ДОДАТОК А

АПРОБАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ

The Neural Network Technologies Effectiveness for Face Detection

Kirill Smelyakov
Department of Electronic Computers
Kharkiv National University of Radio Electronics
Ukraine, Kharkiv, Nauky Ave, 14
kyrylo.smelyakov@nure.ua

Anastasia Chupryna
Department of Software Engineering
Kharkiv National University of Radio Electronics
Ukraine, Kharkiv, Nauky Ave, 14
anastasiya.chupryna@nure.ua

Oleksandr Bohomolov
Department of Software Engineering
Kharkiv National University of Radio Electronics
Ukraine, Kharkiv, Nauky Ave, 14
oleksandr.bohomolov@nure.ua

Igor Ruban
Department of Electronic Computers
Kharkiv National University of Radio Electronics
Ukraine, Kharkiv, Nauky Ave, 14
ihor.ruban@nure.ua

***Abstract* – The paper is devoted to estimation of the effectiveness of the use of modern convolutional neural networks for face detection. On standard open datasets, learning of neural networks and comparison of the effectiveness of their functioning are carried out. Conclusions are drawn regarding the practical application of the neural networks for detecting faces on digital photographs.**

***Keywords* – Convolutional Neural Network, Face Detection, Architecture, Model, Effectiveness.**

I. Introduction

The problem of face recognition has become very important in the field of biometric identification. Facial detection in digital images has many applications in solving real-life problems, including entertainment, healthcare, security, etc. [1]. To this day, there are a lot of approaches and models for solving these tasks. Although older algorithms, such as the Viola-Jones algorithm perform facial recognition tasks fast and accurately in close to optimal conditions, their performance diminishes when input images are distorted, too dark or too bright, or when faces appear in the image under different angles [2]. The development of deep learning and deep convolutional neural networks (NN) allowed researches to invent complex neural network-based machine learning algorithms which, when properly trained, perform much better and reliably on images with faces in different environments, angles and lightning conditions [2]. Also, each of the emerged neural network-based algorithms has its features and characteristics which determine its efficiency when applied to different types of applications.

The problem of facial recognition is a subset of a wider problem, object detection. There are two families of neural network-based algorithms that are used most often for object detection: R-CNN algorithms [3], that use Regional Proposal Networks (RPNs) to identify areas in the image where an object may be, and YOLO (an acronym for you only look once), which is faster, however less accurate, algorithm than R-CNN [4]. There are other approaches and algorithms to object detection, however, the two approaches above are the ones that saw wide usage and application in the industry. Each of these approaches is also incrementally enhanced to improve accuracy and address their respective shortcomings.

There is also a NN that was developed specifically to solve the facial recognition problem, MTCNN. This model consists of three separate NN models, the P-Net, R-Net, and O-Net, which determine not only the bounding box around the face but also the location of eyes, nose, and mouth [5].

In this paper, we compare the productivity of fast NN models: YOLOv3 and MTCNN. We use transfer learning to train the YOLOv3 model on the Wider Face dataset using pre-trained weights and compare it with the fully trained MTCNN model. For comparison, we use various object detection metrics, such as Intersection over Union (IOU), precision, recall, and others, which are described in the following sections. We do not include R-CNN family algorithms in the comparison because the training of these algorithms is computationally expensive and we lack computational resources to fully train them.

The dataset we are using is the Wider Face dataset [6]. This dataset was assembled to benchmark different face detection algorithms. It consists of images of people in different situations, the faces on these images are located in different areas, angles and lighting conditions.

The reliable performance of facial recognition algorithms is important for the functioning of modern informational and communicational systems [7-10].

II. Neural Network Models

So, in this paper, we compare two NN architectures for face detection – YOLOv3 [11] and MTCNN [5].

YOLOv3 is the further development of the original YOLO architecture. Original YOLO was developed with speed and efficiency in mind, because at that time the state-of-the-art model for object detection, R-CNN, was too slow in training and inference to use it in such tasks as real-time detection. However, original YOLO, while being faster and more resource-efficient, was less accurate than R-CNN. Because of this, further iterations of YOLO development improved accuracy but became significantly slower.

Before YOLO, R-CNN models used a pipeline of two different algorithms, one for generating regional proposals, i.e. areas where an object might be, and the other one to determine whether the proposed region includes an object the network is trying to find.

Original YOLO introduced the way to unify two of these algorithms into a single neural network [4]. It was done by dividing the image into a square grid $S \times S$. Each cell is responsible for predicting B bounding boxes and a confidence score for each of those boxes.

If there is no object in the cell, the confidence score should be zero. Otherwise, it should be equal to the intersection over union (IOU) between the predicted box and the ground truth. Each bounding box consists of 5 predictions: x, y, w, h and a confidence score, which was described earlier.

The x and y coordinates are relative to the cell, and w and h are relative to the width and height of the image respectively. Each cell also predicts C class probabilities, where C equals to the number of object classes the network tries to detect.

Original YOLO network architecture consists of 24 convolutional layers and 2 fully connected layers. Next improvement, YOLO v2, introduced a custom architecture darknet-19, a 19-layer network supplemented with additional 11 levels for object detection. However, even with these additions, the improved architecture still lacked important elements that are widely used in modern state-of-the-art neural networks for computer visions, such as residual blocks, skip-connections and upsampling. The next generation of YOLO, YOLOv3, incorporates all of these.

YOLOv3 uses a variant of Darknet architecture (fig. 1), with a 53-layered network, together with 53 more layers for detection. This greatly improves the accuracy of the trained models, however, this also substantially increases the time needed for training and inference. One of the most interesting features of YOLOv3 is that it makes detection on 3 different scales. Because of this, YOLOv3 detects small objects much accurately than older versions of YOLO.

YOLO is a neural network architecture for performing general object detection, i.e., detecting as many different object classes as needed. However, in this paper, we compare different neural network architectures for face detection. For this task, there is a specialized neural network architecture, MTCNN [5].

MTCNN stands for Multi-task Cascaded Convolutional Network (MTCNN). Instead of being a single neural network model as YOLO, MTCNN includes three neural networks, called P-Net, R-Net and O-Net (fig. 2). It returns not only the bounding boxes of detected faces but also 5 facial landmarks: 2 for eyes, 1 for a nose and 2 for a mouth.

MTCNN works the following way. Firstly, it creates an image pyramid of multiple scaled versions of an input image.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 1. Darknet 53 architecture [11]

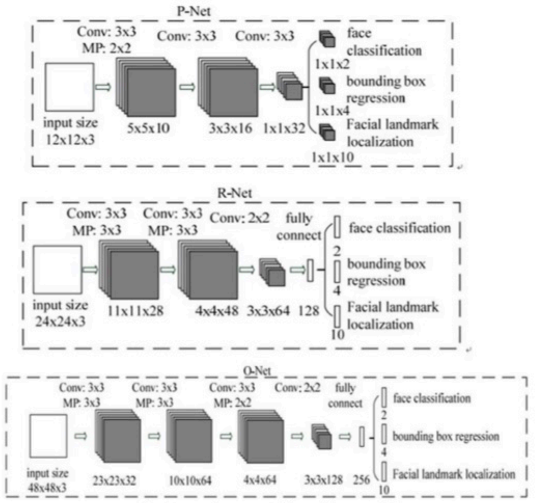


Fig. 2. MTCNN model architecture [5]

Then for of scaled images, it applies a sliding 12x12 kernel with a stride of two. The portion of an image under the kernel is passed to the first neural network, the Proposal Network (P-

Net). P-Net returns the bounding box if it finds the face. It also returns the confidence score for each of the bounding boxes.

