

Липецкий государственный технический университет

Кафедра прикладной математики

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ МАТЕМАТИЧЕСКИХ ИССЛЕДОВАНИЙ

Лекция 2.6

Deep Learning в R

Составитель - Сысоев А.С., к.т.н., доцент

Липецк - 2021

Outline

6.1. Сверточные нейронные сети

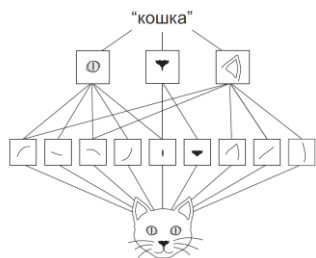
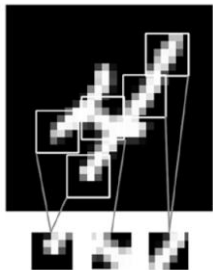
6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных

6.3. Использование предварительно обученной сверточной нейронной сети

6.1. Сверточные нейронные сети

ОПЕРАЦИЯ СВЕРТЫВАНИЯ

Основное отличие полносвязного слоя от сверточного заключается в следующем: полносвязные (dense) слои изучают глобальные шаблоны в пространстве входных признаков (в случае с цифрами – это шаблоны, вовлекающие все пикселы), тогда как сверточные слои изучают локальные шаблоны (в случае с изображениями — шаблоны в небольших двумерных окнах во входных данных).



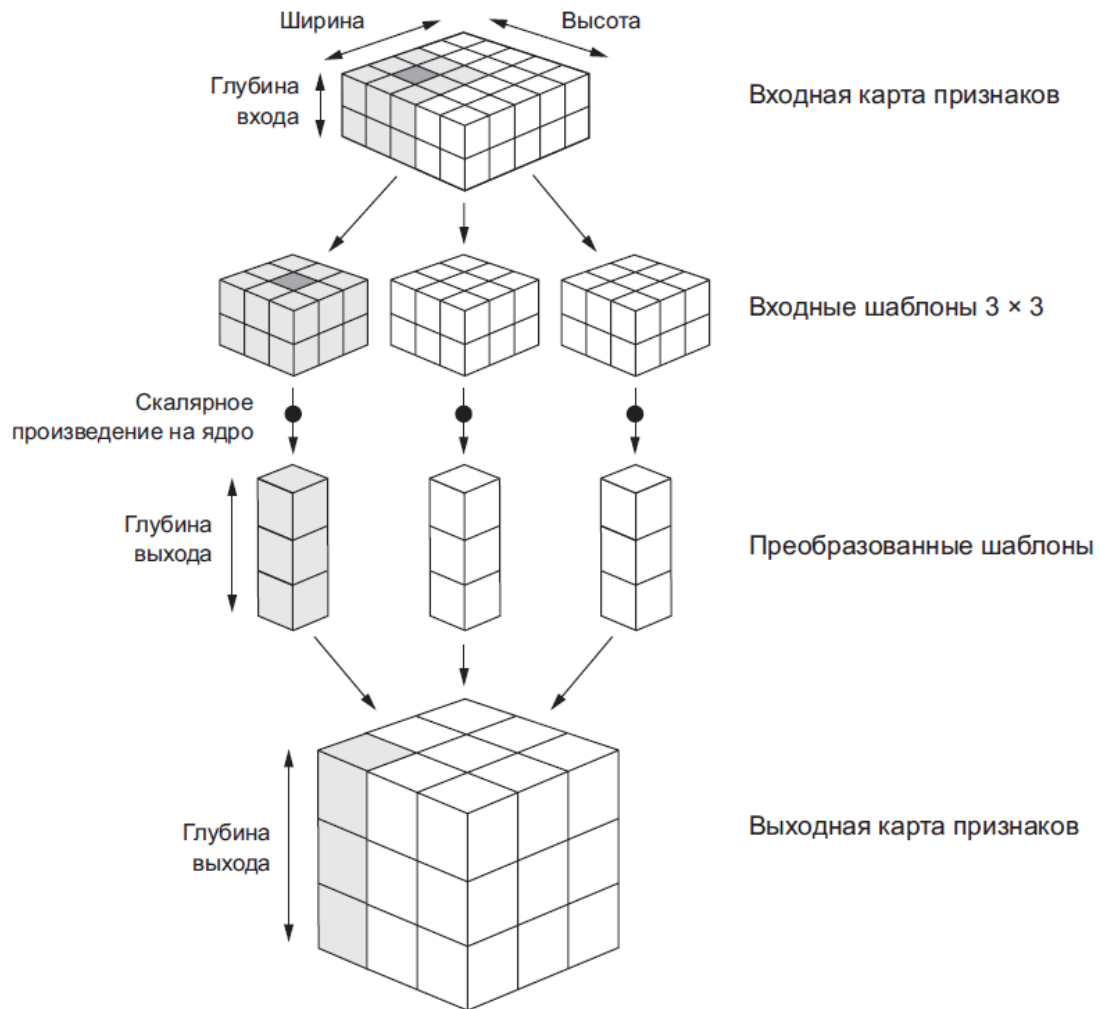
- Шаблоны, которые они изучают, являются инвариантными в отношении переноса. После изучения определенного шаблона в правом нижнем углу картинки сверточная нейронная сеть сможет распознавать его повсюду: например, в левом верхнем углу. Полносвязной сети пришлось бы изучить шаблон заново, если он появляется в другом месте. *Видимый мир по своей сути является инвариантным в отношении переноса.*
- Они могут изучать пространственные иерархии шаблонов. Первый сверточный слой будет изучать небольшие локальные шаблоны, такие как края, второй — более крупные шаблоны, состоящие из признаков, возвращаемых первым слоем, и т.д. Видимый мир формируется пространственными иерархиями видимых модулей: гиперлокальные края объединяются в локальные объекты, такие как глаза или уши, которые, в свою очередь, объединяются в понятия еще более высокого уровня, такие как «кошка».

