to each channel of an original image. Following this way, we can generate three version from an image.

In the second procedure, we have applied the opposite procedure to the first
235 one. Instead of adding the value, we separate the channels of RGN into three gray-scale images as the network works on single channel images (Fig. 10).

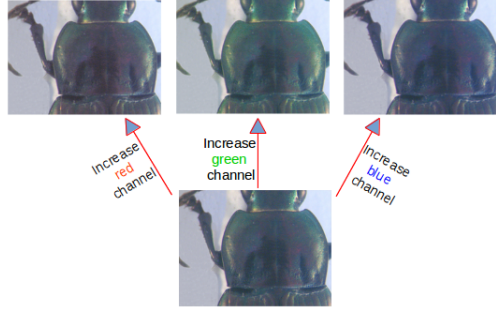


Figure 9: A constant $c = 10$ has been added to each channel of an original image

At the end of the processes, we are able generate six versions from an original image. In total, we have $293 \times 7 = 2051$ images for each anatomical part of beetle (an original image and six generated images). However, we have not used
 240 all images for training and validation. So, we have chosen 260 original images and their generations (1820 images) of each dataset for training and validation processes, the remaining images (33 original images) are used for test process.

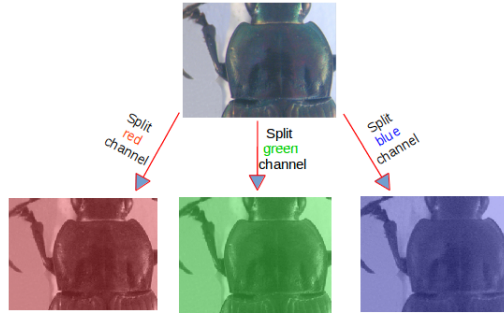


Figure 10: Three channels (red, green, blue) are separated from original image

In practical, to obtain a fast convergence during the computing, it is useful to normalize the brightness of the images to $[0, 1]$ instead of $[0, 255]$ and the
 245 coordinates of the landmarks have been also normalized [36].

5. Network architectures designing

From the beginning of CNN, LeNet [9] can be considered as the first one, which has used to classify digits on hand-written numbers on cheques. The network is very simple by stack together the convolutional layers and max-pooling layers followed by full connected layers. Because of the limit of the resources at that time, this model is also constrained by the availability of computing. Until 2012, when computing resources are improved, AlexNet [10] was born and it has won the challenge by reducing the top-5 error in ImageNet challenge. AlexNet had a similar architecture as LeNet but it was deeper, bigger. Besides, the activation functions have been changed from Sigmoid [37] to ReLU [38] which have been proved more improvement in computing time. Additional, it had supplement the dropout layers to control overfitting. Based on the idea of the improvement, we have tried to design an architecture for landmarking on beetle images. This work is beginning with trying three network models before deciding the final architecture. Like other CNN models, we have employed the classical layers to construct the models, i.e., convolutional layers, maximum pooling layers, dropout layers and full-connected layers.

The first architecture is very classical one, it receives an image with the size of $(1 \times 192 \times 256)$ as the input. Then, the network consists on three repeated strcutre of a convolutional layers followed by a maximum pooling layers. Most CNNs, the hyperparameters of convolutional layers have been set to increase the depth of the images from the first layer to the last layer. That is reflected in the setting of the number of filters at each convolutional layer. So, the depths of convolutional layers increase from 32, 64, and 128 with different size of the kernels: (3×3) , (2×2) and (2×2) , respectively. Inserting pooling layers after a convolutional layers is a common periodcally. The pooling layer effects to progressively reduce the spatial size of the representation to reduce the number of parameters, computation in the network, and it also controls over-fitting. The operations of pooling layers independent on every depth slice of the input. The most common form is a pooling layer with filters of size (2×2) and a stride

of 2. It downsamples every depth by 2 along width and height of the input. Therefore, all the kernels of maximum pooling layers have the same size of (2×2) with a stride of 2 as usual. At the end of the model, three full-connected layers have been added to extract the global relationship between the features and to procedure the outputs. The first of two full-connected layers are set to non-linearity to make sure these nodes interact well and take into account all possible dependencies at the feature level. The outputs of the full-connected layers are 500, 500 and 16. The output of the last full-connected layer corresponds to the coordinates (x and y) of 8 landmarks which we would like to predict.

Fig. 11 shows details of the first model: The orange rectangles represent for convolutional layers while the yellow rectangles represent for maximum pooling layers and three full-connected layers with their parameters are presented at the end of the model.

