

## DEVELOPING A SEMANTIC IMAGE MODEL USING MACHINE LEARNING BASED ON CONVOLUTIONAL NEURAL NETWORKS

*Philip Andon*<sup>a[0000-0001-6546-0826]</sup>, *Andrii Hlybovets*<sup>b[0000-0003-4282-481X]</sup>, *Volodymir Kuryliak*<sup>b</sup>

<sup>a</sup>Institute of Software Systems of the National Academy of Sciences of Ukraine, 03187, Kyiv, 40 Akademika Glushkova Avenue

<sup>b</sup>National University “Kyiv-Mohyla Academy”, 04070, 2 Skovorody vul., Kyiv 04070, Ukraine

У роботі описано основні напрямки досліджень у сфері побудови моделей автоматизації комп'ютерного розпізнавання сутності цифрового зображення. Введено поняття семантичної моделі зображення та описано реалізацію моделі машинного навчання для вирішення задачі автоматичної побудови такої моделі для вхідного зображення. Семантична модель складається зі списку об'єктів, які показано на зображенні, та їх зв'язків. Розроблена модель була порівняна з іншими рішеннями для цієї самої проблеми і показала кращі результати в усіх, за винятком одного, випадків. Ефективність роботи моделі обґрунтована використанням останніх досягнень машинного навчання, зокрема ЗНМ, TL, моделей Faster R-CNN і VGG16. Значна частина зв'язків представлених на зображенні є просторовими зв'язками, таким чином, для кращої роботи моделі, потрібно використовувати цей факт у її проектуванні, що і було зроблено.

Ключові слова: семантична модель зображення, машинне навчання, комп'ютерний зір, згорткові нейронні мережі, зв'язки на зображенні.

В работе описаны основные направления исследований в области построения моделей автоматизации компьютерного распознавания сущности цифрового изображения. Введено понятие семантической модели изображения и описано реализацию модели машинного обучения для решения задачи автоматического построения такой модели для входного изображения. Семантическая модель состоит из списка объектов, которые показаны на изображении, и их связей. Разработанная модель была сравнена с другими решениями для этой самой проблемы и показала лучшие результаты во всех, за исключением одного, случаев. Эффективность работы модели обоснована использованием последних достижений машинного обучения, в частности ЗНС, TL, моделей Faster R-CNN и VGG16. Значительная часть связей представленных на изображении есть пространственными связями, таким образом, для лучшей работы модели, нужно использовать этот факт в ее проектировании, что и было сделано.

Ключевые слова: семантическая модель изображения, машинное обучение, компьютерное зрение, сверточные нейронные сети, связи на изображении.

Understanding (interpretation) of images is one of the most urgent tasks of artificial intelligence. Image understanding we can use in the wide range of possible applications. Visual information is one of the most popular and widespread type of information. As its number increases, the issue of processing automation becomes extremely relevant. Simple tasks, such as automatic annotation of images or search for similar images, can be considered solved, but in terms of a truly deep understanding of the image, modern methods are far from ideal.

In recent years, many researches has been done to approximate the level of understanding of images by humans and automated systems, in particular, to solve problems such as creating a text description or finding objects on the image. However, in order for a computer to fully interpret ("understand") the content of an image (what is represented on it), it is necessary to obtain a formal structured representation of all the information contained in the image. The format and structure of such a presentation requires research. We call this task a semantic image model.

This paper describes the main areas of research in the field of developing computer models for the computer vision. The article provide the concept of the semantic image model. During development of the semantic image model the newest results in computer vision such as CNN, TL, Faster R-CNN, and VGG16 were used. The semantic model consists of a list of objects represented in the image and their relationships. These relations represented as triplets such as "object, relation, and subject". Our model as input received image and as outcome produce a list of triplets. Neural network trained on a collection Visual Genome. As a quality evaluation metric authors used "precision from K". The performance of the model is justified by the use of the latest works in this area. A lot of the relations between objects on the image are spatial links/ This fact were used during model development.