6.1. Сверточные нейронные сети

ОПЕРАЦИЯ СВЕРТЫВАНИЯ

- Свертка применяется к трехмерным тензорам, называемым *картами признаков*, с двумя *пространственными осями* (высота и ширина), а также с *осью глубины* (или *осью каналов*).
- Операция свертывания извлекает шаблоны из своей входной карты признаков и применяет одинаковые преобразования ко всем шаблонам, производя *выходную карту признаков*.
- Глубина карты может иметь любую размерность, потому что выходная глубина является параметром слоя, и разные каналы на этой оси глубины больше не соответствуют конкретным цветам, как во входных данных в формате RGB; они соответствуют *фильтрам*. Фильтры представляют конкретные аспекты входных данных: на верхнем уровне, например, фильтр может соответствовать понятию «присутствие лица на входе».
- Свертки определяются двумя ключевыми параметрами:
 - размером шаблонов, извлекаемых из входных данных — обычно 3 x 3 или 5 x 5;
 - глубиной выходной карты признаков — количеством фильтров, вычисляемых сверткой.
- В Keras эти параметры передаются в слой в первых аргументах:
`layer_conv_2d (выходная_глубина, с (высота_окна, ширина_окна))`

6.1. Сверточные нейронные сети



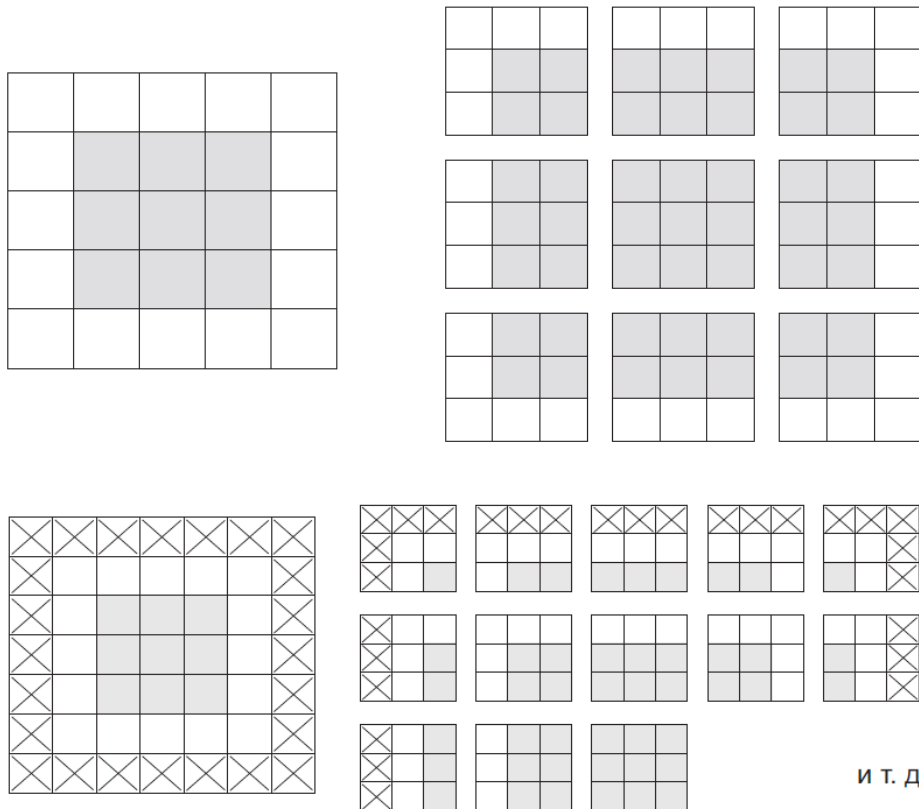
Выходные ширина и высота могут отличаться от входных.

На то есть две причины:

- эффекты границ, которые могут устраняться дополнением входной карты признаков;
- использование шага свертки

6.1. Сверточные нейронные сети

ЭФФЕКТЫ ГРАНИЦ И ДОПОЛНЕНИЕ



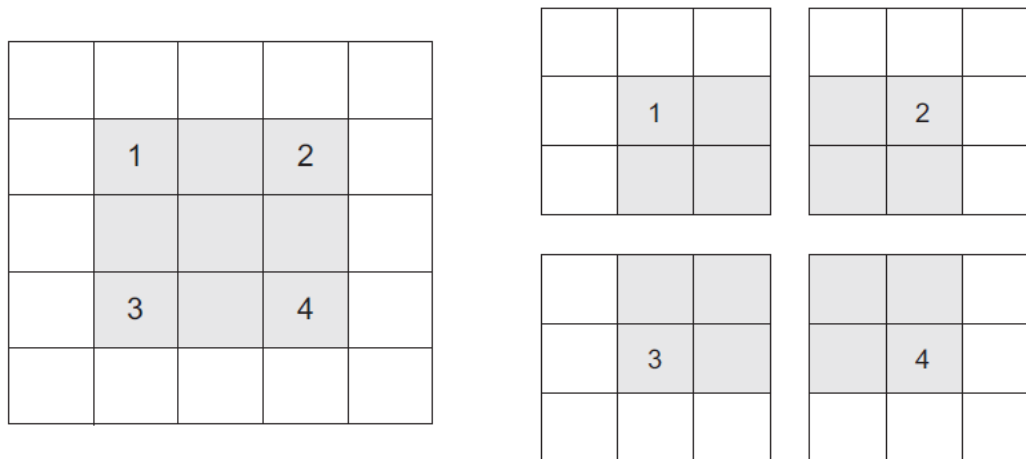
- Карта признаков 5×5 (всего 25 клеток).
- Существует всего 9 клеток, в которых может находиться центр окна 3×3 , образующих сетку 3×3 . Следовательно, карта выходных признаков будет иметь размер 3×3 .
- Она сжатая — ровно на две клетки вдоль каждого измерения.
- Дополнение заключается в добавлении соответствующего количества строк и столбцов с каждой стороны входной карты признаков, чтобы можно было поместить центр окна свертки в каждую входную клетку.

При использовании слоев `layer_conv_2d` дополнение настраивается с помощью аргумента `padding`, который принимает два значения: `valid` (будут использоваться только допустимые местоположения окна), и `same` (дополнить так, чтобы выходная карта признаков имела ту же ширину и высоту, что и входная).

6.1. Сверточные нейронные сети

ШАГ СВЕРТКИ

В общем случае расстояние между двумя соседними окнами является настраиваемым параметром, который называется шагом свертки и по умолчанию равен 1.