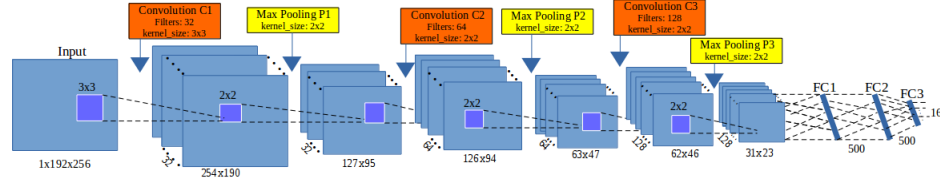


Figure 11: The architecture of the first model

The second architecture is modified from the first model. The layers are kept the same as the first one but the outputs of the first of two full-connected layers are changed from 500 (in the first model) to 1000 (Fig. 12). Increasing the value at full-connected layers is hoping to obtain more features from convolutional layer and to prevent the over-fitting.

To build the third architecture, we have used the definition of *elementary block*. An elementary block is defined as a sequence of convolution (C_i), maximum pooling (P_i) and dropout (D_i) layers (Fig. 13). This significantly reduces overfitting and gives major improvements over other regularization methods [35]. The idea of dropout is to include some variations between different runs. During training phase, dropout samples are done from an exponential number

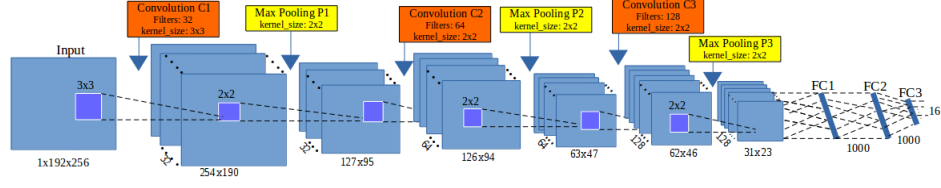


Figure 12: The architecture of the second model

of different “thinned” network. At test phase, it is easy to approximate the effect of averaging the prediction of all thinned networks by simply using a single unthinned network with smaller weights. So, we have modified the architecture by combining some *elementary blocks*.

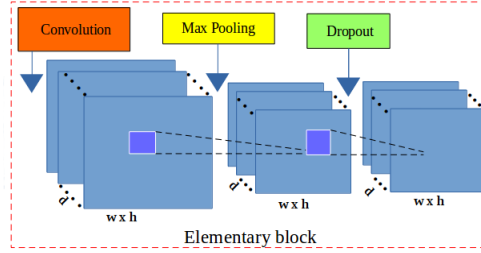


Figure 13: The layers in an elementary block

Fig. 14 illustrates the layers in the third architecture. For our purpose, we have assembled **3 elementary blocks**. The parameters for each layer in each elementary block are as below, the list of values follows the order of elementary blocks ($i = [1..3]$):

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

315

- Kernel filter sizes: (2×2) , (2×2) , and (2×2)
- Stride values: 2, 2, and 2
- No padding is used for POOL layers

- DROP layers:

- Probabilities: 0.1, 0.2, and 0.3

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

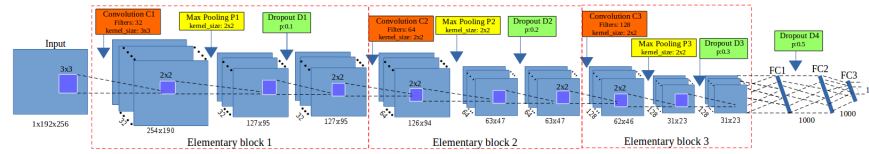


Figure 14: The architecture of the third model

The core of CNN is training over iteration. There are many ways to optimize the learning algorithm, but gradient descent [36] 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 satisfy with the results will remain the same. So, we have chosen gradient descent in the backward phase to update the values of learnable parameters and to increase the accuracy of the network. The networks are designed with a small sharing learning rate and a momentum. The learning rate is initialized at 0.03 and stopped at 0.00001, while the momentum is updated from 0.9 to 0.9999. Their values are updated over training time to fit with the number of epochs ². The implementation of the architectures have been done on Lasagne framework [39] by Python.

