Module6

Andrew Estes

4/22/2022

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.1.2

## Warning: package 'tidyr' was built under R version 4.1.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(keras)
```

```
## Warning: package 'keras' was built under R version 4.1.3
```

```
library(knitr)
```

```
k_clear_session()
```

```
## Loaded Tensorflow version 2.8.0
```

```
model <- application_resnet50(weights = 'imagenet')
summary(model)
```

```
## Model: "resnet50"
## __________________________________________________________________________________
##  Layer (type)                Output Shape      Param #  Connected to
## ==================================================================================
##  input_1 (InputLayer)        [(None, 224, 224  0        []
##                              , 3)]
##
```

```
##  conv1_pad (ZeroPadding2D  (None, 230, 230,  0       ['input_1[0][0]']
##  )                         3)
##
##  conv1_conv (Conv2D)      (None, 112, 112,  9472     ['conv1_pad[0][0]']
##                            64)
##
##  conv1_bn (BatchNormaliza  (None, 112, 112,  256     ['conv1_conv[0][0]']
##  tion)                     64)
##
##  conv1_relu (Activation)  (None, 112, 112,  0        ['conv1_bn[0][0]']
##                            64)
##
##  pool1_pad (ZeroPadding2D  (None, 114, 114,  0       ['conv1_relu[0][0]']
##  )                         64)
##
##  pool1_pool (MaxPooling2D  (None, 56, 56, 6  0       ['pool1_pad[0][0]']
##  )                        4)
##
##  conv2_block1_1_conv (Con  (None, 56, 56, 6  4160    ['pool1_pool[0][0]']
##  v2D)                     4)
##
##  conv2_block1_1_bn (Batch  (None, 56, 56, 6  256     ['conv2_block1_1_conv[0][0]
##  Normalization)           4)                         ']
##
##  conv2_block1_1_relu (Act  (None, 56, 56, 6  0       ['conv2_block1_1_bn[0][0]']
##  ivation)                 4)
##
##  conv2_block1_2_conv (Con  (None, 56, 56, 6  36928   ['conv2_block1_1_relu[0][0]
##  v2D)                     4)                         ']
##
##  conv2_block1_2_bn (Batch  (None, 56, 56, 6  256     ['conv2_block1_2_conv[0][0]
##  Normalization)           4)                         ']
##
##  conv2_block1_2_relu (Act  (None, 56, 56, 6  0       ['conv2_block1_2_bn[0][0]']
##  ivation)                 4)
##
##  conv2_block1_0_conv (Con  (None, 56, 56, 2  16640   ['pool1_pool[0][0]']
##  v2D)                     56)
##
##  conv2_block1_3_conv (Con  (None, 56, 56, 2  16640   ['conv2_block1_2_relu[0][0]
##  v2D)                     56)                        ']
##
##  conv2_block1_0_bn (Batch  (None, 56, 56, 2  1024    ['conv2_block1_0_conv[0][0]
##  Normalization)           56)                        ']
##
##  conv2_block1_3_bn (Batch  (None, 56, 56, 2  1024    ['conv2_block1_3_conv[0][0]
##  Normalization)           56)                        ']
##
##  conv2_block1_add (Add)   (None, 56, 56, 2  0        ['conv2_block1_0_bn[0][0]',
##                           56)                         'conv2_block1_3_bn[0][0]']
##
##  conv2_block1_out (Activa  (None, 56, 56, 2  0       ['conv2_block1_add[0][0]']
##  tion)                    56)
##
```