- Использование шага 2 означает уменьшение ширины и высоты карты признаков за счет уменьшения разрешения в 2 раза (в дополнение к любым изменениям, вызванным эффектами границ).
- Для уменьшения разрешения карты признаков вместо шага часто используется операция выбора максимального значения из соседних (max-pooling).

Из входной карты признаков извлекается окно и из него выбирается максимальное значение для каждого канала. Концептуально это напоминает свертку, но вместо преобразования локальных шаблонов с обучением на линейных преобразованиях (ядро свертки) они преобразуются с использованием жестко заданной тензорной операции выбора максимального значения. Главное отличие от свертки состоит в том, что выбор максимального значения из соседних обычно производится с окном 2×2 и шагом 2, чтобы уменьшить разрешение карты признаков в 2 раза.

6.1. Сверточные нейронные сети

```
model_no_max_pool <- keras_model_sequential() %>%  
  layer_conv_2d(filters = 32, kernel_size = c(3, 3), activation = "relu",  
                input_shape = c(28, 28, 1)) %>%  
  layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%  
  layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu")
```

Сводная информация о модели:

```
> model_no_max_pool
```

Layer (type)	Output Shape	Param #
=====		
conv2d_4 (Conv2D)	(None, 26, 26, 32)	320

conv2d_5 (Conv2D)	(None, 24, 24, 64)	18496

conv2d_6 (Conv2D)	(None, 22, 22, 64)	36928
=====		
Total params: 55,744		
Trainable params: 55,744		
Non-trainable params: 0		

Уменьшение разрешения используется для уменьшения количества коэффициентов в карте признаков для обработки, а также внедрения иерархий пространственных фильтров путем создания последовательных уровней свертки для просмотра все более крупных окон (с точки зрения долей исходных данных, которые они охватывают).

6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных

Классификация изображений собак и кошек из набора данных, содержащего 4000 изображений (2000 кошек, 2000 собак). Мы будем использовать 2000 изображений для обучения, 1000 для проверки и 1000 для контроля.



```
original_dataset_dir <- "~/Downloads/kaggle_original_data"
```

```
base_dir <- "~/Downloads/cats_and_dogs_small"  
dir.create(base_dir)
```

```
train_dir <- file.path(base_dir, "train")  
dir.create(train_dir)
```

6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных

Сверточная нейронная сеть будет организована как стек чередующихся уровней `layer_conv_2d` (с функцией активации `relu`) и `layer_max_pooling_2d`. Сеть будет иметь на одну пару уровней `layer_conv_2d` + `layer_max_pooling_2d` больше. Это увеличит ее емкость и обеспечит дополнительное снижение размеров карт признаков, чтобы они не оказались слишком большими, когда достигнут уровня `layer_flatten`.

```
library(keras)

model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3), activation = "relu",
               input_shape = c(150, 150, 3)) %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3, 3),
               activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3),
               activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3),
               activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dense(units = 512, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")
```

6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных

```
> summary(model)
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
maxpooling2d_1 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18496
maxpooling2d_2 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73856
maxpooling2d_3 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_4 (Conv2D)	(None, 15, 15, 128)	147584
maxpooling2d_4 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 512)	3211776
dense_2 (Dense)	(None, 1)	513

```
Total params: 3,453,121
```

```
Trainable params: 3,453,121
```

```
Non-trainable params: 0
```

6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных

```
model %>% compile(  
  loss = "binary_crossentropy",  
  optimizer = optimizer_rmsprop(lr = 1e-4),  
  metrics = c("acc")  
)
```

ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ

В настоящее время данные хранятся в виде файлов JPEG, поэтому их нужно подготовить для передачи в сеть, выполнив следующие шаги:

1. Прочитать файлы с изображениями.
2. Декодировать содержимое из формата JPEG в таблицы пикселей RGB.
3. Преобразовать их в тензоры с вещественными числами.
4. Масштабировать значения пикселей из диапазона [0, 255] в диапазон [0, 1] (как вы уже знаете, нейронным сетям предпочтительнее передавать небольшие значения).

6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных

```
train_datagen <- image_data_generator(rescale = 1/255)
validation_datagen <- image_data_generator(rescale = 1/255)
```

Масштабировать значения с коэффициентом 1/255

```
train_generator <- flow_images_from_directory(
  train_dir,
  train_datagen,
  target_size = c(150, 150),
  batch_size = 20,
  class_mode = «binary»
)
```

Целевой каталог
Генератор данных обучения
Привести все изображения к размеру 150 × 150
Так как используется функция потерь `binary_crossentropy`, метки должны быть бинарными

```
validation_generator <- flow_images_from_directory(
  validation_dir,
  validation_datagen,
  target_size = c(150, 150),
  batch_size = 20,
  class_mode = «binary»
)
```

```
> batch <- generator_next(train_generator)
> str(batch)
List of 2
 $ : num [1:20, 1:150, 1:150, 1:3] 37 48 153 53 114 194 158 141 255 167 ...
 $ : num [1:20(1d)] 1 1 1 1 0 1 1 0 1 1 ...
```

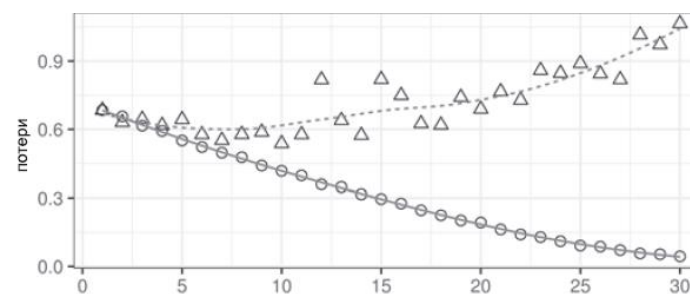
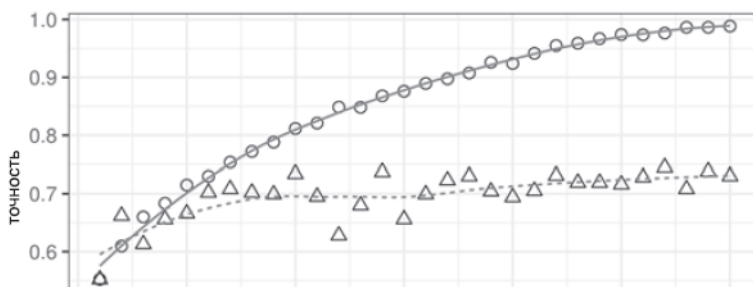
6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных

Обучение модели с использованием генератора пакетов

