DL-inferencing for 3D Cephalometric Landmarks Regression task using OpenVINO*

Evgeny Vasiliev[1][0000−0002−7949−1919], Dmitrii Lachinov[1,2][0000−0002−2880−2887], and Alexandra Getmanskaya[1][0000−0003−3533−1734]

[1] Institute of Information Technologies, Mathematics and Mechanics, Lobachevsky State University

[2] Department of Ophthalmology and Optometry, Medical University of Vienna

evgeny.vasiliev@itmm.unn.ru, dmitry.lachinov@itmm.unn.ru, alexandra.getmanskaya@itmm.unn.ru

Abstract. In this paper, we evaluate the performance of the Intel Distribution of OpenVINO toolkit in practical solving of the problem of automatic three-dimensional Cephalometric analysis using deep learning methods. This year, the authors proposed an approach to the detection of cephalometric landmarks from CT-tomography data, which is resistant to skull deformities and use convolutional neural networks (CNN). Resistance to deformations is due to the initial detection of 4 points that are basic for the parameterization of the skull shape. The approach was explored on CNN for three architectures. A record regression accuracy in comparison with analogs was obtained. This paper evaluates the performance of decision making for the trained CNN-models at the inference stage. For a comparative study, the computing environments PyTorch and Intel Distribution of OpenVINO were selected, and 2 of 3 CNN architectures: based on VGG for regression of cephalometric landmarks and an Hourglass-based model, with the RexNext backbone for the landmarks heatmap regression. The experimental dataset was consist of 20 CT of patients with acquired craniomaxillofacial deformities and was include pre- and post-operative CT scans whose format is 800x800x496 with voxel spacing of 0.2x0.2x0.2 mm. Using OpenVINO showed a great increase in performance over the PyTorch, with inference speedup from 13 to 16 times for a Direct Regression model and from 3.5 to 3.8 times for a more complex and precise Hourglass model.

Keywords: 3D Cephalometry, Automatic Cephalometry, Keypoint Regression, OpenVINO, Deep Learning

1 Introduction

At present time, deep learning models are increasingly used in medicine. They allow solving the problems of analyzing medical images without manual parameter extraction. Biological organisms have great variability in size, structure and shape, which

* This work was supported by Intel and the Russian Foundation for Basic Research, project no. 18-37-00383

does not allow building an accurate mathematical description of biological systems. Deep models during training select and extract deep features on their own and build an internal representation of the objects of analysis, which allows us to solve the problems of processing medical data.

After constructing a deep model, the problem of its implementation in existing software and hardware used in the industry arises. Deep models require a lot of computation, which also imposes restrictions on the possibility of use. To solve the problems of efficient deep model inference on various hardware and embedding in existing software, the Intel® Distribution of OpenVINO™ toolkit [1] is used. The Intel® Distribution of OpenVINO™ toolkit shows significant acceleration of deep learning models in computer vision tasks [2,3] and is also used to accelerate deep learning models in other areas of research in production.

One of the tasks for which it is difficult to construct an exact mathematical description and a deterministic solution algorithm is the problem of finding the key points of the skull.

In this paper we analyse and accelerate the inference performance of state of the art 3D deep convolutional neural network (CNN) based method for keypoint regression to solve the task of three-dimensional cephalometric analysis.

2 Problem statement

The main purpose of our work is to create fast and resistant to variations in the shape of the human skull method for automatic detection cephalometric points. In particular, we are interested in the location of the following 4 landmarks:

1. Left Orbitale.
2. Right Orbitale.
3. Left Porion.
4. Right Porion.

This four points are important because its includes to Frankfort Horizontal determination process. Each point is represented by 3 coordinates in the CT coordinate system.

The task of marking up the source data is complex and ambiguous since different specialists can mark up different positions for landmarks in 3D. In the current formulation of the problem, a deviation of 4 millimeters for the points forming the Frankfort horizontal plane is sufficient accuracy, comparable to the marking accuracy of an average specialist.

