Lab Assignment #11

Nick Noel, Liz Villa, and Cadee Pinkerton

Due Sometime After Midterm 2

Instructions

The purpose of this lab is to introduce neural networks using the `keras` package. In lecture we saw a single-hidden-layer model, but more complicated neural networks (such as CNNs, deep learning, etc.) are usually custom-built using the keras interface.

```
library(ISLR2)
library(dplyr)
library(ggplot2)
library(keras)
```

This lab assignment is worth a total of **10 points**.

Problem 1: Book Code

Part a (Code: 3 pts)

Get `keras` installed on your computer. Then run the example code in Labs 10.9.1, 10.9.2, 10.9.3, and 10.9.4. Notes:

- The first time you try to set up keras, you will have to run `install_keras()` to actually install keras, Tensorflow, and their dependencies. If you do not have Python with Anaconda (or Miniconda) installed already on your device, you may want to follow the instructions in the error messages. If you cannot interpret an error message, please call me over.
- You probably cannot run GPU-based `keras` on your machine and will get a bunch of error messages when you first try to do anything with it. I was able to run the whole lab with CPU-based `keras`.
- In Lab 10.9.2, `predict_classes()` is deprecated and may throw an error. If you cannot figure out how to interpret the error message to get around it, please call me over.
- For Lab 10.9.4, I have found that if I put the `book_images` folder (unzipped) as a subfolder of the directory the lab is in, then everything will work as intended. If you get a "Permission denied" error, then you probably need to change the `img_dir` or move the folder around.
- If all else fails, try running the `torch` version of the lab, which can be found at https://www.statlearning.com/resources-second-edition.

```
Gitters <- na.omit (Hitters)
n <- nrow (Gitters)
set.seed (13)
ntest <- trunc (n / 3)
testid <- sample (1:n, ntest)
```

```
lfit <- lm(Salary ~ ., data = Gitters[-testid , ])
lpred <- predict(lfit , Gitters[testid , ])
with(Gitters[testid, ], mean(abs(lpred - Salary)))
```

```
## [1] 254.6687
```

```
x <- scale ( model.matrix (Salary ~ . - 1, data = Gitters))
y <- Gitters$Salary
```

```
library(glmnet)
```

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

```
## Loading required package: Matrix
```

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

```
## Loaded glmnet 4.1-6
```

```
cvfit <- cv.glmnet(x[-testid , ], y[-testid], type.measure = "mae")
cpred <- predict (cvfit , x[testid , ], s = "lambda.min")
mean ( abs (y[testid] - cpred))
```

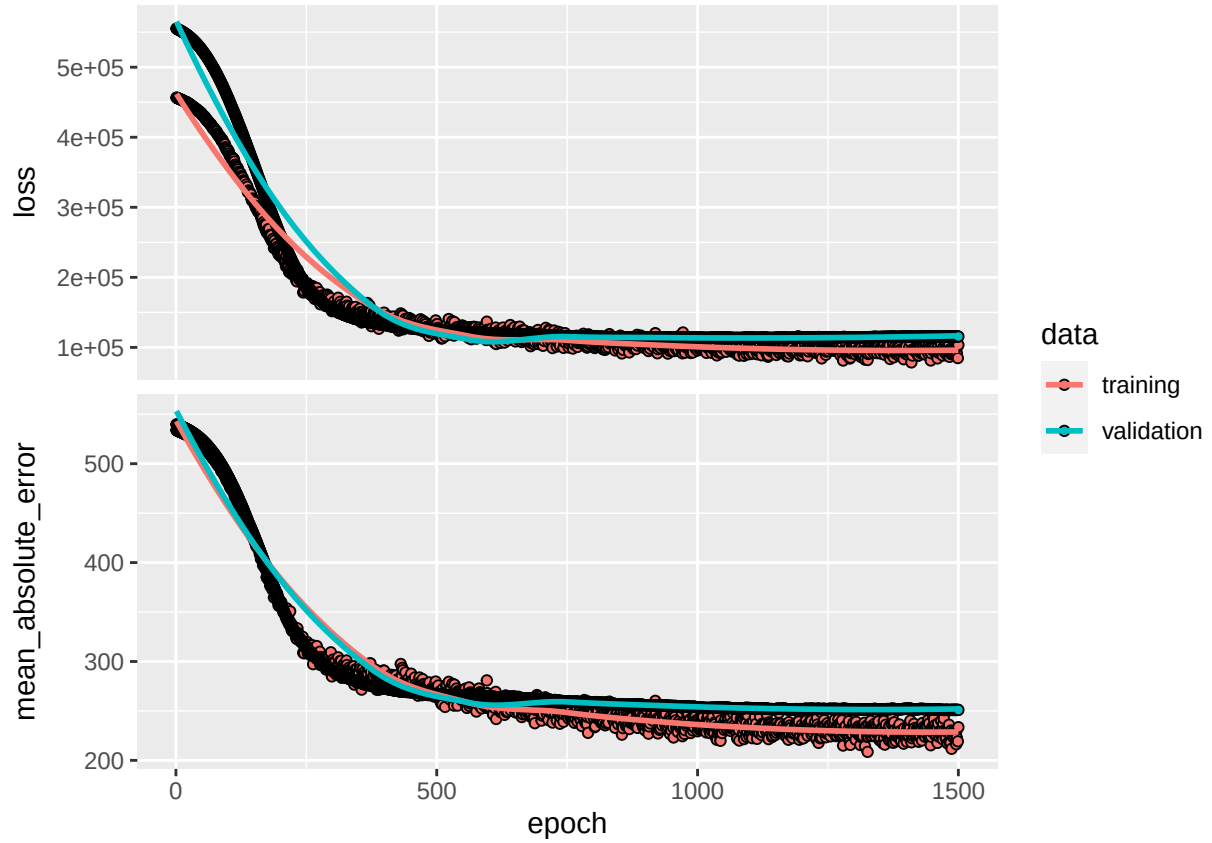
```
## [1] 252.2994
```

```
modnn <- keras_model_sequential () %>%
  layer_dense (units = 50, activation = "relu",input_shape = ncol (x)) %>%
  layer_dropout (rate = 0.4) %>%
  layer_dense (units = 1)
```

```
modnn %>% compile (loss = "mse",
  optimizer = optimizer_rmsprop(),
  metrics = list ("mean_absolute_error"))
```

```
history <- modnn %>% fit (
x[-testid , ], y[-testid], epochs = 1500, batch_size = 32,
validation_data = list (x[testid , ], y[testid]))
```

```
plot(history)
```




```
npred <- predict(modnn , x[testid , ])
mean(abs(y[testid] - npred))
```

```
## [1] 251.0111
```

```
mnist <- dataset_mnist()
x_train <- mnist$train$x
g_train <- mnist$train$y
x_test <- mnist$test$x
g_test <- mnist$test$y
dim (x_train)
```

```
## [1] 60000    28    28
```

```
dim (x_test)
```

```
## [1] 10000    28    28
```

```
x_train <- array_reshape (x_train , c( nrow (x_train), 784))
x_test <- array_reshape (x_test , c( nrow (x_test), 784))
y_train <- to_categorical (g_train , 10)
y_test <- to_categorical (g_test , 10)
```

```
x_train <- x_train / 255
x_test <- x_test / 255
```

```
modelnn <- keras_model_sequential()
modelnn %>%
  layer_dense (units = 256, activation = "relu", input_shape = c (784)) %>%
  layer_dropout (rate = 0.4) %>%
  layer_dense (units = 128, activation = "relu") %>%
  layer_dropout (rate = 0.3) %>%
  layer_dense (units = 10, activation = "softmax")
```

```
summary(modelnn)
```