```
history <- model %>% fit_generator(  
  train_generator,  
  steps_per_epoch = 100,  
  epochs = 30,  
  validation_data = validation_generator,  
  validation_steps = 50  
)
```

Сохранение модели

```
model %>% save_model_hdf5("cats_and_dogs_small_1.h5")
```



данные
—○— обучение
-△- проверка

6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных

РАСШИРЕНИЕ ДАННЫХ

Цель состоит в том, чтобы на этапе обучения модель никогда не увидела одно и то же изображение дважды. Это поможет модели выявить больше особенностей данных и достичь лучшей степени обобщения.

```
datagen <- image_data_generator(  
  rescale = 1/255,  
  rotation_range = 40,  
  width_shift_range = 0.2,  
  height_shift_range = 0.2,  
  shear_range = 0.2,  
  zoom_range = 0.2,  
  horizontal_flip = TRUE,  
  fill_mode = "nearest"  
)
```

`rotation_range` — величина в градусах (0–180), диапазон, в котором будет осуществляться случайный поворот изображения;

`width_shift` и `height_shift` — диапазоны (в долях ширины и высоты), в пределах которых изображения смещаются по горизонтали и по вертикали соответственно;

`shear_range` — для случайного применения сдвигового (shearing) преобразования;

`zoom_range` — для случайного изменения масштаба внутри изображений;

`horizontal_flip` — для случайного переворачивания половины изображения по горизонтали — подходит в случаях отсутствия предположений о горизонтальной асимметрии (например, в изображениях реального мира);

`fill_mode` — стратегия заполнения вновь созданных пикселей, появляющихся после поворота или смещения по горизонтали/вертикали.

6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных

Отображение некоторых обучающих изображений, подвергшихся случайным преобразованиям

```

                                Выбор одного изображения для расширения
fnames <- list.files(train_cats_dir, full.names = TRUE)
img_path <- fnames[[3]]

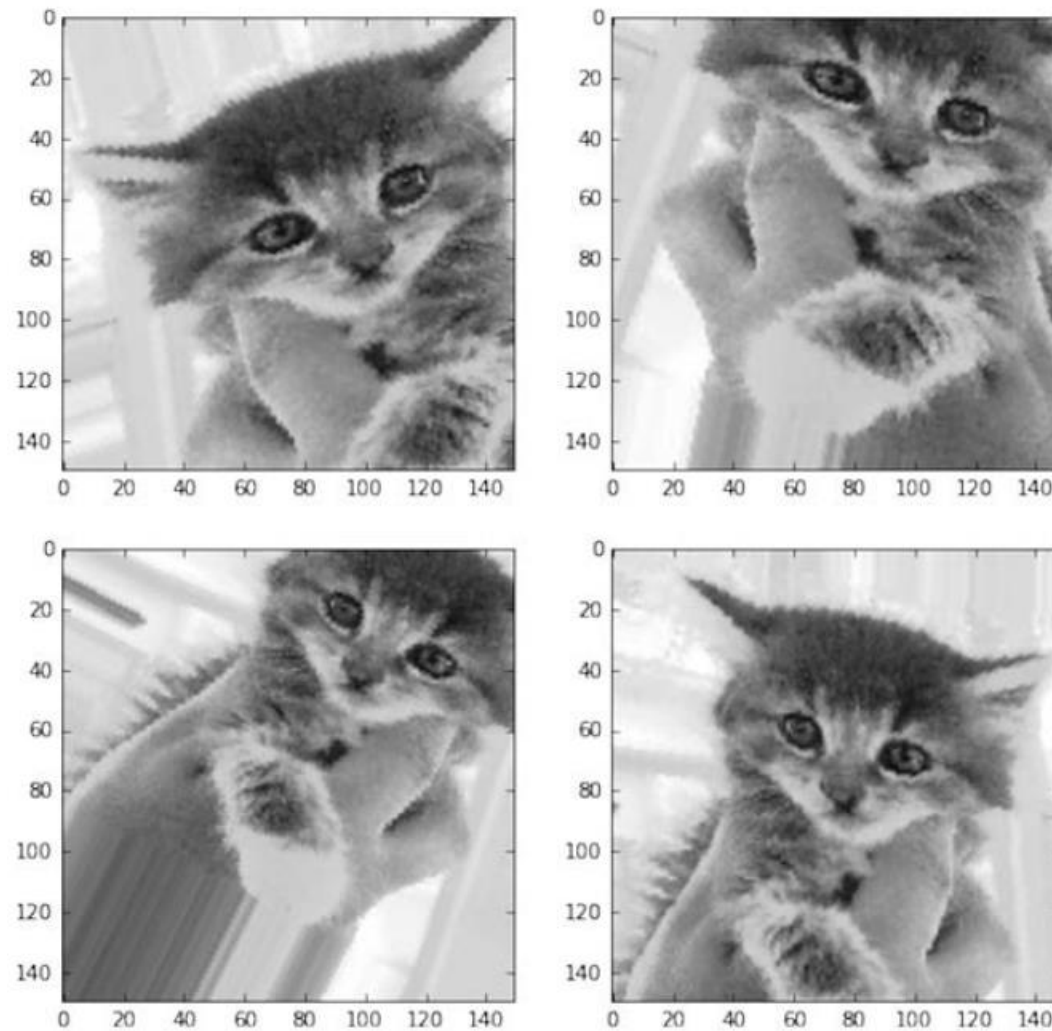
                                Прочитать изображение
                                и изменить его размеры
img <- image_load(img_path, target_size = c(150, 150))
img_array <- image_to_array(img)
                                Преобразовать
                                в массив
                                с формой (150, 150, 3)
img_array <- array_reshape(img_array, c(1, 150, 150, 3))
                                Изменить форму
                                на (1, 150, 150, 3)

augmentation_generator <- flow_images_from_data(
  img_array,
  generator = datagen,
  batch_size = 1
)
                                Сгенерировать пакеты случайно преобразованных
                                изображений. Цикл выполняется бесконечно, поэтому его
                                нужно принудительно прервать в некоторый момент!