```
##  conv2_block2_1_conv (Con  (None, 56, 56, 6  16448   ['conv2_block1_out[0][0]']
##  v2D)                      4)
##
##  conv2_block2_1_bn (Batch  (None, 56, 56, 6  256     ['conv2_block2_1_conv[0][0]
##  Normalization)            4)                        ']
##
##  conv2_block2_1_relu (Act  (None, 56, 56, 6  0       ['conv2_block2_1_bn[0][0]']
##  ivation)                  4)
##
##  conv2_block2_2_conv (Con  (None, 56, 56, 6  36928   ['conv2_block2_1_relu[0][0]
##  v2D)                      4)                        ']
##
##  conv2_block2_2_bn (Batch  (None, 56, 56, 6  256     ['conv2_block2_2_conv[0][0]
##  Normalization)            4)                        ']
##
##  conv2_block2_2_relu (Act  (None, 56, 56, 6  0       ['conv2_block2_2_bn[0][0]']
##  ivation)                  4)
##
##  conv2_block2_3_conv (Con  (None, 56, 56, 2  16640   ['conv2_block2_2_relu[0][0]
##  v2D)                      56)                       ']
##
##  conv2_block2_3_bn (Batch  (None, 56, 56, 2  1024    ['conv2_block2_3_conv[0][0]
##  Normalization)            56)                       ']
##
##  conv2_block2_add (Add)    (None, 56, 56, 2  0        ['conv2_block1_out[0][0]',
##                            56)                        'conv2_block2_3_bn[0][0]']
##
##  conv2_block2_out (Activa  (None, 56, 56, 2  0       ['conv2_block2_add[0][0]']
##  tion)                     56)
##
##  conv2_block3_1_conv (Con  (None, 56, 56, 6  16448   ['conv2_block2_out[0][0]']
##  v2D)                      4)
##
##  conv2_block3_1_bn (Batch  (None, 56, 56, 6  256     ['conv2_block3_1_conv[0][0]
##  Normalization)            4)                        ']
##
##  conv2_block3_1_relu (Act  (None, 56, 56, 6  0       ['conv2_block3_1_bn[0][0]']
##  ivation)                  4)
##
##  conv2_block3_2_conv (Con  (None, 56, 56, 6  36928   ['conv2_block3_1_relu[0][0]
##  v2D)                      4)                        ']
##
##  conv2_block3_2_bn (Batch  (None, 56, 56, 6  256     ['conv2_block3_2_conv[0][0]
##  Normalization)            4)                        ']
##
##  conv2_block3_2_relu (Act  (None, 56, 56, 6  0       ['conv2_block3_2_bn[0][0]']
##  ivation)                  4)
##
##  conv2_block3_3_conv (Con  (None, 56, 56, 2  16640   ['conv2_block3_2_relu[0][0]
##  v2D)                      56)                       ']
##
##  conv2_block3_3_bn (Batch  (None, 56, 56, 2  1024    ['conv2_block3_3_conv[0][0]
##  Normalization)            56)                       ']
##
```

```
##  conv2_block3_add (Add)   (None, 56, 56, 2  0        ['conv2_block2_out[0][0]',
##                           56)                        'conv2_block3_3_bn[0][0]']
##
##  conv2_block3_out (Activa  (None, 56, 56, 2  0       ['conv2_block3_add[0][0]']
##  tion)                    56)
##
##  conv3_block1_1_conv (Con  (None, 28, 28, 1  32896   ['conv2_block3_out[0][0]']
##  v2D)                     28)
##
##  conv3_block1_1_bn (Batch  (None, 28, 28, 1  512     ['conv3_block1_1_conv[0][0]
##  Normalization)           28)                        ']
##
##  conv3_block1_1_relu (Act  (None, 28, 28, 1  0       ['conv3_block1_1_bn[0][0]']
##  ivation)                 28)
##
##  conv3_block1_2_conv (Con  (None, 28, 28, 1  147584  ['conv3_block1_1_relu[0][0]
##  v2D)                     28)                        ']
##
##  conv3_block1_2_bn (Batch  (None, 28, 28, 1  512     ['conv3_block1_2_conv[0][0]
##  Normalization)           28)                        ']
##
##  conv3_block1_2_relu (Act  (None, 28, 28, 1  0       ['conv3_block1_2_bn[0][0]']
##  ivation)                 28)
##
##  conv3_block1_0_conv (Con  (None, 28, 28, 5  131584  ['conv2_block3_out[0][0]']
##  v2D)                     12)
##
##  conv3_block1_3_conv (Con  (None, 28, 28, 5  66048   ['conv3_block1_2_relu[0][0]
##  v2D)                     12)                        ']
##
##  conv3_block1_0_bn (Batch  (None, 28, 28, 5  2048    ['conv3_block1_0_conv[0][0]
##  Normalization)           12)                        ']
##
##  conv3_block1_3_bn (Batch  (None, 28, 28, 5  2048    ['conv3_block1_3_conv[0][0]
##  Normalization)           12)                        ']
##
##  conv3_block1_add (Add)   (None, 28, 28, 5  0        ['conv3_block1_0_bn[0][0]',
##                           12)                        'conv3_block1_3_bn[0][0]']
##
##  conv3_block1_out (Activa  (None, 28, 28, 5  0       ['conv3_block1_add[0][0]']
##  tion)                    12)
##
##  conv3_block2_1_conv (Con  (None, 28, 28, 1  65664   ['conv3_block1_out[0][0]']
##  v2D)                     28)
##
##  conv3_block2_1_bn (Batch  (None, 28, 28, 1  512     ['conv3_block2_1_conv[0][0]
##  Normalization)           28)                        ']
##
##  conv3_block2_1_relu (Act  (None, 28, 28, 1  0       ['conv3_block2_1_bn[0][0]']
##  ivation)                 28)
##
##  conv3_block2_2_conv (Con  (None, 28, 28, 1  147584  ['conv3_block2_1_relu[0][0]
##  v2D)                     28)                        ']
##
```

