Deep learning approach for cerebellum localization in prenatal ultrasound images

Ramos, Rodrigo, Olveres, Jimena, Escalante-Ramírez, Boris, Arámbula Cosío, Fernando

Rodrigo Ramos, Jimena Olveres, Boris Escalante-Ramírez, Fernando Arámbula Cosío, "Deep learning approach for cerebellum localization in prenatal ultrasound images," Proc. SPIE 11353, Optics, Photonics and Digital Technologies for Imaging Applications VI, 1135322 (1 April 2020); doi: 10.1117/12.2556818

SPIE.

Event: SPIE Photonics Europe, 2020, Online Only, France

Deep learning approach for cerebellum localization in prenatal ultrasound images.

Rodrigo Ramos Díaz[a], Jimena Olveres[a], Boris Escalante-Ramírez[a] and Fernando Arámbula Cosío[b]
[a]Posgrado en Ingeniería, Facultad de Ingeniería, Universidad Nacional Autónoma de México, Ciudad de México, México [b]Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, Ciudad de México, México

ABSTRACT

Ultrasound (US) has become one of the most common forms for medical imaging in clinical practice. It is a non-invasive and safe practice that allows obtaining images in real time. It is also a technology with important challenges such as low image quality and high variability (between manufacturers and institutions) [1]. This work aims to apply a fast and accurate deep learning architecture to detect and locate cerebellum in prenatal ultrasound images. Cerebellum biometry is used to estimate fetal age [2] and cerebellum segmentation could be applied to detect malformation [3].
YOLO (You Only Look Once) is a convolutional neural network (CNN) architecture for detection, classification and location of objects in images [4]. YOLO was innovative because it solved a regression problem to predict the location (coordinates and sizes) of bounding boxes and associated classes.
We used 316 ultrasound scans of fetal brains and their respective cerebellar segmentations. From these, 78 images were randomly taken to be treated as test images and the rest were available to feed the trainings. Segmentation masks were converted to numerical descriptions of bounding boxes. To deal with small data set, transfer learning was done by initializing convolutional layers with weights pretrained on Imagenet [5].
We evaluated detection using F1 score and localization using average precision (AP) for 78 test images. Our best AP was 84.8% using 121 divisions or cells per image. Future work will focus on segmentation task assisted by localization.
Keywords: fetal ultrasound, image processing, deep learning, detection, localization

1. INTRODUCTION

1.1 Fetal cerebellum in ultrasound screening

Ultrasound imaging has advantages over other medical imaging modalities. Ultrasound is low-cost, portable and safe technology that delivers internal structures images in real time. It has become ubiquitous in clinical practice and preferable for prenatal screening. X-ray, MRI and CT scan are considered riskier due to ionizing radiation. Gestational age determination is fundamental in pregnancy control. Fetal cerebellum biometry is a trustworthy parameter to determine gestational age in second and third trimester, so fetal cerebellum imaging and measurement is useful for gestational age determination when other used parameters are in doubt. Malformations like Dandy-Walker and Arnold-Chiari are found in 1/30,000 live births. These abnormalities and other like spina bifida can be seen and detected in cerebellum.

1.2 Deep learning in medical ultrasound

Liu et al. reviewed deep learning in medical ultrasound analysis in 2018. Multiple deep learning architectures have been applied in different anatomical structures. Most common architectures are CNN, Deep Belief Network (DBN), Fully Convolutional Network (FCN), Auto-encoders (AE), and hybrid methods (including generative and discriminative components). Most common anatomical structure are fetus, heart, breast, prostate, and thyroid. They report a growth of the field from six papers in 2012 to almost 60 in 2017. Computer-aided Diagnostics (CADx) research is focused on three main tasks: detection (localization), classification, and segmentation. Detection and localization of object of interest is a prerequisite for further processing like classification or segmentation. An example of classification is discriminating between benign and malignant tumors. Segmentation of anatomical structures can be used for quantitative and qualitative parameter estimation like volume and shape related parameters in cardiac and brain analysis.