op <- par(mfrow = c(2, 2), pty = «s», mar = c(1, 0, 1, 0))
for (i in 1:4) {
  batch <- generator_next(augmentation_generator)
  plot(as.raster(batch[1,,,]))
}

                                Вывод изображений
par(op)
```


6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных



6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных

Обучение сверточной нейронной сети с использованием
генераторов расширения данных

```
datagen <- image_data_generator(  
  rescale = 1/255,  
  rotation_range = 40,  
  width_shift_range = 0.2,  
  height_shift_range = 0.2,  
  shear_range = 0.2,  
  zoom_range = 0.2,  
  horizontal_flip = TRUE  
)  
  
test_datagen <-  
  image_data_generator(rescale = 1/255)  
  
train_generator <- flow_images_from_directory(←  
  train_dir, ← Целевой каталог  
  datagen, ← Генератор данных  
  target_size = c(150, 150), ← Привести все изображения  
  batch_size = 32, ← к размеру 150 × 150  
  class_mode = "binary"  
)
```

Обратите внимание, что проверочные
данные не требуется расширять!

Так как используется функция потерь `binary_crossentropy`,
метки должны быть бинарными

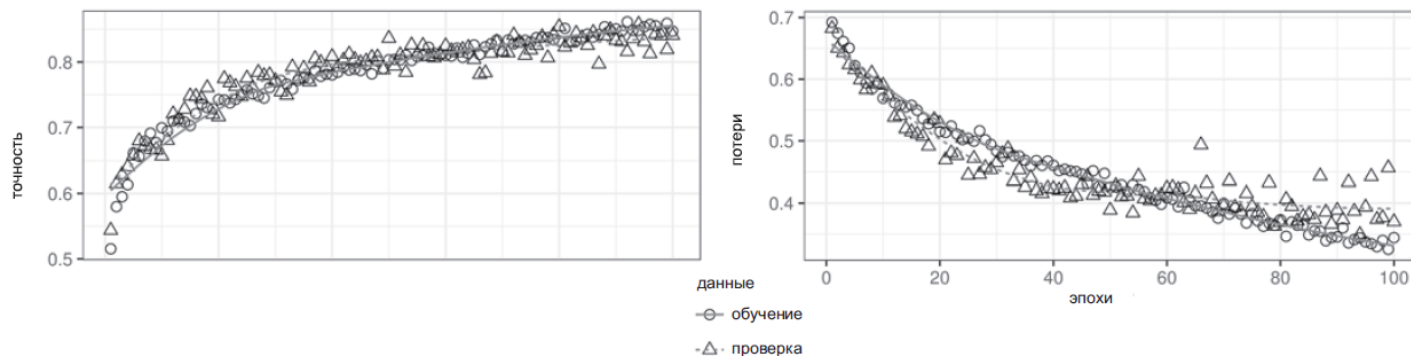
6.2. Обучение сверточной нейронной сети с нуля на небольшом наборе данных

```
validation_generator <- flow_images_from_directory(  
  validation_dir,  
  test_datagen,  
  target_size = c(150, 150),  
  batch_size = 32,  
  class_mode = «binary»  
)
```

```
history <- model %>% fit_generator(  
  train_generator,  
  steps_per_epoch = 100,  
  epochs = 100,  
  validation_data = validation_generator,  
  validation_steps = 50  
)
```

Сохранение модели

```
model %>% save_model_hdf5("cats_and_dogs_small_2.h5")
```

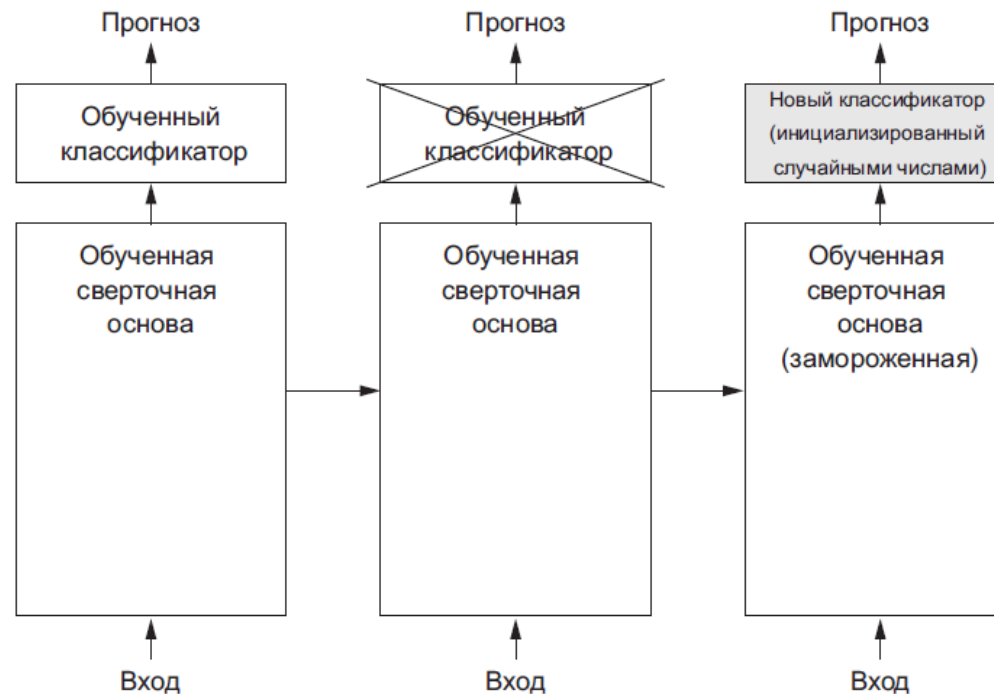


6.3. Использование предварительно обученной сверточной нейронной сети

Есть два приема использования предварительно обученных сетей: выделение признаков (feature extraction) и дообучение (fine-tuning).

ВЫДЕЛЕНИЕ ПРИЗНАКОВ

Выделение признаков заключается в использовании представлений, изученных предыдущей сетью, для выделения признаков из новых образцов, которые затем пропускаются через новый классификатор, обучаемый с нуля.



6.3. Использование предварительно обученной сверточной нейронной сети

Сверточная основа сети VGG16, обученной на данных ImageNet, для выделения полезных признаков из изображений кошек и собак.