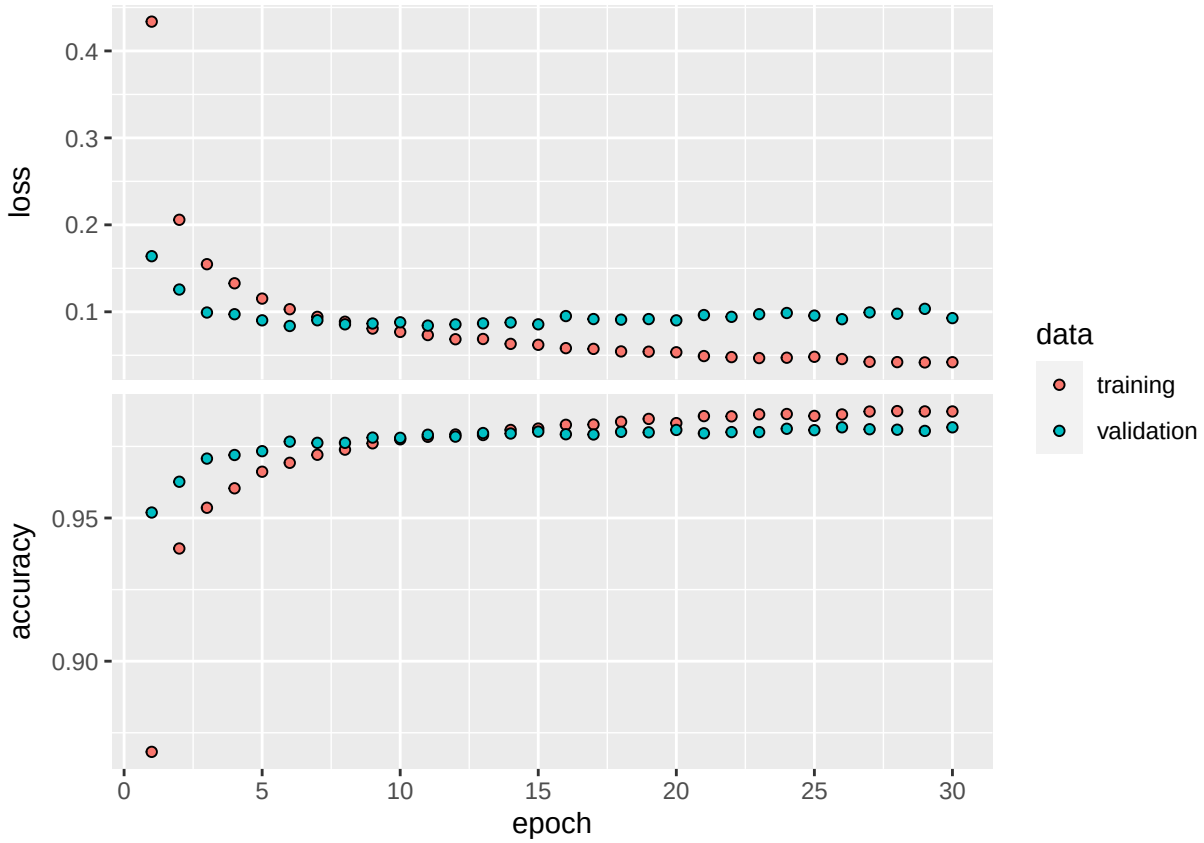
```
## Model: "sequential_1"
## _________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =================================================================================
##  dense_4 (Dense)                    (None, 256)                     200960
##  dropout_2 (Dropout)                (None, 256)                     0
##  dense_3 (Dense)                    (None, 128)                     32896
##  dropout_1 (Dropout)                (None, 128)                     0
##  dense_2 (Dense)                    (None, 10)                      1290
## =================================================================================
## Total params: 235,146
## Trainable params: 235,146
## Non-trainable params: 0
## _________________________________________________________________________________
```

```
modelnn %>% compile (loss = "categorical_crossentropy", optimizer = optimizer_rmsprop(), metrics = c("a
```

```
system.time(
  history <- modelnn %>%
  fit (x_train , y_train , epochs = 30, batch_size = 128, validation_split = 0.2)
  )
```

```
##    user  system elapsed
##  439.73   29.98  148.63
```

```
plot(history, smooth = FALSE)
```




```
accuracy <- function(pred,truth) (
  mean(drop(pred) == drop(truth))
  )
modelnn %>%
  predict(x_test) %>% k_argmax() %>%
  as.numeric() %>%
  accuracy(g_test)
```

```
## [1] 0.9828
```

```
modellr <- keras_model_sequential () %>%
  layer_dense (input_shape = 784, units = 10, activation = "softmax")
summary (modellr)
```

```
## Model: "sequential_2"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                     Param #
## =========================================================================
##  dense_5 (Dense)                    (None, 10)                       7850
## =========================================================================
## Total params: 7,850
## Trainable params: 7,850
## Non-trainable params: 0
## _________________________________________________________________________
```

```
modellr %>% compile(loss = "categorical_crossentropy", optimizer = optimizer_rmsprop(), metrics = c("ac
modellr %>% fit (x_train , y_train , epochs = 30, batch_size = 128, validation_split = 0.2)
```

```
modellr %>%
  predict(x_test) %>% k_argmax() %>%
  as.numeric() %>%
  accuracy(g_test)
```

```
## [1] 0.9274
```

```
cifar100 <- dataset_cifar100()
names (cifar100)
```

```
## [1] "train" "test"
```

```
x_train <- cifar100$train$x
g_train <- cifar100$train$y
x_test <- cifar100$test$x
g_test <- cifar100$test$y
dim (x_train)
```

```
## [1] 50000    32    32     3
```

```
range (x_train[1,,, 1])
```

```
## [1]  13 255
```