Optics, Photonics and Digital Technologies for Imaging Applications VI, edited by
Peter Schelkens, Tomasz Kozacki, Proc. of SPIE Vol. 11353, 1135322 · © 2020 SPIE
CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2556818

1.3 Deep learning and transfer learning

AlexNet [6] is considered the first deep convolutional neural network. It has five convolutional layers and three fully connected with a total of 60 million trainable parameters. It was trained on an ImageNet's subset of 1,2 million images from 1000 categories. Krishevsky's architecture won several competitions. Since then, deep learning has been accumulating state of the art results in fields like computer vision, speech recognition, machine translation, natural language processing, bioinformatics, robotics, cybersecurity, and more. One important deep learning ability is to automatically learn low-level to high-level features. YOLO was also trained on ImageNet and saved weights are available to download as a file. Transfer learning is the possibility to transfer knowledge from a source domain (like ImageNet) into a new target domain (like fetal echography). Source and target domain of knowledge should be related.

1.4 You only look once

YOLO authors compared their architecture with machine learning method Deformable Part Model (DPM)[7]. They also compared with other deep learning architecture: Regional Convolution Neural Network (R-CNN)[8]. YOLO produced results hundreds of times faster than the other two. Speed is YOLO´s principal feature. Their results had accuracy almost as good as R-CNN and two times better than DPM. YOLO paper highlights unified detection as its second feature and describes it as a global reasoning about the full image and all its objects. It explains this is achieved by a grid division of the image and separate predictions for each cell. Third feature is generalization across domains, and it was tested using artwork images. The architecture is composed by twenty-four convolutional layers (extracting complex features from input image) and two fully connected layers (predicting bounding boxes and classes from feature maps).

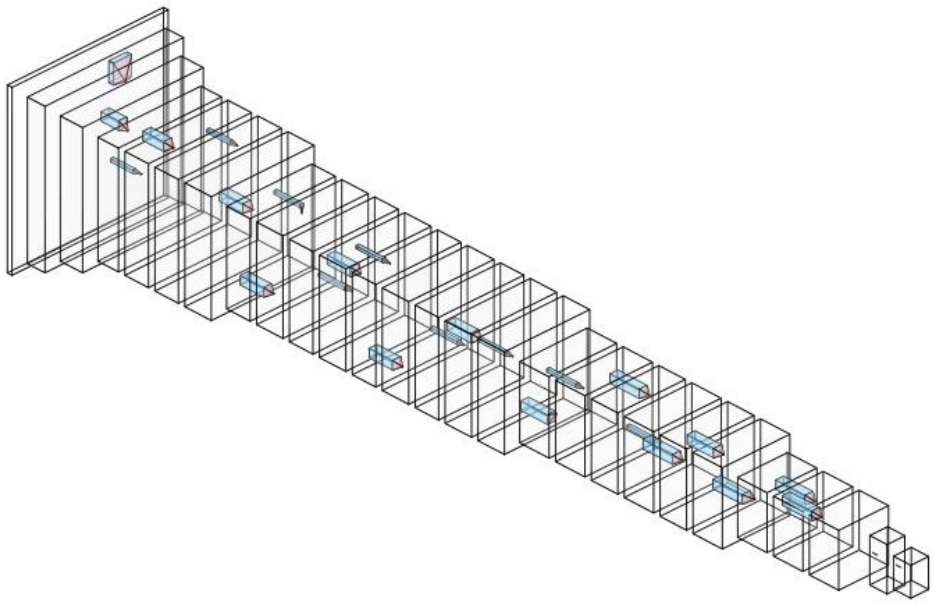


Figure 1. YOLO architecture with 24 convolutional and 2 dense layers. Scales and log scales were applied to have an overall vision of the full architecture. Figure was created using NN-SVG [9] with AlexNet style.

1.5 You only look once (fetal ultrasound images)

Among scientific community, there have been doubts and objections regarding the use of CNN with a reduced data set. The answer to these was transfer learning: CNN models can be initialized with weights pretrained on millions of images. Still there was skepticism: ultrasound medical images have nothing to do with Imagenet! Nevertheless, our results point out that knowledge obtained from natural images (showing daily objects like persons, animals, vehicles, furniture, food, etc.) may be related somehow to ultrasound images (showing textures and speckle). We applied with success deep learning detection and localization of cerebellum in fetal ultrasound images. We also made experiments to practice the art of hyperparameter tuning. We investigated learning rate, momentum coefficient, and number of divisions (for image grid).