3 Related works

Besides the clinical application, the landmark regression is used in a number of different spheres. For instance, facial landmarks regression, human pose estimation, or even crowd counting take a central part in intelligent surveillance systems. In these tasks, detected landmarks can represent different entities like face parts, body parts, or whole

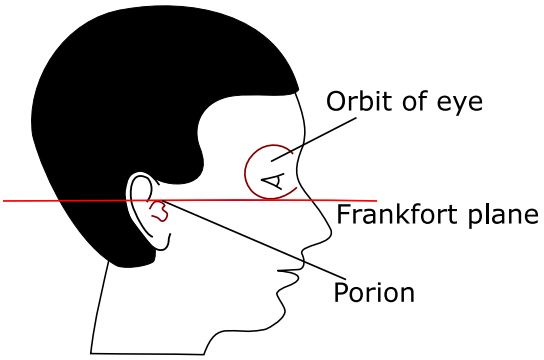

Fig. 1: The Frankfort horizontal plane

human body. In the literature, neural network based methods for solving key point regression task can be split into two groups: the direct regression of target variables and regression through some intermediate representation, for example, heatmap.

Chen et al. [4] used neural networks and genetic algorithms to find areas on the radiograph containing cephalometric points. Osadchy et al. [5] applied a convolutional neural network based approach for mapping face image to a manifold, parametrized by pose. Tompson et al. [6] proposed a convolutional network based human pose estimation method that outputs heatmap. The heatmap per pixel describes the likelihood of a landmark appearing in each spatial position. Newell et al. [7] proposed the new convolutional network architecture for this task called Hourglass networks. The base network operates over all scales of the image. Authors also propose to stack sequentially multiple base networks.

Deep learning based approaches mainly focus on automatic detection of cephalometric landmarks on lateral X-Ray images. Lee et al. [8] utilize patch classification and point estimation neural networks for the identification of 33 landmarks. Hwang et al. [9] proposed the YOLOv3 [10] based system. For 3D CT scans Lee et al. [11] proposed VGG-based [12] method for detecting landmarks on shadowed 2D projections. A completely 3D based approach using 3D convolutional neural network based system was proposed by Kang et al. [13].

In our previous paper Lachinov et al. [14] we proposed 3 convolutional neural networks for cephalometric landmarks regression on 3D CT images. We found out that the Stacked Hourglass network [7] and Softargmax based model [15,16] achieve remarkable accuracy, and Direct Regression model [12] achieve the best performance.

The Root Mean Squared Error (RMSE) values are reported in Table 1. Hereby we can see that Direct Regression has the highest error. The likelihood of landmarks to be within a certain radius from ground truth points is reported in Table 2. We pay special attention to this characteristic since the radius of 4 mm is considered to be a threshold for clinical applications. As we can see, only 61% of the points predicted by Direct Regression fall within 4 mm radius. In contrast, the Stacked Hourglass model achieves 95% of its prediction to fall within 4 mm radius respectively. The high accuracy for 2

mm and 3 mm radius is also notable (Table 2). For more quantitative analysis, see the article [14].

Table 1: The RMSE for landmarks

Model	Or_l	Or_r	Po_l	Po_r	Total
Direct Regression	4.14	3.78	5.03	5.22	**4.58**
Hourglass	1.66	1.24	1.91	1.97	**1.72**

Table 2: The likelihood

Model	2 mm	3 mm	4 mm
Direct Regression	0.20	0.39	0.61
Hourglass	0.81	0.94	0.95

4 Intel® Distribution of OpenVINO™ toolkit

Intel® Distribution of OpenVINO™ toolkit is developed to accelerate and deploy neural network models with a built-in model optimizer and an inference engine runtime for hardware-specific acceleration. Inference optimization is provided through the analysis and optimization of a computational graph, effective planning of data processing and vectorization, and various methods of compression of the deep model. Intel® Distribution of OpenVINO™ toolkit focuses on developing cross-platform solutions of computer vision problems and pays a lot of attention to medical imaging AI workloads. OpenVINO has few dependencies that help to integrate OpenVINO with existing software. A promising feature is a support for the protection of deep models by encrypting them.

