Morphometry landmarks detection by convolutional neural network

Van Linh LE
LaBRI - CNRS 5800, France
ITDLU - Dalat University, Vietnam
van-linh.le@labri.fr/
linhlv@dlu.edu.vn

Marie BEURTON-AIMAR
LaBRI - CNRS 5800
Bordeaux University
Talence, France
beurton@labri.fr

Akka ZEMMARI
LaBRI - CNRS 5800
Bordeaux University
Talence, France
zemmari@labri.fr

Nicolas PARISEY
IGEPP
INRA 1349
Le Rheu, France
nparisey@rennes.inra.fr

***Abstract*—Morphometric analysis is general method applied on organisms and are useful to appraise the covariances between the ecological factors and the organisms(shape, size, form,...). In which, landmark-based morphometric is known as one of the approaches to analyze the characteristics of organisms. Finding enough the landmarks can give to the biologist a comprehensive description of organism shape. In this study, we propose a convolutional neural network (CNN) to predict the landmarks on biological images. The network is designed as a pipeline of the layers, it was trained with a set of manually landmarks examples. Then, the network is used to provide the morphometric landmarks on biological images automatically. The coordinates of predicted landmarks are evaluated by calculating the correlation coefficient with the manual coordinates which given by the biologists. Besides, the evaluations of the distances between predicted and manual landmarks are also given. The network is implemented by Python on Lasagne framework.**

***Index Terms*—Morphometry, biological, landmarks, CNN**

I. Introduction

Morphometry analysis refers to measure the topography of an object, a notion that includes the shape and size. Morphometry analysis is generally applied to organisms. In biology, the biologist can work with several pieces of information from organisms such as lengths, widths, masses, angles,... to analyze the interaction of environment to the developmental of organisms. Besides the traditional information, the landmark is known as one of the characteristics to analyze the shape. Instead of collecting all information, the shape is determined by a finite set of points, called landmarks. The landmarks are the points that store the important information about the shape of the object, *for example*, four corners of the rectangle are four landmarks of a rectangle. Normally, the landmarks are along on the outline of the object but in some special cases, it has been defined inside the object. Morphometry landmarks are a kind of points-of-interest, they are directly linked to the animal anatomy. In our study, the morphometric landmarks are specific points defined by the biologists. They are used in many biological studies and include the classification tasks. Manual landmarks identification is time-consuming and difficult to re-procedure. Therefore, a method that gives automatically the location of landmarks is very interesting.

This work introduces a method for automatic detection the landmarks on biological images. The main idea consists design and train a convolutional neural network[1] with a set of manual landmarks. By this way, the trained network will be able to detect the morphometry landmarks on biological images. The dataset that used to study including 293 beetles images from Brittany lands. All the images are presented in RGB color with two dimensional. For each beetle, the biologists took images of five parts: *left and right mandibles, head, body, and pronotum*. For every image, a set of landmarks has been manually determined by experts. In the last our work, a method has been presented to determine the landmarks on left and right mandibles[2]. This method is based on the image processing techniques[3], combining with principal component analysis[4] and SIFT descriptor[5]. In the context of this work, we work on the dataset of pronotum images (Fig.1). For each pronotum image, a set of 8 manual landmarks have been set by the biologists. The coordinates of manual landmarks were used as the input to train the network. During the first phase, a number of 260 images and their manual landmarks were used to train and validate the network. The remaining images were used to evaluate the output model of the network in the second phase.

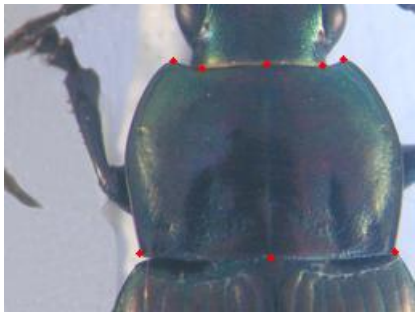

Fig. 1. An example of pronotum with manual landmarks

In the next section, we study several related works to determine the landmarks on 2D images by CNN. Section 3 presents the architecture of the network, its parameters, and the implementation. All the experiments and analysing the results will be detailed in section 4.