```
##  conv3_block2_2_bn (Batch  (None, 28, 28, 1  512     ['conv3_block2_2_conv[0][0]
##  Normalization)           28)                        ']
##
##  conv3_block2_2_relu (Act  (None, 28, 28, 1  0       ['conv3_block2_2_bn[0][0]']
##  ivation)                 28)
##
##  conv3_block2_3_conv (Con  (None, 28, 28, 5  66048   ['conv3_block2_2_relu[0][0]
##  v2D)                     12)                        ']
##
##  conv3_block2_3_bn (Batch  (None, 28, 28, 5  2048    ['conv3_block2_3_conv[0][0]
##  Normalization)           12)                        ']
##
##  conv3_block2_add (Add)   (None, 28, 28, 5  0        ['conv3_block1_out[0][0]',
##                           12)                         'conv3_block2_3_bn[0][0]']
##
##  conv3_block2_out (Activa  (None, 28, 28, 5  0       ['conv3_block2_add[0][0]']
##  tion)                    12)
##
##  conv3_block3_1_conv (Con  (None, 28, 28, 1  65664   ['conv3_block2_out[0][0]']
##  v2D)                     28)
##
##  conv3_block3_1_bn (Batch  (None, 28, 28, 1  512     ['conv3_block3_1_conv[0][0]
##  Normalization)           28)                        ']
##
##  conv3_block3_1_relu (Act  (None, 28, 28, 1  0       ['conv3_block3_1_bn[0][0]']
##  ivation)                 28)
##
##  conv3_block3_2_conv (Con  (None, 28, 28, 1  147584  ['conv3_block3_1_relu[0][0]
##  v2D)                     28)                        ']
##
##  conv3_block3_2_bn (Batch  (None, 28, 28, 1  512     ['conv3_block3_2_conv[0][0]
##  Normalization)           28)                        ']
##
##  conv3_block3_2_relu (Act  (None, 28, 28, 1  0       ['conv3_block3_2_bn[0][0]']
##  ivation)                 28)
##
##  conv3_block3_3_conv (Con  (None, 28, 28, 5  66048   ['conv3_block3_2_relu[0][0]
##  v2D)                     12)                        ']
##
##  conv3_block3_3_bn (Batch  (None, 28, 28, 5  2048    ['conv3_block3_3_conv[0][0]
##  Normalization)           12)                        ']
##
##  conv3_block3_add (Add)   (None, 28, 28, 5  0        ['conv3_block2_out[0][0]',
##                           12)                         'conv3_block3_3_bn[0][0]']
##
##  conv3_block3_out (Activa  (None, 28, 28, 5  0       ['conv3_block3_add[0][0]']
##  tion)                    12)
##
##  conv3_block4_1_conv (Con  (None, 28, 28, 1  65664   ['conv3_block3_out[0][0]']
##  v2D)                     28)
##
##  conv3_block4_1_bn (Batch  (None, 28, 28, 1  512     ['conv3_block4_1_conv[0][0]
##  Normalization)           28)                        ']
##
```

```
##  conv3_block4_1_relu (Act  (None, 28, 28, 1  0       ['conv3_block4_1_bn[0][0]']
##  ivation)                  28)
##
##  conv3_block4_2_conv (Con  (None, 28, 28, 1  147584  ['conv3_block4_1_relu[0][0]
##  v2D)                      28)                       ']
##
##  conv3_block4_2_bn (Batch  (None, 28, 28, 1  512     ['conv3_block4_2_conv[0][0]
##  Normalization)            28)                       ']
##
##  conv3_block4_2_relu (Act  (None, 28, 28, 1  0       ['conv3_block4_2_bn[0][0]']
##  ivation)                  28)
##
##  conv3_block4_3_conv (Con  (None, 28, 28, 5  66048   ['conv3_block4_2_relu[0][0]
##  v2D)                      12)                       ']
##
##  conv3_block4_3_bn (Batch  (None, 28, 28, 5  2048    ['conv3_block4_3_conv[0][0]
##  Normalization)            12)                       ']
##
##  conv3_block4_add (Add)    (None, 28, 28, 5  0        ['conv3_block3_out[0][0]',
##                            12)                        'conv3_block4_3_bn[0][0]']
##
##  conv3_block4_out (Activa  (None, 28, 28, 5  0       ['conv3_block4_add[0][0]']
##  tion)                     12)
##
##  conv4_block1_1_conv (Con  (None, 14, 14, 2  131328  ['conv3_block4_out[0][0]']
##  v2D)                      56)
##
##  conv4_block1_1_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block1_1_conv[0][0]
##  Normalization)            56)                       ']
##
##  conv4_block1_1_relu (Act  (None, 14, 14, 2  0       ['conv4_block1_1_bn[0][0]']
##  ivation)                  56)
##
##  conv4_block1_2_conv (Con  (None, 14, 14, 2  590080  ['conv4_block1_1_relu[0][0]
##  v2D)                      56)                       ']
##
##  conv4_block1_2_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block1_2_conv[0][0]
##  Normalization)            56)                       ']
##
##  conv4_block1_2_relu (Act  (None, 14, 14, 2  0       ['conv4_block1_2_bn[0][0]']
##  ivation)                  56)
##
##  conv4_block1_0_conv (Con  (None, 14, 14, 1  525312  ['conv3_block4_out[0][0]']
##  v2D)                      024)
##
##  conv4_block1_3_conv (Con  (None, 14, 14, 1  263168  ['conv4_block1_2_relu[0][0]
##  v2D)                      024)                      ']
##
##  conv4_block1_0_bn (Batch  (None, 14, 14, 1  4096    ['conv4_block1_0_conv[0][0]
##  Normalization)            024)                      ']
##
##  conv4_block1_3_bn (Batch  (None, 14, 14, 1  4096    ['conv4_block1_3_conv[0][0]
##  Normalization)            024)                      ']
##
```

