EB-Net to landmark 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 Libration, 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 program, and in recent years, it has risen strongly because of improvements in the computation performance. It has been applied to solve 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 new convolutional neural network model based on composition of elementary blocks of layers to predict key points (landmarks) on 2D anatomical biological images. Our proposed model has been trained and evaluated on a dataset including the images of 5 parts of 293 beetles. During the experiments, the network has been tested in two ways: training from scratch and applying fine-tuning process. The quality of predicted landmarks is evaluated by comparing the coordinates distance between predicted landmarks and manual ones which have been set by biologists. The final results have been provided to biologists and they have confirmed that

*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.

the quality of predicted landmarks is statistically good enough to replace the manual landmarks for the different morphometry analysis.

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

1. Introduction

In recent years, deep learning [1] is known as a solution for difficult tasks in different domains. It has been known as 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 output to 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 [2]. Currently, deep learning has many kinds of variant architectures and each of them has found success in such as: Deep Neural Network (DNN) to solve classification or data analysis problems[3, 4]; Convolutional Neural Network (CNN) in computer vision [5, 6, 7]; Recurrent Neural Network (RNN) on time sequences analysis [8, 9, 1, 10]. All of them have exhibited impressive performance comparing to more classical methods. In deep learning architectures, CNN is a specific network for pre-processing data which have grid topology, for examples, time series (1-D), 2D and 3D images, or video. From the first architecture [5] until now, many CNN models have been proposed and have succeeded in different tasks of computer vision such as image classification [5, 6, 7], object recognition [7, 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. In biology, the landmarks are most often provided by

the biologists. Depending on the objective of work and the studied 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. From landmarks coordinates, it is possible to extract object characteristics and to apply measure, for examples, to detect human face [14], human pose [17] or topology of objects in an organism in biology.

In this work, we propose a new composition of layers for a CNN architecture to predict the landmarks on biological species images. The proposed model has been trained on a dataset of 293 beetles images. We have also designed a specific procedure to augment our dataset because several hundred images are usually considered as a modest number to apply 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 deep learning and setting of landmarks on 2D images. Section 3 presents the method to augment our dataset. Section 4 explains the design of new network model. The first experiments of the network on each dataset are presented in Section 5. In the last section, we present a technique to improve the results of the proposed model: fine-tuning.

2. Related works

In the middle of the previous century, deep learning [1] have been introduced as a method for artificial intelligence applications. However, several problems appeared in order to take into account real-world cases because of the limitation of memory size or computing power. Nowadays, huge 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 domains such as **computer vision** [5, 6, 7, 11, 12], **speech recognition** [4, 3], **language translation** [8, 9], **natural language processing** [1, 10, 18], ... In computer vision, deep learning, specifically with CNN, has been used to achieve difficult tasks in image analysis such as image classification, objects or key points detection.

2.1. 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 variations, but in general, it consists of several types of layers: 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 achieve the output of the network from the application of a decision function.

Fig. 1 shows a classical example of a CNN, 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. Now, we will describe the different types of layers, readers familar to tem can jump directly to the next section.

Convolutional (CONV) layer is used 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

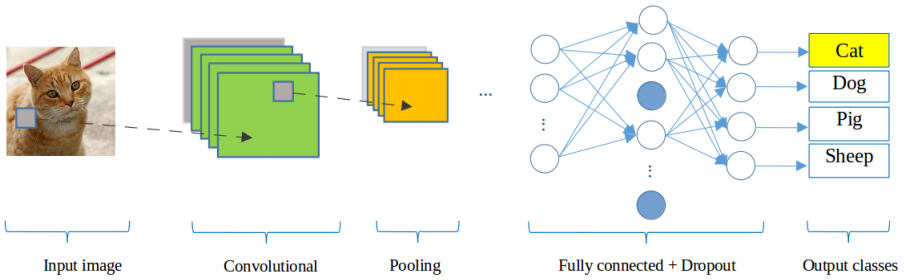


Figure 1: A CNN network for classification problem

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.