After that, all P-Net outputs are collected and the bounding boxes with low confidence scores are removed. Then all the coordinates of bounding boxes are being unscaled to match the original input size. However, at this stage, there is still a lot of bounding boxes, so the Non-Maximum Suppression (NMS) method is applied to remove redundant bounding boxes and merge several closely located bounding boxes into one.

After that, the remaining bounding boxes are fed into another convolutional neural network, the Refine Network (R-Net). This network rejects a large number of region proposals from remaining candidates and performs further calibration of bounding boxes. After that, NMS is applied to the result once again, rejecting false bounding boxes.

The final stage is similar to the second stage. The output of the second stage is fed to the last network, the Output Network (O-Net). This time, in addition to the bounding boxes, this network outputs also locations of facial landmarks.

The neural networks mentioned above perform 3 tasks: face / not face classification, bounding box detection and facial landmark detection. For each of these tasks, there is a corresponding loss function

$$L_i^{det} = -\left(y_i^{det}\log(p_i) + \left(1 - y_i^{det}\right)\left(1 - \log(p_i)\right)\right). \quad (1)$$

In the formula above p_i is the probability with which the input is a face, and y_i^{det} is the ground truth label. This is a binary cross-entropy loss

$$L_i^{box} = \left\|\hat{y}_i^{box} - y_i^{box}\right\|_2^2. \quad (2)$$

The bounding box determination is a regression problem, and for the loss function, we use simple Euclidean loss. In formula 2, $\hat{y}_i^{box}$ is the ground truth bounding box, and y_i^{box} is the bounding box which is the output of the neural network

$$L_i^{landmark} = \left\|\hat{y}_i^{landmark} - y_i^{landmark}\right\|_2^2. \quad (3)$$

The facial landmark detection is also the regression problem, so the loss function for facial landmark detection (3) is the same as the formula (2).

Both YOLO and MTCNN models are capable of doing face detection. The main advantage of YOLO is that together with face detection it can perform detection of any other types of objects. This is useful if, for instance, you need to detect faces together with road signs or automobiles. It is also useful if you have to detect different types of faces, for instance, faces with hats or sunglasses on. The main drawback of YOLO in comparison with MTCNN is that YOLO requires much more computational resources to be trained efficiently.

On the other hand, MTCNN can detect only faces. However, it also performs facial landmark detection, which YOLO doesn't do. Although MTCNN uses 3 neural networks instead of a single one in YOLO, its neural networks are less complex and therefore require much less time and resources to train. Also, MTCNN already ships with pre-trained weights, so unless the problem requires fine-tuning the weights to the specific dataset, no additional training is required.

The advantages and disadvantages of both models are listed above. The rest of the paper will describe the experiment and its results that show how both networks compare in terms of face detection efficiency.

III. Experimental Results Analysis

For comparison of the two models, YOLO and MTCNN, we will use the Wider Face dataset [6]. It is a face detection benchmark dataset which consists of 32,203 selected images from publicly available WIDER image dataset. The images in the Wider Face dataset are diverse. As well as portrait images, the dataset also contains group images, images from sporting events, etc. The diversity of images helps us to compare the performance of both algorithms under less-than-optimal conditions.

Due to the hardware limitations and lack of computational resources, in the experiment, we will use only the subset of the dataset. We will use 10,000 images for training and 1,000 images for inference. The small size of the dataset will not affect the experiment result significantly, because we will apply transfer learning by using already pre-trained YOLO weights. We are not going to train MTCNN because this model is created specifically for face detection and already has fully trained weights available.

We are using YOLOv3 and MTCNN model implementations that use TensorFlow framework and Keras library [12-14].

For comparison of two algorithms, we use the averaged precision as a metric and precision/recall curves [15].

To calculate precision and recall, we must determine what the true positive, false positive and false negative are in our task. For this, we have to use Intersection Over Union measure (IOU).

IOU is a measure that evaluates the overlap between two bounding boxes. It requires a ground truth bounding box B_{gt} and a predicted bounding box B_p. The formula for IOU (4) is listed below

$$IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}. \quad (4)$$

In simpler terms, IOU is a fraction between an area of overlap and an area of union (fig. 3).

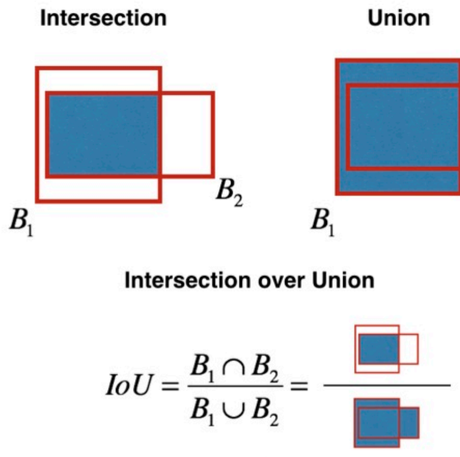
We can use IOU to determine if detection is a true positive, false positive, or a false negative.

If IOU is bigger than a threshold, the detection is correct and therefore is a true positive.

If IOU is less than the threshold, the detection is a false positive.

The detection is a false negative if no object was detected.

Using true positives, false positives and false negatives, we can calculate the precision and the recall [15].



b)
Fig. 3. Intersection Over Union [15, 16]

Precision is the ability of the model to identify only the relevant objects. It is the percentage of correct positive predictions

$$Precision = \frac{TP}{TP+FP} = \frac{TP}{all\ detections}. \quad (5)$$

Recall is the ability of the model to find all the relevant cases (all ground truth bounding boxes). It is the percentage of true positive detected among all relevant ground truths

$$Recall = \frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truths}. \quad (6)$$

To illustrate the performance of the two models we use Precision x Recall curves.

It is a good way to evaluate the performance of an object detector as the confidence is changed by plotting a curve. The model can be considered good if precision stays high as recall increases.

To compare the two models, we use the Average Precision metrics. We get average precision by calculating the area under the curve (AUC) of the Precision x Recall curve [15].

We calculate metrics for both training and validation data. For YOLO this can demonstrate if the network overfits during training. For MTCNN, it is irrelevant, because we use an already fully trained MTCNN model.

Below we list the experiment outcomes (tab. 1), and the precision x recall curves for each experiment (fig. 4-fig. 7).

TABLE I.
RESULTS OF FEATURE POINTS DETECTING FOR FACE WITH TRANSFORMATION

	YOLO	MTCNN
Training dataset AP	32.55%	41.71%
Validation dataset AP	30.88%	38.35%

Below we list the inference time that was measured during the experiment (tab. 2).