2. METHOD

2.1 Framework

Its preparation starts with the installation of an appropriate operating system. Next steps are installation of GPU driver, CUDA[10], and OpenCV[11] (to draw bounding boxes, if desired). Final step is the compilation of Darknet[12], which is an open source neural network library written in C and CUDA. CUDA means Compute Unified Device Architecture and it includes programming language, compiler and toolkit for massive parallel data processing in GPU. NVidia GPU with at least 4 GB of VRAM is required. We used NVidia K4200 (with 1344 cores), CUDA 10.1, and Ubuntu 18.04. Darknet can be executed from the command line or from Python using the OS library and the system command. It is advisable not to execute Darknet from Jupyter Notebook, for training, or the system may run out of memory. It can also be recommended not to update any of the following software in order to keep framework functional: Linux kernel, CUDA toolkit, GPU driver, and compiler.

2.2 Dataset

The National Institute of Perinatology (INPER) in Mexico provided us with 315 ultrasound scans of fetal brains and their corresponding masks with cerebellum expert annotations or segmentations. The images are registered in transcerebellar plane with cerebellum always in right side. Standard fetal brain screening should include three planes: transventricular, transthalamic, and transcerebellar. Images are grayscale bitmap format and the segmentations are MATLAB binary (.mat) format. They come from 21 ultrasound volumes or 21 different patients, so we have 15 parallel slices for each cerebellum specimen.

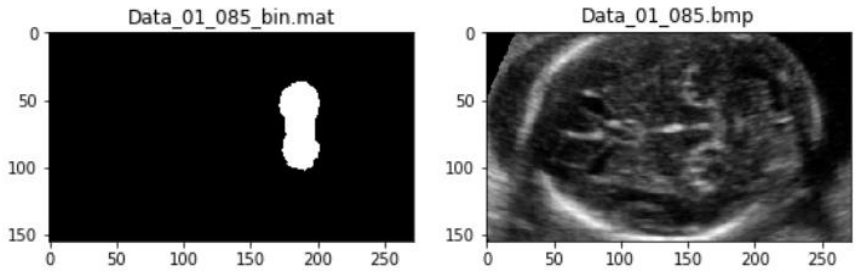


Figure 2. An example of a binary segmentation mask (.mat) and its corresponding grayscale image (.bmp).

2.3 Annotations and data partitions

The training requires a text file with annotations for each image. Annotation text files should contain 5 values separated by spaces: number of class (c), coordinates of the center of the bounding box (x, y), width and height (w, h). When an image contains several objects, each group of annotations ends with a line break. In this case we want to detect and localize only one class: cerebellum. A script was written in Python to process the .mat files and convert binary segmentation matrices into text file annotations**.** 78 images were randomly selected and taken apart to be treated as test images, so the rest remained available to feed the trainings. Python scripts were also written to create text files with lists of training and test images for data partitions.

2.4 Training

Darknet library features YOLO for PASCAL-VOC [13] with 20 classes of objects. In this work we have a single class: cerebellum, so it was necessary to modify the source file yolo.c and recompile. All the gray-scale images were transformed to the Matplotlib [14] "viridis" color map so we could meet the input requirement of three channel images. Such color map is supposed to be colorful, perceptually uniform, and robust to colorblindness. The configuration file was also modified to have the last dense layer output size corresponding to one single class predictions. The output tensor shows the structure of all the numbers we want the model to predict. All these values are also to be provided during training as annotations or ground truth.