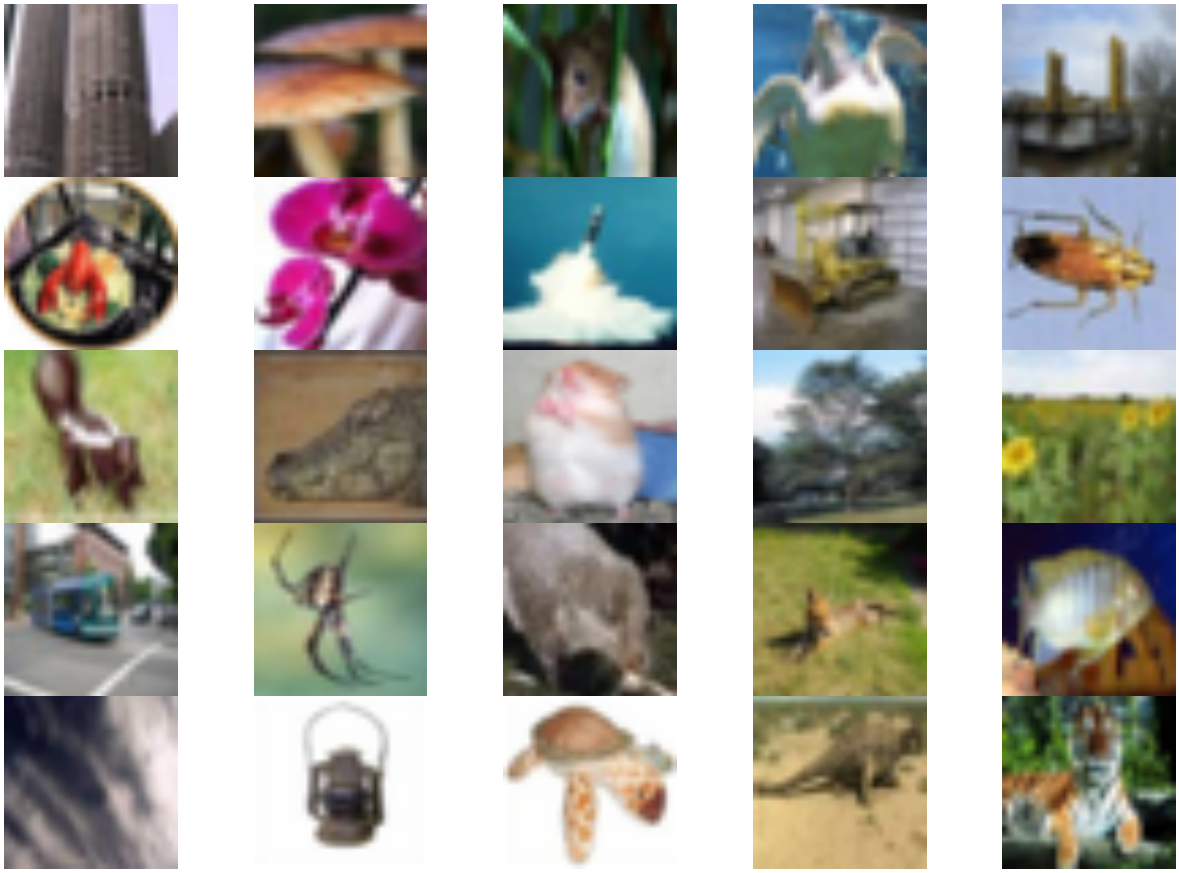
```
x_train <- x_train / 255
x_test <- x_test / 255
y_train <- to_categorical (g_train , 100)
dim (y_train)
```

```
## [1] 50000   100
```

```
library(jpeg)
```

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

```
par (mar = c(0, 0, 0, 0), mfrow = c(5, 5))
index <- sample ( seq(50000), 25)
for (i in index) plot (as.raster (x_train[i,,, ]))
```



```
model <- keras_model_sequential () %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3), padding = "same", activation = "relu", input_shape
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3, 3),padding = "same", activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3),padding = "same", activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 256, kernel_size = c(3, 3),padding = "same", activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense (units = 512, activation = "relu") %>%
  layer_dense (units = 100, activation = "softmax")
summary (model)
```

```
## Model: "sequential_3"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  conv2d_3 (Conv2D)                  (None, 32, 32, 32)              896
##  max_pooling2d_3 (MaxPooling2D)     (None, 16, 16, 32)              0
##  conv2d_2 (Conv2D)                  (None, 16, 16, 64)              18496
##  max_pooling2d_2 (MaxPooling2D)     (None, 8, 8, 64)                0
##  conv2d_1 (Conv2D)                  (None, 8, 8, 128)               73856
##  max_pooling2d_1 (MaxPooling2D)     (None, 4, 4, 128)               0
##  conv2d (Conv2D)                    (None, 4, 4, 256)               295168
```

```
##  max_pooling2d (MaxPooling2D)       (None, 2, 2, 256)              0
##  flatten (Flatten)                  (None, 1024)                   0
##  dropout_3 (Dropout)                (None, 1024)                   0
##  dense_7 (Dense)                    (None, 512)                    524800
##  dense_6 (Dense)                    (None, 100)                    51300
## ================================================================================
## Total params: 964,516
## Trainable params: 964,516
## Non-trainable params: 0
## --------------------------------------------------------------------------------
```

```
model %>% compile (loss = "categorical_crossentropy", optimizer = optimizer_rmsprop(), metrics = c("acc
history <- model %>% fit (x_train , y_train , epochs = 30, batch_size = 128, validation_split = 0.2)
model %>% predict(x_test) %>% k_argmax() %>%
  as.numeric() %>%
  accuracy(g_test)
```

```
## [1] 0.4476
```

```
img_dir <- "book_images"
image_names <- list.files(img_dir)
num_images <- length(image_names)
x <- array(dim = c(num_images , 224, 224, 3))
for (i in 1:num_images) {
  img_path <- paste (img_dir , image_names[i], sep = "/")
  img <- image_load (img_path, target_size = c(224, 224))
  x[i,,, ] <- image_to_array(img)
}
x <- imagenet_preprocess_input(x)
```

