Fine-tuning CNN on Beetles dataset

LE Van Linh and BEURTON-AIMAR Marie

March, 2018

Abstract

In this study, we continue applying the convolutional neural network (CNN) to predict the landmarks on other parts of beetles, i.e, pronotum, head and body. The proposed model was continuing to use to predict the landmarks. However, instead of training from scratch on each dataset, the model has been training with a combined data of three parts: pronotum, head and body part (called training stage); then, the trained model is used to fine-tune on each dataset and predict the landmarks (called fine-tuning stage). In training stage, 260 images have been chosen randomly on each part. Then, they are enlarged to obtain 5460 images ($260 \times 7 \times 3$) which are used to train the network. In fine-tuning stage, the trained model has been fine-tuned on each dataset. After fine-tuning, the model is used to predict the landmarks of the images in the test set. The coordinates of predicted landmarks are evaluated by calculating the distance between manual landmarks and corresponding predicted landmarks. The model is implemented by Python on Lassagne framework[1].

1 Model architecture and training stage

1.1 Architecture

The network includes three convolutional(CONV) layers followed by three maximum pooling(POOL) layers, four dropouts(DROP) layers, and three full connected(FC) layers (Fig.1). The input of the network is the gray-scale image with the size of 256×192. The depth of network can be expressed by increasing of the deep at each convolutional layer. They are increased from $32, 64$, and 128 from the first CONV layer to the third CONV layer with different filter sizes. While, the filter sizes are kept in the same size for every POOL layers. The dropout ratios of the DROP layers increase from the first to the end: $0.1, 0.2, 0.3$, and 0.5. At the end of the network, three full connected are set up to predict the landmarks. The first two FC layers have the same outputs(1000) while the output at the last FC has been change to correspond with the number of landmarks. The detail parameters at each layer are presented in Appendix **??**. The model is implemented by Lassagne framework[1].

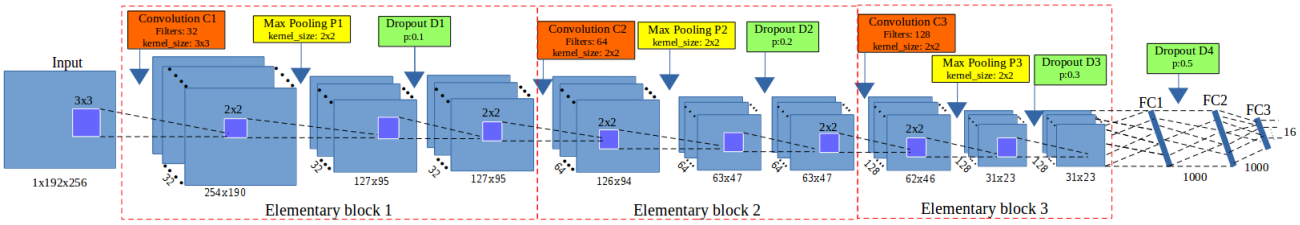

Figure 1: The illustration of the convolutional neural network

1.2 Parameters

The model is trained with 10000 `epochs` and `batch size` of 128 (default value of Lasagne). For each epoch, the dataset is randomly split into training set and validation set with the ratio of 0.6 : 0.4. The `learning rate` and `momentum` are initialized to 0.01 and 0.9, respectively. During training, they are re-calculated to adjust with the remaining epochs. All the initial parameters are shown in the Table 1.

Parameter	Initial value	End value
Epochs	10000	
Training batch size	128	
Testing batch size	128	
Learning rate	0.01	0.0001
Momentum	0.9	0.9999

Table 1: The network parameters in proposed model

1.3 Training data

The training data is combined from the images of three parts: *pronotum, body, head*. For each part, 260 original images have been chosen randomly ($260 \times 3 = 780$). Then, the number of images have been enlarged by applying two procedures.

The **first** procedure is adding a constant value to a channel of RGB image, we will have a new RGB image. For example, from an original RGB image, if we add 10 to red channel, we will have a new image $(R+10)GB$. Then, we apply the same rule with blue and green channel, we will obtain two new images: $R(G+10)B$ and $RG(B+10)$. By that way, from an RGB image, we can generate three RGB images by adding a constant to each channel(each time just change to a channel).

The **second** procedure is splitting the channels of RGB image (because the models work on gray-scale). It means that we can generate six versions from an original image. At the end, the number of the image in the training data is $260 \times 3 \times 7 = 5460$ images (six versions and original). Before giving to the models, the images are down-sampled with the size of 256×192. The number of the images in training set and validation set are splitted automatically by the model's parameter.

After generating the images, the dataset is large enough to train with CNN. However, another problem has occurred that the number of landmarks of each part is different: *8 landmarks on pronotum part, 11 landmarks on body part, and 10 landmarks on head*. Because the location of the landmarks has a signification with biologists, so, adding the landmarks are more difficult than removing the landmarks. Therefore, we kept the number of landmarks on pronotum as a reference and we suppressed the landmarks on body and head part. Specifically, we have removed 3 landmarks on body part $(1^{st}, 6^{th}, 9^{th})$ and 2 landmarks on head part$(5^{th}, 6^{th})$.