II. Related works

In recent years, deep learning is known as a solution for computer vision literature. Using convolutional network to learn the vision features or to detect the important features on the images have achieved better results in many domains such as image classification[6][7], face detection and pose estimation[8]–[11], handwritten detection[12][13] and key points (landmarks) identification [14]–[16]. The landmarks are not only used in biology, they appear in many domains with many applications. Until now, most of the presented methods for identifying the landmarks that based on the image processing techniques[17][18][2]. In this section, we introduce some studies that have applied the convolutional neural network to predict the landmarks automatically on 2D images.

In the field of facial keypoint detection, Yi Sun et al.[14] proposed cascaded convolutional networks to predict five facial points (points stay on the human face): *left eye center, right eye center, nose, left mouth corner, and right mouth corner*. They cascade three levels of the convolutional networks to predict the facial points: In the first level, the networks are designed to predict several landmarks together by covering the whole face; the networks in the second and the third level are used to predict each landmark on the face. They take the patches centered at the predicted positions of previous levels as input and try to improve the accuracy of predicted positions. Zhanpeng Zhang et al[15] proposed a *Tasks-Constrained Deep Convolutional Network* to optimize facial landmarks detection. The model determined the facial landmarks with a set of related tasks such as head pose estimation, gender classification, age estimation, face recognition, or facial attribute inference. Firstly, all the images are used as the input, their output spaces the images into several tasks. Then, the network is applied to determine the landmarks on the images of each task.

In biology field, Cintas et al[16] has introduced a network to predict the landmarks on human ears. The network was designed to receive the images with the size of 96×96 as the inputs. After training, the network has the ability to predict 45 landmarks and semi-landmarks on human ears. In their proposed architecture, the network with three times repeated of a structure includes two convolutional layers with the filters, followed by maximum pooling and dropout layers. The structures then adding two full-connected layers and a dropout layer. At the end, an output layer with 90 output units corresponding with 45 landmarks is hired to provide the position of the predicted landmarks.

III. Method and implementation

In previous section, we have introduced the methods using CNN to predict the landmarks. In this section, we describe the architecture of proposed network that use to predict the landmarks on biological images. Besides, the pratical techniques that using in deep learning are also recommended.

A. Convolutional neural networks

Deep learning allows computational model composed of multiple processing layers to learn representations of data with multiple levels of abstraction[19]. Each layer extracts the representation of the input data from the previous layer and computes a new representation for the next layer. In the hierarchy of model, higher layers of representation enlarge aspects of the input that is important for discrimination and suppress irrelevant variations. Each level of representations is corresponding to the different level of abstraction. Almost the algorithms in deep learning learn are supervised (i.e classification) or unsupervised (i.e pattern analysis) practices. During training, it uses the forms of gradient descent to update the learnable parameters via backpropagation. The development of deep learning open the promises result of the artificial intelligent on hight dimensional data, therefore applicable to many domains: image recognition and classification[6], [7], [20], speech recognition[21]–[23], question answering[24] and language translation[25][26].

Convolutional neural networks (CNNs) contributes to resolve the state of the art in many computer vision such as classification[6][7] or recognition[8][27]. Normally, a CNN consists a number of connected layers. The layers of CNN has neurons arranged in three dimensions: *width, height, and depth* with learnable parameters. The common layers in CNN are *convolutional and pooling* layers. *Convolutional* layer computes a dot product between their weights and a small region in the input. At the output, the results of connected local regions are combined. Convolution layer uses a set of learnable filters as parameters. Each filter is small spatially but extends the depth of the input. *Pooling* layer uses to reduce the spatial size of the input. It usually to down-sampling the input. The main purpose of this layer is to reduce the computational cost in remaining layers, reducing the spatial of feature maps and control overfit.

From the beginning of deep learning until now, many deep learning frameworks have been developed. Almost the framework was served a specific aim and was open source. According to the written programming languages, the frameworks can be separated into two main groups: C++ and Python. In the group written by C++, the outstanding frameworks include Caffe[28], Deeplearning4j, Microsoft Cognitive Toolkit[29]; whereas Keras[30], Theano[31] are known as the good frameworks that implemented by Python. Besides the main groups, we also have some frameworks that written in both programming languages such as Caffe2[28], TensorFlow[32]. In these frameworks, Caffe, Theano, and TensorFlow are known as good frameworks for convolutional neural networks. They provide the details of install, documentation with many examples in different domains such as classification, digit recognition.

