Д. И. КОНАРЕВ, А. А. ГУЛАМОВ
Юго-Западный государственный университет, Курск

ПОВЫШЕНИЕ ТОЧНОСТИ ПРЕДВАРИТЕЛЬНО ОБУЧЕННЫХ НЕЙРОННЫХ СЕТЕЙ ПУТЁМ ТОНКОЙ НАСТРОЙКИ

В работе рассмотрены методы повышения точности работы предварительно обученных сетей. Сети построены и дообучены с использованием библиотек машинного обучения Keras и TensorFlow. Описана тонкая настройка (Fine Tuning) предварительно обученных свёрточных искусственных нейронных сетей для задач распознавания образов. Проведена тонкая настройка сетей VGG16 и VGG19 из модуля Keras Applications. Точность работы сети VGG16 при тонкой настройке последнего свёрточного блока возросла с 94.38 % до 95,21 %, что составляет прирост всего на 0,83 %. Точность работы сети VGG19 при тонкой настройке последнего свёрточного блока возросла с 92,97 % до 96,39 %, что составляет 3,42 %.

Введение. Исходя из цели разработки методов и системы программно-технических средств сбора, обработки, хранения, анализа параметров судоходных каналов, обеспечения навигации грузо- и пассажироперевозок в рамках информационно- телекоммуникационной системы мониторинга и управления судоходными каналами [1, 2, 3] одной из основных задач является распознавание судов на изображении или видео. Эффективность применения нейронных сетей для решения этой задачи была показана в [4].

Постановка задачи. В работе [5] была рассмотрена технология переноса обучения. Тонкая настройка (Fine Tuning) – следующий шаг, с помощью которого можно повысить точность работы предварительно обученной сети. Если на этапе дообучения изменяется только новая часть, классификатор, то на этапе тонкой настройки дообучается и свёрточная часть тоже. За счет этого происходит более точное выделение характерных признаки тех объектов, которые входят в набор данных новой задачи [6]. Тонкую настройку можно применять только после того, как обучен классификатор на новом наборе данных.

Реализация, предложенная в статье. В работе [5] использовались предварительно обученные сети из библиотеки Keras Applications. На примере сети VGG16 был продемонстрирован процесс заморозки свёрточной части сети и дообучение созданного классификатора на новом наборе данных. В составной сети около 2 миллионов параметров, находящихся в полносвязанных слоях классификатора. Была получена точность 94.38 %.

На этапе тонкой настройки необходимо разморозить несколько слоев или целых блоков нейронной сети. Выбранное количество слоев или блоков, которые необходимо разморозить и дообучить, зависит от того, насколько сильно новый набор данных отличается от набора данных на котором производилось предварительное обучение нейронной сети [7]. Набор данных ImageNet, на котором обучались сети из библиотеки Keras Applications, включает изображения морских судов. Поэтому новый набор данных слабо отличается от набора данных ImageNet и можно обучать небольшое количество слоев. Для обучения будет разморожен последний блок предварительно обученных сетей. На примере сети VGG16 последний блок предварительно обученной части выделен на рис. 1.

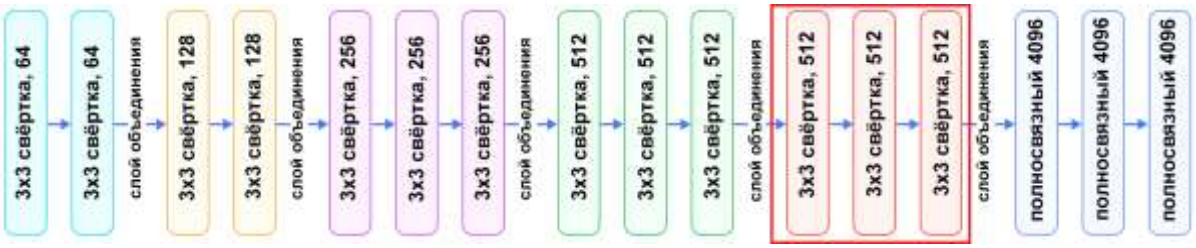

Рис. 1. Архитектура сети VGG16

На рис. 2 представлена информация о сети, где указан перечень слоев и количество весов нейронов в них. Название слоев состоят из двух частей, первая часть – это название блока,

вторая – тип и номер слоя в блоке. Должны быть разморожены слои с префиксом «block5» в имени.

```
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
```

Рис. 2. Информация о загруженной сети VGG16

Во-первых, необходимо разрешить обучение на уровне сети. Для этого поле trainable устанавливается в значение True. Затем создаётся флаг trainable, который изначально установлен в значение False и означает, обучается тот или иной слой или нет. Перебором слоёв сети находится первый слой пятого блока с именем «block5_conv». Для этого и всех последующих слоёв параметр trainable устанавливается в значение True (рис. 3).