```
library(keras)

conv_base <- application_vgg16(
  weights = "imagenet",
  include_top = FALSE,
  input_shape = c(150, 150, 3)
)
```

- `weights` определяет источник весов для инициализации модели.
- `include_top` определяет необходимость подключения к сети полносвязного классификатора.
- `input_shape` определяет форму тензоров с изображениями, которые будут подаваться на вход сети.

Далее можно пойти двумя путями:

- Пропустить набор данных через сверточную основу, записать получившийся массив на диск и затем использовать его как входные данные для отдельного полносвязного классификатора. Это быстрое и недорогое решение, потому что требует запускать сверточную основу только один раз для каждого входного изображения, а сверточная основа — самая дорогостоящая часть конвейера.
- Дополнить имеющуюся модель (`conv_base`) полносвязными слоями и пропустить все входные данные. Этот путь позволяет использовать расширение данных, потому что каждое изображение проходит через сверточную основу каждый раз, когда попадает в модель. Но по той же причине этот путь намного дороже первого.

6.3. Использование предварительно обученной сверточной нейронной сети

БЫСТРОЕ ВЫДЕЛЕНИЕ ПРИЗНАКОВ БЕЗ РАСШИРЕНИЯ ДАННЫХ

```
base_dir <- "~/Downloads/cats_and_dogs_small"
train_dir <- file.path(base_dir, "train")
validation_dir <- file.path(base_dir, "validation")
test_dir <- file.path(base_dir, "test")

datagen <- image_data_generator(rescale = 1/255)
batch_size <- 20

extract_features <- function(directory, sample_count) {

  features <- array(0, dim = c(sample_count, 4, 4, 512))
  labels <- array(0, dim = c(sample_count))

  generator <- flow_images_from_directory(
    directory = directory,
    generator = datagen,
    target_size = c(150, 150),
    batch_size = batch_size,
    class_mode = "binary"
  )
```

6.3. Использование предварительно обученной сверточной нейронной сети

БЫСТРОЕ ВЫДЕЛЕНИЕ ПРИЗНАКОВ БЕЗ РАСШИРЕНИЯ ДАННЫХ

```
i <- 0
while(TRUE) {
  batch <- generator_next(generator)
  inputs_batch <- batch[[1]]
  labels_batch <- batch[[2]]
  features_batch <- conv_base %>% predict(inputs_batch)

  index_range <- ((i * batch_size)+1):((i + 1) * batch_size)
  features[index_range,,] <- features_batch
  labels[index_range] <- labels_batch
  i <- i + 1
  if (i * batch_size >= sample_count)
    break
}

list(
  features = features,
  labels = labels
)
}
```

Обратите внимание: поскольку генераторы возвращают данные в цикле до бесконечности, мы должны прервать цикл после передачи всех изображений

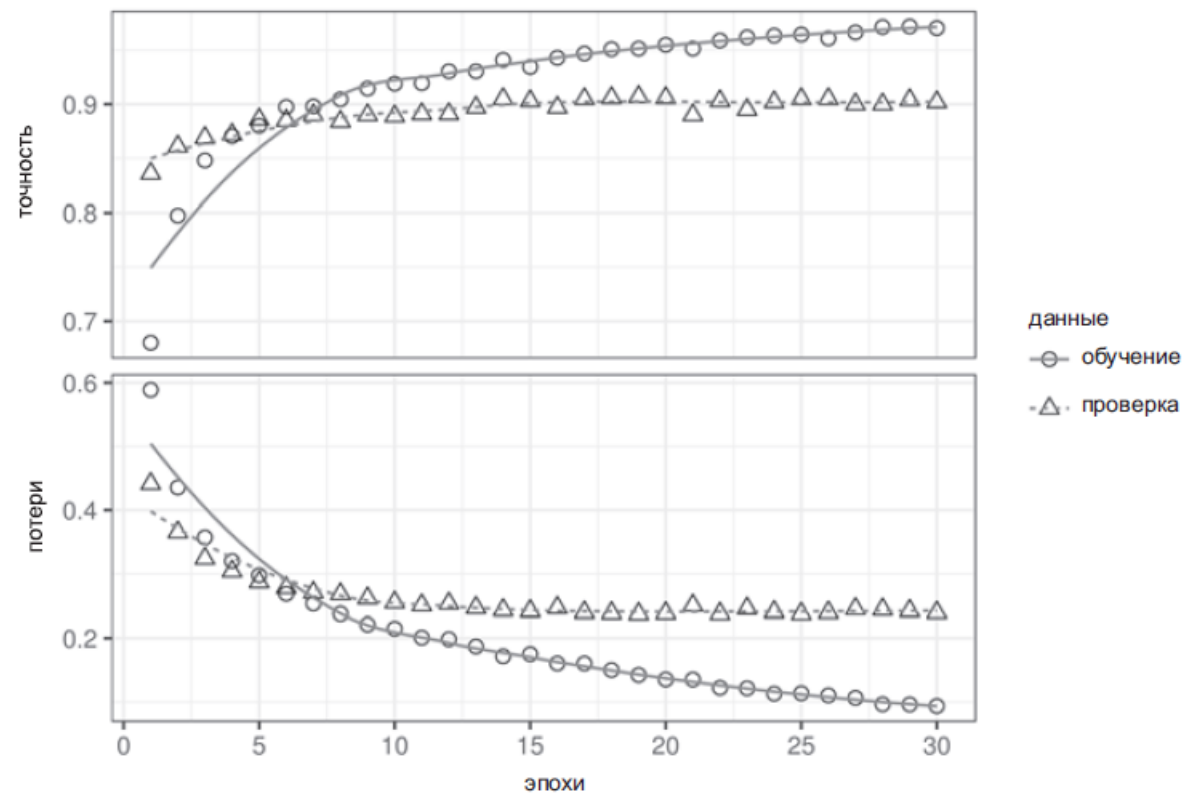
```
train <- extract_features(train_dir, 2000)
validation <- extract_features(validation_dir, 1000)
test <- extract_features(test_dir, 1000)
```

6.3. Использование предварительно обученной сверточной нейронной сети

БЫСТРОЕ ВЫДЕЛЕНИЕ ПРИЗНАКОВ БЕЗ РАСШИРЕНИЯ ДАННЫХ