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 a common practice to use average pooling to propagate the average of all the inputs to the next layer. However, in more recent models [6, 19, 12], maximum pooling function 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 a stride of 2, if we apply to the yellow region an average pooling, the output will be 36.25 and a maximum pooling will return the value 122.

Dropout (DROP) [20] is a technique used to prevent the over-fitting during the training. The term dropout mentions dropping some output units and their connections (incoming and outgoing) belonging 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 [20]. So, training a neural network with dropout looks like training a collection of thinned networks. The dropouts layers are most often placed after the fully connected layers, but

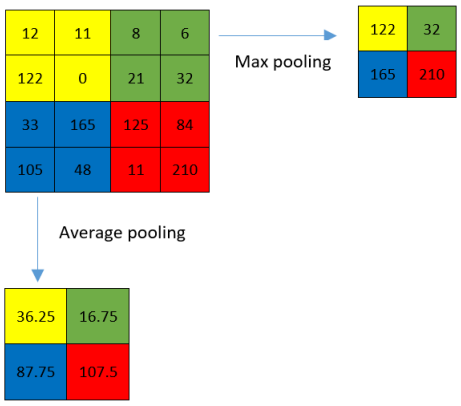


Figure 2: The results of different pooling

it is possible to use them after the pooling layers to introduce 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 choosing the activation function is depended on which kind of problem that network solves.

2.2. State of the arts in deep learning and key points detection

LeNet [5] model is considered as the first architecture of CNN. LeCun et al. [5] 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 two fully connected layers. But 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. In the last ten years, as the capabilities to compute have been drastically improved and in

the same time, a huge amount of data became available, new models of neural networks appear well adapted to this new environment. One of the first ones is AlexNet [6], which is similar to LeNet [5] but with a deeper structure: LeNet has 2 convolutional layers and 1 fully connected layer while AlexNet has 5 and 3, respectively. Furthermore, in AlexNet the activation functions have been changed 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 [21], GoogLeNet [7], VGGNet [22], or ResNet-50 [23]. The main difference between these networks is that their architectures became deeper and deeper by adding more layers, e.g. ResNet-50, which won the champion of ILSVRC 2015, is deeper 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 multiple tasks in the estimation step. Shaoli Huang et al. [17]

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

have introduced a coarse-fine network to locate 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 [7]) which are used to focus on capturing local characteristics and modeling spatial dependencies between 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.

From AlexNet period to ResNet-50, the obtained success stories [6, 23] have proved that CNN models produce better results on a large dataset. To use this technique, the size of dataset remains as a bottleneck. The next section is turn to the description of the method we have designed to augment the size of the dataset.

3. 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 operations like translation, rotation or scaling which are well known to be efficient to generate new version of existing images. However, this kind of operations are not useful in our case because the analysis of images by CNN (convoluted) are most often invariant to translation or rotation. So, we have designed another method to obtain misshapen images.

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

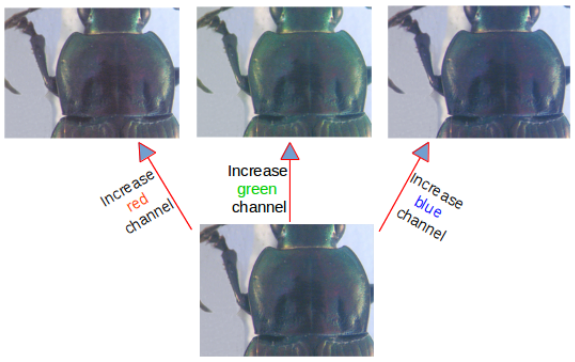


Figure 3: A constant $c = 10$ has been added to each channel of an original image

In the second procedure, each channel is considered separately and one gray image is generated for it (Fig. 4). Consequently, we obtain 3 new images (single channel) from an original one. At the end of the process, 6 versions of an original image are made. In total, the new data set contains $293 \times 7 = 2051$ images for each anatomical part of beetle (an original image and six misshapen ones).

