тоже могут организовываться ассоциативные связи.

Перечисленное даёт фундамент для построения математической модели иерархической ассоциативной памяти для искусственных когнитивных агентов, причём в случае успешной реализации на достаточном объёме данных и вычислительных ресурсов такая память может стать основой для создания искусственного интеллекта общего уровня. Применение этой модели памяти в рамках гибридной парадигмы искусственного интеллекта [Душкин & Андронов, 2019] открывает разнообразные возможности для реализации искусственных когнитивных агентов различного предназначения.

Модель иерархической ассоциативной памяти для искусственных когнитивных агентов общего назначения основана на множестве специфических структурных элементов, каждый из которых специальным образом кодирует одно понятие общего характера или специфическое для проблемной области, в которой функционирует когнитивный агент. Кодирование понятия осуществляется при помощи наиболее общего способа в виде пары типа (наименование, значение). Такой способ кодирования позволяет осуществлять кодирование любых сущностей проблемных областей любого рода — объектов, атрибутов и предикатов.

Вместе с тем к каждому такому структурному элементу памяти в предлагаемой к рассмотрению модели приписывается некоторое количество специальной и служебной информации, а именно:

1.Множество ассоциативных связей, каждая из которых имеет наименование и список вспомогательных атрибутов. Фактически при помощи этого множества формируется семантическая сеть на хранящихся в памяти понятиях.

2.Структура вспомогательной информации о том, когда и при каких условиях было получено понятие, которое встроено в описанную в предыдущем пункте семантическую сеть.

3.Список архивных значений понятия, который может использоваться для самореференсной памяти о том, как происходило обучение и актуализация понятийной базы когнитивного агента.

Описанная структура представляет собой расширенную семантическую сеть, которая, однако, должна быть динамической и постоянно актуализироваться в процессе функционирования когнитивного агента, особенно при использовании методов машинного обучения и, в частности, обучения с подкреплением при активном взаимодействии со средой. В этом заключается отличие предлагаемой модели от обычной семантической сети.

Литература

1.Brézillon P. (1999) Context in Artificial Intelligence // Computing and Informatics / Computers and Artificial Intelligence — CAI. — TRANSLIBRIS, May 1999. — P. 321-340.

2.Hawkins J. (2004) On Intelligence (1st ed.). Times Books. — P. 272. — ISBN 978-0805074567.

3.Paquette P. (2020) A Road Map to Strong Intelligence // Preprint, February 2020. — URL: https://bit.ly/_aRMtSI.

4.Takagi M., Sakurai A., Hagiwara M. (2019) Quality Recovery for Image Recognition // IEEE Access. 7. 1-1. — DOI: 10.1109/ACCESS.2019.2932726.

5.Душкин Р. В. (2019) Искусственный интеллект. — М.: ДМК-Пресс, 2019. — 280 с. — ISBN 978-5-97060-787-9.

6.Душкин Р. В., Андронов М. Г. (2019) Гибридная схема построения искусственных интеллектуальных систем // Кибернетика и программирование. — 2019. — № 4. — С. 51-58. — DOI: 10.25136/2644-5522.2019.4.29809. — URL: http://e-notabene.ru/kp/article_29809.html.

7.Шумский С. А. (2020) Машинный интеллект. Очерки по теории машинного обучения и искусственного интеллекта. — М.: РИОР, 2020. — 340 с. — ISBN: 978-5-369-01832-3.

ACCURACY IMPROVING OF PRE-TRAINED NEURAL NETWORKS BY FINE TUNING

DOI: 10.31618/ESU.2413-9335.2021.5.82.1231

Konarev D.I.

Department of Space Instrument-making and Communication Systems,
Southwestern State University, 94, 50 let Oktyabrya, Kursk, Russia

Gulamov A.A.

Department of Space Instrument-making and Communication Systems,
Southwestern State University, 94, 50 let Oktyabrya, Kursk, Russia

ABSTRACT

Methods of accuracy improving of pre-trained networks are discussed. Images of ships are input data for the networks. Networks are built and trained using Keras and TensorFlow machine learning libraries. Fine tuning of previously trained convoluted artificial neural networks for pattern recognition tasks is described. Fine tuning of VGG16 and VGG19 networks are done by using Keras Applications. The accuracy of VGG16 network with fine-tuning of the last convolution unit increased from 94.38% to 95.21%. An increase is only 0.83%. The accuracy of VGG19 network with fine-tuning of the last convolution unit increased from 92.97% to 96.39%, which is 3.42%.

1. Introduction

One of the main tasks is to recognize vessels in an image or video. It is based on the purpose of methods and system development of software and hardware, collection, processing, storage, analysis of shipping channels parameters, navigation of cargo and passenger traffic in information and telecommunication system of monitoring and management of shipping channels [1; 2; 3] The effectiveness of using neural networks to solve this problem has been shown in [4].

2. Task setting

Technology of training transfer was described. Fine tuning is the next step in accuracy improving of a pre-trained network. If only a new part, the classifier, changes at the stage of additional training, then at the stage of fine tuning the convolution part is also trained. This results in a more accurate selection of characteristics of those objects that are part of a data set of a new task [5]. Fine tuning can only be applied after the classifier on a new dataset has been trained.

3. Implementation proposed in the article

Pre-trained networks from Keras Applications library were used. Freezing process of network convoluted part and additional training of created classifier on a new data set was demonstrated using an example of VGG16 network. There are about 2 million parameters in the composite network. They are located in fully connected layers of the classifier. An accuracy of 94.38% was obtained.