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

```
## Model: "resnet50"
## ________________________________________________________________________________
##  Layer (type)         Output Shape   Param #  Connected to           Trainable
## ================================================================================
##  input_1 (InputLayer)  [(None, 224,   0       []                     Y
##                       224, 3)]
##  conv1_pad (ZeroPaddi  (None, 230, 2  0       ['input_1[0][0]']      Y
##  ng2D)                30, 3)
##  conv1_conv (Conv2D)  (None, 112, 1  9472     ['conv1_pad[0][0]']    Y
##                       12, 64)
##  conv1_bn (BatchNorma  (None, 112, 1  256     ['conv1_conv[0][0]']   Y
##  lization)            12, 64)
##  conv1_relu (Activati  (None, 112, 1  0       ['conv1_bn[0][0]']     Y
##  on)                  12, 64)
##  pool1_pad (ZeroPaddi  (None, 114, 1  0       ['conv1_relu[0][0]']   Y
##  ng2D)                14, 64)
##  pool1_pool (MaxPooli  (None, 56, 56  0       ['pool1_pad[0][0]']    Y
##  ng2D)                , 64)
##  conv2_block1_1_conv   (None, 56, 56  4160    ['pool1_pool[0][0]']   Y
##  (Conv2D)             , 64)
##  conv2_block1_1_bn (B  (None, 56, 56  256     ['conv2_block1_1_conv  Y
##  atchNormalization)   , 64)                   [0][0]']
##  conv2_block1_1_relu   (None, 56, 56  0       ['conv2_block1_1_bn[0  Y
```

```
##  (Activation)          , 64)                    ][0]']
##  conv2_block1_2_conv   (None, 56, 56  36928    ['conv2_block1_1_relu  Y
##  (Conv2D)              , 64)                    [0][0]']
##  conv2_block1_2_bn (B  (None, 56, 56  256      ['conv2_block1_2_conv  Y
##  atchNormalization)    , 64)                    [0][0]']
##  conv2_block1_2_relu   (None, 56, 56  0        ['conv2_block1_2_bn[0  Y
##  (Activation)          , 64)                    ][0]']
##  conv2_block1_0_conv   (None, 56, 56  16640    ['pool1_pool[0][0]']   Y
##  (Conv2D)              , 256)
##  conv2_block1_3_conv   (None, 56, 56  16640    ['conv2_block1_2_relu  Y
##  (Conv2D)              , 256)                   [0][0]']
##  conv2_block1_0_bn (B  (None, 56, 56  1024     ['conv2_block1_0_conv  Y
##  atchNormalization)    , 256)                   [0][0]']
##  conv2_block1_3_bn (B  (None, 56, 56  1024     ['conv2_block1_3_conv  Y
##  atchNormalization)    , 256)                   [0][0]']
##  conv2_block1_add (Ad  (None, 56, 56  0        ['conv2_block1_0_bn[0  Y
##  d)                    , 256)                   ][0]',
##                                                  'conv2_block1_3_bn[0
##                                                 ][0]']
##  conv2_block1_out (Ac  (None, 56, 56  0        ['conv2_block1_add[0]  Y
##  tivation)             , 256)                   [0]']
##  conv2_block2_1_conv   (None, 56, 56  16448    ['conv2_block1_out[0]  Y
##  (Conv2D)              , 64)                    [0]']
##  conv2_block2_1_bn (B  (None, 56, 56  256      ['conv2_block2_1_conv  Y
##  atchNormalization)    , 64)                    [0][0]']
##  conv2_block2_1_relu   (None, 56, 56  0        ['conv2_block2_1_bn[0  Y
##  (Activation)          , 64)                    ][0]']
##  conv2_block2_2_conv   (None, 56, 56  36928    ['conv2_block2_1_relu  Y
##  (Conv2D)              , 64)                    [0][0]']
##  conv2_block2_2_bn (B  (None, 56, 56  256      ['conv2_block2_2_conv  Y
##  atchNormalization)    , 64)                    [0][0]']
##  conv2_block2_2_relu   (None, 56, 56  0        ['conv2_block2_2_bn[0  Y
##  (Activation)          , 64)                    ][0]']
##  conv2_block2_3_conv   (None, 56, 56  16640    ['conv2_block2_2_relu  Y
##  (Conv2D)              , 256)                   [0][0]']
##  conv2_block2_3_bn (B  (None, 56, 56  1024     ['conv2_block2_3_conv  Y
##  atchNormalization)    , 256)                   [0][0]']
##  conv2_block2_add (Ad  (None, 56, 56  0        ['conv2_block1_out[0]  Y
##  d)                    , 256)                   [0]',
##                                                  'conv2_block2_3_bn[0
##                                                 ][0]']
##  conv2_block2_out (Ac  (None, 56, 56  0        ['conv2_block2_add[0]  Y
##  tivation)             , 256)                   [0]']
##  conv2_block3_1_conv   (None, 56, 56  16448    ['conv2_block2_out[0]  Y
##  (Conv2D)              , 64)                    [0]']
##  conv2_block3_1_bn (B  (None, 56, 56  256      ['conv2_block3_1_conv  Y
##  atchNormalization)    , 64)                    [0][0]']
##  conv2_block3_1_relu   (None, 56, 56  0        ['conv2_block3_1_bn[0  Y
##  (Activation)          , 64)                    ][0]']
##  conv2_block3_2_conv   (None, 56, 56  36928    ['conv2_block3_1_relu  Y
##  (Conv2D)              , 64)                    [0][0]']
##  conv2_block3_2_bn (B  (None, 56, 56  256      ['conv2_block3_2_conv  Y
##  atchNormalization)    , 64)                    [0][0]']
##  conv2_block3_2_relu   (None, 56, 56  0        ['conv2_block3_2_bn[0  Y
```

```