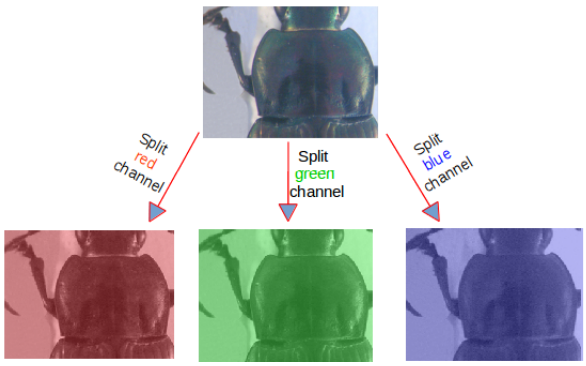


Figure 4: Three channels (red, green, blue) are separated from original image

4. Network architectures designing

As we have presented previously, several CNN architectures are available from literature and tools libraries. It is always possible to adapt them to a specific application by changing the parameters values or by modifying the arrangement of layers. By the way, several trials have been achieved before to obtain a satisfying model dedicated to landmarks estimation. In this section, we present three versions of the model that we have designed to solve this task. As usual, we have combined the classical layer types to build the model, i.e., convolutional, maximum pooling, dropout, and full-connected layers.

The first architecture has been a very classical one (Fig. 5). It receives an input image with the size of $(1 \times 192 \times 256)$, then it is composed by three repeated structures of a convolutional (CONV) layer followed by a maximum pooling (POOL) one. In most of CNNs, the parameters of CONV layers have been set to increase the depth of the images from the first to the last layer. This is done by setting the number of filters at each CONV layer. In this first model, the depths of the CONV layers increase from $32, 64$, to 128 and with different size of the kernels: (3×3), (2×2) and (2×2), respectively. Inserting POOL layers after a convolutional layers is usually done. The POOL layer

effects to progressively reduce the spatial size of the representation to reduce the number of parameters, computation in the network, and also to control over-fitting. The operation of POOL layers is independent for each depth slice of their inputs. In our model, we have used the most common form for one POOL layer: a filter with size of (2×2) and a stride of 2 pixels. At the end of the model, three FC layers have been added to extract the global relationship between the features and to proceed the outputs. The first two FC layers have been applied the activation functions to make sure these nodes interact well and to 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. Nevertheless, the obtained results with this architecture has not been considered as enough good to continue to use it. One of the main problems is the presence of over-fitting during the training process (Detailed results will be discussed in Section 5).

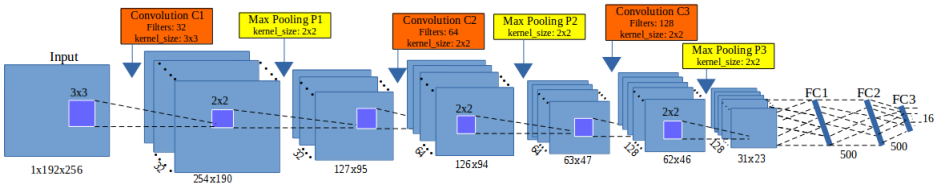


Figure 5: The architecture of the first model

The second model has kept the same architecture but the number of output of the two FC layers has been increased to 1000. Increasing the value at FC layers could allow to get more features from CONV layer without requirements of computing resources. However, the obtained results remained not satisfying, it will be discussed in the result section (Section 5).

To build the third architecture, we have defined a new concept: the *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. 6). The Dropout layer has been added to prevent over-fitting by addition of a step of removal of some nodes. This significantly reduces overfitting and gives major improvements over other regularization methods [20]. The final architecture is a composition of elementary blocks.