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

6. Experiments and results

335 Before widely applying to all anatomical parts, we have firstly tried with
prontum part to evaluate the performance. The networks have been trained
in 5,000 epochs on Ubuntu machine by using NVIDIA TITAN X cards. The
set of images that used for training and validation are merged together. During
the training, the images are chosen randomly from the dataset with a ratio of
340 60% for training and 40% for validation. The training step takes into account
a pair of information (*images, manual landmarks coordinates*) as training data.
In the context of deep learning, landmark prediction can be seen as a regres-
sion problem. So, we have used Root Mean Square Error (RMSE) to compute
the loss of implemented architectures. At the test phase, images without land-
345 marks are given to the trained network to produce output coordinates of the
predicted landmarks. The results then evaluated by comparing with the manual
landmarks coordinates provided by biologists which have been seen as ground
truth.

Fig. 15 shows the training errors and the validation errors during traning
350 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. 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
355 that this solution is still not suitable.

Then, we have continued to train the third model on the same dataset of
prontum images. Fig. 16 illustrates the losses during the training of the third
model. As the same meaning in Fig. 15, the blue line is training loss, the
green line is validation loss. In the opposite with two previous models, the
360 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

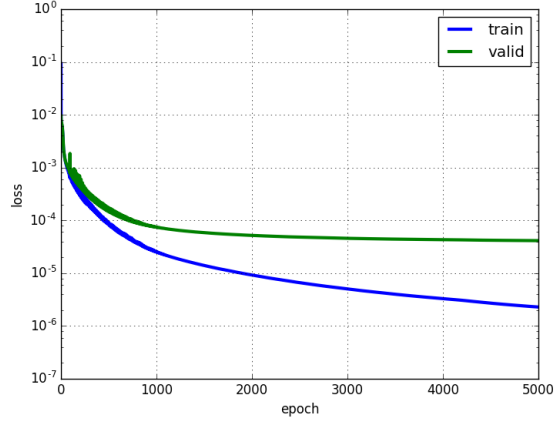


Figure 15: The training and validation losses of the first model

the model. *So, we have decided to keep the architecture of the third model for our landmarking problem.*

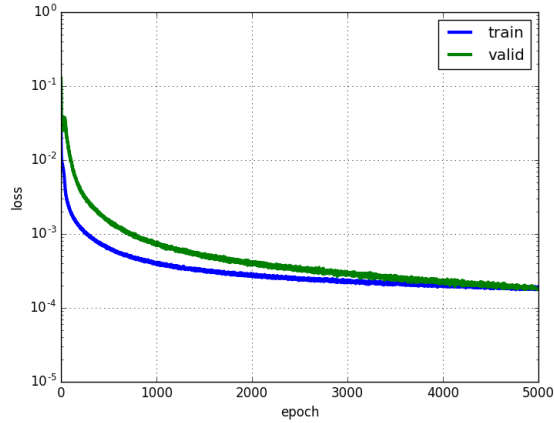


Figure 16: The training and validation losses of the third model

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 time, we have chosen a different fold of 33 images as testing images, the remaining images are used as training and validation images

370 (293/33 \approx 9 rounds). Following that, the network will be trained with different
 datasets, then the trained model will be used to predicted the lanmarks on the
 images in the corresponding test set. Table. 1 resumes the losses of 9 rounds
 when we trained the third model on pronotum images. Clearly, the training
 loss/validation loss among rounds are not so large and the RMSE values are
 375 looking 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
 have been computed the correlation between the manual landmarks and their
 corresponding predicted one. Table. 2 shows the correlation scores of 3 metrics
 (using *scikit-learn* [40]), i.e, coefficient of determination (r^2), explained variance
 380 (EV), and Pearson correlation. All of three metrics have the same possibility.
 The best score is 1.0 if the correlation data is good, lower values are worse.
 It means that our predicted coordinates are very close with the ground truth.
 However, the measure is not enough good to provide a useful result to biologists.
 Moreover standing on the side of image processing, we are looking forward to
 385 seeing the predicted coordinates than the statistical results.

The main goal of computing is to predict the coordinates of landmarks,
 so the distances (in pixels) between the coordinates of manual landmarks and

| Metric | r^2 | EV | Pearson |
|--------|---------------|---------------|---------------|
| Score | 0.9952 | 0.9951 | 0.9974 |

Table 2: Correlation scores between manual landmarks and predicted landmarks

corresponding predicted landmarks have been taken into account on all images. Then, the average of distances are computed by landmarks. Table. 3 shows the average distances by landmarks on all images of pronotum dataset. With
390 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. 17 shows the distribution of the distances on the first landmark of all
395 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
400 pixels: 3.07%.