Key words: semantic image model, machine learning, computer vision, convolutional neural networks, image links.

### Introduction

Understanding (interpretation) of images is nowadays one of the most important tasks of artificial intelligence [1]. The interest of many researchers to the topic is conditioned by a wide range of its possible applications. Visual information is one of the most popular and common type of information, on the Internet in particular. As the amount of visual information increases, the issue of its automatic processing becomes extremely relevant. Such simple tasks, as automatic annotation of images or search for similar images, can be considered solved [2, 3, 4, 5], however, in terms of a truly deep understanding of images, modern methods are far from perfect.

In recent years, much research has been done to approximate levels of understanding of images – by humans and by automated systems, in particular, to solve such problems as creating a text description or finding some objects in the

image [1, 5]. However, in order that a computer could fully interpret ("understand") the content of an image (i.e. objects represented in an image), it is crucial for it to receive a formal structured representation of all the information an image contains. The format and structure of such a representation requires research. We name this task a **semantic image model**. A semantic image model includes a list of the objects represented in the image and their relationships.

Therefore, the aim of this paper is to represent our vision of automated generation of image description. We introduce here the term "semantic model" to formalize the representation of an image in the picture. The presence of the term "semantic" means that such a representation should help to reveal the essence of the image based on an approach similar to the mechanism of human interpretation of images. And the word "model" sets requirements for clarity and formalization of the structure of the representation. These comments are important because our goal is to automate the process of computer image processing.

There are several approaches aimed at solving the problem of "understanding images" or related to it partially.

Among the main lines of research in the field of developing automated models for computer recognition of the contents of digital images, there are related tasks of image classification [6], text description of images [7], determining the relationships in the image [8]. Recent studies have focused on promising approaches of determining relationships in an image with the help of visual phrases "Recognition Using Visual Phrases" [9], "Visual Relationship Detection With Language Priors" [10], "Deep Relational Network" [8]. The latter one is designed specifically to effectively use the statistical characteristics of interdependence between objects and their relationships in the image in machine learning on the basis of convolutional neural networks.

In this paper, we are going to articulate our approach to solving the problem of building a semantic image model. The model is machine learning (ML) oriented. It must receive an image at the input and return a set of subjects, objects, and their relationships at the output.

### **Development of a semantic model**

In ML, expert knowledge of the subject area allows you to better understand the problem and come to the best possible solution. That is why a model design development begins with data analysis and finding the right collections. We are also going to start by finding the right data collection.

**Training set.** In order to solve our task, we used the Visual Genome collection [11]. It is created and successfully applied to resolve various computer vision challenges related to improving the "understanding" of images. On the official website [12] one can find training collections, in particular, for such tasks as finding objects, defining relationships between them, creating a text description. There are about 108,000 images. It contains a list of objects present in an image, with their positions and relationships between them, for each of the images. Therefore, this data is exactly what is necessary to solve our task.

The Visual Genome data format is rather redundant; it contains a lot of noise and, in large, is not so easy to work with. Therefore, many researchers use modifications of this collection or convert it to another format. We particularly used the format proposed in [8]. In it, all the data is divided into training and test sets. Lists of object classes and connections between objects are provided separately. For each image, there is its location (`img_path`); classes of objects depicted on it (`classes`); positions of objects that act as subjects (`ix2`) and as objects (`ix2`); bounding rectangulars for each localized object (`boxes`); and also a class of relationships for each subject-object pair (`rel_classes`). This information can be visualized by displaying an image, all its localized objects with their classes, and specifying all the relationships between them.

**Convolutional neural networks.** In this work, we are going to use convolutional neural networks (CNNs) with **transfer learning** (TL). They were developed for finding the optimal solution of image processing challenges, applying the methods of neural networks.

The model we used for this work is VGG16 [3]. This is a CNN, which was designed to solve the problem of classifying the ImageNet collection into 1000 classes. It had only 7.5% error rate. This result was mostly ensured by the use of deep architecture. VGG16 contains 16 hidden layers, which is a lot, even by today's standards. Over time, it has proven the ability to be easily transformed for solving new tasks and has become the de facto standard for TL in solving tasks of computer vision. After VGG16, several other networks were proposed and they showed even better results for the same task, but they are less successful for TL, in part, because the interpretation of their results can sometimes be quite difficult.