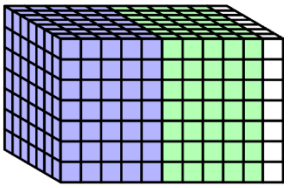


Figure 3. Graphic representation of output tensor with 7 by 7 grid, dual bounding box prediction per cell, and single class detection. First object's predictions in purple, second object's predictions in green, and class prediction in white.

$$Output\ size\ =\ S * S * (B * 5 + C) \tag{1}$$

S is number of divisions by side of grid, B is number of bounding boxes predicted per cell, and C is the number of classes. Each bounding box is defined by centroid coordinates, width, height and a confidence value or probability of object. Four experiments or trainings were made using different values of S: 7, 9, 11, and 13. The respective output sizes are 539, 891, 1331, and 1859.

The original configuration file proposes to finish the training after processing 40,000 batches of images. In our case 1000 batches seemed to be enough. Optimized weights were saved every 100 batches, so we have 10 weight files or checkpoints per training. We used a batch size of 64 images. Considering we have 315-78=237 images for training, 1000 iterations equal 270 epochs.

Transfer learning was implemented by loading pretrained weights as initialization for convolutional layers. This means we expected to transfer knowledge for automatic feature extraction task only, not for detection and localization.

2.5 Experiments with tiny YOLO

Tiny YOLO is an architecture with 8 convolutional layers instead of 24. In addition, the size of the entrance was changed from 448 x 448 to 224 x 224 pixels. The latter was done considering the size of the images in the data set that have, on average, a width of 255 and a height of 173. The new network has less trainable elements (and lower computational cost) so its training takes less time. Four experiments were done using learning rate ε of 5*10-4 and different values of momentum coefficient μ: 0.3, 0.5, 0.7, 0.9. Same four experiments were repeated using a learning rate ε of 5*10-3. With these experiments we wanted to verify if momentum enables stochastic gradient descent (SGD) to converge faster.

$$v_{t+1} = \mu v_t - \varepsilon \nabla L(\theta_t) \tag{2}$$

$$\theta_{t+1} = \theta_t + v_{t+1} \tag{3}$$

Equation 3 shows that the new weights or parameters are equal to past weights plus a velocity vector (which accelerates gradient descent). Equation 2 explains that the velocity vector equals the past velocity vector (multiplied by momentum coefficient μ) minus the product of the learning rate ε and the gradient (of the loss function L).

3. RESULTS

3.1 Results with YOLO

We remember that 4 trainings were executed, each one generating 10 checkpoints or models, so forty models are evaluated with 78 test images. The average precision and F1 score were calculated. The average precision (AP) is the average of the Intersection over Union (IoU) values for 78 test images. IoU is the area of the intersection (between the annotated and the predicted box) divided by the union of both areas. The perfect precision (for localization) would be IoU = 1 and this would mean that the intersection and the union are equal. F1 score is the harmonic mean of precision and recall. Perfect precision and recall (for detection) is F1=1.

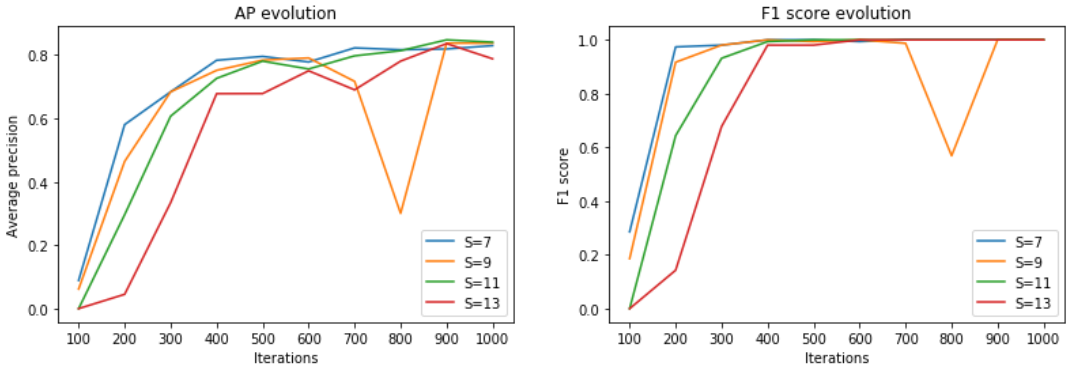


Figure 4. Average precision vs number of iterations (left) and F1 score vs number of iterations (right)

We note that increasing the number of divisions per side (hyperparameter S) can achieve greater accuracy but requires more training. At the end the difference is not very significative. Could this be due to the number of parameters more than the number of divisions? Number of trainable parameters can also be increased with the number of predictions (B) per cell. More experiments should be done varying hyperparameter B while keeping S constant.