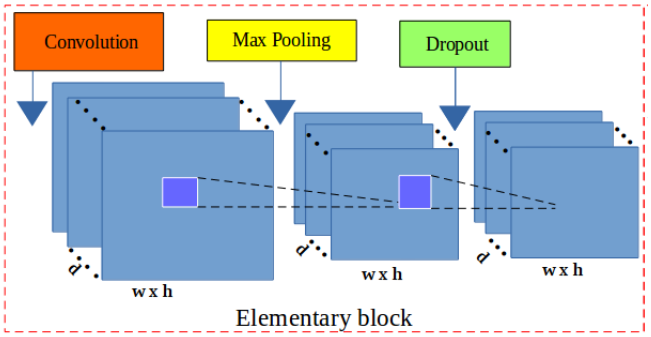


Figure 6: The layers in an elementary block. It includes a CONV layer (red), a maximum POOL layer (yellow) and a DROP layer (green).

Fig. 7 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 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 FC layers 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.

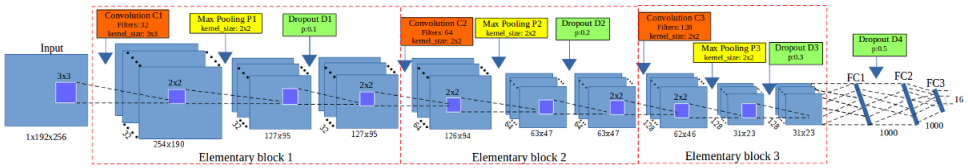


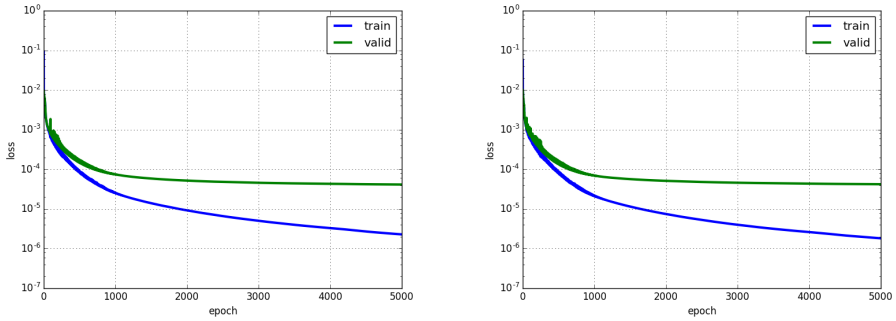
Figure 7: The architecture of the third model

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 reduce the loss in neural network. The core idea is to follow the gradient until to reach stable state. 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 to use the same 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] by applying parameters adjustment during the training. The three architectures implementations 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`

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

5. Experiments and results

This work is a part of a project about automatized morphology. The choice to turn to deep learning process has been motivated by the high difficulty to segment some parts of the beetle images and consequently to apply classical image processing methods. The pronotum was the first part we have analyzed with deep learning. The networks have been trained in $5,000$ epochs on Linux OS 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. For each image, a set of 8 manual landmarks are available. They have been set by biologists and are considered as the ground truth for the evaluation. In deep learning, many kinds of loss expressions can be considered depending on the class of problem solving by the network, for example, Root Mean Square Error (RMSE) is usually used for regression problems where the outputs are not discrete values. In the context of deep learning, landmark prediction can be seen as a regression problem because the coordinates of landmarks do not belong to discrete classes. Therefore, RMSE has been used to compute the losses of architectures during the training process.



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

Figure 8: The losses (training and validation) of the models

Fig. 8a 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, i.e., training losses are able to decrease but validation losses are stable. In the second model (Section 4), the parameters of full-connected layers have been modified to prevent the over-fitting but it seems that this solution is still not satisfying, the results are very similar to the previous ones (over-fitting is still appears).