To illustrate this purpose, Fig. 18 shows the predicted landmarks on two test

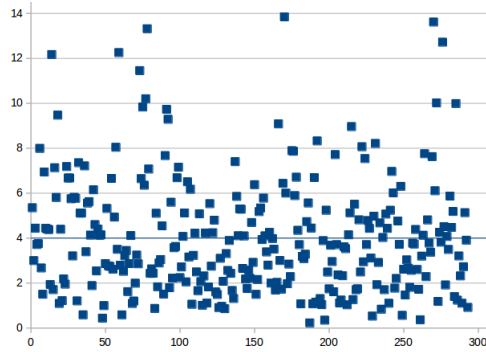


Figure 17: The distribution of the distances on the first landmark. The blue line is the average value of all distances.

images. One can note that even some predicted landmarks (Fig. 18a) are closed to the manual ones, in some case (Fig. 18b) the predicted ones are far from the expect results. This result explains why the average distance by landmarks are
 405 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.

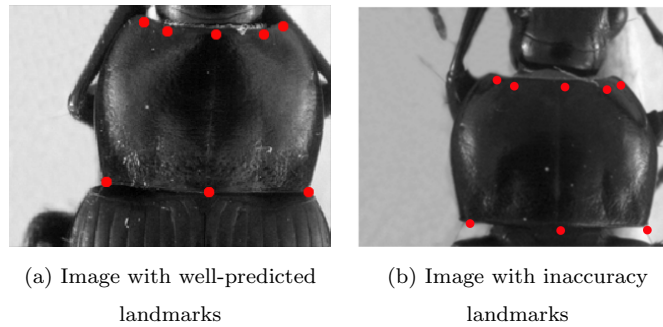


Figure 18: The predicted landmarks, in red, on the images in test set.

From the success of the third architecture on pronotum dataset, we apply the same procedures (data augmentation, training, . . .) on other parts of beetle: *left and right mandibles, elytra and head*. However, we have modified the number of
 410 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.

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 also not enough precise, and the average error are also still high (of course, we have the distances are higher than the average distances).

In order to reach more acceptable results for biologists, we have broadened model with another step of deep learning: **transfer learning**. That is a method enables to re-uses the model developed for a specific task/dataset to lead another task (called *target task*) with another dataset. This process allows rapid process and improves the performance of the model on the target task [41]. The most popular example has been given with the project ImageNet of Google [42] 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 [43]. 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

| 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 left mandible, right mandible, elytra and head images.

but also fine-tunes the weights of a trained model by continuing the backpropagation. Unfortunately, some rapid tests have shown that re-using ImageNet features has not been relevant for our application. We have designed a way to reproduce the method with our own data. It is worth noting that of course the size of data to pre-train has drastically decreased. For our pre-training step, the network has been trained on the whole dataset including the images of three parts of beetle *i.e* *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 training dataset is combined from the images of three sets: *pronotum*, *elytra*, and *head* (after augmentation). When applying the training from the scratch, we have used cross-validation to select the data (9 folds). It means that for each dataset, we have some different training data and corresponding testing data. So, the images that use to train the model are just select from one of the folds in each dataset. Specifically, we have taken 1,820 images of each part. In total, it includes 5,460 images ($260 \times 7 \times 3$).

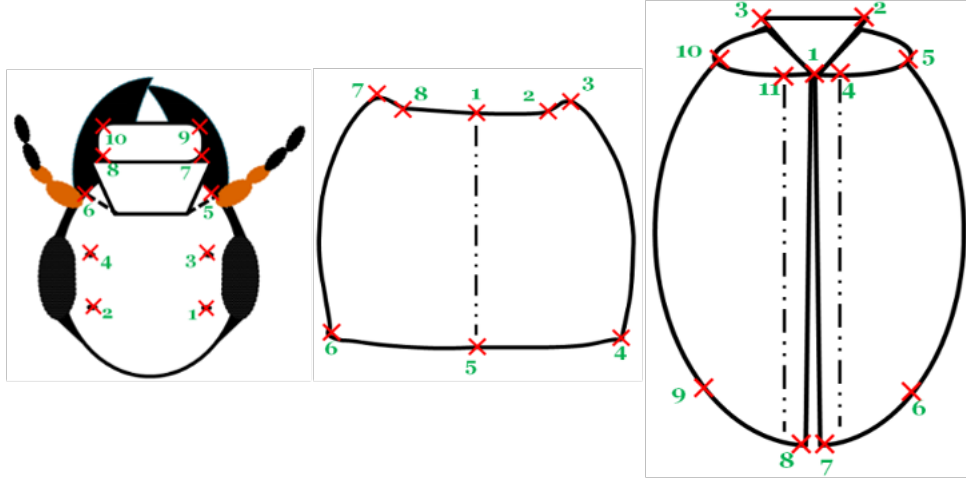


Figure 19: A presentation of head, pronotum and elytra part with corresponding manual landmarks