Theano is an open source developed by machine learning group at the University of $Montréal$. It is a Python library that allows to define, optimize and evaluate mathematical expressions relating multi-dimensional arrays efficiently by using a Numpy package. Theano supports compiled on either CPU

or GPU architectures. Lasagne[33] is a lightweight library in Theano. It allows to build and train the neural networks. In this work, we used Lasagne to implement the proposed neural network.

B. Data and preprocessing data

The images come from a collection of 293 beetles from Brittany lands. All the images are taken with the same camera under same conditions with a 3264×2448 resolutions. For each specific part, a set of manual landmarks has been determined by biologists, *for examples, 8 landmarks for pronotum, 16 landmarks for the left mandible, 18 landmarks for right mandible*. In the content of this study, we work on pronotum part of beetle. The provided dataset contains 293 images, each image with 8 landmarks provided by biologists. The dataset was split into a training set with 260 images (training and validation) and a testing set of 33 images. During the training, the network learned the information through a pair of *(image, landmarks)* in training set. At the testing phase, the image without landmarks was given to the trained network and the predicted landmarks will be given at the output.

Because the resolution of the image is large, it becomes a difficulty for the network. During training and testing, the images are down-sampling to the new resolution of 256×192. Certainly, the landmark coordinates of the image are also scaled to suit their new resolution. In practical when we work with CNN, convergence is usually faster if the average of each input variable over the training set is close to zero. Because the values of the pixels and the coordinates of the landmarks are positive. If we consider that we stay at a layer of the network, and the weights are updated by an amount proportional to δx(δ is the scalar error at the layer and x is the input vector). When the input vectors are positive, the updates of weights that feed into the layer will be the same sign($sign(\delta)$), it means that the weights can only all decrease or all increase together for a given input. That, if the weight vector change direction, it can only do by zigzagging which is inefficient and thus slow down learning. Therefore, it is good to shift the inputs so that the average over the training set is close to zero. Moreover, when the input is set closed with zero, it will more suitable with the sigmoid activation function[34]. Therefore, the brightness of the image is normalized to $[0, 1]$, instead of $[0, 255]$. And, the coordinates of the landmarks are normalized to $[-1, 1]$, instead of $[0, 256]$ and $[0, 192]$ before giving to the network.

C. Network architecture and training

The CNN was designed and trained for performing an automatic landmarks detection on pronotum images. The network consists several common layers in the neural network with different learnable parameters. It takes a single channel pronotum image with the size of 256×192 as the input. Before entering the network, the brightness of pixels in the image are normalized to $[0, 1]$ and the coordinates of landmarks are normalized to $[-1, 1]$. Fig.2 shows the architecture of proposed network. The structure consists of a convolution layer with square filter, followed with a *maximum* pooling and dropout layer. This structure is repeated three times in the architecture with the differences of filter size, number of features and probability values. The difference of layer parameters provides to the network the ability to study the features at different levels. In this structure, all *max* pooling layers have the same size 2×2 for the filters, while the number of filters and the size of filters at convolution layers are different. The number of filters is increased from the first convolution layer to the third convolution layer. The number of filters are 32, 64, and 128, respectively. Besides, the size of filters are also not the same: 3×3, 2×2, and 2×2. The probability values used for dropout layers are 0.1, 0.2, and 0.3.

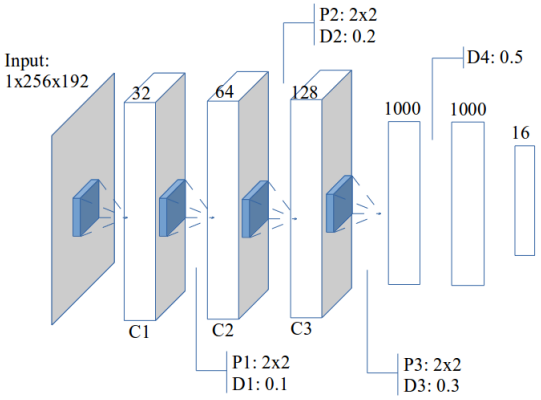

Fig. 2. The architecture of proposed network

After three feature extraction structures, the architecture contains two full connected layers with 1000 units per each layer and a dropout layer between them. The probability value of this layer is set to 0.5. The output layer contains 16 units corresponding to the coordinates of eight predicted landmarks. The detail of parameters at each layer are shown in Appendix section. The implementation of this architecture used Python on Lasagne framework[33] which allows to train the network on GPU. The training process took around 3 hours using NVIDIA TITAN X cards. The design of the network is available on GitHub[1].