4.1 Components

The Intel® Distribution of OpenVINO™ toolkit is an actively developing product in which new functions are developed. The current version consists of several major parts [1].

1. **Deep learning for computer vision**. This part includes the Deep Learning Deployment Toolkit to make a high-performance inference of pretrained deep neural network models using a high-level application programming interface.
2. **Traditional computer vision**. This part includes accelerated computer vision library OpenCV [17].

3. **Additional packages** to perform optimized inference using different hardware (Intel® FPGA, Intel® Movidius™ Neural Compute Stick, Intel® GMM-GNA) and media encode/decode functions for improving the performance of processing graphics and video.
4. **Open Model Zoo** is a public repository of more than 180 pretrained models for solving various problems of computer vision, samples and demos.
5. **Post-Training Optimization Tool** designed to accelerate the inference of DL models by converting them into a more hardware-friendly representation by applying specific methods that do not require re-training, for example, post-training quantization.

4.2 Scheme of using The Intel® Distribution of OpenVINO™ toolkit

The basic variant of using OpenVINO involves the following steps:

1. Train a deep neural network model trained using any popular deep learning frameworks (Caffe, TensorFlow, Keras, PyTorch etc.) or download pretrained model using Model Downloader.
2. Convert the model to the intermediate representation (IR) by calling Model Optimizer.
3. Load input data for the model and infer the model using Inference Engine efficiently, receive the model output for the subsequent interpretation.

4.3 Model conversion

To perform the inference, the Inference Engine does not operate with the original model, but with its Intermediate Representation (IR), which is optimized for execution on endpoint target devices. To generate an IR for your trained model, the Model Optimizer tool is used. The Model Optimizer loads a model into memory, reads it, builds the internal representation of the model, optimizes it, and produces the Intermediate Representation. OpenVINO supports the next framework formats: Caffe, MXNet, TensorFlow, Kaldi, ONNX. OpenVINO supports all common deep learning layers, but new layers are invented every day. OpenVINO contains a mechanism for adding your custom layers. To infer PyTorch model using OpenVINO, you should convert it to ONNX first, it is a simple operation. The next step will be convert ONNX model to Intermediate respesentation. The conversion of proposed model from ONNX format is listed below:

```
cd %OPENVINO_DIR%/deployment_tools/model_optimizer
python mo.py --input_model vgglike.onnx \
    --input_shape [1,1,128,128,64] \
    --output_dir %YOUR_DIR%
```

4.4 Chosen Models

Direct regression. We define a direct regression as a convolutional neural network followed by global pooling and fully connected layer. The output of a fully connected

layer matches the target variables. The graph of the model is presented in Fig. 2. The network is in line with the VGG-based model introduced by Simonyan et al. [12]. It takes 3D CT scan as input and processes it with a series of 3x3x3 convolutional blocks, instance normalization [18] layers and ReLU activation. At the end of the fully convolution part of the network global average pooling is performed that is followed by two fully connected layers with activations. The number of outputs of the final layer corresponds to the number of regressing values. In our case it equal to 4 points with 3 coordinates each, 12 in total.

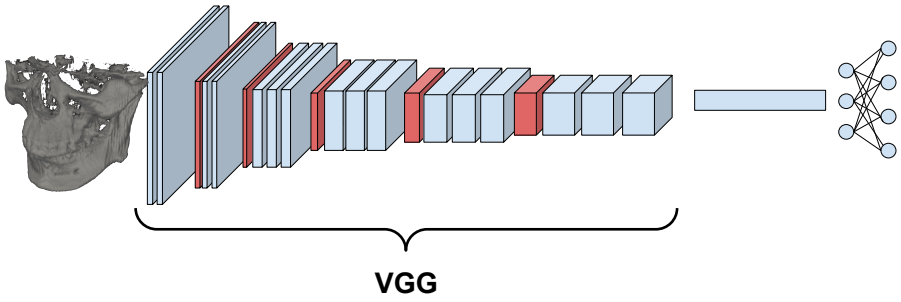

Fig. 2: The VGG based architecture for direct regression. Each blue block corresponds to convolution followed by normalization and ReLU activation. The red blocks have strided convolutions.