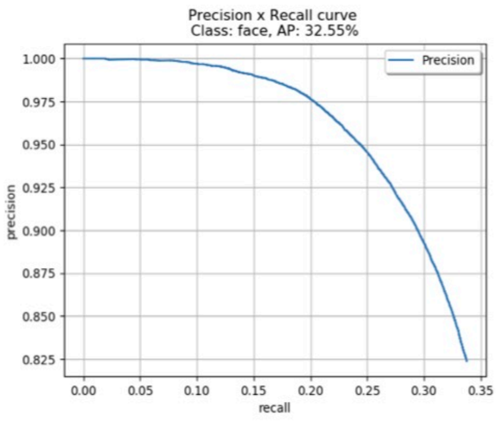


Fig. 4. YOLO training dataset precision x recall curve

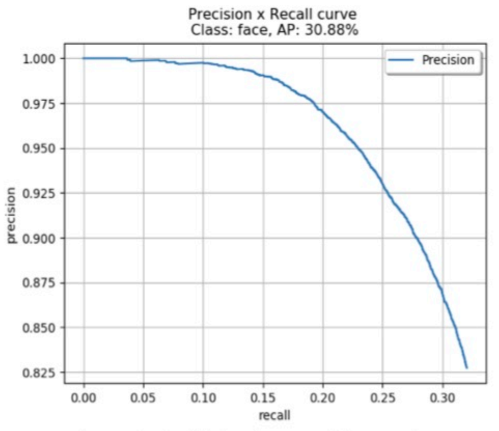


Fig. 5. YOLO validation dataset precision x recall curve

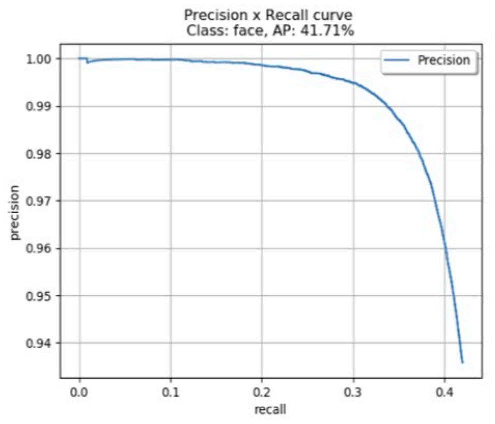


Fig. 6. MTCNN training dataset precision x recall curve

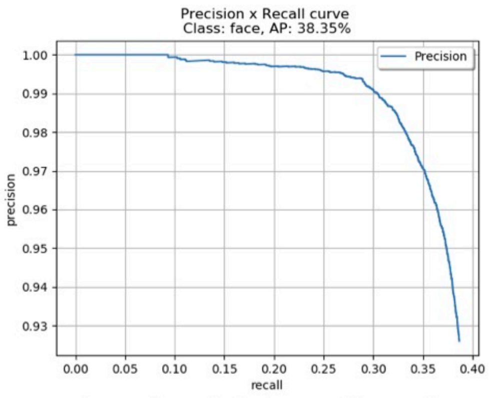


Fig. 7. MTCNN validation dataset precision x recall curve

TABLE II.
RESULTS OF FEATURE POINTS DETECTING FOR FACE WITH TRANSFORMATION

	YOLO	MTCNN
Training dataset (10,000 examples), seconds	978.45	2475
Validation dataset (2,500 examples), seconds	269.77	732.59

As we can say from the experiment outcomes and precision x recall curves listed above, the MTCNN model performs better and more accurately than the YOLO model. MTCNN model was developed specifically for face detection problems, while YOLO was developed as a general object detection algorithm. MTCNN neural networks are also much less complex than YOLO neural network.

Despite the fact that neural networks in MTCNN models are much less complex than YOLO neural network, in the inference the MTCNN model is about 3 times slower than the YOLO model. This is because MTCNN includes 3 neural networks and uses the Non-Maximum Suppression method.

The final conclusion we can make is that in most cases MTCNN should be used for face detection tasks. YOLO should be used when you have to detect other objects rather than just faces or your dataset is very specific and you have to use transfer learning to adjust the weights. Also, YOLO is preferable when fast inference speed is required. In other cases, MTCNN is a good solution for face detection tasks.

CONCLUSION

In this work, we compared two neural network-based models applied to the task of face detection, YOLO and MTCNN. We did not include another popular neural network-based model Faster R-CNN into the comparison because we lacked computational resources to train it properly. YOLO is a network for general object detection, it can detect any objects, not just faces. MTCNN is a model designed specifically for face detection.

YOLOv3 is the latest iterative improvement to the YOLO model. It contains a single but complex neural network that requires a lot of computational resources to train. MTCNN contains 3 much simpler neural networks, but the MTCNN algorithm is complicated and performs a lot of transformations on input data.

The experiment was performed using the Wider Face dataset. The YOLO model was trained using pre-trained weights. The MTCNN model was used with fully pre-trained weights. Average precision and precision x recall curves were used as comparison metrics.

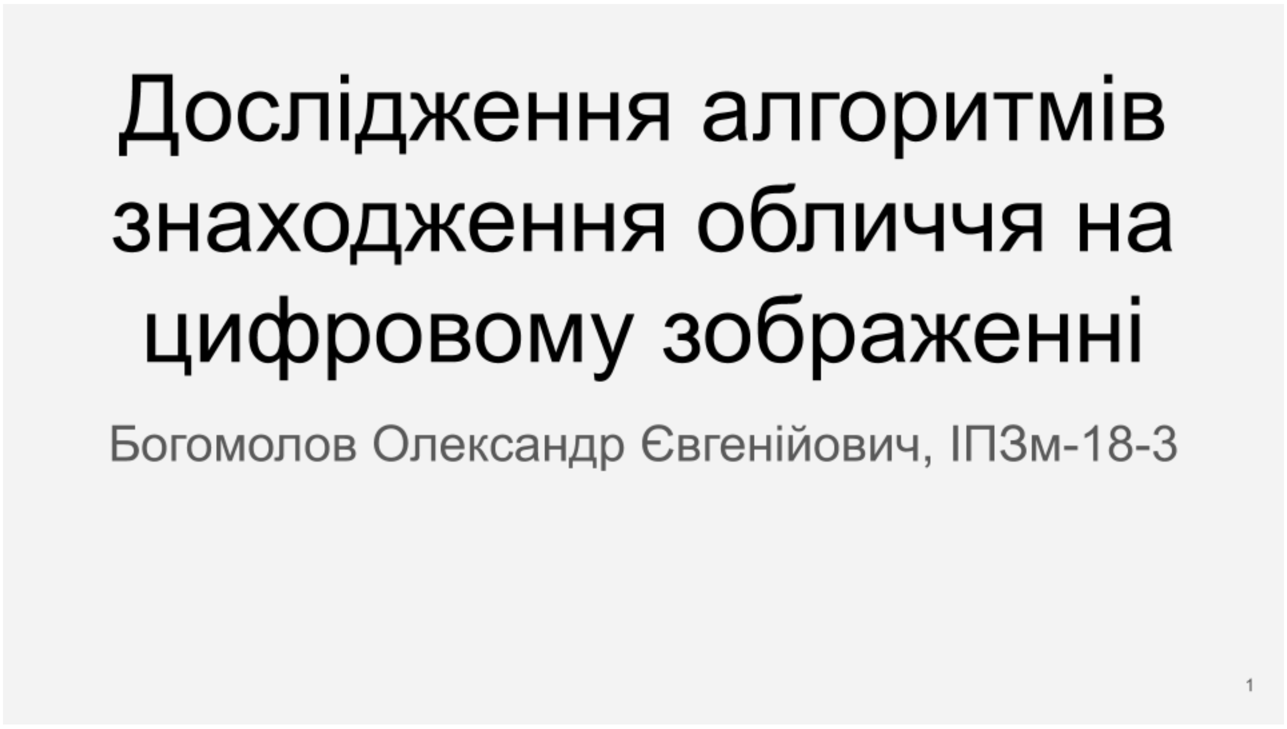
The outcomes of the experiment show that MTCNN performs better on the task of face detection. In most cases, MTCNN should be used for face detection tasks. YOLO should be used if there is a requirement to detect not only faces but other objects as well, fast inference speed is required, or if the dataset is very specific and transfer learning has to be used to adjust the weights. The MTCNN model also detects facial landmarks. This is useful for other advanced facial recognition tasks, such as emotion recognition.