VGG16 is included in many machine learning libraries and frameworks, which makes it significantly easier to apply. In PyTorch, there is a special mechanism for loading the data in the process of work with machine learning models, its function is to name the Dataset class [13]. In the course of our work we used this mechanism for data loading as well as for images normalization. This is a required element for working with the model [14].

To test the model, you can send there an image, i.e. from the Visual Genome training set. For this, you need to set the model to evaluation mode. The result of the model's work will be an estimation of the probability that the image in the picture belongs to a certain class of images. Accordingly, it is necessary to receive the information on the position of the largest element and find which class corresponds to the found index.

As the model was used to classify images, we received one value, not several, as it was in the Visual Genome training set. It is also worth mentioning that VGG16 uses a different set of classes (ImageNet classes), that is why, in order to get the classes of objects that were in the training set, it will have to be retrained.

**Adding a RoI Pool layer.** The first modification we made to the VGG16 model was that we added a RoI Pool layer. RoI Pool (a regions of interest layer) is a Pool-layer of a CNN of the fixed size (it is a parameter of the layer) for a certain region of the previous layer of a CNN [5]. It was proposed in the Faster R-CNN model [15].

This layer allows to select certain features from a specific region of the layer and to clearly locate objects. In Faster R-CNN, it was used for objects detection. We, however, need to predict the classes of relationships, so the regions where these relationships are found can be potentially useful. The question is how to determine that the relationship is in a particular region? In order to receive an answer to that question, one can first analyze a simpler problem.

Knowing the positions of the objects (from the training set), we can apply to them the RoI Pool layer.

As we can use a fixed number of objects, for example,  $k$ , for image identification, then applying this layer will result in  $k$  sets of images. So, if earlier the model returned classes of one object at an output, now we can return classes of  $k$  number of objects.

The VGG16 model returns the result as a matrix with a size  $(1, 1000)$ , as there are 1000 classes for which it was trained. For each of these classes, we get understanding of the probability that this class is the class of the object contained in the image. Now, when it returns  $k$  classes, the result will have the dimension  $(k, 1000)$ . However, we need to predict classes from the training set, not ImageNet classes. Therefore, in the model, we need to replace the last fully connected layer with the layer that will return the correct number of classes and retrain the model. It is worth noting that we leave all weights of the model, except for the weights of this last layer, as they were in the trained VGG16 model and are not going to optimize them (carry out the back propagation of the error), as these weights correspond to the layers that are not directly related to classification, and are responsible for defining features (feature extraction).

We have replaced the existing Adaptive Average Pool layer with a RoI Pooling layer, so that the majority of the model elements remain unchanged. In order not to have to change the FC layers that follow after the Pool layer, we also used the same template size as in the Pool layer  $(7 \times 7)$ . All this allowed us to get a new model to solve our problem with little effort.

Since we are making changes to the forward pass on the network, we had to create a new model and redefine the forward method in it. The code for this model is shown in Listing 1.

```

1  class Net(nn.Module):
2      def __init__(self, obj_num):
3          super(Net, self).__init__()
4
5          tl_model = vgg16(pretrained=True)
6          for param in tl_model.parameters():
7              param.requires_grad = False
8
9          self.features = tl_model.features
10         self.roi_pool = RoIPool((7, 7), 1 / 16.0)
11
12         classifier = list(tl_model.classifier.children())[:-1]
13         classifier.extend([nn.Linear(4096, obj_num)])
14         self.classifier = nn.Sequential(*classifier)
15
16     def forward(self, x, boxes):
17         x = self.features(x)
18         x = self.roi_pool(x, boxes)
19         x = x.view(x.size(0), -1)
20         x = self.classifier(x)
21         return x