Fig. 9 illustrates the losses during the training of the third model, one can note that after several epochs, the two-loss values become closed and the over-fitting disappears. The sequence of Dropout addition inside elementary block works well to prevent over-fitting and improve the accuracy of the model greatly. This third model has been selected to compute automatically landmarks.

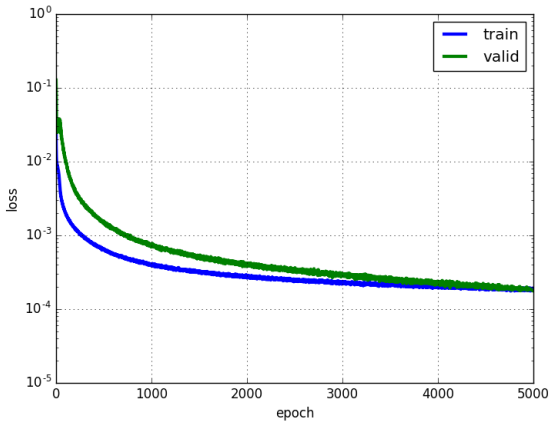


Figure 9: The losses (training and validation) of the third model

In order to extract predicted landmarks from all pronotum images, we have applied *cross-validation* procedure to choose the test images, we call it *round*. For each round, we have decided to choose 33 images for testing step. In order to predict landmarks for all available images, we will do 9 rounds. The 260 remaining images are used as training and validation images. Of course, this

dataset will be augmented as described before to provide 1820 images for these 2 steps. 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 and stable.

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 ones have been computed. Table. 2 shows the correlation scores of 3 metrics (using *scikit-learn* [26]), e.g. 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 good enough to replace manual landmarks in statistical analysis of pronotum morphology. However, standing on the side of image processing, seeing the real coordinates on images is more appropriate than statistical results. So, the distances (in pixels) between manual coordinates and predicted coordinates have been calculated for all images. Then, the average distance for each landmark has been computed.

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 the images resolution 256 × 192, we can consider that an error of 1% (corresponding to 2 pixels) 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 in the worse case, landmark 6.

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. 10 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: best results, the images having distance less than the average value (4 pixels): 56.66%; acceptable results, the images having the distance from the average value to 7 pixels (standard deviation error): 31.40%; and the images which are clearly in error with the distance greater than 7 pixels: 11.94%.

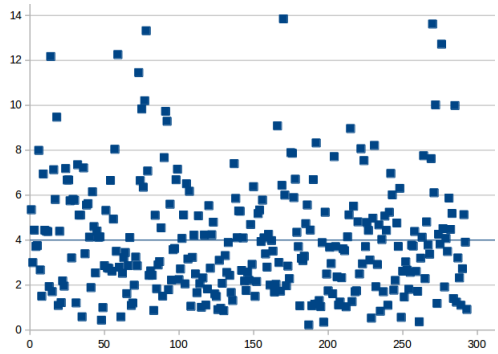
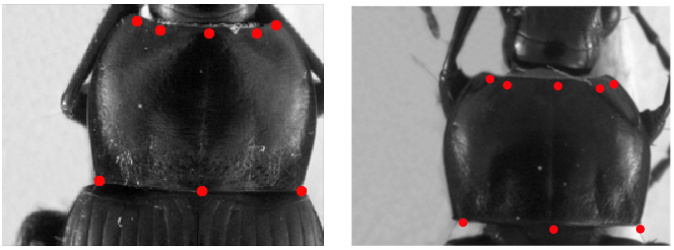


Figure 10: The distribution of the distances on the first landmark. The blue line is the average value of all distances.

To illustrate this purpose, Fig. 11 shows the predicted landmarks on two test images. One can note that even some predicted landmarks (Fig. 11a) are closed to the manual ones, in some case (Fig. 11b) the predicted ones are far from the expected results. 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 11: The predicted landmarks, in red, on the images in test set.

6. Resulting improvement by fine-tuning

The proposed network (third architecture) presented in Section 4 have been trained from scratch on five datasets of beetles (left mandible, right mandible, pronotum, elytra, and head). At this 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.