REFERENCES

[1] Asit Kumar Datta, Madhura Datta, Pradipta Kumar Banerjee Face Detection and Recognition. - CRC Press, 2015. - 352p.
[2] G. Blokdyk Facial Recognition A Complete Guide. - 5STARCooks, 2018. - 126p.
[3] R-CNN, Fast R-CNN, Faster R-CNN, YOLO: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
[4] J. Redmon, S. Divvala, R. Girshick, A. Farhadi You Only Look Once: Unified, Real-Time Object Detection // IEEE Conference on Computer Vision and Pattern Recognition (CVPR) – 2015.
[5] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Yu Qiao Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks // Signal Processing Letters. - Vol. 23 , Iss. 10, 2016. - P. 1499-1503.
[6] WIDER FACE: A Face Detection Benchmark: http://shuoyang1213.me/WIDERFACE
[7] V. Mukhin, Y. Romanenkov, J. Bilokin, A. Rohovyi, A. Kharazii, V. Kosenko, N. Kosenko, J. Su. The method of variant synthesis of information and communication network structures on the basis of the graph and set-theoretical models / International Journal of Intelligent Systems and Applications(IJISA). – 2017. – № 9 (11). – P. 42-51.
[8] Boiko J., Tolubko V., Barabash O., Eromenko O., Havrylko Ye. Signal processing with frequency and phase shift keying modulation in telecommunications / TELKOMNIKA Telecommunication, Computing, Electronics and Control. – 2019. – Vol. 17, № 4. – P. 2025-2038.
[9] Yeremenko, O., Lebedenko, T., Vavenko, T. and Semenyaka, M. Investigation of queue utilization on network routers by the use of dynamic models / Second International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), October 2015. – P. 46-49.
[10] Tkachov, V., Tokariev, V., Dukh, Y., & Volotka, V. Method of Data Collection in Wireless Sensor Networks Using Flying Ad Hoc Network / International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), October, 2018. – P. 197-201.
[11] J. Redmon, A. Farhadi YOLOv3: An Incremental Improvement // ArXiv – 2018.
[12] TensorFlow: https://www.tensorflow.org
[13] YOLOv3 implementation: https://github.com/qqwweee/keras-yolo3
[14] MTCNN implementation: https://github.com/ipazc/mtcnn
[15] Metrics for object detection https://github.com/rafaelpadilla/Object-Detection-Metrics
[16] YOLOv3 implementation: https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193

ДОДАТОК Б

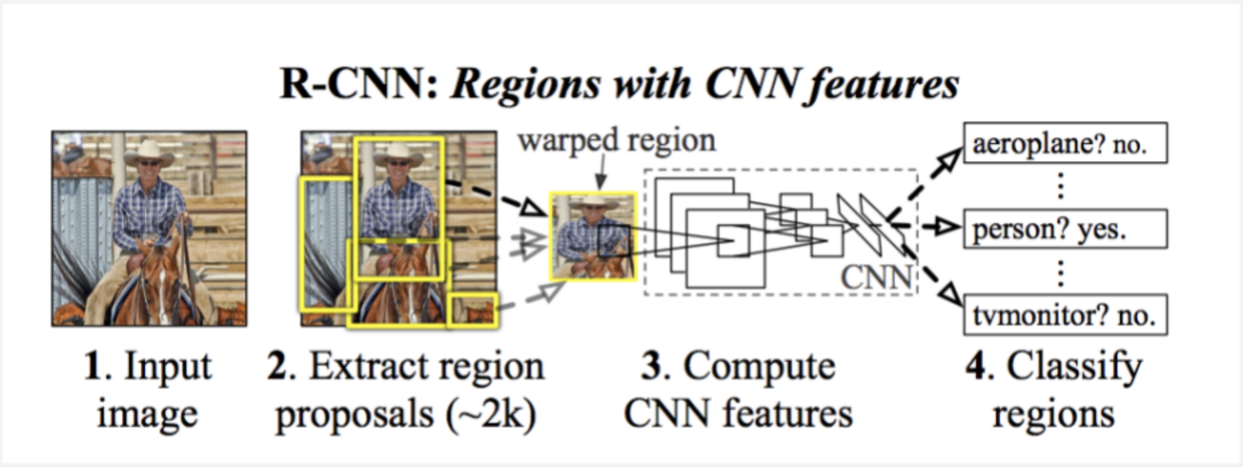
СЛАЙДИ ПРЕЗЕНТАЦІЇ



Дослідження алгоритмів знаходження обличчя на цифровому зображенні

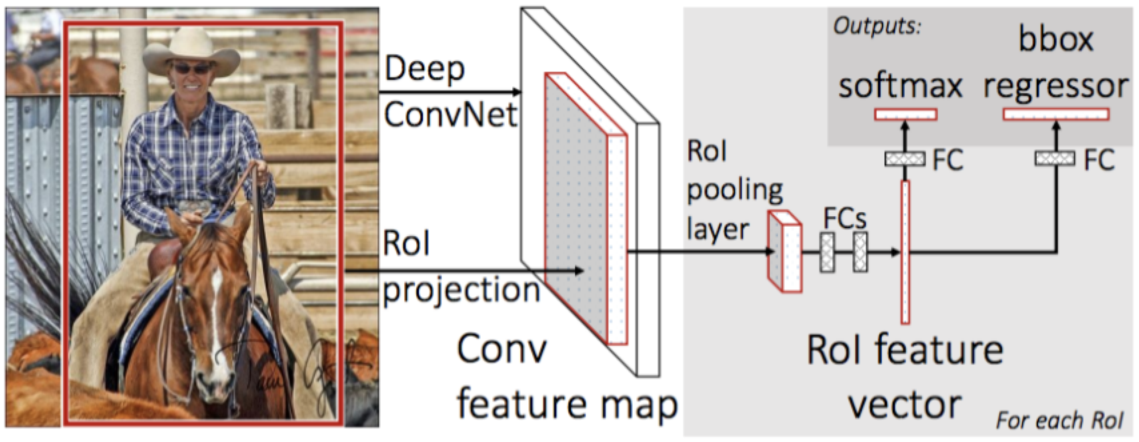
Богомолов Олександр Євгенійович, ІПЗм-18-3