```

Listing 1. Network using VGG16 with the help of TL

For the model to work, first one needs to create all the layers so that it would be possible to use them during the forward pass. Therefore, in the constructor, we first get the VGG16 model with trained weights (line 5) and turn off the gradient propagation for all its parameters (lines 6-7). We leave the set of features layers unchanged (line 9), and add a new RoI Pool layer with the required template size (line 10). We only change the last layer in the classifier set (lines 12-14). The changed layer will have the number of output layers that equal to the number of object classes of the training set (obj\_num), which will be defined in the constructor. It is worth mentioning that the last layer of the classifier set is the only one that has the value for the requires\_grad parameters equal to true (this is the default value), so this will be the only layer for which weights will be optimized.

```

1 class IsmDataset(Dataset):
2
3     def __init__(self, data):
4         self.data = data
5         self.img_transform = Compose([
6             ToTensor(),
7             Normalize([0.485, 0.456, 0.406],
8                 [0.229, 0.224, 0.225])
9         ])
10
11    def __len__(self):
12        return len(self.data)
13
14    def __getitem__(self, index):
15        data = self.data[index]
16
17        if (data == None):
18            return ()
19
20        file_path = data['img_path']
21
22        if (not os.path.exists(file_path)):
23            return ()
24
25        with open(file_path, 'rb') as f:
26            img = Image.open(f)
27            img_tensor = self.img_transform(img)
28
29        objects = data['classes'].astype(np.int64)
30
31        raw_boxes = data['boxes']
32        boxes = np.zeros((raw_boxes.shape[0], 5), dtype=np.float32)
33        boxes[:, 1:5] = raw_boxes
34
35        return img_tensor, boxes, objects

```

Listing 2. Updated class for data loading after changes to VGG16.

In addition, it is necessary to modify the class to load the training set, since one still needs to rotate the positions of the objects (bounding rectangulars). The modified form is shown in Listing 2.

It was necessary to add 0 to each list of bounding boxes in lines 31-33. This is conditioned by the specifics of RoI Pool layer work with object positions.

The training set was divided into 2 sets: a training one and a test one. For this purpose the class 8, defined in the listing, and also the DataLoader class were used. The class code is shown in Listing 3.

```

1 data_sets = { x: IsmDataset(read_dataset(x)) for x in ['train', 'test'] }
2 data_loaders = { x: DataLoader(data_sets[x]) for x in ['train', 'test'] }

```

Listing 3. Dividing the training set into a training one and a test one

With all the necessary elements, it is possible to start the learning process. To do this, we defined the function that can simultaneously train and test, depending on what part of the training set it receives. This approach has been adapted on the basis of official recommendations [16].

The function calculated the values received from the forward pass on the network as well as losses, made a return loss offer and optimized the parameters. The rest of the code was responsible for checking the training phase, saving execution time, accumulating losses and saving the model that showed the best result.

The function has several parameters that have not yet been mentioned. The first one is the loss function criterion. Since we needed to predict multiple classes, we used MultiLabelMarginLoss, an SVM-based loss function, but for multiple classes. Optimizer and scheduler are the objects that will optimize the values of the network weights after the back propagation of the error. The optimizer added to the weights a gradient value with a certain step, and the scheduler changed the learning rate to increase the efficiency of learning. The code to run the training process is shown in Listing 4.

```

1 net = Net(len(obj_classes)).to(device)
2 criterion = nn.MultiLabelMarginLoss()
3 last_layer_params = list(net.classifier.children())[-1].parameters()
4 optimizer = optim.Adam(last_layer_params, lr=0.001, momentum=0.9)
5 scheduler = lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)

```

Listing 4. Running the model's training

**Adding the remaining layers.** Returning to the question of receiving classes of relationships, it is necessary to make a resume of the information we have at the input, of what we have during the operation of the network now and what should be in the output.