Training a network from scratch is not the only way to work in Deep learning. It is possible to initialize parameter values with values extract from another experiment with another dataset. It is called transfer learning [27]. In order to improve these results, we have broadened model with this technique. The hypothesis is that we decrease the risk to fall, the most popular example has

been given by the project ImageNet of Google [28] which has labeled several millions of images. The obtained parameter values have been used in another context to classify other datasets [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 the dataset to pre-train was not at all no huge than this one of ImageNet.

6.1. Data preparation and training

The images are combined from the training images of three sets: *pronotum, elytra, and head* (after augmentation). As previously, 9 rounds have been built from the images of each part to train the model from scratch. In this step, to create the pre-train dataset, we have just selected the train images from one round of each part. The final size of the training dataset is 260 (original images) $\times$ 7 (data augmentation) $\times$ 3 (parts: head, elytra and pronotum).

However, wanting to merge all beetle's parts in one dataset created a new problem: the number of landmarks is not the same for each part: 8 *landmarks on pronotum part,* 10 *landmarks on head part, and* 11 *landmarks on elytra part* (Fig. 12). Because of the meaning of landmarks on each anatomical part for biologists, we cannot insert new landmarks arbitrary. So, we have decided to keep the smallest number 8 of pronotum as reference and to remove some on elytra and head parts instead of adding. 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

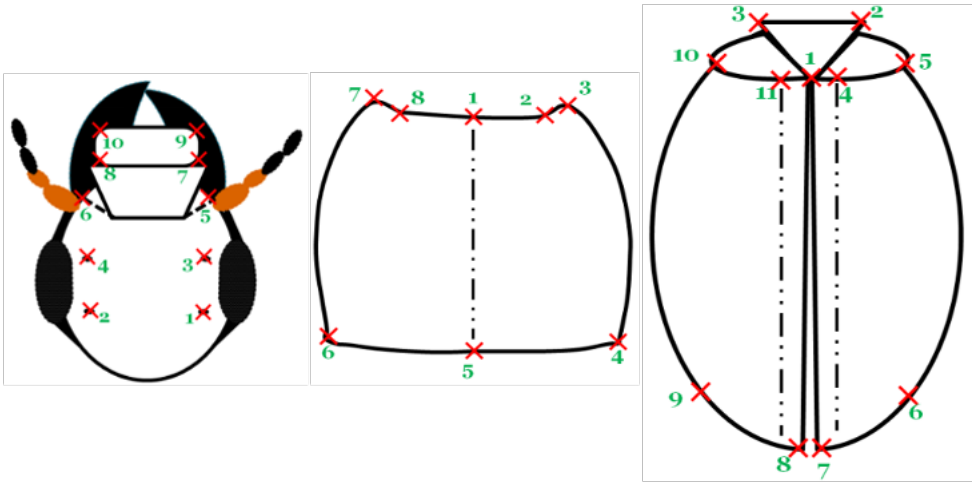


Figure 12: A presentation of head, pronotum and elytra part with corresponding manual landmarks

training from scratch but the number of epochs are increased to $10,000$ instead of $5,000$ to achieve better learning on the parameters. We have also shuffled the training data because it will be helped the model to work with different anatomical parts rather than the same anatomical samples in each training time.

6.2. Final result for all parts

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 the losses during fine-tuning. *For example*, Table. 4, 6, 8 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 man-

ual landmarks have been also computed. Tables. 5, 7, 9, 10, and 11 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. 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 4: 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 5: The average error distances per landmark of two deep learning processes on pronotum images

In another view, Fig. 13 shows the comparison of the average distance distributions on each dataset, the **blue** curves present for the average distances on each landmarks when we train the model from scratch and when we fine-tuned the model: **orange** curves. In the case of 2 mandibles (left and right), we have previously predicted landmarks with convolutional methods (hough transform, PCAI [30]), the black curves (Fig. 14) present these results in order to appreciate them. It is worth to remind that this method requires a segmentation step which has not been "applicable" on the other parts.