It is necessary to unfreeze several layers or entire blocks of a neural network at a fine tuning stage. Selected number of layers or blocks to be thawed and trained depends on how much a new data set differs from a data set on which the neural network was previously trained [6]. The ImageNet dataset, which was used to train networks from Keras Applications library, includes images of naval vessels. So, a new dataset is slightly different from ImageNet dataset and a small number of layers can be trained. The last block of pre-trained networks will be thawed for training. Using an example of VGG16 network, the last unit of pre-trained part is highlighted in Figure 1.

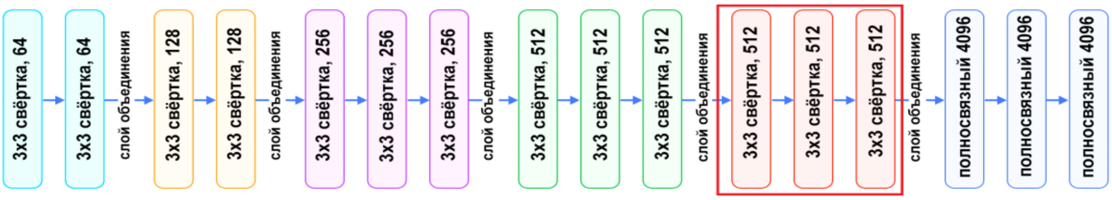

Fig. 1. VGG16 network architecture

Figure 2 shows information about the network, with the list of layers and the number of neurons in them. The name of the layers consists of two parts. The first part is the name of a block and the second is a type and number of a layer in a block. Layers with "block5" prefix in the name must be unlocked.

```
vgg16_net.summary()

Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
```

Fig. 2. Information about the loaded VGG16 network

At first, it is necessary to allow training at the network level. Trainable field is set as «True» for this purpose. Then trainable flag is created, which is initially set as «False» and means that a particular layer is being trained or not. There is the first layer of the fifth block with the name "block5_conv" in network. Trainable parameter is set as «True» for this and others subsequent layers (Figure 3).

```
vgg16_net.trainable = True
trainable = False
for layer in vgg16_net.layers:
    if layer.name == 'block5_conv1':
        trainable = True
    layer.trainable = trainable
```

Fig. 3. Unlock of the last block of VGG16 network

The network must then be recompiled. It is necessary to use a small training rate parameter in order not to lose the results of pre-training because the network is already pre-trained [7]. The learning rate parameter lr (learning rate) is specified when the optimizer is created. In this case, Adam optimizer is used (Figure 4)

```
model.compile(loss='binary_crossentropy',
              optimizer=Adam(lr=1e-5),
              metrics=['accuracy'])
```

Fig. 4. Network recompiling with a slow learning rate

The compiled network is trained in the same way as the composite network classifier was trained. It is not necessary to use a large number of eras for fine-tuning of the network. In this case, two eras are used. When the training is completed, the quality of the network is checked using data that was not used during the training.

Network accuracy on test data is 95.21% and increased compared to the network without fine tuning by 0.83%. A small increase is due to the fact that new data set is not very different from ImageNet data set and convoluted part of VGG16 neural network already identifies characteristic features of sea vessels. If dataset contained objects that were not part of the original dataset for training, the efficiency of network fine-tuning on such dataset would be significantly higher. Fine-tuning of VGG19 network was carried out similarly and the accuracy of operation was 96.39%, which is more than that of VGG16. Although the accuracy of VGG19 to fine tuning was less than that of VGG16. It was 92.97%. So it is increased by 3.42%.

4. Results and conclusions

The use of pre-trained neural networks to classify objects on an image and a method for improving their accuracy were described. At the first stage, when transferring training from a pre-trained neural network, the part responsible for objects classifying is removed and a new part is added. This part provides classification of objects in a given task. A fine-tuning stage follows after training a new classifier. At this stage both the classifier and the convolution part are trained. This allows the neural network to determine the characteristic features of objects on a new dataset better. The effectiveness of fine tuning depends on how different the datasets are. And it is more effective if the new dataset contains objects that were not in pre-training dataset. So fine-tuning of VGG16 network showed an increase in accuracy of 0.83% and fine-tuning of VGG19 network showed an increase in accuracy of 3.42%.

References

[1] Goltsova I. A., Gulamov A. A. Information support of the railway section, Proceeding of the Southwest State University. Series: Control, Computer engineering, Information science. Medical instruments engineering, 2017, vol. **7**, no. 2(23), pp. 6–11. https://swsu.ru/izvestiya/seriesivt/archiv/2_2017.pdf

[2] Maklakov E.S., Gulamov A.A. The collection of information control center. Proceedings of the Southwest State University. 2018;**22**(6):136-142. (In Russ.) https://doi.org/10.21869/2223-1560-2018-22-6-136-142

[3] Maklakov E. S., Gulamov A.A. Optimizaciya «poslednih mil'» do udalennyh uzlov dostupa putem primeneniya tekhnologii LCAS. Modelirovanie, optimizaciya i informacionnye tekhnologii. Nauchnyj zhurnal, 2019.**7**(3). (In Russ.). http://moit.vivt.ru/. https://doi.org/ 10.26102/2310-6018/2019.26.3.039.

[4] Konarev D.I., Gulamov A.A. Synthesis of Neural Network Architecture for Recognition of Sea-Going Ship Images. Proceedings of the Southwest State University. 2020;**24**(1):130-143. (In Russ.) https://doi.org/10.21869/2223-1560-2020-24-1-130-143

[5] Bastiaan Sjardin, Luca Massaron, Alberto Boschetti Large Scale Machine Learning with Python, Packt Publishing, 2016. – 420 p.

[6] Weiss K., Khoshgoftaar T.M., Wang D. A Survey of Transfer Learning Journal of Big Data. 2016. Vol. **3**, No. 1. pp. 1-9.

DOI: 10.1186/s40537-016-0043-6

[7] Russakovsky, O., Deng, J., Su, H., et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV). Vol **115**, Issue 3, 2015, pp. 211–252