However, another problem has been appeared when we combined the images

from different dataset. That is the different number of landmarks on each part:
 8 landmarks on pronotum part, 10 landmarks on head part, and 11 landmarks on
 460 elytra part. Fig. 19 shows the position 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
 465 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 pa-
 rameters of the network (learning rate, momentum, ...) are kept the same as
 470 training from scratch but the number of epochs are increased to 10,000 instead
 of 5,000 to achieve better learning on the parameters (weights). Additional,
 we have shuffled the training set. Because the neural network learns the faster
 from the most unexpected sample. It is advisable to choose a sample at each
 iteration that is the most unfamiliar to the system. Shuffling the examples will
 475 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 out-
 puts (8 landmarks). Then, the trained model is used to fine-tuning on each
 480 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, respec-
 tively. Comparing with the losses when we trained the model from scratch,
 485 *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 av-

erage error based on the distances between predicted and corresponding manual
 landmarks have been also computed. Tables. 6, 8, 10, 11, and 12 show the aver-
 490 age 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 land-
 marks with the help of fine-tuning is more precise than training from scratch.
 495 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 | Validation |
|-------|----------|------------|
| | loss | 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 distance per landmark of two processes on pronotum images

In another view, Fig. 20 shows the comparison of the average distance
 distributions on each dataset in two procedures (from scratch and fine-tuning).
 500 In which:

- **Blue** curves: present for the average distances on each landmarks when we train the model from scratch.

| 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

| 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 | 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 distance per landmark of two processes on elytra images

| #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 distance per landmark of two 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 distance per landmark of two 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 distance per landmark of two processes on head images

- **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): illustrate for the average distances when we applied the image processing procedures to predict the landmarks on segmentable images.

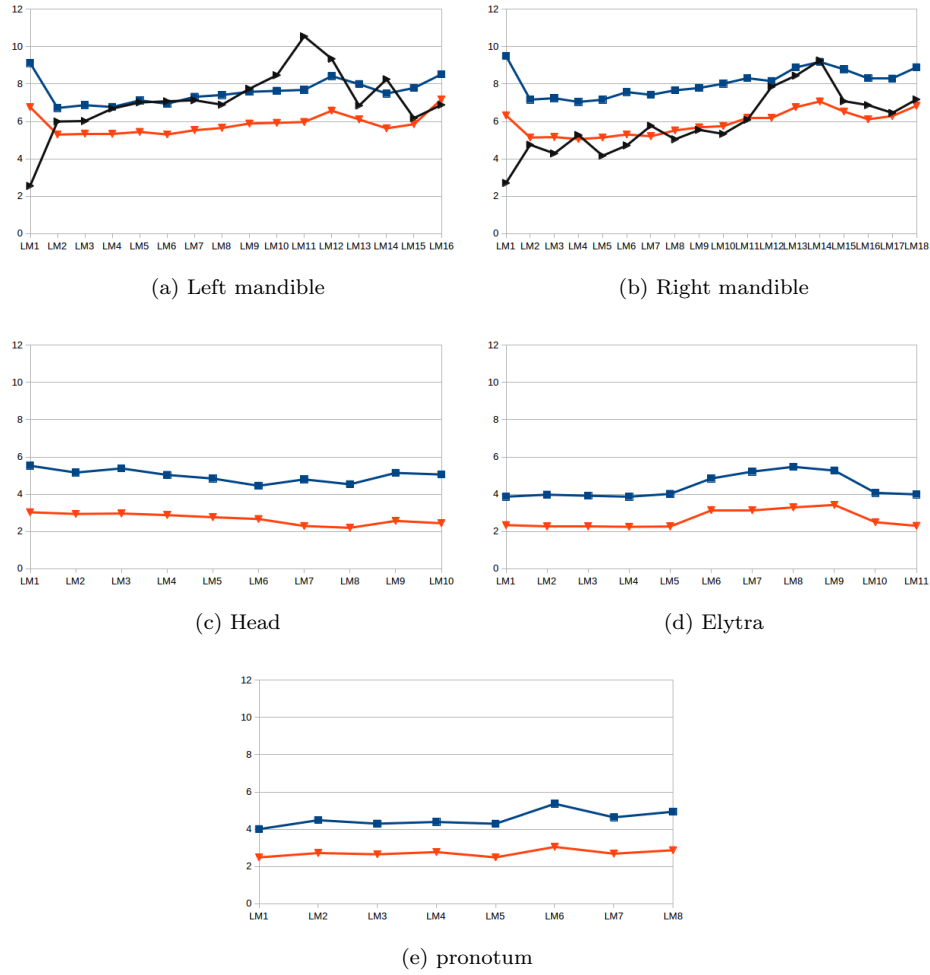


Figure 20: The distribution of average distances on each landmark of each beetle's anatomical