```
##  conv4_block1_add (Add)   (None, 14, 14, 1  0        ['conv4_block1_0_bn[0][0]',
##                           024)                        'conv4_block1_3_bn[0][0]']
##
##  conv4_block1_out (Activa  (None, 14, 14, 1  0       ['conv4_block1_add[0][0]']
##  tion)                    024)
##
##  conv4_block2_1_conv (Con  (None, 14, 14, 2  262400  ['conv4_block1_out[0][0]']
##  v2D)                     56)
##
##  conv4_block2_1_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block2_1_conv[0][0]
##  Normalization)           56)                        ']
##
##  conv4_block2_1_relu (Act  (None, 14, 14, 2  0       ['conv4_block2_1_bn[0][0]']
##  ivation)                 56)
##
##  conv4_block2_2_conv (Con  (None, 14, 14, 2  590080  ['conv4_block2_1_relu[0][0]
##  v2D)                     56)                        ']
##
##  conv4_block2_2_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block2_2_conv[0][0]
##  Normalization)           56)                        ']
##
##  conv4_block2_2_relu (Act  (None, 14, 14, 2  0       ['conv4_block2_2_bn[0][0]']
##  ivation)                 56)
##
##  conv4_block2_3_conv (Con  (None, 14, 14, 1  263168  ['conv4_block2_2_relu[0][0]
##  v2D)                     024)                       ']
##
##  conv4_block2_3_bn (Batch  (None, 14, 14, 1  4096    ['conv4_block2_3_conv[0][0]
##  Normalization)           024)                       ']
##
##  conv4_block2_add (Add)   (None, 14, 14, 1  0        ['conv4_block1_out[0][0]',
##                           024)                        'conv4_block2_3_bn[0][0]']
##
##  conv4_block2_out (Activa  (None, 14, 14, 1  0       ['conv4_block2_add[0][0]']
##  tion)                    024)
##
##  conv4_block3_1_conv (Con  (None, 14, 14, 2  262400  ['conv4_block2_out[0][0]']
##  v2D)                     56)
##
##  conv4_block3_1_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block3_1_conv[0][0]
##  Normalization)           56)                        ']
##
##  conv4_block3_1_relu (Act  (None, 14, 14, 2  0       ['conv4_block3_1_bn[0][0]']
##  ivation)                 56)
##
##  conv4_block3_2_conv (Con  (None, 14, 14, 2  590080  ['conv4_block3_1_relu[0][0]
##  v2D)                     56)                        ']
##
##  conv4_block3_2_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block3_2_conv[0][0]
##  Normalization)           56)                        ']
##
##  conv4_block3_2_relu (Act  (None, 14, 14, 2  0       ['conv4_block3_2_bn[0][0]']
##  ivation)                 56)
##
```