D. Overfitting problem

The proposed network has a large number of learnable parameters. In addition, the size of the dataset is limited, this means that overfitting will occur during the training process. Because the network works on gray-scale image and mostly the processes compute the value from the pixels. So, we have applied two rules to enlarge the size of the dataset.

The first rule was applied to change the value of each channel in the original image. According to this rule, a constant is adding to a channel of RGB image and for each time, we just change the value of one of three channels. For example, from an original RGB image, if we add a constant $c = 10$ to the red channel, we will obtain a new image with the values at red channel are greater than the red channel of

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

original image a value of 10. By this way, we can generate three new RGB images from a RGB image.

The second rule is splitting the channels of RGB images. It means that we separate the channels of RGB into three gray-scale images. This work seems like right because the network works on single-channel images. At the end, we can generate six version from an image, the total number of images that used to train and validate is $260 \times 7 = 1820$ images (six versions and original image). The number of images that used for training and validation is split randomly by a ratio that has been set during setup the network.

Besides enlarge the dataset, we also used dropout layers to prevent the overfitting. The idea of dropout is to randomly drop units from the neural network during training. It deters units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" network. At test times, it is easy to approximate the effect of averaging the prediction of all thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods[35].

IV. Results

The dataset has been built by the biologists. It includes the images and manual landmarks. So, we can use the manual landmarks coordinates as ground truth to evaluate the coordinates of predicted landmarks. In the context of deep learning, landmark prediction can see as the regression problem. Therefore, the quality metric is used to evaluate the results. In particular, we use root mean square error (RMSE) to compute the accuracy of the implemented architecture. To have the predicted coordinates of all landmarks in all the images, the network was trained many times with different training dataset and was tested on the corresponding test set.

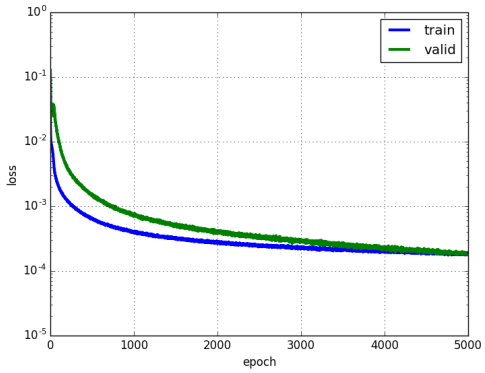

Fig. 3. Learning curves of the network. The blue curve presents RMSE on training set, the green curve presents the validation error

The learning curves in Fig.3 show the training error and the validation error of one training (and validation) time of the network. Fig.4 shows the predicted landmarks on an unseen image in the test set.

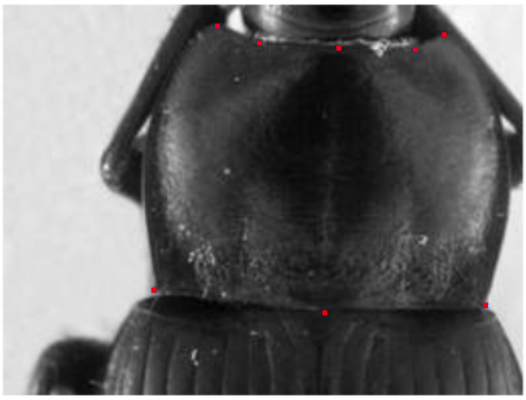

Fig. 4. The predicted landmarks on an image in test set (red points)

For a comprehensive evaluation of the efficacy of the network. The correlation coefficient on the coordinates of predicted landmarks is calculated by using three methods: Pearson[36], Spearman[37], and Kendall[38]. These results are shown in Table.I.