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
510 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
515 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. Clearly, the result with fine-tuning has improved the location of
estimated landmarks. Even the average distances which obtained from scratch
520 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 learn-
525 ing, we have run the test on an image of all parts in both methods. Then, we
have calculated the distances between manual and predicted landmarks (in both
cases). Fig. 21 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
530 processing procedures and the **green points** are predicted landmarks which
provided by deep learning. Fig. 22 shows the distances of each part that we
have obtained. In these charts, the **blue lines** present the distances when we
apply the calculation on image processing procedures; the **orange and yellow**
lines show the distances with deep learning: training from scratch and fine
535 tuning, respectively.

As described in [32], 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

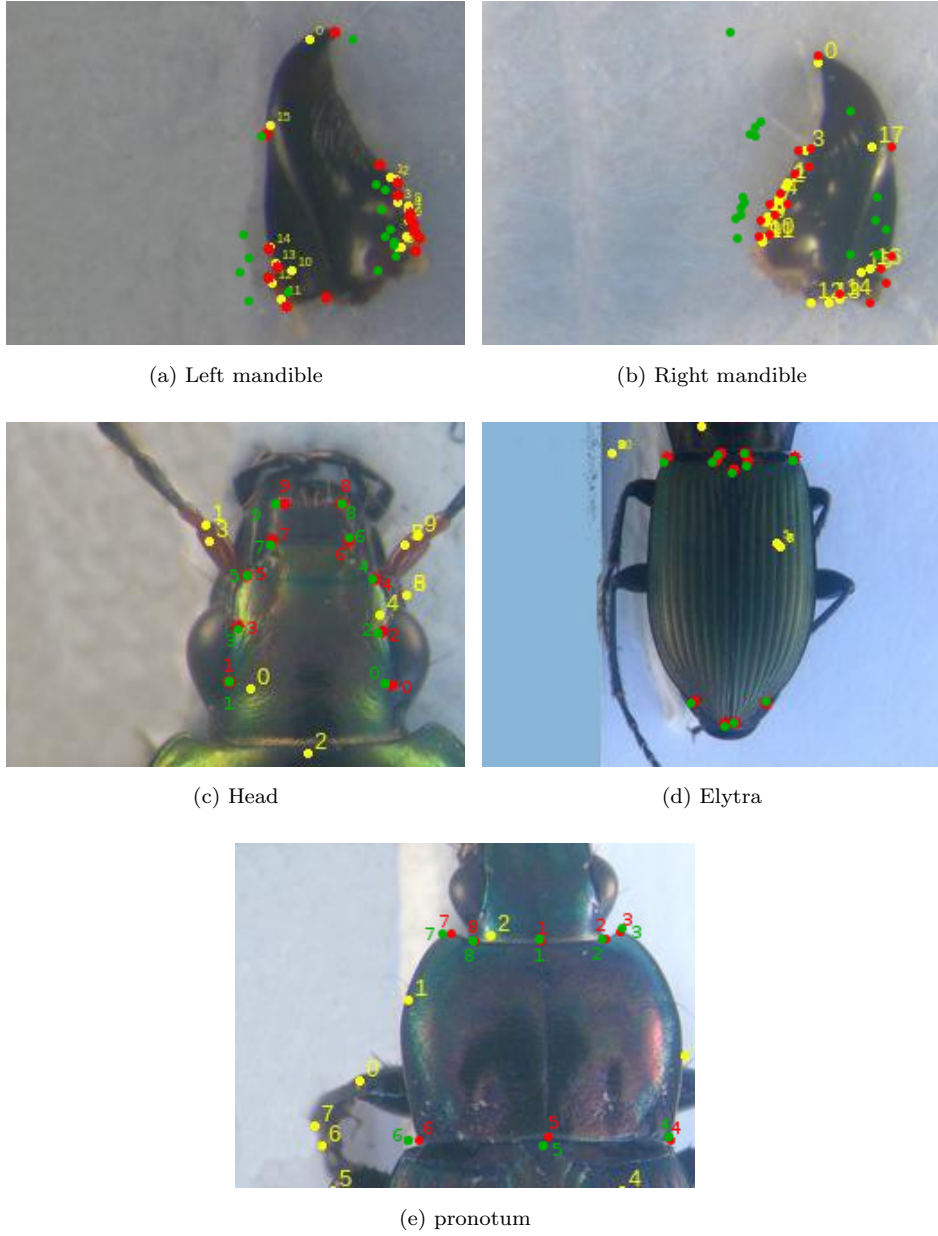