Heatmap regression. Unlike the previous model, in which we tried to directly predict the target variable, here we focus on predicting the probability of a key point for each voxel (Heatmap). Ground truth heatmaps are generated by a probability density function of a Gaussian distribution with an average value in the target landmark. In CNN design, we follow the Stacked Hourglass network architecture proposed by Newell et al. [7]. It combines multiple subnetworks stacked one after another (Fig. 3). The subnetworks consist of encoder and decoder that are connected by the means of additive skip-connections. The architecture of an individual network is displayed in Fig. 3. In this model we use 3 stacked Hourglass networks with ResNext blocks [19] and Group Normalization. [20]. The output layer consists of a single convolution and sigmoid activation. At the end of each network in the stack, we provide additional supervision by attaching auxiliary output layers with the corresponding loss function.

In the basic implementation of the Hourglass model, the upscaling operation through trilinear interpolation was not fast enough for 3D data processing using OpenVINO. OpenVINO has a simple and convenient system for analyzing the execution time of individual layers of the model. A question was asked in the OpenVINO repository on GitHub how this operation can be optimized, and an Intel engineer proposed the solution to replace upscaling with a specific deconvolution layer, which gives absolutely the same result [21].

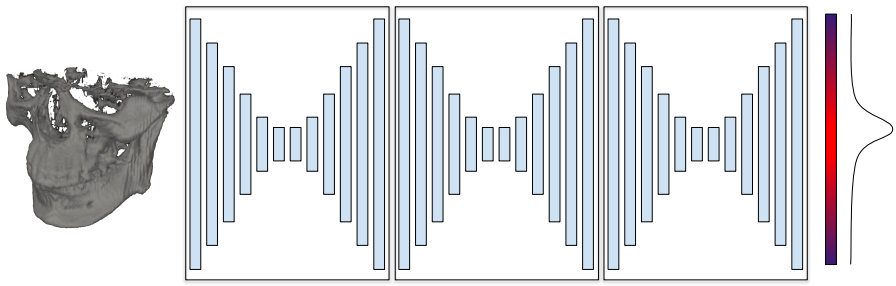

Fig. 3: The Stacked hourglass network consists of the consecutive individual hourglass networks.

A step-by-step tutorial on how to create code to infer deep models using OpenVINO can be found in the article [3], which details the sequence of actions and provides the source code of tutorial for working with OpenVINO.

5 Experiments

5.1 Infrastructure

The following hardware was used as a test infrastructure for measuring performance:
CPU: Intel® Core™ i7-8700 CPU @ 3.20GHz
RAM size: 64 GB
OS version: Linux-5.3.0-51-generic-x86-64-with-Ubuntu-18.04-bionic
Python version: 3.7.7
OpenVINO version: 2020.3.194
PyTorch version: 1.5.1 (from anaconda)

5.2 Data

In our experiments, we use the dataset consisting of 20 CT images of patients with acquired cranio-maxillofacial deformities. All scans were taken with two multispiral CT devices. The dataset consists of pre- and post-operative CT scans. The resolution of each image is 800x800x496 with voxel spacing of 0.2x0.2x0.2 mm. For every image 4 cephalometric landmarks were annotated: left and right orbitale and porion.

As a preprocessing step, we downsample images to the size of 128x128x64 with voxel spacing of 1.25x1.25x1.55 mm. Then we perform the z-score normalization of the image I by subtracting mean μ and diving by standard deviating σ: $I_z = \frac{I-\mu_I}{\sigma_I}$.

5.3 Performance Analysis

VGG and Hourglass models contain $1.33 \cdot 10^6$ and $25.5 \cdot 10^6$ parameters and need $3.8 \cdot 10^9$ and $6.9 \cdot 10^9$ operations respectively. During the experiment on the inference

of the VGG model using OpenVINO, OpenVINO showed a great increase in performance over the original framework. We have got inference speedup from 13 to 16 times for a simpler VGG model and from 3.5 to 3.8 times faster for a more complex Hourglass model. The Hourglass model has more complex architecture and contains many more parameters, and that is why OpenVINO did not show as much acceleration as the VGG model. Trilinear upscaling interpolation for 3D data in the decoder is the longest computational operation in the Hourglass model.