Although YOLO architecture is supposed to reason globally at once, we can observe that task of prediction (for detection and localization) is very redundant. One wonders if the part of redundancy related to number of divisions has the same relevance to the task of prediction as to the task of feature extraction. We find this question pertinent since grid division seems to have a role in shaping spatial features within the task of feature extraction. For future research we propose to train with many divisions, then freeze the convolutional layers and retrain with few divisions. Then we compare to a model that was trained with few divisions from the beginning.

Table 1. Best models of YOLO (24 layers) after evaluating all trained models (checkpoints).

Divisions	**49**	**81**	**121**	**169**
Batches	1000	1000	900	900
Average precision	82.97 %	83.77%	84.8%	83.6%

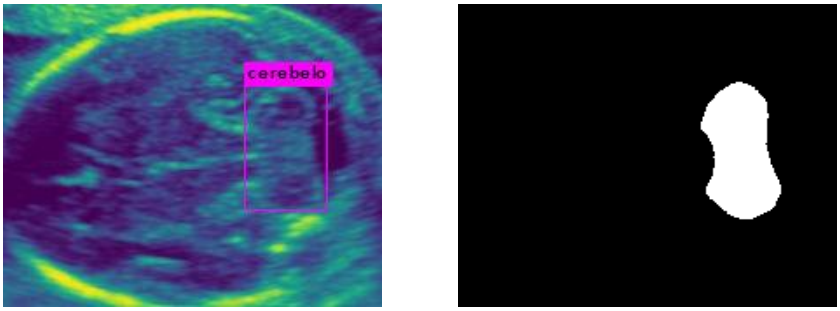


Figure 5. Image with cerebellum detection and localization in left column and corresponding expert annotations in right column. Model with 49 divisions after 500 batches was loaded.

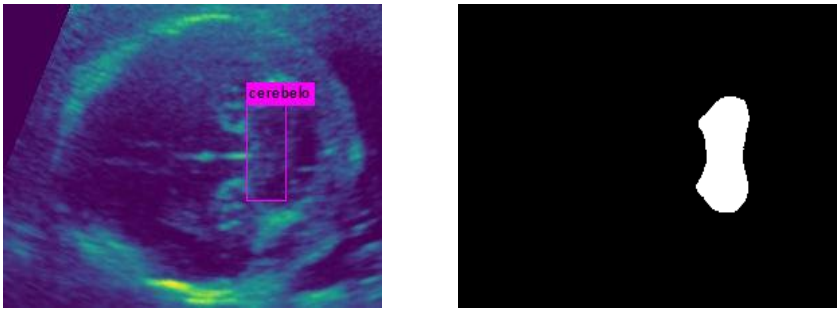


Figure 6. Image with cerebellum detection and localization in left column and corresponding expert annotations in right column. Model with 81 divisions after 600 batches was loaded.

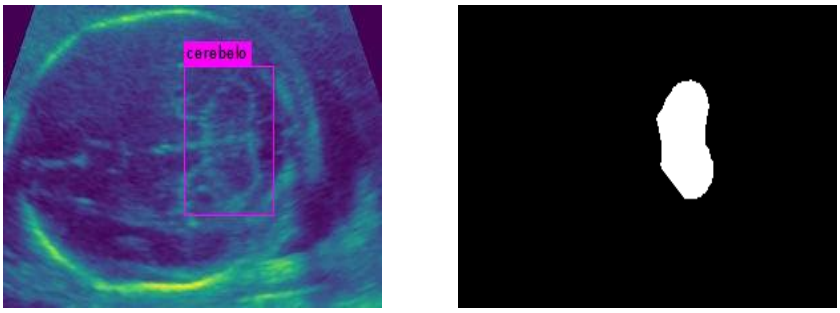


Figure 7. Image with cerebellum detection and localization in left column and corresponding expert annotations in right column. Model with 121 divisions after 600 batches was loaded.