1.4 Training process

The model has been trained on the dataset including the images of 3 parts. The training process was finished in 10000 epochs. Figure.2 shows the losses during training process (training loss and validation loss). At the beginning, the validation loss is always higher than the tranining loss, but from the 2000 epochs, the training loss begins stable while the validation loss continue to decrease. At the end of training, the losses values are 0.00029 and 0.00009 for training and validation, respectively.

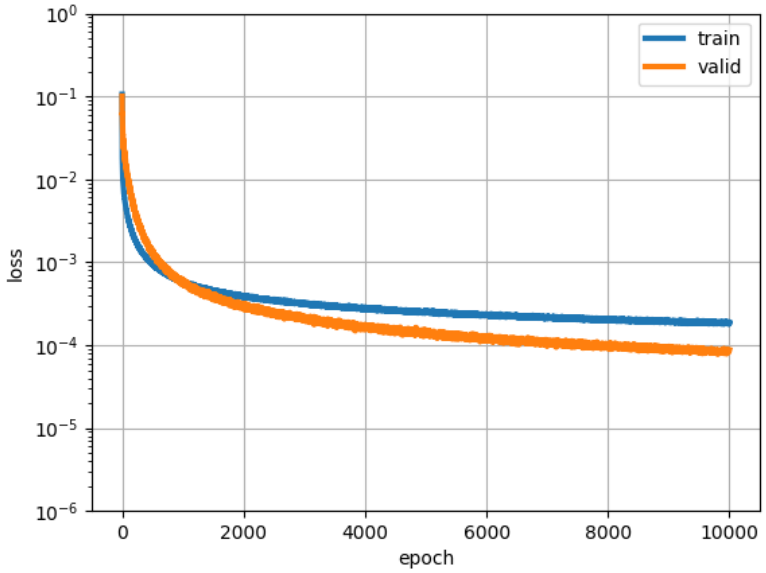

Figure 2: The losses during training process

2 Fine-tuning stage

The trained model have been continued to fine-tune[] on each dataset: pronotum, body, and head. For each part, to get all predicted landmarks for all images, a scenario (called round) has been applied to choose 33 images for the test set, the remaining images have been put to the training set. After fine-tuning, the model has abled to predict the landmarks corresponds to each part of beetles.

In the next, each sub-section includes two tables: the first table presents the losses during fine-tuning on each dataset; the second table presents the comparison based on the average distances (in pixels) of two different procedures (fine-tuning and training from scratch). The average distances are given by each landmarks.

2.1 Pronotum dataset

Table.2 shows the losses during fine-tuning the trained model on pronotum dataset.

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 2: The losses during fine-tuning model on pronotum dataset

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

Table 3: The average error distance per landmark.

To have a comparison with the last result (training from scratch), the fine-tuned model has

been used to predict the landmarks on protum images. Then, the distance between the manual landmarks and corresponding predicted landmarks has been computed. Table.3 shows the average distances which given by each landmark. The values is **Fine-tune** columns presents for the average distance of fine-tuned model, while **From scratch** columns presents for the average distance of the last result (train model from scratch on pronotum dataset).

2.2 Body dataset

Table.4 shows the losses during fine-tuning the trained model on body dataset. Table.5 shows the comparison on average distance between two periods.

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 4: The losses during fine-tuning model on body dataset

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

Table 5: The average error distance per landmark.

2.3 Head dataset

Table.6 shows the losses during fine-tuning the trained model on head dataset. Table.7 shows the comparison on average distance between two periods.

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

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

Table 7: The average error distance per landmark.

3 Conclusions

In this study, we proposed a CNN to predict the landmarks on beetles images. The model is trained on a combining data of 3 parts: pronotum, head, and body. Then, the trained model has been used to fine-tune on each dataset. The fine-tuned models are used to predict the landmarks on the correspoding testing dataset. The coordinates of predicted landmarks are evaluated by calculating the distance to the manual ones. The study presents also a comparison between two different procedures for each dataset (training from scratch and fine-tuning). The experiments result can be download from GitHub[1]. The explaination about structure are given in appendix part.

References

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

[1] https://github.com/linhlevandlu/cnnBeetles

APPENDIX

Experiment results

- **data** folder: contains the results when we train the model from scratch (the last result). All of the experiment results is provided on GitHub. Each folder contains the results corresponding to each beetle's part. In each folder, we provide the losses curves, result images(first sixteen images of each test set), and the prediction landmarks(in csv file). Each folder contains:
  - **Test** folder: contains the loss curves during training (and validation) and the prediction on test images (top 16) for each training round,
  - **CSV** file: contains the information of image, index of landmark, coordinates of manual landmark($x1, y1$), and coordinates of predicted landmark($x2, y2$).
  - The output trained model can be obtained by requesting the authors.
- **fine_tuning** folder: contains the result when we **fine-tune the trained model** on each separate dataset. It includes:
  - **csv** folder: contains the distances between the predicted and manual landmarks of all images, which given by each landmark.
  - **data** folder: contains some predicted examples (*images*) and the coordinates of all predicted landmarks (*predicted_landmarks* folder)
  - **ods** folder: contains all **ods** files which provide the statistic processes.
- **latex** folder: contains the latex source of summary documents.