```
##  conv4_block3_3_conv (Con  (None, 14, 14, 1  263168  ['conv4_block3_2_relu[0][0]
##  v2D)                      024)                      ']
##
##  conv4_block3_3_bn (Batch  (None, 14, 14, 1  4096    ['conv4_block3_3_conv[0][0]
##  Normalization)            024)                      ']
##
##  conv4_block3_add (Add)    (None, 14, 14, 1  0       ['conv4_block2_out[0][0]',
##                            024)                       'conv4_block3_3_bn[0][0]']
##
##  conv4_block3_out (Activa  (None, 14, 14, 1  0       ['conv4_block3_add[0][0]']
##  tion)                     024)
##
##  conv4_block4_1_conv (Con  (None, 14, 14, 2  262400  ['conv4_block3_out[0][0]']
##  v2D)                      56)
##
##  conv4_block4_1_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block4_1_conv[0][0]
##  Normalization)            56)                       ']
##
##  conv4_block4_1_relu (Act  (None, 14, 14, 2  0       ['conv4_block4_1_bn[0][0]']
##  ivation)                  56)
##
##  conv4_block4_2_conv (Con  (None, 14, 14, 2  590080  ['conv4_block4_1_relu[0][0]
##  v2D)                      56)                       ']
##
##  conv4_block4_2_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block4_2_conv[0][0]
##  Normalization)            56)                       ']
##
##  conv4_block4_2_relu (Act  (None, 14, 14, 2  0       ['conv4_block4_2_bn[0][0]']
##  ivation)                  56)
##
##  conv4_block4_3_conv (Con  (None, 14, 14, 1  263168  ['conv4_block4_2_relu[0][0]
##  v2D)                      024)                      ']
##
##  conv4_block4_3_bn (Batch  (None, 14, 14, 1  4096    ['conv4_block4_3_conv[0][0]
##  Normalization)            024)                      ']
##
##  conv4_block4_add (Add)    (None, 14, 14, 1  0       ['conv4_block3_out[0][0]',
##                            024)                       'conv4_block4_3_bn[0][0]']
##
##  conv4_block4_out (Activa  (None, 14, 14, 1  0       ['conv4_block4_add[0][0]']
##  tion)                     024)
##
##  conv4_block5_1_conv (Con  (None, 14, 14, 2  262400  ['conv4_block4_out[0][0]']
##  v2D)                      56)
##
##  conv4_block5_1_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block5_1_conv[0][0]
##  Normalization)            56)                       ']
##
##  conv4_block5_1_relu (Act  (None, 14, 14, 2  0       ['conv4_block5_1_bn[0][0]']
##  ivation)                  56)
##
##  conv4_block5_2_conv (Con  (None, 14, 14, 2  590080  ['conv4_block5_1_relu[0][0]
##  v2D)                      56)                       ']
##
```

```
##  conv4_block5_2_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block5_2_conv[0][0]
##  Normalization)           56)                        ']
##
##  conv4_block5_2_relu (Act  (None, 14, 14, 2  0       ['conv4_block5_2_bn[0][0]']
##  ivation)                 56)
##
##  conv4_block5_3_conv (Con  (None, 14, 14, 1  263168  ['conv4_block5_2_relu[0][0]
##  v2D)                     024)                       ']
##
##  conv4_block5_3_bn (Batch  (None, 14, 14, 1  4096    ['conv4_block5_3_conv[0][0]
##  Normalization)           024)                       ']
##
##  conv4_block5_add (Add)   (None, 14, 14, 1  0        ['conv4_block4_out[0][0]',
##                           024)                        'conv4_block5_3_bn[0][0]']
##
##  conv4_block5_out (Activa  (None, 14, 14, 1  0       ['conv4_block5_add[0][0]']
##  tion)                    024)
##
##  conv4_block6_1_conv (Con  (None, 14, 14, 2  262400  ['conv4_block5_out[0][0]']
##  v2D)                     56)
##
##  conv4_block6_1_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block6_1_conv[0][0]
##  Normalization)           56)                        ']
##
##  conv4_block6_1_relu (Act  (None, 14, 14, 2  0       ['conv4_block6_1_bn[0][0]']
##  ivation)                 56)
##
##  conv4_block6_2_conv (Con  (None, 14, 14, 2  590080  ['conv4_block6_1_relu[0][0]
##  v2D)                     56)                        ']
##
##  conv4_block6_2_bn (Batch  (None, 14, 14, 2  1024    ['conv4_block6_2_conv[0][0]
##  Normalization)           56)                        ']
##
##  conv4_block6_2_relu (Act  (None, 14, 14, 2  0       ['conv4_block6_2_bn[0][0]']
##  ivation)                 56)
##
##  conv4_block6_3_conv (Con  (None, 14, 14, 1  263168  ['conv4_block6_2_relu[0][0]
##  v2D)                     024)                       ']
##
##  conv4_block6_3_bn (Batch  (None, 14, 14, 1  4096    ['conv4_block6_3_conv[0][0]
##  Normalization)           024)                       ']
##
##  conv4_block6_add (Add)   (None, 14, 14, 1  0        ['conv4_block5_out[0][0]',
##                           024)                        'conv4_block6_3_bn[0][0]']
##
##  conv4_block6_out (Activa  (None, 14, 14, 1  0       ['conv4_block6_add[0][0]']
##  tion)                    024)
##
##  conv5_block1_1_conv (Con  (None, 7, 7, 512  524800  ['conv4_block6_out[0][0]']
##  v2D)                     )
##
##  conv5_block1_1_bn (Batch  (None, 7, 7, 512  2048    ['conv5_block1_1_conv[0][0]
##  Normalization)           )                          ']
##
```