As far as we receive objects at the input of the model, and we want to get classes of relationships between them at the output, the original data format can be represented as a relationship between each pair of objects. In order to be able to conduct optimization, we need to encode relationship classes in a one-hot vector. It means that if we have  $k$  objects and  $n$  possible classes of relationships, the result will be the matrix of dimension  $k \times (k-1) \times n$ . The advantage of this method is that in this way we come to the task of multiclass classification, and, consequently, it is possible to use for its solving the experience of the previous section.

However, there are some difficulties. We used the RoI Pool layer to classify objects in order to obtain visual features for each object, but now we need to get visual features for each possible relationship between objects. To solve this problem, we are going to use the combined features of a subject and an object. To get them, it is necessary to find where the bounding boxes of a subject and an object overlap. This is a simple operation, it must be performed for each pair of objects (except an object itself, as an object in the training sample cannot have relationships with itself) and in this way we get bounding boxes for each relationship we need to predict.

According to the VGG16 architecture, there should be 3 FC layers after the Pool layer, in order the model could be trained to classify relationships. After that, it is possible to add the last FC-layer, whose output dimension will be equal to the number of classes of relationships, and we will again receive a multiclass classifier.

The performance of the proposed model can be improved if we use spatial features in addition to visual ones. We borrowed this idea from the previously mentioned work [8] and adapted it to our solution. Adding spatial information facilitates work of the network, because a large part of the relationships are spatial ("under", "on", "near", etc.) and clearly distinguishing them, we simplify their correct definition. It is difficult to identify these types of relationships with only the visual features of the combined region of a subject and an object. Therefore, using a new type of information, we improve the performance of the model.

There are several ways to represent spatial relationships between objects, and for us the most suitable one was their representation in the form of masks. Image masks for an object and a subject are created, so that only the pixels that are present in the bounding box of each of them have a value of 1, the rest equal 0. The two masks received as a result of this appear to be a representation of spatial relationships. The advantage of using this method is a simple possibility to present these masks as an input layer for a CNN and use the strengths of a CNN to train the model to determine the spatial relationships between them. Since the input layer will contain both masks, the layer depth will be 2 and all convolution operations will take place on both layers. The fact that both of them have a common area will be taken into account.

To design the convolutional layers that will be used for spatial features, we also took the VGG16 convolutional layers as a basis. There were 3 of them, and after them there was one FC-layer.

Now that we have two kinds of features, we need to combine them somehow. The standard solution in this case is to add another fully connected layer that will receive both sets of features and combine them. This means that before the last FC layer, which was mentioned earlier, there will be another FC-layer, which will deal with combining the results of the layers for visual and spatial features.

For training the final model the same code that was used to train the intermediate model was applied.

## Results

To assess the quality of work of the developed model, we used the "recall at  $K$ " metric. As it was already mentioned, it was proposed in [10] and became the standard for solving this class of tasks. It determines the proportion of correctly predicted values among the best  $K$  predictions. Basically, for  $K$ , values 50 and 100 are used. Thus, we get two metrics: recall at 50 and recall at 100.

The developed model receives objects and their positions at the input and returns relationships between them at the output. In order to assess the quality of its work according to the mentioned metrics, we need to use the test data set from the Visual Genome collection. However, the task to receive both objects and relationships between them only from an input image is much more interesting. Moreover, it is one of the conditions we set before developing the model.

To solve this problem, we used the existing model for finding objects and their positions, and used the results we

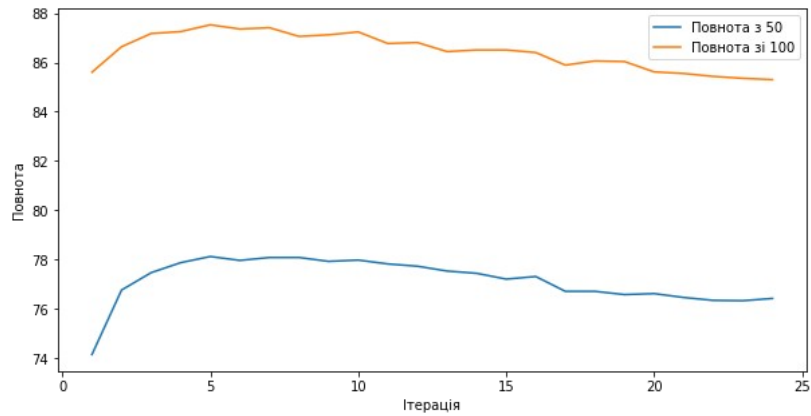


Figure 1. Dependence of recall on iterations of the Predicate Detection task

received as the input data for our model. Thus, the work of the model consisted of two stages. A similar approach was used in a number of other works [8, 9].