```
reshape_features <- function(features) {  
  array_reshape(features, dim = c(nrow(features), 4 * 4 * 512))  
}  
train$features <- reshape_features(train$features)  
validation$features <- reshape_features(validation$features)  
test$features <- reshape_features(test$features)  
  
model <- keras_model_sequential() %>%  
  layer_dense(units = 256, activation = "relu",  
              input_shape = 4 * 4 * 512) %>%  
  layer_dropout(rate = 0.5) %>%  
  layer_dense(units = 1, activation = "sigmoid")  
  
model %>% compile(  
  optimizer = optimizer_rmsprop(lr = 2e-5),  
  loss = "binary_crossentropy",  
  metrics = c("accuracy")  
)  
  
history <- model %>% fit(  
  train$features, train$labels,  
  epochs = 30,  
  batch_size = 20,  
  validation_data = list(validation$features, validation$labels)  
)
```


6.3. Использование предварительно обученной сверточной нейронной сети



6.3. Использование предварительно обученной сверточной нейронной сети

ВЫДЕЛЕНИЕ ПРИЗНАКОВ С РАСШИРЕНИЕМ ДАННЫХ

Добавление полносвязного классификатора
поверх сверточной основы

```
model <- keras_model_sequential() %>%  
  conv_base %>%  
  layer_flatten() %>%  
  layer_dense(units = 256, activation = "relu") %>%  
  layer_dense(units = 1, activation = "sigmoid")  
  
> model
```

Layer (type)	Output Shape	Param #
vgg16 (Model)	(None, 4, 4, 512)	14714688
flatten_1 (Flatten)	(None, 8192)	0
dense_1 (Dense)	(None, 256)	2097408
dense_2 (Dense)	(None, 1)	257

=====
Total params: 16,812,353
Trainable params: 16,812,353
Non-trainable params: 0

Перед компиляцией и обучением модели очень важно заморозить сверточную основу. Замораживание одного или нескольких слоев предотвращает изменение весовых коэффициентов в них в процессе обучения. Если этого не сделать, тогда представления, прежде изученные сверточной основой, изменятся в процессе обучения на новых данных.

6.3. Использование предварительно обученной сверточной нейронной сети

ВЫДЕЛЕНИЕ ПРИЗНАКОВ С РАСШИРЕНИЕМ ДАННЫХ

```
train_datagen = image_data_generator(  
    rescale = 1/255,  
    rotation_range = 40,  
    width_shift_range = 0.2,  
    height_shift_range = 0.2,  
    shear_range = 0.2,  
    zoom_range = 0.2,  
    horizontal_flip = TRUE,  
    fill_mode = "nearest"  
)
```

```
test_datagen <-  
    image_data_generator(rescale = 1/255)
```

```
train_generator <- flow_images_from_directory(  
    train_dir,  
    train_datagen,  
    target_size = c(150, 150),  
    batch_size = 20,  
    class_mode = «binary»  
)
```

Обратите внимание, что
проверочные данные не
требуется расширять!

← Целевой каталог

← Генератор данных

← Привести все изображения
к размеру 150 × 150

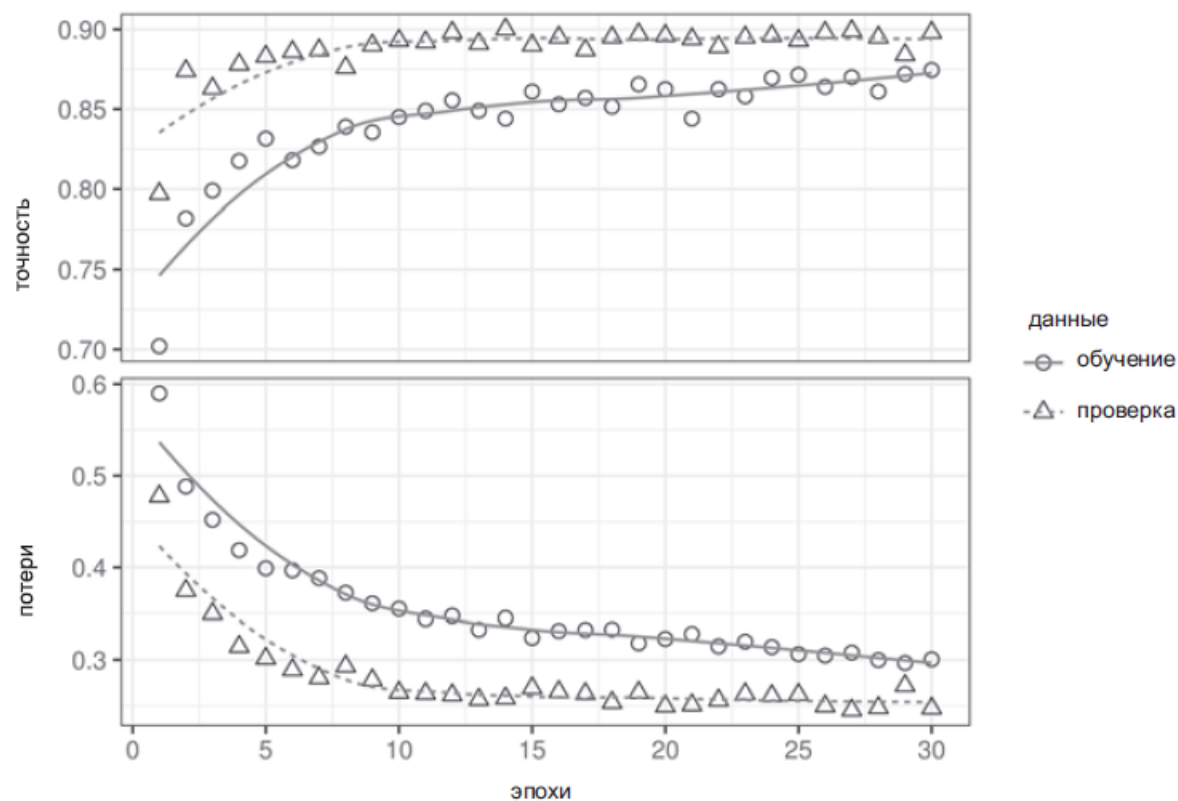
6.3. Использование предварительно обученной сверточной нейронной сети

ВЫДЕЛЕНИЕ ПРИЗНАКОВ С РАСШИРЕНИЕМ ДАННЫХ