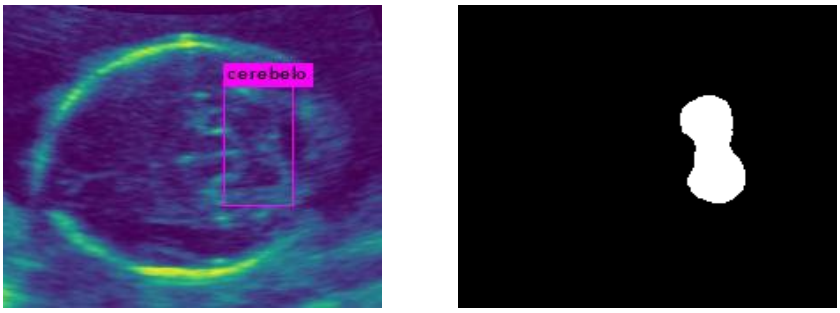
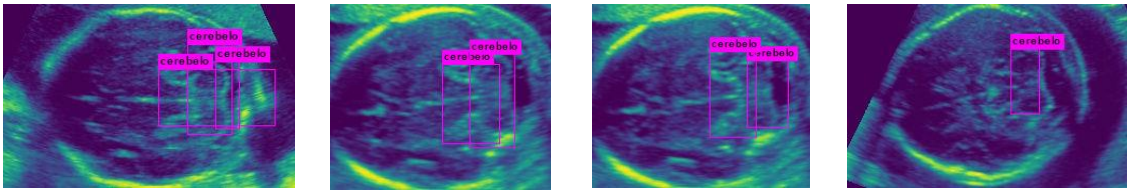


Figure 8. Image with cerebellum detection and localization in left column and corresponding expert annotations in right column. Model with 169 divisions after 1000 batches was loaded.

3.2 Results with tiny YOLO

We remember that 8 trainings were executed, each one generating 10 checkpoints or models, so eighty models are evaluated with 78 test images. Average precision and F1 score were again calculated for every loaded model.



μ=0.3 μ=0.5 μ=0.7 μ=0.9

Figure 9. Predictions by tiny YOLO models with best learning rate, after 1000 iterations, for different momentum coefficient μ values.

Table 2. Best models of Tiny YOLO (8 layers) after evaluating all trained models (checkpoints).

Momentum	**0.3**		**0.5**		**0.7**		**0.9**	
Learning rate	5*10-4	5*10-3	5*10-4	5*10-3	5*10-4	5*10-3	5*10-4	5*10-3
Average precision	0.45	0.48	0.42	0.31	0.45	0.29	0.40	0.66
F1 score	0.96	0.96	0.99	0.99	0.94	0.94	1	1

AP and F1 score evaluations casted very unstable behavior. AP results are unsatisfactory. It seems that reduced number of layers in model disabled localization precision. But there are several models that showed good results in detection.