##  (Activation)         , 64)                    ][0]']
##  conv2_block3_3_conv   (None, 56, 56  16640    ['conv2_block3_2_relu  Y
##  (Conv2D)             , 256)                   [0][0]']
##  conv2_block3_3_bn (B  (None, 56, 56  1024     ['conv2_block3_3_conv  Y
##  atchNormalization)   , 256)                   [0][0]']
##  conv2_block3_add (Ad  (None, 56, 56  0        ['conv2_block2_out[0]  Y
##  d)                   , 256)                   [0]',
##                                                 'conv2_block3_3_bn[0
##                                                ][0]']
##  conv2_block3_out (Ac  (None, 56, 56  0        ['conv2_block3_add[0]  Y
##  tivation)            , 256)                   [0]']
##  conv3_block1_1_conv   (None, 28, 28  32896    ['conv2_block3_out[0]  Y
##  (Conv2D)             , 128)                   [0]']
##  conv3_block1_1_bn (B  (None, 28, 28  512      ['conv3_block1_1_conv  Y
##  atchNormalization)   , 128)                   [0][0]']
##  conv3_block1_1_relu   (None, 28, 28  0        ['conv3_block1_1_bn[0  Y
##  (Activation)         , 128)                   ][0]']
##  conv3_block1_2_conv   (None, 28, 28  147584   ['conv3_block1_1_relu  Y
##  (Conv2D)             , 128)                   [0][0]']
##  conv3_block1_2_bn (B  (None, 28, 28  512      ['conv3_block1_2_conv  Y
##  atchNormalization)   , 128)                   [0][0]']
##  conv3_block1_2_relu   (None, 28, 28  0        ['conv3_block1_2_bn[0  Y
##  (Activation)         , 128)                   ][0]']
##  conv3_block1_0_conv   (None, 28, 28  131584   ['conv2_block3_out[0]  Y
##  (Conv2D)             , 512)                   [0]']
##  conv3_block1_3_conv   (None, 28, 28  66048    ['conv3_block1_2_relu  Y
##  (Conv2D)             , 512)                   [0][0]']
##  conv3_block1_0_bn (B  (None, 28, 28  2048     ['conv3_block1_0_conv  Y
##  atchNormalization)   , 512)                   [0][0]']
##  conv3_block1_3_bn (B  (None, 28, 28  2048     ['conv3_block1_3_conv  Y
##  atchNormalization)   , 512)                   [0][0]']
##  conv3_block1_add (Ad  (None, 28, 28  0        ['conv3_block1_0_bn[0  Y
##  d)                   , 512)                   ][0]',
##                                                 'conv3_block1_3_bn[0
##                                                ][0]']
##  conv3_block1_out (Ac  (None, 28, 28  0        ['conv3_block1_add[0]  Y
##  tivation)            , 512)                   [0]']
##  conv3_block2_1_conv   (None, 28, 28  65664    ['conv3_block1_out[0]  Y
##  (Conv2D)             , 128)                   [0]']
##  conv3_block2_1_bn (B  (None, 28, 28  512      ['conv3_block2_1_conv  Y
##  atchNormalization)   , 128)                   [0][0]']
##  conv3_block2_1_relu   (None, 28, 28  0        ['conv3_block2_1_bn[0  Y
##  (Activation)         , 128)                   ][0]']
##  conv3_block2_2_conv   (None, 28, 28  147584   ['conv3_block2_1_relu  Y
##  (Conv2D)             , 128)                   [0][0]']
##  conv3_block2_2_bn (B  (None, 28, 28  512      ['conv3_block2_2_conv  Y
##  atchNormalization)   , 128)                   [0][0]']
##  conv3_block2_2_relu   (None, 28, 28  0        ['conv3_block2_2_bn[0  Y
##  (Activation)         , 128)                   ][0]']
##  conv3_block2_3_conv   (None, 28, 28  66048    ['conv3_block2_2_relu  Y
##  (Conv2D)             , 512)                   [0][0]']
##  conv3_block2_3_bn (B  (None, 28, 28  2048     ['conv3_block2_3_conv  Y
##  atchNormalization)   , 512)                   [0][0]']
##  conv3_block2_add (Ad  (None, 28, 28  0        ['conv3_block1_out[0]  Y
```

```
##  d)                  , 512)                 [0]',
##                                              'conv3_block2_3_bn[0
##                                             ][0]']
##  conv3_block2_out (Ac  (None, 28, 28  0      ['conv3_block2_add[0]  Y
##  tivation)           , 512)                  [0]']
##  conv3_block3_1_conv   (None, 28, 28  65664  ['conv3_block2_out[0]  Y
##  (Conv2D)            , 128)                  [0]']
##  conv3_block3_1_bn (B  (None, 28, 28  512    ['conv3_block3_1_conv  Y
##  atchNormalization)  , 128)                  [0][0]']
##  conv3_block3_1_relu   (None, 28, 28  0      ['conv3_block3_1_bn[0  Y
##  (Activation)        , 128)                  ][0]']
##  conv3_block3_2_conv   (None, 28, 28  147584 ['conv3_block3_1_relu  Y
##  (Conv2D)            , 128)                  [0][0]']
##  conv3_block3_2_bn (B  (None, 28, 28  512    ['conv3_block3_2_conv  Y
##  atchNormalization)  , 128)                  [0][0]']
##  conv3_block3_2_relu   (None, 28, 28  0      ['conv3_block3_2_bn[0  Y
##  (Activation)        , 128)                  ][0]']
##  conv3_block3_3_conv   (None, 28, 28  66048  ['conv3_block3_2_relu  Y
##  (Conv2D)            , 512)                  [0][0]']
##  conv3_block3_3_bn (B  (None, 28, 28  2048   ['conv3_block3_3_conv  Y
##  atchNormalization)  , 512)                  [0][0]']
##  conv3_block3_add (Ad  (None, 28, 28  0      ['conv3_block2_out[0]  Y
##  d)                  , 512)                  [0]',
##                                              'conv3_block3_3_bn[0
##                                             ][0]']
##  conv3_block3_out (Ac  (None, 28, 28  0      ['conv3_block3_add[0]  Y
##  tivation)           , 512)                  [0]']
##  conv3_block4_1_conv   (None, 28, 28  65664  ['conv3_block3_out[0]  Y
##  (Conv2D)            , 128)                  [0]']
##  conv3_block4_1_bn (B  (None, 28, 28  512    ['conv3_block4_1_conv  Y
##  atchNormalization)  , 128)                  [0][0]']
##  conv3_block4_1_relu   (None, 28, 28  0      ['conv3_block4_1_bn[0  Y
##  (Activation)        , 128)                  ][0]']
##  conv3_block4_2_conv   (None, 28, 28  147584 ['conv3_block4_1_relu  Y
##  (Conv2D)            , 128)                  [0][0]']
##  conv3_block4_2_bn (B  (None, 28, 28  512    ['conv3_block4_2_conv  Y
##  atchNormalization)  , 128)                  [0][0]']
##  conv3_block4_2_relu   (None, 28, 28  0      ['conv3_block4_2_bn[0  Y
##  (Activation)        , 128)                  ][0]']
##  conv3_block4_3_conv   (None, 28, 28  66048  ['conv3_block4_2_relu  Y
##  (Conv2D)            , 512)                  [0][0]']
##  conv3_block4_3_bn (B  (None, 28, 28  2048   ['conv3_block4_3_conv  Y
##  atchNormalization)  , 512)                  [0][0]']
##  conv3_block4_add (Ad  (None, 28, 28  0      ['conv3_block3_out[0]  Y
##  d)                  , 512)                  [0]',
##                                              'conv3_block4_3_bn[0
##                                             ][0]']
##  conv3_block4_out (Ac  (None, 28, 28  0      ['conv3_block4_add[0]  Y
##  tivation)           , 512)                  [0]']
##  conv4_block1_1_conv   (None, 14, 14  131328 ['conv3_block4_out[0]  Y
##  (Conv2D)            , 256)                  [0]']
##  conv4_block1_1_bn (B  (None, 14, 14  1024   ['conv4_block1_1_conv  Y
##  atchNormalization)  , 256)                  [0][0]']
##  conv4_block1_1_relu   (None, 14, 14  0      ['conv4_block1_1_bn[0  Y
```

```
##  (Activation)         , 256)                  ][0]']
##  conv4_block1_2_conv   (None, 14, 14  590080  ['conv4_block1_1_relu  Y
##  (Conv2D)             , 256)                  [0][0]']
##  conv4_block1_2_bn (B  (None, 14, 14  1024    ['conv4_block1_2_conv  Y
##  atchNormalization)   , 256)                  [0][0]']
##  conv4_block1_2_relu   (None, 14, 14  0       ['conv4_block1_2_bn[0  Y
##  (Activation)         , 256)                  ][0]']
##  conv4_block1_0_conv   (None, 14, 14  525312  ['conv3_block4_out[0]  Y
##  (Conv2D)             , 1024)                 [0]']
##  conv4_block1_3_conv   (None, 14, 14  263168  ['conv4_block1_2_relu  Y
##  (Conv2D)             , 1024)                 [0][0]']
##  conv4_block1_0_bn (B  (None, 14, 14  4096    ['conv4_block1_0_conv  Y
##  atchNormalization)   , 1024)                 [0][0]']
##  conv4_block1_3_bn (B  (None, 14, 14  4096    ['conv4_block1_3_conv  Y
##  atchNormalization)   , 1024)                 [0][0]']
##  conv4_block1_add (Ad  (None, 14, 14  0       ['conv4_block1_0_bn[0  Y
##  d)                   , 1024)                 ][0]',
##                                                'conv4_block1_3_bn[0
##                                               ][0]']
##  conv4_block1_out (Ac  (None, 14, 14  0       ['conv4_block1_add[0]  Y
##  tivation)            , 1024)                 [0]']
##  conv4_block2_1_conv   (None, 14, 14  262400  ['conv4_block1_out[0]  Y
##  (Conv2D)             , 256)                  [0]']
##  conv4_block2_1_bn (B  (None, 14, 14  1024    ['conv4_block2_1_conv  Y
##  atchNormalization)   , 256)                  [0][0]']
##  conv4_block2_1_relu   (None, 14, 14  0       ['conv4_block2_1_bn[0  Y
##  (Activation)         , 256)                  ][0]']
##  conv4_block2_2_conv   (None, 14, 14  590080  ['conv4_block2_1_relu  Y
##  (Conv2D)             , 256)                  [0][0]']
##  conv4_block2_2_bn (B  (None, 14, 14  1024    ['conv4_block2_2_conv  Y
##  atchNormalization)   , 256)                  [0][0]']
##  conv4_block2_2_relu   (None, 14, 14  0       ['conv4_block2_2_bn[0  Y
##  (Activation)         , 256)                  ][0]']
##  conv4_block2_3_conv   (None, 14, 14  263168  ['conv4_block2_2_relu  Y
##  (Conv2D)             , 1024)                 [0][0]']
##  conv4_block2_3_bn (B  (None, 14, 14  4096    ['conv4_block2_3_conv  Y
##  atchNormalization)   , 1024)                 [0][0]']
##  conv4_block2_add (Ad  (None, 14, 14  0       ['conv4_block1_out[0]  Y
##  d)                   , 1024)                 [0]',
##                                                'conv4_block2_3_bn[0
##                                               ][0]']
##  conv4_block2_out (Ac  (None, 14, 14  0       ['conv4_block2_add[0]  Y
##  tivation)            , 1024)                 [0]']
##  conv4_block3_1_conv   (None, 14, 14  262400  ['conv4_block2_out[0]  Y
##  (Conv2D)             , 256)                  [0]']
##  conv4_block3_1_bn (B  (None, 14, 14  1024    ['conv4_block3_1_conv  Y
##  atchNormalization)   , 256)                  [0][0]']
##  conv4_block3_1_relu   (None, 14, 14  0       ['conv4_block3_1_bn[0  Y
##  (Activation)         , 256)                  ][0]']
##  conv4_block3_2_conv   (None, 14, 14  590080  ['conv4_block3_1_relu  Y
##  (Conv2D)             , 256)                  [0][0]']
##  conv4_block3_2_bn (B  (None, 14, 14  1024    ['conv4_block3_2_conv  Y
##  atchNormalization)   , 256)                  [0][0]']
##  conv4_block3_2_relu   (None, 14, 14  0       ['conv4_block3_2_bn[0  Y
```

```
##  (Activation)         , 256)                 ][0]']
##  conv4_block3_3_conv   (None, 14, 14  263168  ['conv4_block3_2_relu  Y
##  (Conv2D)             , 1024)                [0][0]']
##  conv4_block3_3_bn (B  (None, 14, 14  4096    ['conv4_block3_3_conv  Y
##  atchNormalization)   , 1024)                [0][0]']
##  conv4_block3_add (Ad  (None, 14, 14  0       ['conv4_block2_out[0]  Y
##  d)                   , 1024)                [0]',
##                                               'conv4_block3_3_bn[0
##                                              ][0]']
##  conv4_block3_out (Ac  (None, 14, 14  0       ['conv4_block3_add[0]  Y
##  tivation)            , 1024)                [0]']
##  conv4_block4_1_conv   (None, 14, 14  262400  ['conv4_block3_out[0]  Y
##  (Conv2D)             , 256)                 [0]']
##  conv4_block4_1_bn (B  (None, 14, 14  1024    ['conv4_block4_1_conv  Y
##  atchNormalization)   , 256)                 [0][0]']
##  conv4_block4_1_relu   (None, 14, 14  0       ['conv4_block4_1_bn[0  Y
##  (Activation)         , 256)                 ][0]']
##  conv4_block4_2_conv   (None, 14, 14  590080  ['conv4_block4_1_relu  Y
##  (Conv2D)             , 256)                 [0][0]']
##  conv4_block4_2_bn (B  (None, 14, 14  1024    ['conv4_block4_2_conv  Y
##  atchNormalization)   , 256)                 [0][0]']
##  conv4_block4_2_relu   (None, 14, 14  0       ['conv4_block4_2_bn[0  Y
##  (Activation)         , 256)                 ][0]']
##  conv4_block4_3_conv   (None, 14, 14  263168  ['conv4_block4_2_relu  Y
##  (Conv2D)             , 1024)                [0][0]']
##  conv4_block4_3_bn (B  (None, 14, 14  4096    ['conv4_block4_3_conv  Y
##  atchNormalization)   , 1024)                [0][0]']
##  conv4_block4_add (Ad  (None, 14, 14  0       ['conv4_block3_out[0]  Y
##  d)                   , 1024)                [0]',
##                                               'conv4_block4_3_bn[0
##                                              ][0]']
##  conv4_block4_out (Ac  (None, 14, 14  0       ['conv4_block4_add[0]  Y
##  tivation)            , 1024)                [0]']
##  conv4_block5_1_conv   (None, 14, 14  262400  ['conv4_block4_out[0]  Y
##  (Conv2D)             , 256)                 [0]']
##  conv4_block5_1_bn (B  (None, 14, 14  1024    ['conv4_block5_1_conv  Y
##  atchNormalization)   , 256)                 [0][0]']
##  conv4_block5_1_relu   (None, 14, 14  0       ['conv4_block5_1_bn[0  Y
##  (Activation)         , 256)                 ][0]']
##  conv4_block5_2_conv   (None, 14, 14  590080  ['conv4_block5_1_relu  Y
##  (Conv2D)             , 256)                 [0][0]']
##  conv4_block5_2_bn (B  (None, 14, 14  1024    ['conv4_block5_2_conv  Y
##  atchNormalization)   , 256)                 [0][0]']
##  conv4_block5_2_relu   (None, 14, 14  0       ['conv4_block5_2_bn[0  Y
##  (Activation)         , 256)                 ][0]']
##  conv4_block5_3_conv   (None, 14, 14  263168  ['conv4_block5_2_relu  Y
##  (Conv2D)             , 1024)                [0][0]']
##  conv4_block5_3_bn (B  (None, 14, 14  4096    ['conv4_block5_3_conv  Y
##  atchNormalization)   , 1024)                [0][0]']
##  conv4_block5_add (Ad  (None, 14, 14  0       ['conv4_block4_out[0]  Y
##  d)                   , 1024)                [0]',
##                                               'conv4_block5_3_bn[0
##                                              ][0]']
##  conv4_block5_out (Ac  (None, 14, 14  0       ['conv4_block5_add[0]  Y
```

```
##  tivation)            , 1024)                 [0]']
##  conv4_block6_1_conv   (None, 14, 14  262400  ['conv4_block5_out[0]  Y
##  (Conv2D)             , 256)                  [0]']
##  conv4_block6_1_bn (B  (None, 14, 14  1024    ['conv4_block6_1_conv  Y
##  atchNormalization)   , 256)                  [0][0]']
##  conv4_block6_1_relu   (None, 14, 14  0       ['conv4_block6_1_bn[0  Y
##  (Activation)         , 256)                  ][0]']
##  conv4_block6_2_conv   (None, 14, 14  590080  ['conv4_block6_1_relu  Y
##  (Conv2D)             , 256)                  [0][0]']
##  conv4_block6_2_bn (B  (None, 14, 14  1024    ['conv4_block6_2_conv  Y
##  atchNormalization)   , 256)                  [0][0]']
##  conv4_block6_2_relu   (None, 14, 14  0       ['conv4_block6_2_bn[0  Y
##  (Activation)         , 256)                  ][0]']
##  conv4_block6_3_conv   (None, 14, 14  263168  ['conv4_block6_2_relu  Y
##  (Conv2D)             , 1024)                 [0][0]']
##  conv4_block6_3_bn (B  (None, 14, 14  4096    ['conv4_block6_3_conv  Y
##  atchNormalization)   , 1024)                 [0][0]']
##  conv4_block6_add (Ad  (None, 14, 14  0       ['conv4_block5_out[0]  Y
##  d)                   , 1024)                 [0]',
##                                                'conv4_block6_3_bn[0
##                                               ][0]']
##  conv4_block6_out (Ac  (None, 14, 14  0       ['conv4_block6_add[0]  Y
##  tivation)            , 1024)                 [0]']
##  conv5_block1_1_conv   (None, 7, 7,   524800  ['conv4_block6_out[0]  Y
##  (Conv2D)             512)                    [0]']
##  conv5_block1_1_bn (B  (None, 7, 7,   2048    ['conv5_block1_1_conv  Y
##  atchNormalization)   512)                    [0][0]']
##  conv5_block1_1_relu   (None, 7, 7,   0       ['conv5_block1_1_bn[0  Y
##  (Activation)         512)                    ][0]']
##  conv5_block1_2_conv   (None, 7, 7,   2359808  ['conv5_block1_1_relu  Y
##  (Conv2D)             512)                    [0][0]']
##  conv5_block1_2_bn (B  (None, 7, 7,   2048    ['conv5_block1_2_conv  Y
##  atchNormalization)   512)                    [0][0]']
##  conv5_block1_2_relu   (None, 7, 7,   0       ['conv5_block1_2_bn[0  Y
##  (Activation)         512)                    ][0]']
##  conv5_block1_0_conv   (None, 7, 7,   2099200  ['conv4_block6_out[0]  Y
##  (Conv2D)             2048)                   [0]']
##  conv5_block1_3_conv   (None, 7, 7,   1050624  ['conv5_block1_2_relu  Y
##  (Conv2D)             2048)                   [0][0]']
##  conv5_block1_0_bn (B  (None, 7, 7,   8192    ['conv5_block1_0_conv  Y
##  atchNormalization)   2048)                   [0][0]']
##  conv5_block1_3_bn (B  (None, 7, 7,   8192    ['conv5_block1_3_conv  Y
##  atchNormalization)   2048)                   [0][0]']
##  conv5_block1_add (Ad  (None, 7, 7,   0       ['conv5_block1_0_bn[0  Y
##  d)                   2048)                   ][0]',
##                                                'conv5_block1_3_bn[0
##                                               ][0]']
##  conv5_block1_out (Ac  (None, 7, 7,   0       ['conv5_block1_add[0]  Y
##  tivation)            2048)                   [0]']
##  conv5_block2_1_conv   (None, 7, 7,   1049088  ['conv5_block1_out[0]  Y
##  (Conv2D)             512)                    [0]']
##  conv5_block2_1_bn (B  (None, 7, 7,   2048    ['conv5_block2_1_conv  Y
##  atchNormalization)   512)                    [0][0]']
##  conv5_block2_1_relu   (None, 7, 7,   0       ['conv5_block2_1_bn[0  Y
```