Figure 21: The presentation of manual and predicted landmarks on each beetle's part by applying two methods: image processing procedures and deep learning

540 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
545 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. 22). In the case of mandibles which have good results from
image processing techniques then the results from deep learning are not much
difference.

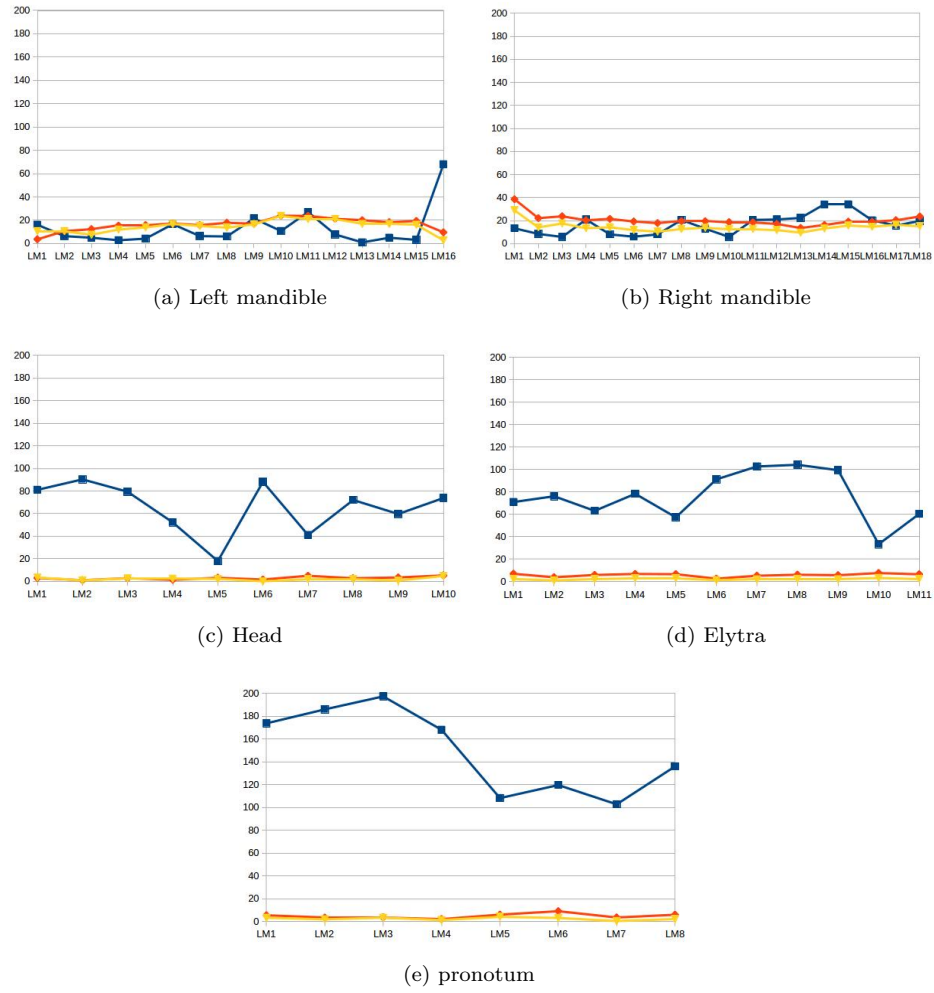


Figure 22: The distances between manual and predicted landmarks of a test image when applying different techniques of each beetle's anatomical

550 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
555 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
560 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
565 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.

570 References

References

- [1] A. Sonnenschein, D. VanderZee, W. R. Pitchers, S. Chari, I. Dworkin, Supporting material and data for "an image database of drosophila melanogaster wings for phenomic and biometric analysis" (2015). doi:
575 10.5524/100141.
- [2] C. Cintas, et al., Automatic ear detection and feature extraction using geo-

metric morphometrics and convolutional neural networks, *IET Biometrics* 6 (3) (2016) 211–223.