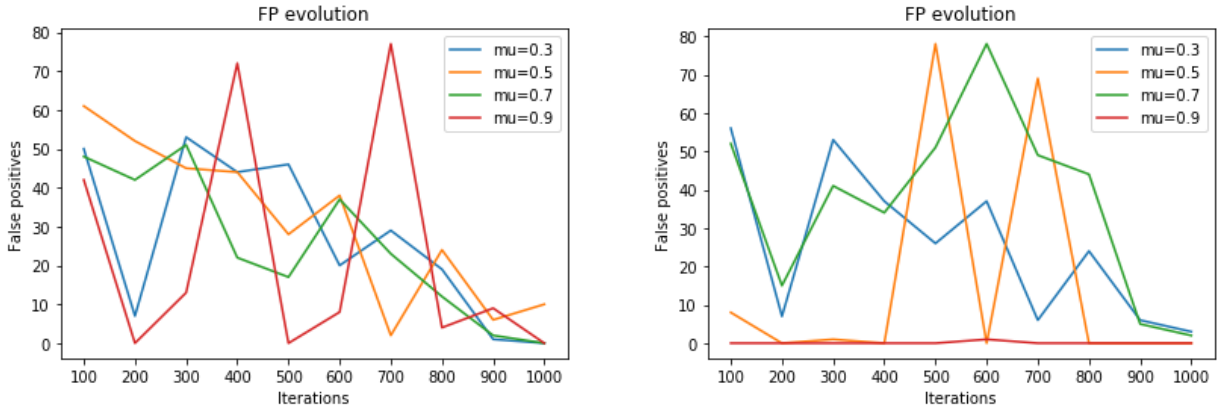


Figure 10. False positives vs. iterations, with learning rate of 5*10-4 (left), with learning rate of 5*10-3 (right).

Despite general unstable behavior during training, we show evolution of false positives. We expected FP to decrease while increasing number of iterations, but this trend is only shown by $\mu = 0.3$ to $\mu = 0.7$ for lower learning rate (left) and by $\mu = 0.5$ for higher learning rate. Half of experiments show expected trend. The other half is rather unstable. We observe that the accumulation of "momentum" introduces instability in learning as the momentum coefficient increases. But it is also observable that better results are achieved with most increased values of μ. Finally, it seems that choosing the right learning rate may have greater impact than the use of momentum in obtaining the best possible results.

4. CONCLUSIONS

Cerebellum detection and localization in medical ultrasound images was achieved by training a deep learning architecture model. It is verified that the number of layers or depth play a very important role in CNN architectures. Transfer learning was implemented to deal with a reduced data set. Despite differences between natural and medical ultrasound images, we found that some knowledge may have been transferred across domains, at least for feature extraction/detection task. Open questions are posed to explore hyperparameter tuning and its relation to neural network interpretability. In the future, we will work in segmentation assisted by localization expecting to allow or improve the detection of rare malformations that are hard to diagnose.

ACKNOWLEDGEMENTS

Rodrigo Ramos Díaz would like to thank CONACYT (Consejo Nacional de Ciencia y Tecnologia) for the financial support (CVU: 927245) during his graduate studies.

Authors would like to thank grants: UNAM PAPIIT IA103119 and SECTEI/202/2019, as well as Instituto Nacional de Perinatología for providing the ultrasound scans.

REFERENCES

[1] S. Liu, Y. Wang, X. Yang, B. Lei, L. Liu, S. X. Li, D. Ni, and T. Wang, "Deep Learning in Medical Ultrasound Analysis: A Review," Engineering, vol. 5, no. 2, pp. 261–275, 2019.

[2] Álvaro Sepúlveda-Martínez, Verónica Sepúlveda, "Biometría de cerebelo fetal: ¿Parámetro útil en edad gestacional dudosa?," Revista chilena de Ultrasonografía, no. 10, pp. 122-127, 2007.

[3] Victor Bunduki, Marcelo Zugaib, "Atlas of Fetal Ultrasound: Normal Imaging and Malformations," pp. 29-49, 2018.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision (IJCV), 2015.

[6] Krizhevsky, A., Sutskever, I., and Hinton, G. E. "ImageNet classification with deep convolutional neural networks," In NIPS, pp. 1106–1114, 2012.

[7] J. Yan, Z. Lei, L. Wen, and S. Z. Li. "The fastest deformable part model for object detection," 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2497–2504. IEEE, 2014.

[8] R. B. Girshick. "Fast R-CNN," International Conference on Computer Vision, 2015.

[9] LeNail. NN-SVG: Publication-Ready Neural Network Architecture Schematics. Journal of Open Source Software, 4(33), 747, 2019, https://doi.org/10.21105/joss.00747 (29 February 2020)

[10] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron, "Scalable Parallel Programming with CUDA," Queue 6, 2, pp. 40–53, March 2008, https://doi.org/10.1145/1365490.1365500 (29 February 2020)

[11] Bradski, G. "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

[12] Joseph Redmon, "Darknet: Open Source Neural Networks in C," 2013, http://pjreddie.com/darknet/ (29 February 2020)

[13] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. "The pascal visual object classes challenge: A retrospective," International Journal of Computer Vision, 111(1):98–136, Jan. 2015.

[14] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.