1



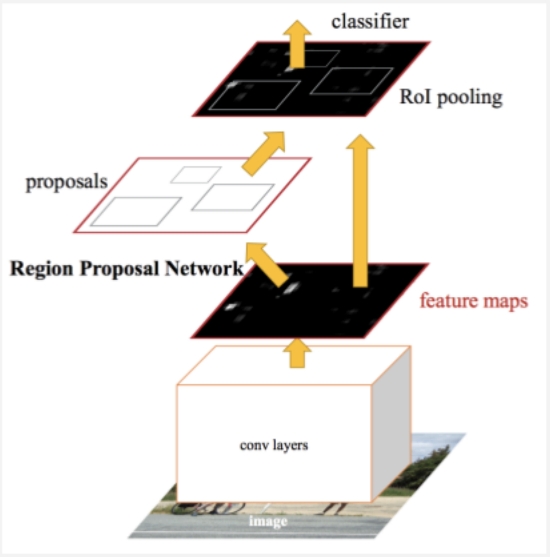
Принцип роботи R-CNN

2



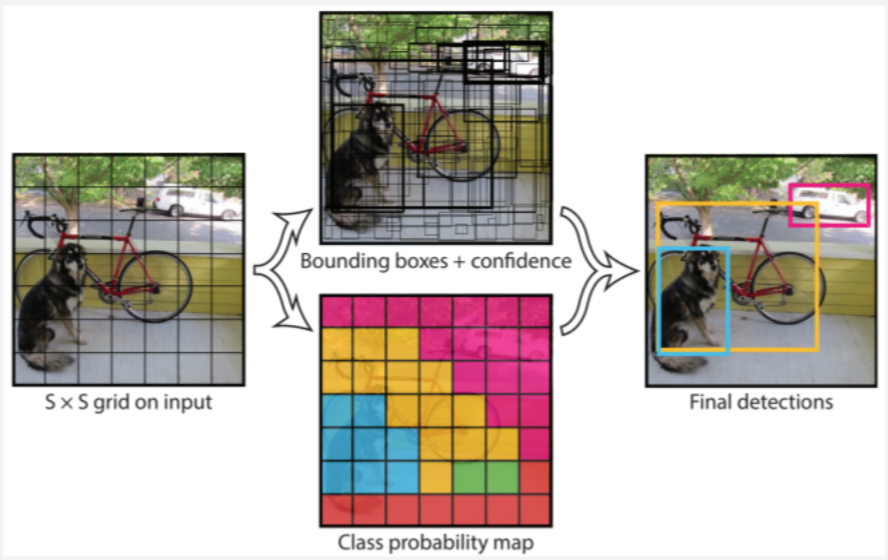
Принцип роботи Fast R-CNN

3

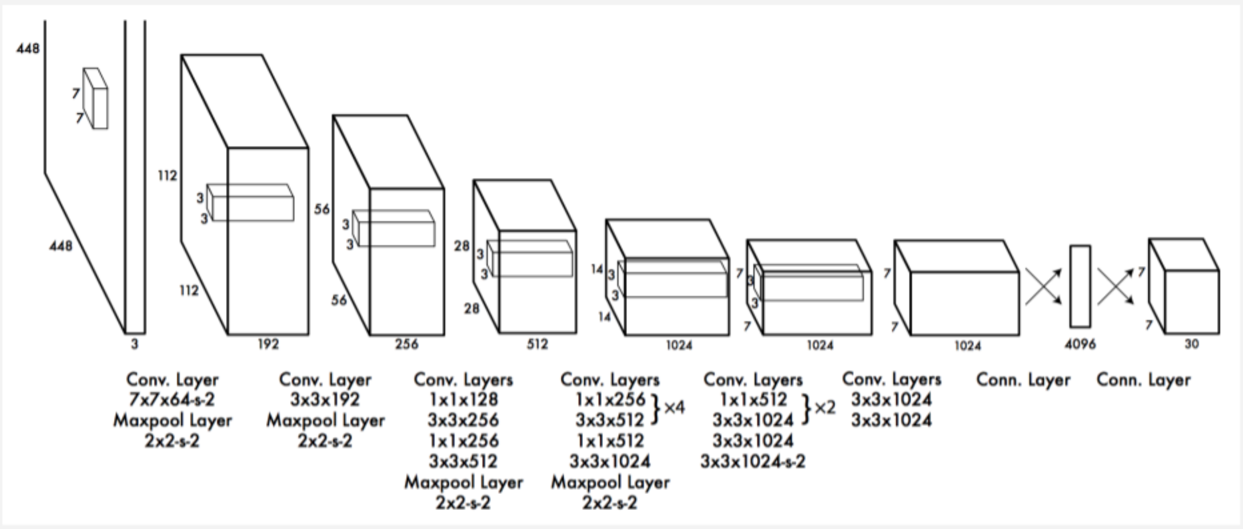


Принцип роботи Faster R-CNN

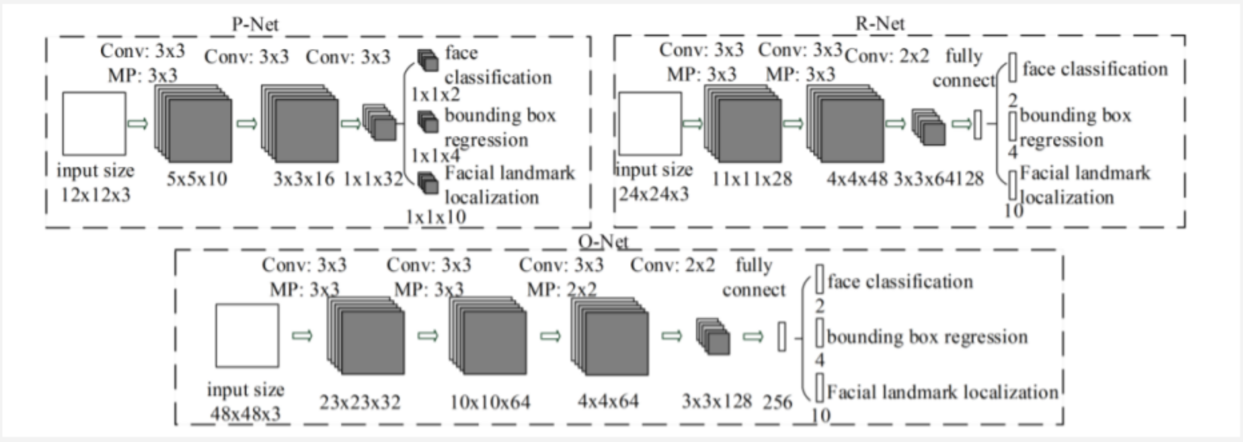
4



Принцип роботи YOLO

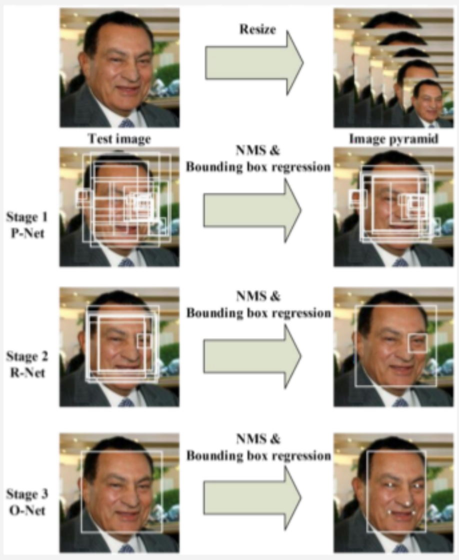


Архітектура YOLO



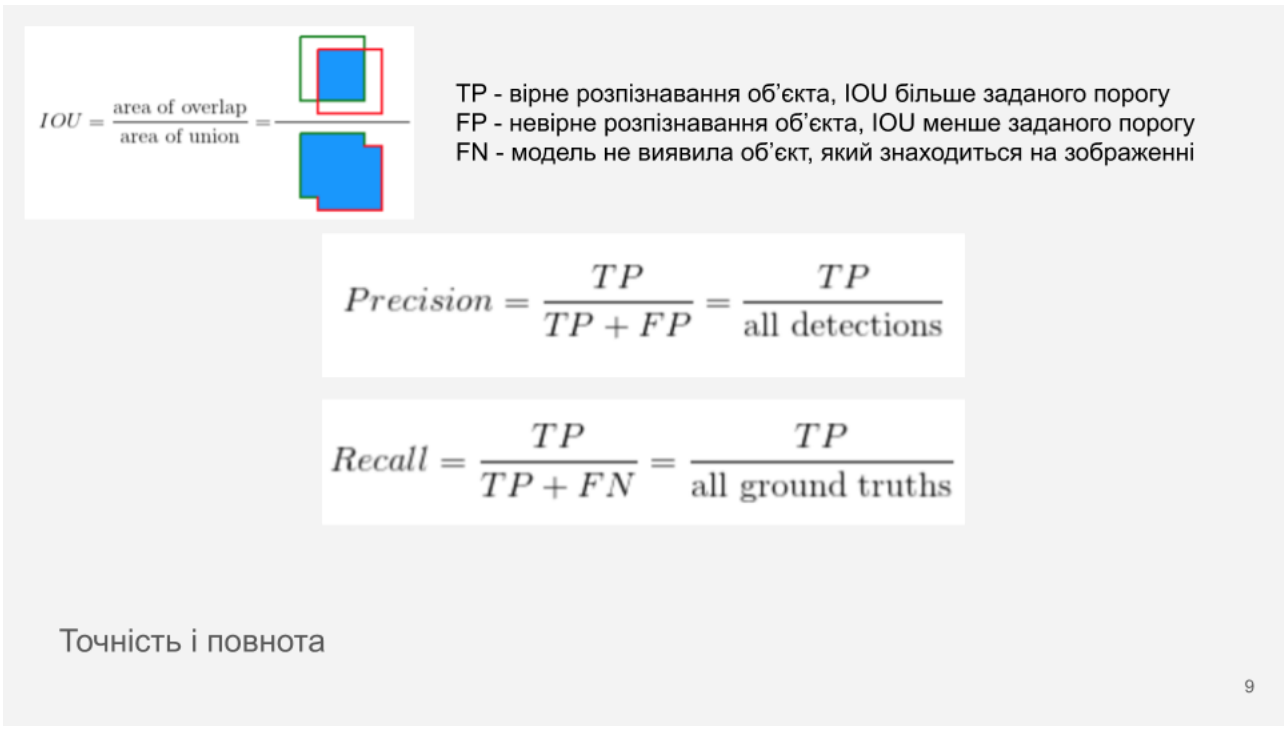
Архітектура MTCNN

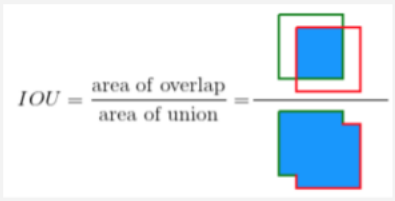
7


Resize
Test image
Image pyramid
Stage 1 P-Net
NMS & Bounding box regression
Stage 2 R-Net
NMS & Bounding box regression
Stage 3 O-Net
NMS & Bounding box regression

Принцип роботи MTCNN

8





$$IOU = \frac{\text{area of overlap}}{\text{area of union}}$$

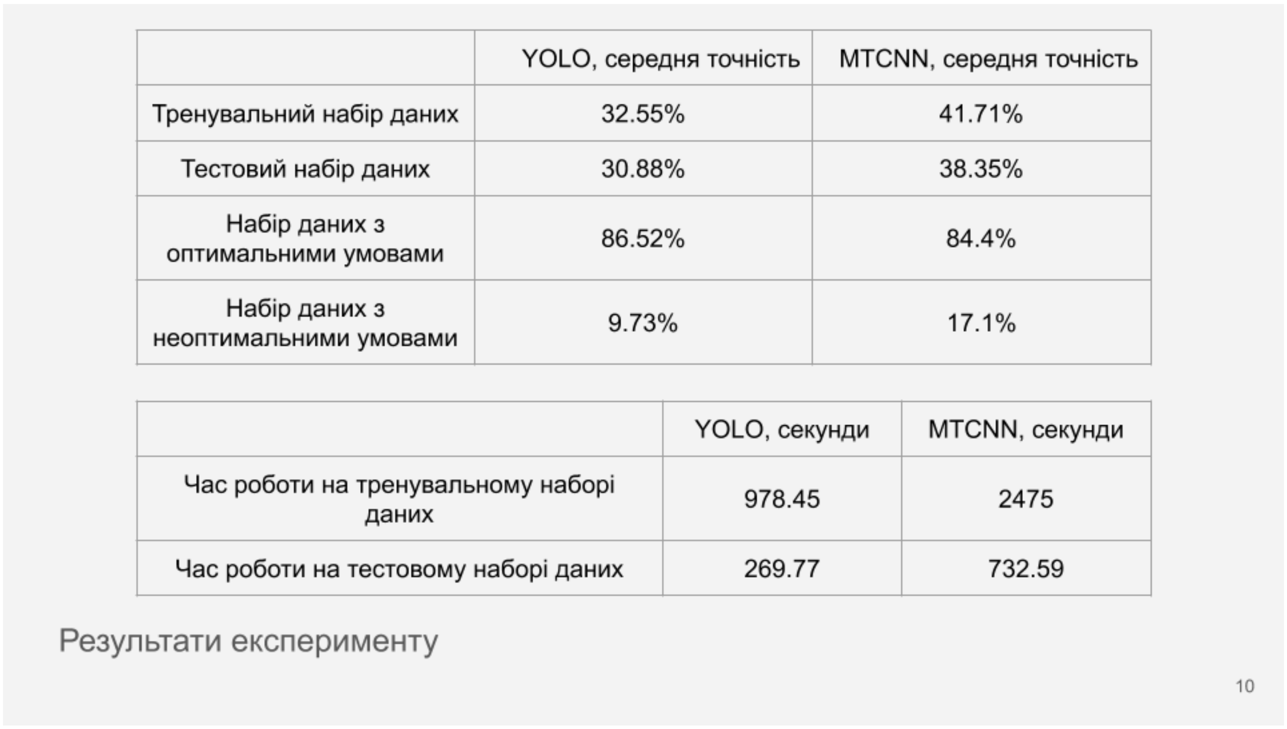