The fine-tuning process has improved the results of the proposed architecture

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 6: 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

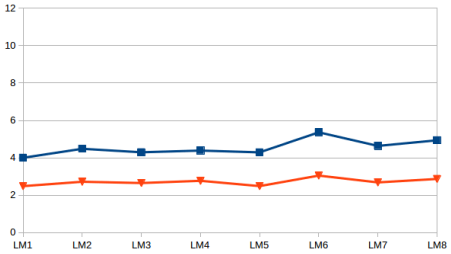
Table 7: The average error distances per landmark of two deep learning processes on elytra images

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 8: 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 9: The average error distances per landmark of two deep learning processes on head images



(a) Pronotum

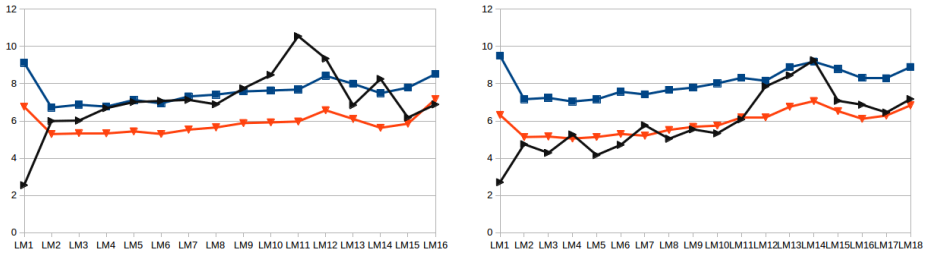
Figure 13: The distribution of average distances on each landmark of three beetle's parts. Blue and orange lines present for the results when train the model from scratch and apply fine-tuning, respectively.

#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 10: 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 11: The average error distances per landmark of two deep learning processes on right mandible images



(a) Left mandible

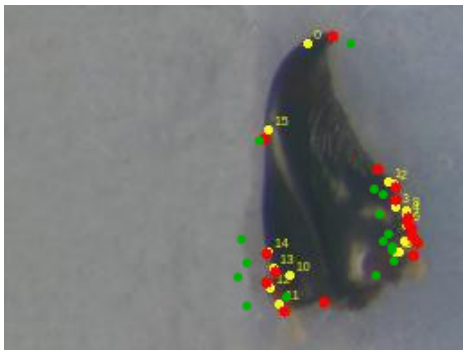
(b) Right mandible

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

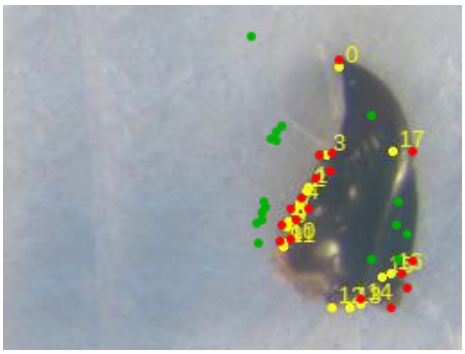
on both 5 datasets: left, right mandible, pronotum, elytra and head. All the average distances have significantly decreased: $\approx$ 26.9% on left mandible, $\approx$ 22.8% on right mandible, $\approx$ 40.3% on pronotum, $\approx$ 39.8% on elytra, and $\approx$ 46.4% on head part based on considering 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 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.