```
##  conv5_block1_1_relu (Act  (None, 7, 7, 512  0       ['conv5_block1_1_bn[0][0]']
##  ivation)                 )
##
##  conv5_block1_2_conv (Con  (None, 7, 7, 512  2359808  ['conv5_block1_1_relu[0][0]
##  v2D)                     )                          ']
##
##  conv5_block1_2_bn (Batch  (None, 7, 7, 512  2048    ['conv5_block1_2_conv[0][0]
##  Normalization)           )                          ']
##
##  conv5_block1_2_relu (Act  (None, 7, 7, 512  0       ['conv5_block1_2_bn[0][0]']
##  ivation)                 )
##
##  conv5_block1_0_conv (Con  (None, 7, 7, 204  2099200  ['conv4_block6_out[0][0]']
##  v2D)                     8)
##
##  conv5_block1_3_conv (Con  (None, 7, 7, 204  1050624  ['conv5_block1_2_relu[0][0]
##  v2D)                     8)                         ']
##
##  conv5_block1_0_bn (Batch  (None, 7, 7, 204  8192    ['conv5_block1_0_conv[0][0]
##  Normalization)           8)                         ']
##
##  conv5_block1_3_bn (Batch  (None, 7, 7, 204  8192    ['conv5_block1_3_conv[0][0]
##  Normalization)           8)                         ']
##
##  conv5_block1_add (Add)   (None, 7, 7, 204  0        ['conv5_block1_0_bn[0][0]',
##                           8)                         'conv5_block1_3_bn[0][0]']
##
##  conv5_block1_out (Activa  (None, 7, 7, 204  0       ['conv5_block1_add[0][0]']
##  tion)                    8)
##
##  conv5_block2_1_conv (Con  (None, 7, 7, 512  1049088  ['conv5_block1_out[0][0]']
##  v2D)                     )
##
##  conv5_block2_1_bn (Batch  (None, 7, 7, 512  2048    ['conv5_block2_1_conv[0][0]
##  Normalization)           )                          ']
##
##  conv5_block2_1_relu (Act  (None, 7, 7, 512  0       ['conv5_block2_1_bn[0][0]']
##  ivation)                 )
##
##  conv5_block2_2_conv (Con  (None, 7, 7, 512  2359808  ['conv5_block2_1_relu[0][0]
##  v2D)                     )                          ']
##
##  conv5_block2_2_bn (Batch  (None, 7, 7, 512  2048    ['conv5_block2_2_conv[0][0]
##  Normalization)           )                          ']
##
##  conv5_block2_2_relu (Act  (None, 7, 7, 512  0       ['conv5_block2_2_bn[0][0]']
##  ivation)                 )
##
##  conv5_block2_3_conv (Con  (None, 7, 7, 204  1050624  ['conv5_block2_2_relu[0][0]
##  v2D)                     8)                         ']
##
##  conv5_block2_3_bn (Batch  (None, 7, 7, 204  8192    ['conv5_block2_3_conv[0][0]
##  Normalization)           8)                         ']
##
```

```
##  conv5_block2_add (Add)   (None, 7, 7, 204  0        ['conv5_block1_out[0][0]',
##                           8)                          'conv5_block2_3_bn[0][0]']
##
##  conv5_block2_out (Activa  (None, 7, 7, 204  0       ['conv5_block2_add[0][0]']
##  tion)                    8)
##
##  conv5_block3_1_conv (Con  (None, 7, 7, 512  1049088  ['conv5_block2_out[0][0]']
##  v2D)                     )
##
##  conv5_block3_1_bn (Batch  (None, 7, 7, 512  2048    ['conv5_block3_1_conv[0][0]
##  Normalization)           )                          ']
##
##  conv5_block3_1_relu (Act  (None, 7, 7, 512  0       ['conv5_block3_1_bn[0][0]']
##  ivation)                 )
##
##  conv5_block3_2_conv (Con  (None, 7, 7, 512  2359808  ['conv5_block3_1_relu[0][0]
##  v2D)                     )                          ']
##
##  conv5_block3_2_bn (Batch  (None, 7, 7, 512  2048    ['conv5_block3_2_conv[0][0]
##  Normalization)           )                          ']
##
##  conv5_block3_2_relu (Act  (None, 7, 7, 512  0       ['conv5_block3_2_bn[0][0]']
##  ivation)                 )
##
##  conv5_block3_3_conv (Con  (None, 7, 7, 204  1050624  ['conv5_block3_2_relu[0][0]
##  v2D)                     8)                         ']
##
##  conv5_block3_3_bn (Batch  (None, 7, 7, 204  8192    ['conv5_block3_3_conv[0][0]
##  Normalization)           8)                         ']
##
##  conv5_block3_add (Add)   (None, 7, 7, 204  0        ['conv5_block2_out[0][0]',
##                           8)                          'conv5_block3_3_bn[0][0]']
##
##  conv5_block3_out (Activa  (None, 7, 7, 204  0       ['conv5_block3_add[0][0]']
##  tion)                    8)
##
##  avg_pool (GlobalAverageP  (None, 2048)     0        ['conv5_block3_out[0][0]']
##  ooling2D)
##
##  predictions (Dense)      (None, 1000)      2049000  ['avg_pool[0][0]']
##
## ==================================================================================
## Total params: 25,636,712
## Trainable params: 25,583,592
## Non-trainable params: 53,120
## ----------------------------------------------------------------------------------
```

1

Downloads the CIFAR-10 image set and saves the dog and frog photos in the appropriate directories. (This can be lifted verbatim from the code I used. Once you've used it once, you can comment it out or otherwise set it not to run so that you're no re-creating the directories more than once.)