TP - вірне розпізнавання об’єкта, IOU більше заданого порогу
FP - невірне розпізнавання об’єкта, IOU менше заданого порогу
FN - модель не виявила об’єкт, який знаходиться на зображенні

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}}$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}}$$

Точність і повнота

9

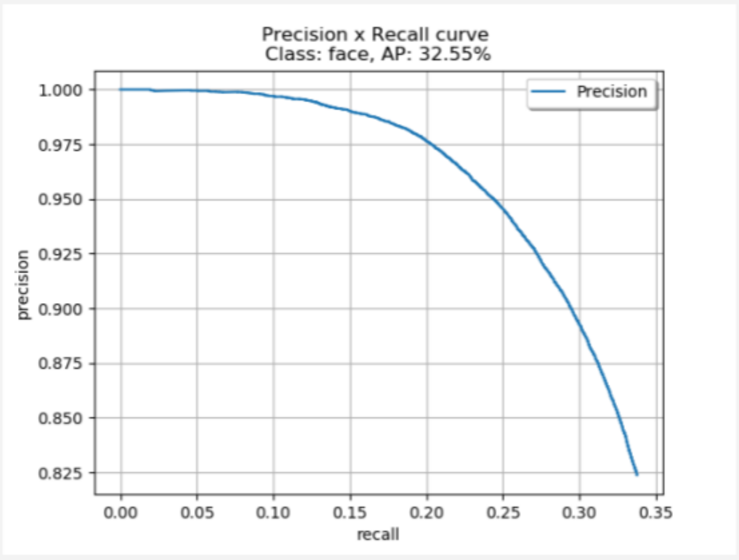


	YOLO, середня точність	MTCNN, середня точність
Тренувальний набір даних	32.55%	41.71%
Тестовий набір даних	30.88%	38.35%
Набір даних з оптимальними умовами	86.52%	84.4%
Набір даних з неоптимальними умовами	9.73%	17.1%

	YOLO, секунди	MTCNN, секунди
Час роботи на тренувальному наборі даних	978.45	2475
Час роботи на тестовому наборі даних	269.77	732.59