TABLE I
THE CORRELATION COEFFICIENT OF PREDICTED LANDMARKS ON SEVERAL CORRELATION METHODS

Method	**x-correlation**	**y-correlation**
Pearson	0.9970585	0.9978605
Spearman	0.9942475	0.9859642
Kendall	0.9430501	0.9067739

Besides the coefficient, the distance from predicted landmarks to manual landmarks deserves attention also. Firstly, the distance between them is calculated. Then, the standard deviation[39] is used to quantify the dispersion of a set of distance. Table.II shows the average error distance given on each landmark. Fig.5 shows the proportion of acceptable landmarks. In our case, a predicted landmark is acceptable if the distance between it and corresponding manual landmarks is less than the average distance plus a value of standard deviation. Most of the landmarks have been detected with the accuracy greater than 70%. However, we can see a vast difference between the correlation coefficient results and the proportions on each landmark.

TABLE II
THE AVERAGE DISTANCE ON EACH LANDMARK

#Landmark	**Distance**
1	4.002
2	4.4831
3	4.2959
4	4.3865
5	4.2925
6	5.3631
7	4.636
8	4.9363

At the test phase, the trained network is used to predict the landmarks on a set of unseen images (test set). With a bit of

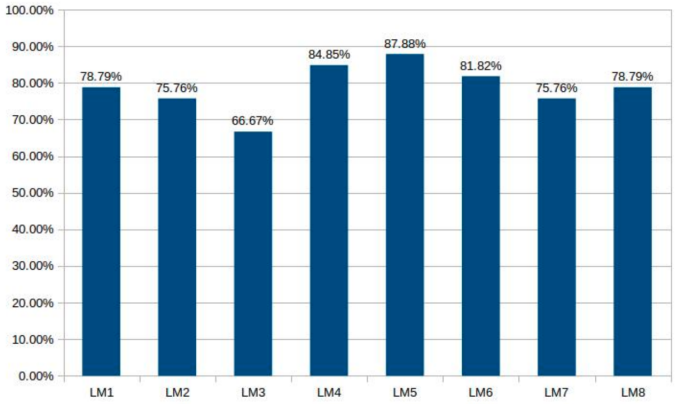

Fig. 5. The proportion of acceptable predicted landmarks

Python code, the program outputs the predicted-landmarks of the images as TPS files; in additional, it also fills and displays the predicted-landmarks on sixteenth firstly images of test data. With the outputs are TPS files, the user can use MAELab[2] framework[2] to display the landmarks on the images.

V. Conclusion and Future Works

We present a method for landmarks prediction on beetle images, specifically, pronotum images. The method used the convolutional neural network for automatic detection landmarks. It includes three times repeated of a structure consists of a convolutional layer, a max pooling layer, and a dropout layer, followed by the connected layers. During the training phase, the techniques are used to prevent overfitting, a common issue of the neural network. The network was trained in several times in different selection of training data. After training with the manual landmarks which given by the biologist, the network is possible to predict the landmarks on the set of unseen images. The model is implemented using open source tools and available on GitHub.

The results from the testing period are evaluated by several different methods: The correlations have been used to calculate the coefficients on coordinates of predicted landmarks. Besides, the distance between manual landmark and corresponding predicted-landmark were also computed. Then, the standard deviation based on the distance is used to correct the acceptable landmarks. The average of distance errors on each landmark was also considered. Some of the methods have been given the good coefficient. The quality of prediction can be used to replace for manual landmarks in some aspect.

Finally, using the convolutional network to predict the landmarks on biological images promising the good results. However, when we expect more about the accuracy of predicted landmarks, the result of this work is not enough. Therefore, future research in landmarking identification appears as a improve the worth exploring.

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

References

[1] Y. LeCun, K. Kavukcuoglu, and 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.

[2] V. L. LE, M. BEURTON-AIMAR, A. KRÄHENBÜHL, and N. PARISEY, "MAELab: a framework to automatize landmark estimation," in *WSCG 2017*, ser. 25th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2017, Plzen, Czech Republic, May 2017. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01571440

[3] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[4] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

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

[7] D. Ciregan, U. Meier, and 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.

[8] H. Li, Z. Lin, X. Shen, J. Brandt, and 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.

[9] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[10] S. S. Farfade, M. J. Saberian, and L.-J. Li, "Multi-view face detection using deep convolutional neural networks," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. ACM, 2015, pp. 643–650.