```
cifar <- dataset_cifar10()
cifar$train$x <- cifar$train$x[cifar$train$y %in% c(5, 6), , , ]/255
cifar$train$y <- cifar$train$y[cifar$train$y %in% c(5, 6)]
cifar$train$y <- factor(if_else(cifar$train$y == 5, "Dog", "Frog"))
cifar$test$x <- cifar$test$x[cifar$test$y %in% c(5, 6), , , ]/255
cifar$test$y <- cifar$test$y[cifar$test$y %in% c(5, 6)]
cifar$test$y <- factor(if_else(cifar$test$y == 5, "Dog", "Frog"))
```

```
par(mfrow=c(1,2))
plot.new(); plot.window(xlim=c(0,1), ylim=c(0,1), asp=1)
rasterImage(cifar$train$x[1, , , ], 0, 0, 1, 1)
text(x=0.5, y=0.1, label=cifar$train$y[1], cex=2, col="white")
plot.new(); plot.window(xlim=c(0,1), ylim=c(0,1), asp=1)
rasterImage(cifar$train$x[7, , , ], 0, 0, 1, 1)
text(x=0.5, y=0.1, label=cifar$train$y[7], cex=2, col="white")
```

```
#dir.create("Train")
#dir.create("Test")
#for(lab in levels(cifar$train$y)){
#  dir.create(paste0("Train/", lab))
#  dir.create(paste0("Test/", lab))
#}
#for(i in 1:dim(cifar$train$x)[1]){
#  png::writePNG(cifar$train$x[i, , , ],
#                paste0("Train/", cifar$train$y[i], "/img", i, ".png"))
#}
#for(i in 1:dim(cifar$test$x)[1]){
#  png::writePNG(cifar$test$x[i, , , ],
#                paste0("Test/", cifar$test$y[i], "/img", i, ".png"))
#}
#rm(cifar)
```

```
training_image_gen <- image_data_generator(
  width_shift_range = 0.2,
  height_shift_range = 0.2,
  # I really want to try to train on the serif/sans details, so the next two
  # ensure that letters are in a variety of orientations.
  rotation_range = 20,
  horizontal_flip = TRUE,
  validation_split=0.2
)
training_image_flow <- flow_images_from_directory(
  directory = "Train/",
  generator = training_image_gen,
  subset = "training",
  class_mode = "binary",
  batch_size = 20,
  target_size = c(224, 224),
  # Randomizing inputs is important so that the NN doesn't get stuck in a rut.
  # It's the default, but I've put it in explicitly.
  shuffle = TRUE
)
# A separate directory and generator could be used with a fixed validation set.
```

```
validation_image_flow <- flow_images_from_directory(
  directory = "Train/",
  generator = training_image_gen,
  subset = "validation",
  class_mode = "binary",
  batch_size = 20,
  target_size = c(224, 224),
  shuffle = FALSE
)
```

2

Train a regular (non-convolutional) network on the image data for at least 30 epochs (you can go farther if you want to–how high does validation accuracy go if you've got a lot of time?).

```
freeze_weights(model)

dfmodel <- keras_model_sequential() %>%
  # Pre-built resnet50 model.
  model %>%
  # My additions to find dogs and frogs.
  layer_dense(units=256, activation="relu") %>%
  layer_dropout(rate=0.2) %>%
  layer_dense(units=1, activation="sigmoid") %>%
  compile(
    loss = "binary_crossentropy",
    optimizer = "adam",
    metrics = "accuracy")

dfmodel %>% fit(
  x=training_image_flow,
  validation_data=validation_image_flow,
  epochs=2
)

#accuracy from 77.4% to 81.92%

#1153 seconds

pred <- model %>% predict(validation_image_flow)
pred.classes <- bind_rows(imagenet_decode_predictions(pred, top=1))
table(pred.classes$class_description)
```

```
##
##            affenpinscher                ambulance                    Angora
##                       10                        2                         5
##               Arctic_fox                   ashcan       Australian_terrier
##                        2                        1                         1
##                   badger                bannister                   beaker
##                        3                        2                         1
##     Bernese_mountain_dog         Blenheim_spaniel           Border_terrier
##                        1                        3                         6
##                   borzoi              Boston_bull     Bouvier_des_Flandres
##                        2                        1                        12
##                      cab                   candle              car_mirror
##                      229                        1                         2
##             cash_machine Chesapeake_Bay_retriever          cocktail_shaker
##                        6                        2                         8
##           container_ship          crossword_puzzle           Dandie_Dinmont
##                        2                        8                         1
##            digital_clock             dining_table                  doormat
##                       25                        5                         1
##                   dugong             electric_ray           English_setter
```