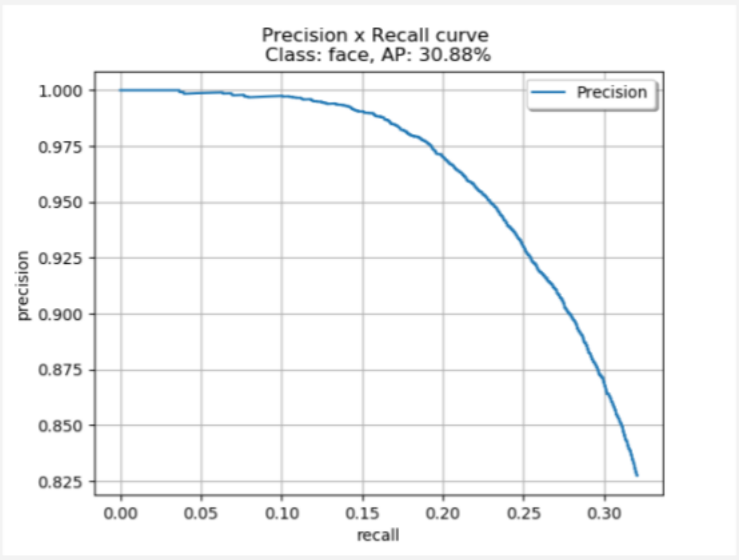
Результати експерименту

10



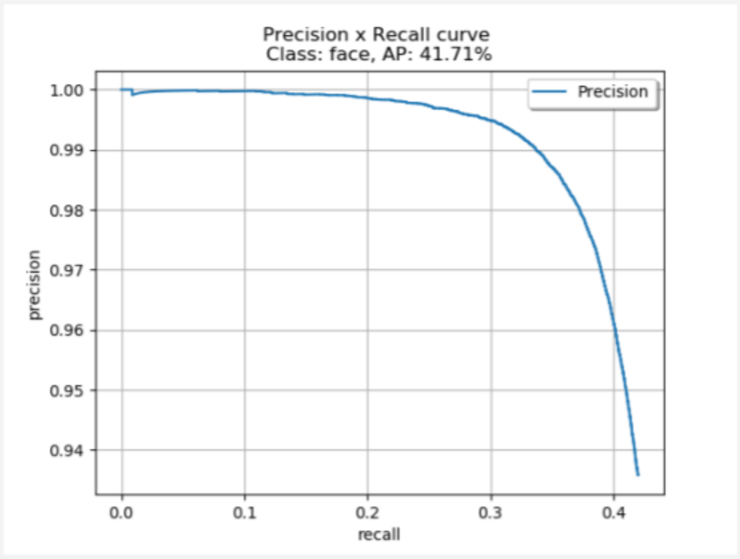
Крива точності і повноти для тренувального набору даних і моделі YOLO

11



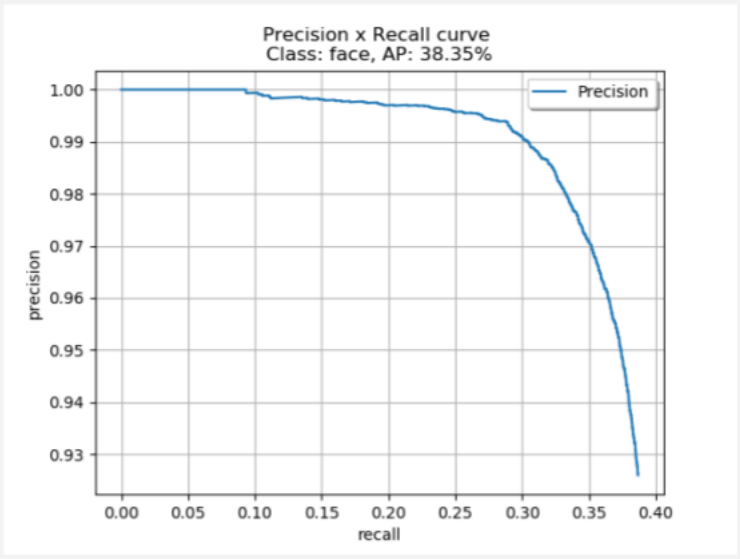
Крива точності і повноти для тестового набору даних і моделі YOLO

12



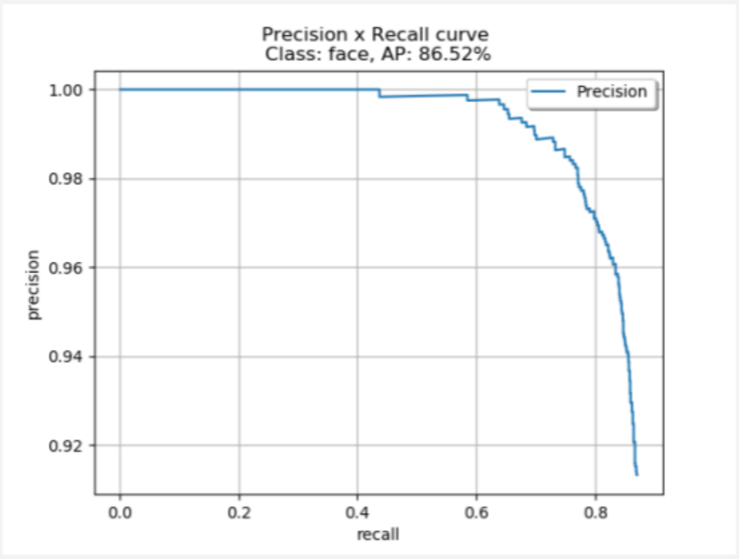
Крива точності і повноти для тренувального набору даних і моделі MTCNN

13



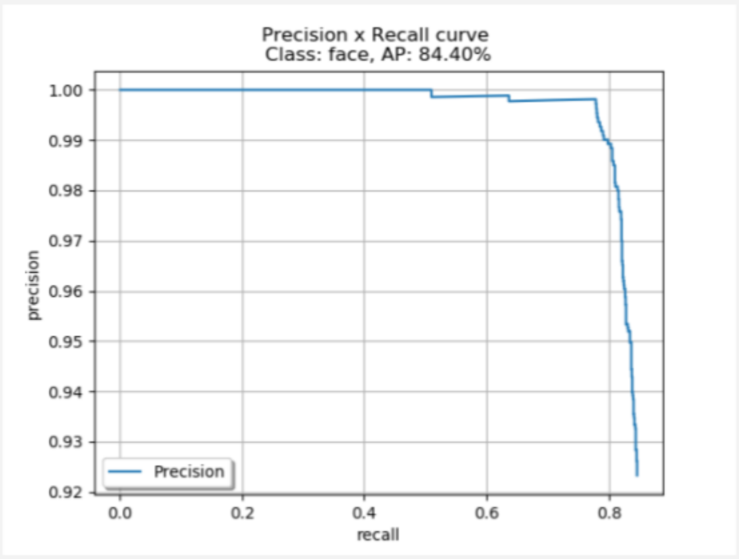
Крива точності і повноти для тестового набору даних і моделі MTCNN

14



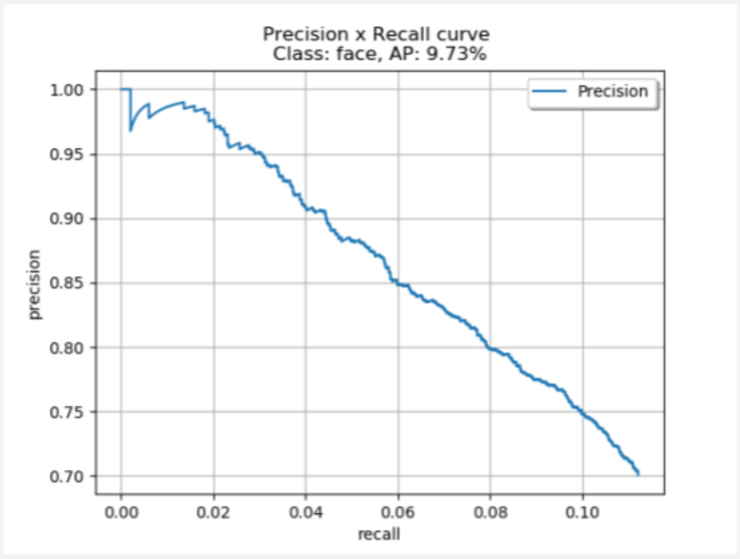
Крива точності і повноти для набору даних з оптимальними умовами і моделі YOLO