[11] M. Osadchy, Y. L. Cun, and M. L. Miller, "Synergistic face detection and pose estimation with energy-based models," *Journal of Machine Learning Research*, vol. 8, no. May, pp. 1197–1215, 2007.

[12] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.

[13] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in neural information processing systems*, 2009, pp. 545–552.

[14] Y. Sun, X. Wang, and 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, P. Luo, C. C. Loy, and X. Tang, "Facial landmark detection by deep multi-task learning," in *European Conference on Computer Vision*. Springer, 2014, pp. 94–108.

[16] C. Cintas, M. Quinto-Sánchez, V. Acuña, C. Paschetta, S. de Azevedo, C. C. S. de Cerqueira, V. Ramallo, C. Gallo, G. Poletti, M. C. Bortolini *et al.*, "Automatic ear detection and feature extraction using geometric morphometrics and convolutional neural networks," *IET Biometrics*, vol. 6, no. 3, pp. 211–223, 2016.

[17] S. Palaniswamy, N. A. Thacker, and C. P. Klingenberg, "Automatic identification of landmarks in digital images," *IET Computer Vision*, vol. 4, no. 4, pp. 247–260, 2010.

[18] A. Kaur and C. Singh, "Automatic cephalometric landmark detection using zernike moments and template matching," *Signal, Image and Video Processing*, vol. 9, no. 1, pp. 117–132, 2015.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[21] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černockỳ, "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.

[22] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[23] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8614–8618.

[24] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," *arXiv preprint arXiv:1406.3676*, 2014.

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

[26] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, "On using very large target vocabulary for neural machine translation," *arXiv preprint arXiv:1412.2007*, 2014.

[27] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Advances in neural information processing systems*, 2014, pp. 1799–1807.

[28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[29] D. Yu, A. Eversole, M. Seltzer, K. Yao, Z. Huang, B. Guenter, O. Kuchaiev, Y. Zhang, F. Seide, H. Wang *et al.*, "An introduction to computational networks and the computational network toolkit," *Microsoft Technical Report MSR-TR-2014–112*, 2014.

[30] F. Chollet *et al.*, "Keras," 2015.

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

[32] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[33] S. Dieleman, J. Schlter, C. Raffel, E. Olson, S. K. Snderby, D. Nouri *et al.*, "Lasagne: First release." Aug. 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.27878

[34] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[35] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[36] J. Pallant, *SPSS survival manual*. McGraw-Hill Education (UK), 2013.

[37] J. L. Myers, A. Well, and R. F. Lorch, *Research design and statistical analysis*. Routledge, 2010.

[38] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[39] J. M. Bland and D. G. Altman, "Statistics notes: measurement error," *Bmj*, vol. 313, no. 7059, p. 744, 1996.

APPENDIX

The proposed network consists 13 layers of convolutional, max-pooling, dropout and full-connected layers. The layers are interleaved arrangement as discussed in section III. In which:

- Input size: $1 \times 256 \times 192$
- Convolutional layers: 3 layers
- Max pooling layers: 3 layers
- Dropout layers: 4 layers
- Full-connected layers: 3 layers

The detail parameters in each layer are shown in Table III.

TABLE III
THE PARAMETERS AT EACH LAYER OF THE PROPOSED NETWORK

Layers	Type of layer and parameters
input	$1 \times 256 \times 192$
layer 1	CONV(32,3,1,0)
layer 2	POOL(2,2,0)
layer 3	**DROP(0.1)**
layer 4	CONV(64,2,1,0)
layer 5	POOL(2,2,0)
layer 6	**DROP(0.2)**
layer 7	CONV(128,2,1,0)
layer 8	POOL(2,2,0)
layer 9	**DROP(0.3)**
layer 10	FC(1000)
layer 11	**DROP(0.5)**
layer 12	FC(1000)
layer 13	FC(16)

Which:

- CONV(x,y,z,t): convolutional layer with the parameters: *x = number of filters, y = size of filter matrix, z = stride value, t = padding value*
- POOL(y,z,t): maximum pooling layer with: *y = size of filter, z = stride value, t = padding value*
- DROP(p): dropout layer with *p is the dropout ratio*
- FC(x): full-connected layer with *x is the number of output*