We used the Faster R-CNN, mentioned earlier, as a model for finding objects. At the present time, its gets state-of-the-art results in solving such tasks[15].

So, instead of one task, here were two. The first one is to predict the relationships between predefined objects. The second one is to predict subject-object-relationship sets. These tasks are called Predicate Detection and Relationship Prediction, respectively. Some works have also suggested solutions to these problems, and we are comparing our results with them. Figure 1 shows graphs of the dependence of the selected metrics on the number of iterations for each of the tasks, respectively. It can be seen that in both cases, the highest value was received at the 5th iteration and after that the model started to retrain itself. Table 1 compares the best results our model had with the test set and the results of researches conducted by other authors.

In order to better understand what kind of results the developed model returns and analyze their features, one can consider some results generated by the developed model and shown in Table 2. For each image, we offer both the result generated with known objects (predicate detection) and without them (relationship prediction). For the second case, it was necessary to send the input image to the Faster R-CNN detector, in the first place, and then send the results to the model.


For each image, we selected 20 predicted triplets with the highest value of the evaluation function.

**Table 1.** Comparing the results of the developed model

Model	Predicate Detection		Relationship Prediction	
	Recall at 50	Recall at 100	Recall at 50	Recall at 100
[16] 2015	0.97	1.91	-	-
[4] 2016	47.87	47.87	13.86	14.70
[13] 2018	80.78	81.90	17.73	20.88
Our model	78.12	87.53	18.56	22.05

Table 2. Some examples of the model's work

Image	Predicate Detection	Relationship Prediction
	<p>sunglasses - on - person  watch - on - person  jeans - on - person  building - behind - dog  person - wear - jeans  person - wear - sunglasses  person - wear - watch  building - behind - person  glasses - on - person  dog - in the front of - building  person - wear - glasses  person - in the front of - building  bag1 - behind - glasses  bag - behind - glasses  glasses - next to - bag  glasses - next to - bag1  person - hold - bag  watch - near - glasses  person - hold - bag1  bag1 - next to - dog</p>	<p>shirt - on - person  shirt - on - person2  shirt - on - person1  jeans - on - person2  glasses - on - person2  jeans - on - person  glasses - on - person  jeans - on - person1  glasses - on - person1  jacket - on - person2  jacket - on - person1  person1 - wear - jacket  person2 - wear - jacket  jacket - on - person  person - wear - jacket  person1 - wear - jeans  person1 - wear - shirt  person - wear - jeans  person2 - wear - shirt  person2 - wear - jeans</p>
	<p>shirt - on - person  jeans - on - person  shorts - on - person  person - wear - jeans  person - wear - shorts  person - wear - shirt  shirt - above - shorts  cabinet - above - bag  shirt - above - jeans  cabinet - behind - person  shorts - below - shirt  shirt - behind - bag  bag - above - shorts  person - hold - bag  jeans - below - shirt  bag - above - jeans  shorts - next to - bag  person - in the front of - cabinet  bag - below - cabinet  bag - near - shirt</p>	<p>shirt - on - person1  shirt1 - on - person  shirt - on - person  shirt1 - on - person1  pants - on - person1  pants - on - person  person - wear - shirt  person1 - wear - shirt  person - wear - shirt1  person1 - wear - shirt1  person1 - wear - pants  person - wear - pants  shirt - above - pants  shirt1 - above - pants  pants - below - shirt  pants - below - shirt1  person - next to - person1  person1 - next to - person  shirt - behind - shirt1  shirt1 - behind - shirt</p>
	<p>jeans - on - chair2  jeans - on - chair1  jeans - on - chair  chair1 - next to - chair  chair1 - next to - chair2  chair2 - next to - chair  chair - next to - chair2  chair - next to - chair1  chair2 - next to - chair1  bed - near - chair  bed - near - chair2  bed - near - chair1  chair1 - near - bed  chair - near - bed  chair2 - near - bed  bed - has - jeans</p>	<p>plant - on - table1  plant - on - table  lamp - on - table1  table - on - street  table1 - on - street  chair - on - street  table1 - has - plant  lamp - on - table  chair - next to - table1  chair - next to - table  table - has - plant  lamp - behind - plant  lamp - next to - chair  street - below - lamp  street - under - table  street - under - table1</p>