```
validation_generator <- flow_images_from_directory(  
  validation_dir,  
  test_datagen,  
  target_size = c(150, 150),  
  batch_size = 20,  
  class_mode = "binary"  
)  
  
model %>% compile(  
  loss = "binary_crossentropy"  
  optimizer = optimizer_rmsprop(lr = 2e-5),  
  metrics = c("accuracy")  
)  
  
history <- model %>% fit_generator(  
  train_generator,  
  steps_per_epoch = 100,  
  epochs = 30,  
  validation_data = validation_generator,  
  validation_steps = 50  
)
```

6.3. Использование предварительно обученной сверточной нейронной сети

ВЫДЕЛЕНИЕ ПРИЗНАКОВ С РАСШИРЕНИЕМ ДАННЫХ



6.3. Использование предварительно обученной сверточной нейронной сети

ДООБУЧЕНИЕ

Дообучение заключается в размораживании нескольких верхних уровней замороженной модели, использовавшейся для выделения признаков, и совместное обучение вновь добавленной части модели (в данном случае полносвязного классификатора) и этих верхних уровней.



6.3. Использование предварительно обученной сверточной нейронной сети

ДООБУЧЕНИЕ

Для дообучения сети требуется выполнить следующие шаги:

1. Добавить свою сеть поверх обученной базовой сети.
2. Заморозить базовую сеть.
3. Обучить добавленную часть.
4. Разморозить несколько слоев в базовой сети.
5. Обучить эти слои и добавленную часть вместе.

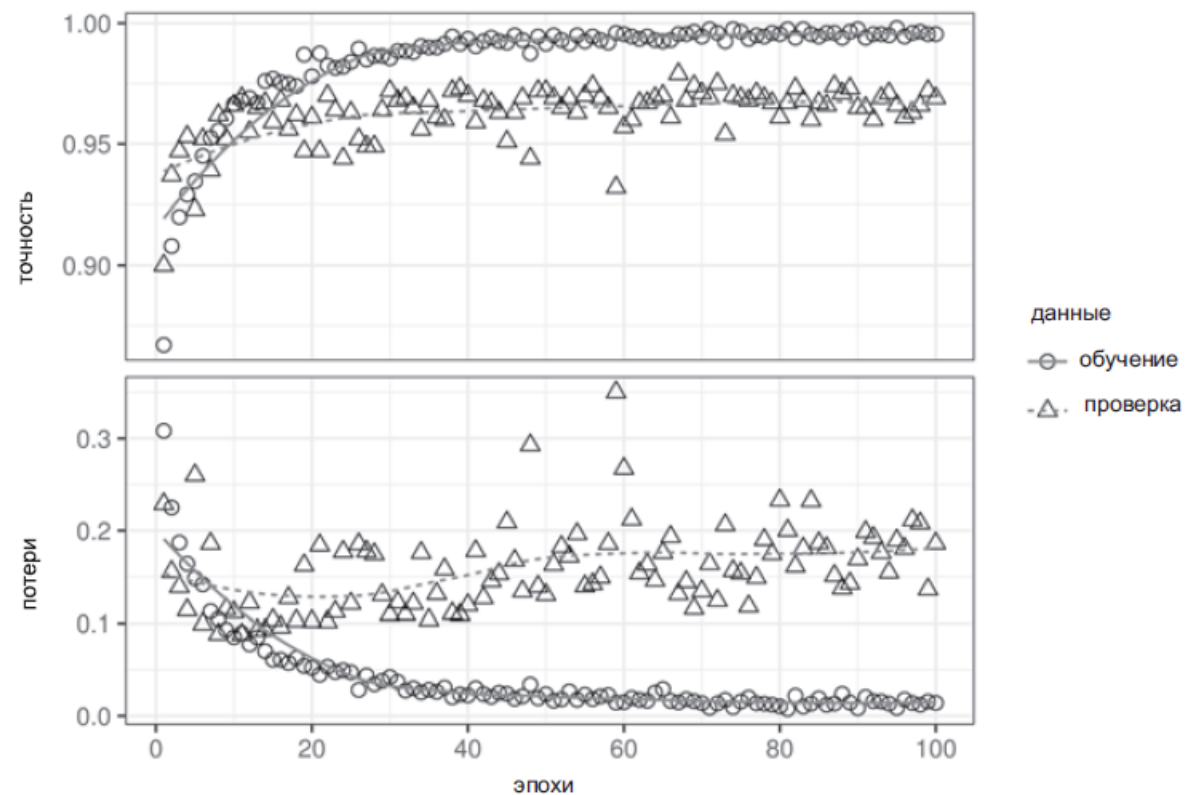
```
model %>% compile(  
  loss = "binary_crossentropy",  
  optimizer = optimizer_rmsprop(lr = 1e-5),  
  metrics = c("accuracy")  
)  
  
history <- model %>% fit_generator(  

```

```
  train_generator,  
  steps_per_epoch = 100,  
  epochs = 100,  
  validation_data = validation_generator,  
  validation_steps = 50  
)
```

6.3. Использование предварительно обученной сверточной нейронной сети

ДООБУЧЕНИЕ



Итоги

- Сверточные нейронные сети — лучший тип моделей машинного обучения для задач распознавания образов. Вполне можно обучить такую сеть с нуля на очень небольшом наборе данных и получить приличный результат.
- Когда объем данных ограничен, главной проблемой становится переобучение. Расширение данных — мощное средство борьбы с переобучением при работе с изображениями.
- Существующую сверточную нейронную сеть с легкостью можно повторно использовать на новом наборе данных, применив прием выделения признаков. Этот прием особенно ценен при работе с небольшими наборами изображений.
- В дополнение к выделению признаков можно использовать прием дообучения, который адаптирует к новой задаче некоторые из представлений, ранее полученных существующей моделью. Он еще больше повышает качество модели.