Table 3: Performance of VGG-like model with batch=1.

Framework and inference mode	Latency (s)	Total time (s)	FPS	Speedup
PyTorch	0.183	187.788	5.453	1.0
OpenVINO sync	0.013	13.655	74.990	13.752
OpenVINO async (4 requests)	-	**11.180**	**91.595**	**16.797**

Table 4: Performance of VGG-like model using different batches.

	PyTorch		OpenVINO sync mode		OpenVINO async mode	
Batch size	Total time (s)	FPS	Total time (s)	FPS	Total time (s)	FPS
1	375.814	5.450	27.354	74.871	**22.722**	**90.135**
2	294.316	6.959	28.503	71.852	22.928	89.323
4	259.146	7.903	28.910	70.842	23.511	87.110
8	251.736	8.136	29.215	70.100	23.353	87.697
16	243.928	8.396	29.288	69.927	23.805	86.032
32	259.44	7.894	29.413	69.630	24.894	82.268
64	168.322	12.167	29.471	69.492	24.834	82.467
128	141.724	14.451	29.662	69.044	26.67	76.790
256	140.164	14.611	29.807	68.709	28.493	71.878

Table 5: Performance of Hourglass model with batch=1.

Framework and inference mode	Latency (s)	Total time (s)	FPS	Speedup
PyTorch	2.421	308.830	0.414	1.0
OpenVINO sync	0.686	87.798	1.458	3.518
OpenVINO async (4 requests)	-	**79.829**	**1.603**	**3.828**

A common way to increase productivity and utilize processor power is to process data with large batches. This approach is often used during training of deep networks or during remote data processing, for example, server processing data from several cameras simultaneously. Using a large batch allows you to achieve greater processor

performance in the tasks of image classification, detection of objects in images, and other tasks.

In our case of the VGG-like model, the use of a large batch did not entail an increase in system performance. The main problem for increasing productivity by increasing the batch is a large amount of input for the model. In the cephalometry problem, the input tensor for the VGG model is an order of magnitude larger than the size of the input image for the Resnet-50 classification model.

Tables 3 and 4 gives the performance of the Direct Regression on generated data set of 2048 objects using batch 1 and analysis of performance using different batches, tables 5 and 6 gives the performance of the Hourglass method on generated data set of 128 objects respectively. Dashes indicate experiments that were not finished due to out of memory error on the target device.

Table 6: Performance of Hourglass model using different batches.

	PyTorch		OpenVINO sync mode		OpenVINO async mode	
Batch size	Total time (s)	FPS	Total time (s)	FPS	Total time (s)	FPS
1	308.830	0.414	87.798	1.458	80.673	1.587
2	303.234	0.422	89.439	1.431	79.155	1.617
4	293.672	0.436	89.581	1.429	**79.030**	**1.620**
8	309.207	0.414	89.639	1.428	82.291	1.555
16	320.290	0.400	89.416	1.432	86.469	1.480
32	-	-	90.005	1.422	91.701	1.396
64	-	-	90.904	1.408	-	-

Analyzing the results of performance measurements, we can see that OpenVINO can significantly accelerate the inference of deep neural networks on user hardware without using of specialized computing devices, only by optimizing the calculations.

6 Conclusion

This article presented a study of using the Intel® Distribution of OpenVINO™ toolkit in a complex medical problem. An overview of the Intel® Distribution of OpenVINO™ toolkit has been shown. The practical application of the OpenVINO™ toolkit has been demonstrated on the problem of cephalometric landmark regression on 3D computed tomography data. The solution of this problem using deep neural networks have described and the inference speedup results on typical hardware have been shown. The quality and inference performance experiments with two CNN model have been performed: based on VGG for direct regression of cephalometric landmarks and an Hourglass-based model, with the RexNext backbone for the landmarks heatmap regression, which contain $1.33 \cdot 10^6$ and $25.5 \cdot 10^6$ parameters respectively. The total RMSE values 4.58 for VGG model and 1.72 for Hourglass model were obtained. OpenVINO inference showed a significant speedup of the model execution over the PyTorch, the speedup was from 13 to 16 times for VGG model and from 3.5 to 3.8 times for a more complex and precise Hourglass model.