	jeans - on - bed chair2 - has - jeans chair - has - jeans chair1 - has - jeans	plant - next to - lamp table1 - has - lamp street - under - chair table - has - lamp
	shoes - on - person person - wear - jeans person - wear - hat person - wear - shoes jeans - on - person hat - above - shoes hat - on - person jeans - above - shoes roof - above - person hat - over - jeans jeans - near - hat shoes - beneath - jeans person - on - elephant shoes - on the right of - hat roof - above - shoes person - under - roof elephant - next to - person roof - above - jeans shoes - on - elephant roof - above - elephant	shirt1 - on - person1 shirt1 - on - person3 shirt - on - person2 shirt - on - person3 shirt - on - person1 shirt1 - on - person2 shirt - on - person trees - behind - elephant trees - behind - person trees - behind - person2 trees - behind - person3 person2 - wear - shirt1 person - wear - shirt building - behind - person2 shirt1 - on - person building - behind - person person3 - wear - shirt1 person2 - wear - shirt person - wear - shirt1 building - behind - person

As we can see, the predictions received for the first image during the first task are better. They describe the image in more detail and contain various relationships, such as "hold", "wear", "in front of", "next to", while the results for the second task have only "on" and "wear". It becomes obvious that the use of the detector, instead of the data from the training set directly, negatively affects the results.

Along with that, sometimes such simple predictions can indicate good quality of an image, like in the case with images 2 and 3. Here, both tasks were solved properly, although, in general, they also use only spatial relationships and a predicate "wear". Kinds of relationships that one can receive depend on the type of an image. This is also a problem of the training set itself. The "on" and "wear" relationships were the most popular in the training set, so the model tends to anticipate them. Another factor is that we clearly used spatial features in the model, so the fact that most predictions are related to the positions of the objects is not surprising.

Regarding the third image, it should be added that the developed model is able to distinguish in this image several objects of the same class. In particular, we can see that it successfully found relationships for three chairs.

In the last image, it is important to pay attention at the difference between the predictions for different tasks. In the case of predicate detection, the focus is on the person sitting on the elephant and everything around it; while the results for relationships prediction do not even contain an elephant, but describe several people in the background. This once again proves that for the second task, the quality definitely depends on the quality of the detector's work.

## Conclusions

This work introduces the concept of a semantic image model and describes the implementation of the machine learning model for solving tasks of automatic development of such a model for an input image. A semantic model consists of a list of objects represented in an image and their relationships.

In addition, this work considered other possible and existing approaches to representing a semantic model. The analysis of these approaches has shown that the relationship-based approach is the most detailed and the most suitable one for solving various kinds of tasks related to understanding of images.

The developed model was compared to other solutions for the same problem and showed the best results in all but one case. The efficiency of the model's operation is based on the use of the latest achievements of machine learning, in particular, CNN, TL, Faster R-CNN and VGG16 models.

On the basis of the results of the model, we can come to the conclusion that the model's performance in solving such type of tasks greatly depends on the quality of the training set. The more varied and complete it is, the better the results will be. Moreover, a very significant part of the relationships shown in the image are spatial relationships, so in order to ensure better results, one needs to take this fact into account in the process of development, which was done in this work.