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

Рис. 3. Разморозка последнего блока сети VGG16

После этого сеть необходимо перекомпилировать. Так как сеть уже предварительно обучена, необходимо использовать малый параметр скорости обучения, чтобы не потерять результаты предварительного обучения [8]. Параметр скорости обучения lr (learning rate) указывается при создании оптимизатора. В данном случае используется оптимизатор Adam (рис. 4)

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

Рис. 4. Перекомпиляция сети с малой скоростью обучения

Скомпилированная сеть обучается так же, как обучался классификатор составной сети. Для тонкой настройки сети, как правило, не требуется большое количество эпох. В данном случае используется две эпохи. После завершения обучения качество работы сети проверяется на тех данных, которые не использовались в процессе обучения.

Точность работы сети на тестовых данных составляет 95,21 % и выросла по сравнению с сетью без тонкой настройки на 0,83 %. Небольшой прирост объясняется тем, что новый набор данных не сильно отличается от набора данных ImageNet и свёрточная часть нейронной сети VGG16 уже хорошо выделяет характерные признаки морских судов. Если бы набор данных содержал объекты, которые не входят в исходный набор данных для обучения, то эффективность тонкой настройки сети на таком наборе данных была бы значительно выше. Аналогично проведена тонкая настройка Сети VGG19 и точность работы составила 96,39 %, что больше, чем у VGG16. Хотя точность VGG19 до тонкой настройки была меньше, чем у VGG16 и составляла 92,97 %, то есть выросла на 3,42 %.

Обсуждение результатов и заключение. Было рассмотрено использование предварительно обученных нейронных сетей для классификации объектов на изображении и метод повышения точности их работы. На первом этапе, при переносе обучения из предварительно обученной нейронной сети удаляется часть, отвечающая за классификацию объектов, и добавляется новая часть, которая обеспечивает классификацию объектов в поставленной задаче. После обучения нового классификатора следует этап тонкой настройки, при котором обучается как классификатор, так и свёрточная часть. Это позволяет нейронной сети лучше определять характерные особенности объектов на новом наборе данных. Эффективность тонкой настройки зависит от того, насколько сильно отличаются наборы данных и наиболее эффективна в случае, если новый набор данных содержит объекты, которых не было в наборе данных для предварительного обучения. Так тонкая настройка сети VGG16 показала прирост точности 0,83 %, а тонкая настройка сети VGG19 – прирост точности 3,42 %.

ЛИТЕРАТУРА

1. **Гольцова И.А., Гуламов А.А**. Информационное обеспечение участка железной дороги // Известия Юго-Западного государственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2017. Т. 7, № 2(23). С. 6–11. https://swsu.ru/izvestiya/seriesivt/archiv/2_2017.pdf
2. **Маклаков Е.С., Гуламов А.А**. Узел сбора информации диспетчерского центра // Известия Юго-Западного государственного университета. 2018. Т. 22, № 6(81). С. 136-142. https://doi.org/ 10.21869/2223-1560-2018-22-6-136-142.
3. **Маклаков Е.С., Гуламов А.А**. Оптимизация «последних миль» до удаленных узлов доступа путем применения технологии LCAS // Моделирование, оптимизация и информационные технологии. Научный журнал, Том 7, № 3, 2019. http://moit.vivt.ru/. https://doi.org/ 10.26102/2310-6018/2019.26.3.039.
4. **Конарев Д.И**. Синтез архитектуры нейронной сети для распознавания образов морских судов / Д.И. Конарев, А.А. Гуламов // Известия Юго-Западного государственного университета. 2020. Т. 24, № 1.
5. **Конарев Д.И., Гуламов А.А., Маклаков Е.С**. Синтез архитектуры нейронной сети для распознавания образов морских судов на базе предварительно обученной нейронной сети // Известия Юго-Западного государственного университета. 2020. В печати.
6. **Bastiaan Sjardin, Luca Massaron, Alberto Boschetti** Large Scale Machine Learning with Python, Packt Publishing, 2016. 420 p.
7. Weiss K., Khoshgoftaar T.M., Wang D. A Survey of Transfer Learning // Journal of Big Data. 2016. Vol. 3, No. 1. P. 1-9. DOI: 10.1186/s40537-016-0043-6
8. 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

D.I. Konarev, A.A. Gulamov (Southwest State University, Kursk)

Increasing the Accuracy of Pre-trained Neural Networks by Fine Adjustments

The paper discusses methods for improving the accuracy of pretrained networks. The networks were built and retrained using the Keras and TensorFlow machine learning libraries. The fine tuning of pretrained convolutional artificial neural networks for pattern recognition tasks is described. Fine-tuned VGG16 and VGG19 networks from the Keras Applications module. The accuracy of the VGG16 network with fine tuning of the last convolutional block increased from 94.38 % to 95.21 %, which is an increase of only 0.83 %. The accuracy of the VGG19 network when fine-tuning the last convolutional block has increased from 92.97 % to 96.39 %, which is 3.42 %.