```
##                     3                     8                     2
##        English_springer              envelope          fire_screen
##                     4                     3                     6
##          folding_chair             forklift           frying_pan
##                     2                     3                     1
##               gasmask               geyser          giant_panda
##                     1                     2                     1
##             golf_ball                  gong        Great_Pyrenees
##                     2                     5                     3
##            guillotine            hair_slide                hamper
##                     3                     4                     1
##          hand_blower                  hare               hatchet
##                     1                     2                     1
##                   hog       Japanese_spaniel              joystick
##                     1                     2                     1
##             keeshond                kelpie                kuvasz
##                     5                     1                     4
##             lacewing                 ladle       Lakeland_terrier
##                     1                     2                     1
##            lampshade               library               lighter
##                     8                    14                    20
##          loudspeaker          Maltese_dog                maraca
##                     3                    29                     1
##                  maze            microphone              milk_can
##                    15                     1                     3
##          mobile_home               monitor          mosquito_net
##                     1                     8                    72
##       mountain_bike              nematode    Old_English_sheepdog
##                     1                     4                    16
##          oxygen_mask              papillon          parallel_bars
##                    22                     3                     1
##                patas              Pekinese            Persian_cat
##                     1                     1                     1
##          photocopier           picket_fence               pinwheel
##                    35                    12                     2
##             platypus             Pomeranian            projectile
##                    15                     6                     3
##            projector                puffer                   pug
##                    26                    10                     3
##             radiator          rocking_chair                Samoyed
##                     7                     1                     3
##                  sax             schipperke        Scotch_terrier
##                     2                     2                     2
##     Scottish_deerhound                screen                 screw
##                     3                     2                     1
##           screwdriver                shield              Shih-Tzu
##                     3                     8                     1
##                shovel         shower_curtain         Siberian_husky
##                     1                   367                     3
##          silky_terrier                 skunk           sliding_door
##                     1                     5                    11
##               snorkel              snowplow         soap_dispenser
##                     1                     2                     2
##            solar_dish          space_shuttle             spotlight
```

```
##                   281                     2                    82
##                 stage           steel_drum              strainer
##                     1                     1                    18
##           street_sign         Sussex_spaniel                 swing
##                    20                     1                     1
##               syringe            television       three-toed_sloth
##                     1                    99                     1
##           tiger_shark                  titi               toaster
##                     2                     1                     1
##           toilet_seat         traffic_light                  tray
##                     9                     3                     1
##             turnstile                vacuum                  vase
##                     7                     1                    14
##                 vault             washbasin            Weimaraner
##                     1                     3                     1
##  Welsh_springer_spaniel               whistle         window_screen
##                     1                     9                   230
##          window_shade            worm_fence                 wreck
##                    20                     1                     4
```

4 and 5

Train a convolutional neural network on the same data set for at least 30 epochs (again, more if you want). Plot validation accuracy and evaluate accuracy on the test set for this model.

```
dfnn <- keras_model_sequential() %>%
  # Note that layer_dense expects a "flat" vector, not an image, so we add
  # layer_flatten first.
  layer_flatten() %>%
  layer_dense(units=256, activation="relu", input_shape=c(224, 224, 3)) %>%
  layer_dropout(rate=0.2) %>%
  layer_dense(units=1, activation="sigmoid") %>%
  compile(
    loss = "binary_crossentropy",
    optimizer = "adam",
    metrics = "accuracy")

dfnn %>% fit(
  x=training_image_flow,
  validation_data=validation_image_flow,
  epochs=15
)
#1712 second with 15 epochs
#max accuracy of 50.15%
```

6

Write a paragraph, along with a table of results, that compares how well each of these models did in accurately predicting dog or frog. The table should have columns for accuracy, number of epochs and time it took to run on your computer. (Time can be can just be added up by hand from the on-screen output of the fit command.)

```
Title <- c("Accuracy", "Epochs", "Seconds", "Minutes/Epoch")
RegularNN <- c(81.92, 2, 1153, round(1153/60/2))
Convolutional <- c(51.15, 15, 1712, round(1712/60/2))


results <- data.frame(Title, RegularNN, Convolutional)
kable(results)
```

Title	RegularNN	Convolutional
Accuracy	81.92	51.15
Epochs	2.00	15.00
Seconds	1153.00	1712.00
Minutes/Epoch	10.00	14.00

A regular neural network outperformed a convolutional neural network by all measurements (timeliness and accuracy). This is unsurprising as the book this problem comes from notes a 46% accuracy. Fine tuning the CNN model can increase the accuracy to 75%, which is still less than a non-convolutional neural network and requires much effort and time.

A 30-epoch assessment would take 7 hours for the CNN to process and 6 hours for the non-convolutional network to run. Given time considerations, I am concluding this prior to running more in-depth analysis.