References

1. Intel® Distribution of OpenVINO™ toolkit. https://docs.openvinotoolkit.org/latest/index.html, last accessed 30 Jun 2020 2, 4
2. Kustikova, V., Vasiliev, E., Khvatov, A., Kumbrasiev, P., Rybkin, R., Kogteva, N.: Dli: Deep learning inference benchmark. Communications in Computer and Information Science 1129 CCIS, 542–553 (2019) 2
3. Kustikova, V., Vasiliev, E., Khvatov, A., Kumbrasiev, P., Vikhrev, I., Utkin, K., Dudchenko, A., Gladilov, G.: Intel distribution of openvino toolkit: a case study of semantic segmentation. AIST: International Conference on Analysis of Images, Social Networks and Texts 11832 LNCS, 11–23 (2019) 2, 7
4. Chen, Y.J., Chen, S.K., Chang, H.F., Chen, K.C.: Comparison of landmark identification in traditional versus computer-aided digital cephalometry. The Angle Orthodontist 70(5), 387–392 (2000) 3
5. Osadchy, M., Le Cun, Y., Miller, M.L.: Synergistic Face Detection and Pose Estimation with Energy-Based Models, pp. 196–206. Springer Berlin Heidelberg, Berlin, Heidelberg (2006) 3
6. Tompson, J., Goroshin, R., Jain, A., LeCun, Y., Bregler, C.: Efficient object localization using convolutional networks. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 648–656 (June 2015) 3
7. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: ECCV (2016) 3, 6
8. Lee, C., Tanikawa, C., Lim, J.Y., Yamashiro, T.: Deep learning based cephalometric landmark identification using landmark-dependent multi-scale patches. ArXiv abs/1906.02961 (2019) 3
9. Hwang, H.W., Park, J.H., Moon, J.H., Yu, Y., Kim, H., Her, S.B., Srinivasan, G., Aljanabi, M.N.A., Donatelli, R.E., Lee, S.J.: Automated identification of cephalometric landmarks: Part 2-might it be better than human? The Angle Orthodontist 90(1), 69–76 (2020) 3
10. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. ArXiv abs/1804.02767 (2018) 3
11. Lee, S.M., Kim, H.P., Jeon, K., Lee, S.H., Seo, J.K.: Automatic 3d cephalometric annotation system using shadowed 2d image-based machine learning. Physics in Medicine & Biology 64(5), 055002 (feb 2019) 3
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014) 3, 6
13. Kang, S.H., Jeon, K., Kim, H.J., Seo, J.K., Lee, S.H.: Automatic three-dimensional cephalometric annotation system using three-dimensional convolutional neural networks: a developmental trial. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization 8(2), 210–218 (2020) 3
14. Lachinov, D., Getmanskaya, A., Turlapov, V.: Cephalometric landmark regression with convolutional neural networks on 3d computed tomography data abs/2007.10052 (2020) 3, 4
15. Nibali, A., He, Z., Morgan, S., Prendergast, L.: Numerical coordinate regression with convolutional neural networks. ArXiv abs/1801.07372 (2018) 3
16. Luvizon, D.C., Tabia, H., Picard, D.: Human pose regression by combining indirect part detection and contextual information. Comput. Graph. 85, 15–22 (2017) 3
17. OpenCV, Open Source Computer Vision Library. http://opencv.org, last accessed 30 Jun 2020 4
18. Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Instance normalization: The missing ingredient for fast stylization. ArXiv abs/1607.08022 (2016) 6

19. Xie, S., Girshick, R.B., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 5987–5995 (2017) 6
20. Wu, Y., He, K.: Group normalization. In: ECCV (2018) 6
21. Issue about slow trilinear upscaling interpolation in OpenVINO. Last accessed 30 Jun 2020 6