15



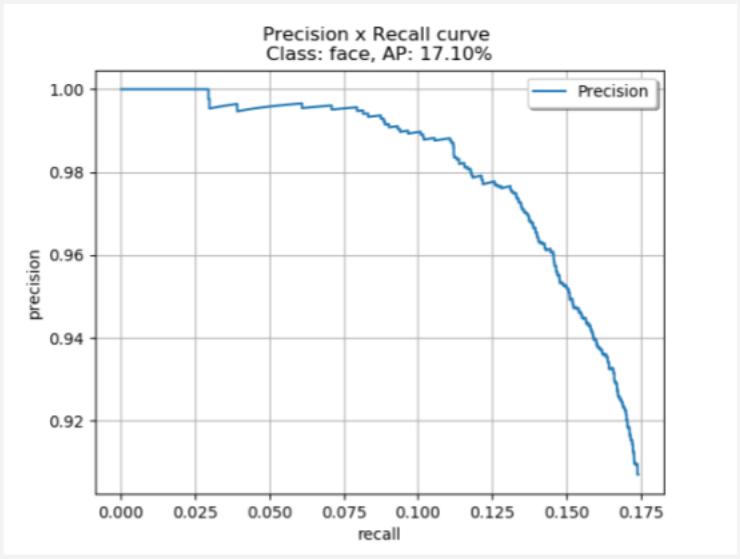
Крива точності і повноти для набору даних з оптимальними умовами і моделі MTCNN

16



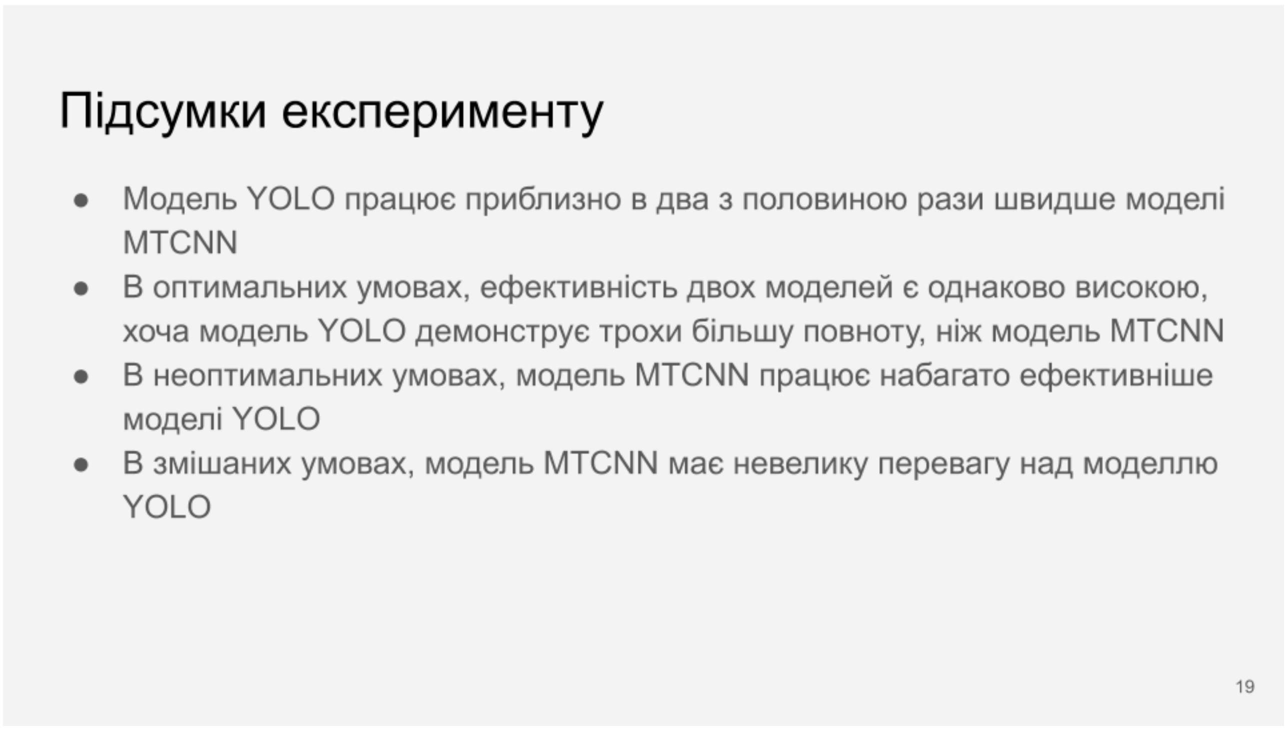
Крива точності і повноти для набору даних з неоптимальними умовами і моделі YOLO

17



Крива точності і повноти для набору даних з неоптимальними умовами і моделі MTCNN

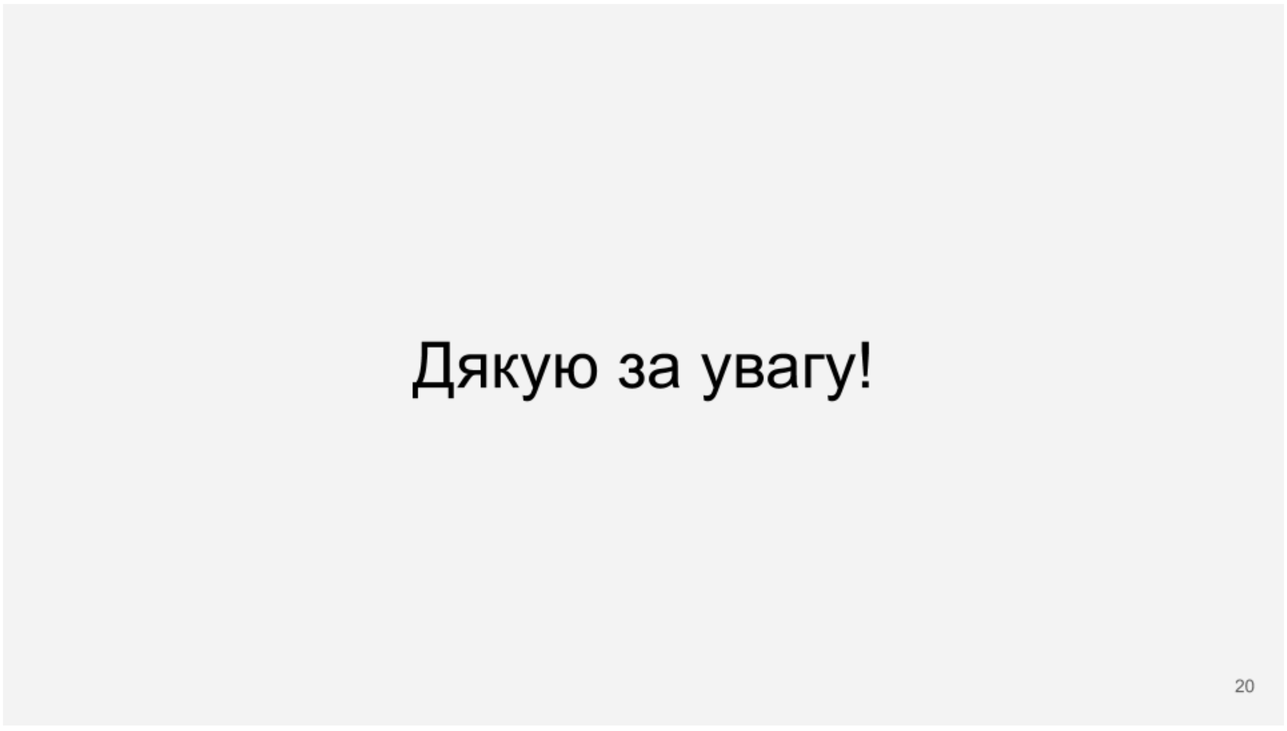
18



Підсумки експерименту

- Модель YOLO працює приблизно в два з половиною рази швидше моделі MTCNN
- В оптимальних умовах, ефективність двох моделей є однаково високою, хоча модель YOLO демонструє трохи більшу повноту, ніж модель MTCNN
- В неоптимальних умовах, модель MTCNN працює набагато ефективніше моделі YOLO
- В змішаних умовах, модель MTCNN має невелику перевагу над моделлю YOLO

19



Дякую за увагу!

20