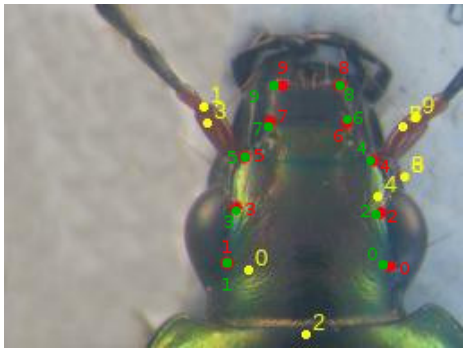
Fig. 16 shows the distances by landmarks when we did a test on one image of each part. In these charts, the **black lines** present the distances when we apply the calculation on image processing procedures; the **blue and orange lines** show the distances with deep learning: training from scratch and fine tun-



(a) Left mandible

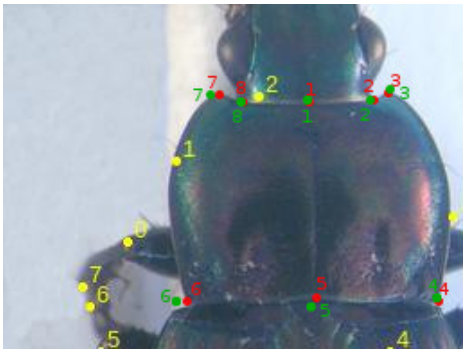


(b) Right mandible



(c) Head

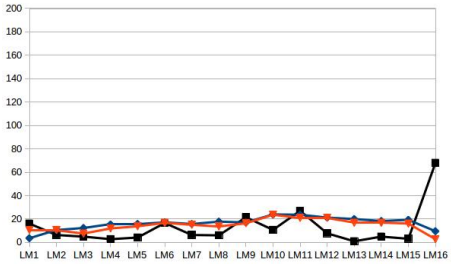
(d) Elytra



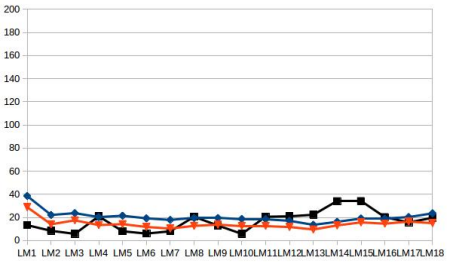
(e) Pronotum

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.

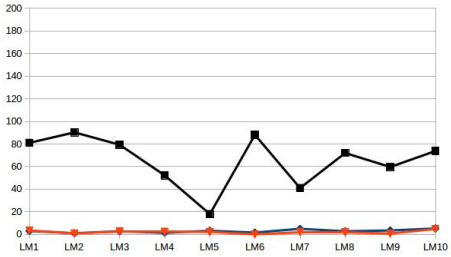
ing, respectively. As described in [30], we have combined some image processing procedures to output the predicted landmarks and this has become also a disadvantage of this method. If one procedure of them provides a bad result, it will affect the final result, i.e. if the result of segmentation step 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.



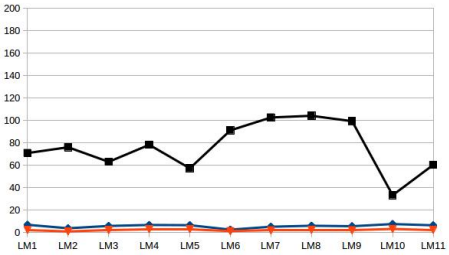
(a) Left mandible



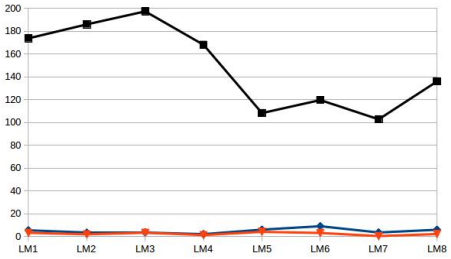
(b) Right mandible



(c) Head



(d) Elytra



(e) Pronotum

Figure 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.

7. Conclusion

In this work, we have presented how to apply convolutional neural network to predict the landmark on 2D anatomical images of beetles. After 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

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

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

[3] 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.

[4] 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.

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

[6] 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.

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

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

[9] 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.

[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] 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.

[18] 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.

[19] 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.

[20] 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.

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

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

[23] 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.

[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