- 580 [3] D. G. Lowe, Distinctive image features from scale-invariant keypoints, *International journal of computer vision* 60 (2) (2004) 91–110.
- [4] H. Bay, T. Tuytelaars, L. Van Gool, Surf: Speeded up robust features, in: *European conference on computer vision*, Springer, 2006, pp. 404–417.
- [5] S. Palaniswamy, N. A. Thacker, C. P. Klingenberg, Automatic identification of landmarks in digital images, *IET Computer Vision* 4 (4) (2010) 247–260.
- 585 [6] Y. LeCun, K. Kavukcuoglu, C. Farabet, Convolutional networks and applications in vision, in: *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, IEEE, 2010, pp. 253–256.
- [7] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: *Advances in neural information processing systems*, 2014, pp. 3320–3328.
- 590 [8] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *Nature* 521 (7553) (2015) 436–444.
- [9] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE* 86 (11) (1998) 2278–2324.
- 595 [10] 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.
- [11] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: *European conference on computer vision*, Springer, 2014, pp. 818–833.
- 600 [12] C. Szegedy, et al., Going deeper with convolutions, *Cvpr*, 2015.

- [13] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.
- 605 [14] 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.
- [15] 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.
- 610 [16] 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.
- [17] 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.
- 615 [18] 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.
- [19] S. Jean, K. Cho, R. Memisevic, Y. Bengio, On using very large target vocabulary for neural machine translation, arXiv preprint arXiv:1412.2007.
- 620 [20] 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.
- [21] 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.
- 625 [22] R. Collobert, J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning, in: Proceedings of

- 630 the 25th international conference on Machine learning, ACM, 2008, pp. 160–167.
- [23] M. Valstar, B. Martinez, X. Binefa, M. Pantic, Facial point detection using boosted regression and graph models, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 2729–
635 2736.
- [24] M. Dantone, J. Gall, G. Fanelli, L. Van Gool, Real-time facial feature detection using conditional regression forests, in: Computer vision and pattern recognition (CVPR), 2012 IEEE conference on, IEEE, 2012, pp. 2578–2585.
- 640 [25] X. P. Burgos-Artizzu, P. Perona, P. Dollár, Robust face landmark estimation under occlusion, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1513–1520.
- [26] T. F. Cootes, G. J. Edwards, C. J. Taylor, Active appearance models, IEEE Transactions on Pattern Analysis & Machine Intelligence (6) (2001)
645 681–685.
- [27] X. Liu, Generic face alignment using boosted appearance model, in: Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on, IEEE, 2007, pp. 1–8.
- [28] X. Yu, J. Huang, S. Zhang, W. Yan, D. N. Metaxas, Pose-free facial landmark fitting via optimized part mixtures and cascaded deformable shape
650 model, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1944–1951.
- [29] Y. Sun, X. Wang, X. Tang, Deep convolutional network cascade for facial point detection, in: Proceedings of the IEEE conference on computer vision
655 and pattern recognition, 2013, pp. 3476–3483.
- [30] Z. Zhang, et al., Facial landmark detection by deep multi-task learning, in: European Conference on Computer Vision, Springer, 2014, pp. 94–108.

- [31] 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.
- [32] 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>
- [33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, corr abs/1409.4842, URL <http://arxiv.org/abs/1409.4842>.
- [34] 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.
- [35] 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.
- [36] Y. A. LeCun, et al., Efficient backprop, in: Neural networks: Tricks of the trade, Springer, 2012, pp. 9–48.
- [37] J. Han, C. Moraga, The influence of the sigmoid function parameters on the speed of backpropagation learning, in: International Workshop on Artificial Neural Networks, Springer, 1995, pp. 195–201.
- [38] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.
- [39] S. Dieleman, et al., Lasagne: First release. (Aug. 2015). doi:10.5281/zenodo.27878.
URL <http://dx.doi.org/10.5281/zenodo.27878>

- 685 [40] P. et al, Scikit-learn: Machine learning in python, Journal of machine learning research 12 (Oct) (2011) 2825–2830.
- [41] L. Torrey, J. Shavlik, Transfer learning, Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques 1 (2009) 242.
- 690 [42] J. Deng, et al., ImageNet: A Large-Scale Hierarchical Image Database, in: CVPR09, 2009.
- [43] 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–
 695 349. arXiv:<https://doi.org/10.1080/21681163.2015.1061448>, doi:
 10.1080/21681163.2015.1061448.
 URL <https://doi.org/10.1080/21681163.2015.1061448>