The use of a multi-stage architecture leads to the fact that the quality of performance on the last stages depends a



lot on the quality of the previous ones. This is what happened when the model used the Faster R-CNN detector. Hence, we can come to the conclusion that in order to improve the results, one can work on improving the quality of solving the task of objects detection.

## List of References

1. Karpathy A., Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions [Electronic resource]. – available at: <https://cs.stanford.edu/people/karpathy/deepimagesent/>
2. A visual proof that neural nets can compute any function [Electronic resource]. – available at: <http://neuralnetworksanddeeplearning.com/chap4.html>
3. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition [Electronic resource]. – available at: <https://arxiv.org/pdf/1409.1556.pdf>
4. Image Captioning [Electronic resource]. – Mode of access: <http://shikib.com/captioning.html>
5. Dai J. R-FCN: Object Detection via Region-based Fully Convolutional Networks [Electronic resource]. – available at: <https://arxiv.org/pdf/1605.06409.pdf>
6. VGG16 – Convolutional Network for Classification and Detection [Electronic resource]. – available at: <https://neurohive.io/en/popular-networks/vgg16/>
7. Vinyals O. Show and Tell: A Neural Image Caption Generator [Electronic resource]. – available at: <https://arxiv.org/pdf/1411.4555.pdf>
8. Dai B. Detecting Visual Relationships with Deep Relational Networks [Electronic resource]. – available at: <https://arxiv.org/pdf/1704.03114.pdf>
9. Sadeghi M. Recognition Using Visual Phrases [Electronic resource]. – available at: [http://vision.cs.uiuc.edu/phrasal/recognition\\_using\\_visual\\_phrases.pdf](http://vision.cs.uiuc.edu/phrasal/recognition_using_visual_phrases.pdf)
10. Lu C. Visual Relationship Detection with Language Priors [Electronic resource]. – available at: <https://arxiv.org/pdf/1608.00187.pdf>
11. Krishna R. Visual Genome Connecting Language and Vision Using Crowdsourced Dense Image Annotations [Electronic resource]. – available at: <https://arxiv.org/pdf/1602.07332.pdf>
12. Visual Genome [Electronic resource]. – available at: <https://visualgenome.org>
13. Data Loading and Processing Tutorial [Electronic resource]. – available at: [https://pytorch.org/tutorials/beginner/data\\_loading\\_tutorial.html](https://pytorch.org/tutorials/beginner/data_loading_tutorial.html)
14. TorchVision Models [Electronic resource]. – available at: <https://pytorch.org/docs/stable/torchvision/models.html>
15. Ren S. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [Electronic resource]. – available at: <https://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf>
16. Chilamkurthy S. Transfer Learning Tutorial [Electronic resource]. – available at: [https://pytorch.org/tutorials/beginner/transfer\\_learning\\_tutorial.html](https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html)

### About the authors:

**Andon Philip I.**, Academician of the National Academy of Science (NAS) of Ukraine, Director of the Institute of Software Systems of the NAS of Ukraine, 03187, Kyiv, 40 Akademika Glushkova Avenue. The number of scientific publications in Ukrainian editions - 400. The number of scientific publications in foreign indexed editions - 10. <http://orcid.org/0000-0001-6546-0826>.

**Hlybovets Andriy M.**, Doctor of Technical Sciences, Dean of the Faculty of Computer Science, National University of Kyiv-Mohyla Academy, 04070, Kyiv, 2 Skovorody street. Number of scientific publications in Ukrainian editions - 40. Number of scientific publications in foreign indexed editions - 5. <https://orcid.org/0000-0003-4282-481X>

**Kurylyak Volodymyr V.**, Master of the Faculty of Computer Science of National University of the Kyiv-Mohyla Academy, 04070, Kyiv, 2 Skovorody street. Number of scientific publications in Ukrainian editions - 0. Number of scientific publications in foreign indexed editions - 0

Contact person: Hlybovets Andriy M., +380674094355, [a.glybovets@ukma.edu.ua](mailto:a.glybovets@ukma.edu.ua)