```
##  (Activation)         512)                    ][0]']
##  conv5_block2_2_conv   (None, 7, 7,   2359808  ['conv5_block2_1_relu  Y
##  (Conv2D)             512)                    [0][0]']
##  conv5_block2_2_bn (B  (None, 7, 7,   2048    ['conv5_block2_2_conv  Y
##  atchNormalization)   512)                    [0][0]']
##  conv5_block2_2_relu   (None, 7, 7,   0       ['conv5_block2_2_bn[0  Y
##  (Activation)         512)                    ][0]']
##  conv5_block2_3_conv   (None, 7, 7,   1050624  ['conv5_block2_2_relu  Y
##  (Conv2D)             2048)                   [0][0]']
##  conv5_block2_3_bn (B  (None, 7, 7,   8192    ['conv5_block2_3_conv  Y
##  atchNormalization)   2048)                   [0][0]']
##  conv5_block2_add (Ad  (None, 7, 7,   0       ['conv5_block1_out[0]  Y
##  d)                   2048)                   [0]',
##                                                'conv5_block2_3_bn[0
##                                               ][0]']
##  conv5_block2_out (Ac  (None, 7, 7,   0       ['conv5_block2_add[0]  Y
##  tivation)            2048)                   [0]']
##  conv5_block3_1_conv   (None, 7, 7,   1049088  ['conv5_block2_out[0]  Y
##  (Conv2D)             512)                    [0]']
##  conv5_block3_1_bn (B  (None, 7, 7,   2048    ['conv5_block3_1_conv  Y
##  atchNormalization)   512)                    [0][0]']
##  conv5_block3_1_relu   (None, 7, 7,   0       ['conv5_block3_1_bn[0  Y
##  (Activation)         512)                    ][0]']
##  conv5_block3_2_conv   (None, 7, 7,   2359808  ['conv5_block3_1_relu  Y
##  (Conv2D)             512)                    [0][0]']
##  conv5_block3_2_bn (B  (None, 7, 7,   2048    ['conv5_block3_2_conv  Y
##  atchNormalization)   512)                    [0][0]']
##  conv5_block3_2_relu   (None, 7, 7,   0       ['conv5_block3_2_bn[0  Y
##  (Activation)         512)                    ][0]']
##  conv5_block3_3_conv   (None, 7, 7,   1050624  ['conv5_block3_2_relu  Y
##  (Conv2D)             2048)                   [0][0]']
##  conv5_block3_3_bn (B  (None, 7, 7,   8192    ['conv5_block3_3_conv  Y
##  atchNormalization)   2048)                   [0][0]']
##  conv5_block3_add (Ad  (None, 7, 7,   0       ['conv5_block2_out[0]  Y
##  d)                   2048)                   [0]',
##                                                'conv5_block3_3_bn[0
##                                               ][0]']
##  conv5_block3_out (Ac  (None, 7, 7,   0       ['conv5_block3_add[0]  Y
##  tivation)            2048)                   [0]']
##  avg_pool (GlobalAver  (None, 2048)  0        ['conv5_block3_out[0]  Y
##  agePooling2D)                                [0]']
##  predictions (Dense)  (None, 1000)   2049000  ['avg_pool[0][0]']     Y
## ==================================================================================
## Total params: 25,636,712
## Trainable params: 25,583,592
## Non-trainable params: 53,120
## __________________________________________________________________________________
```

```
pred6 <- model %>% predict(x) %>%
  imagenet_decode_predictions(top = 3)
names(pred6) <- image_names
print(pred6)
```

```
## $flamingo.jpg
##   class_name class_description       score
```

```
## 1  n02007558          flamingo 0.926349699
## 2  n02006656         spoonbill 0.071699418
## 3  n02002556       white_stork 0.001228211
##
## $hawk.jpg
##   class_name class_description     score
## 1  n03388043          fountain 0.2788653
## 2  n03532672              hook 0.1785545
## 3  n03804744              nail 0.1080728
##
## $hawk_cropped.jpeg
##   class_name class_description      score
## 1  n01608432              kite 0.72270989
## 2  n01622779    great_grey_owl 0.08182549
## 3  n01532829       house_finch 0.04218861
##
## $huey.jpg
##   class_name           class_description      score
## 1  n02097474             Tibetan_terrier 0.50929791
## 2  n02098413                       Lhasa 0.42209798
## 3  n02098105 soft-coated_wheaten_terrier 0.01695857
##
## $kitty.jpg
##   class_name    class_description      score
## 1  n02105641 Old_English_sheepdog 0.83266050
## 2  n02086240             Shih-Tzu 0.04513872
## 3  n03223299              doormat 0.03299759
##
## $weaver.jpg
##   class_name class_description      score
## 1  n01843065           jacamar 0.49795407
## 2  n01818515             macaw 0.22193304
## 3  n02494079   squirrel_monkey 0.04287856
```

Part b (Explanation: 1 pt)

In Lab 10.9.2, it is claimed that their neural network with zero hidden layers is equivalent to a multinomial logistic regression model. Explain why this is the case.

This is the case because neural networks output linear combinations of the inputs with coefficient estimates that are computed as class probabilities using the softmax activation function.

Part c (Code and/or Explanation: 1 pt)

Using the ideas in part (b), how would you revise the code creating `modellr` to perform *linear* regression instead? (HINT: look up the documentation for `layer_dense`)

If we wanted to perform linear regression we would set activation = "linear" or not include an argument for activation and it will default to the linear function.

Part d (Explanation: 1 pt)

Briefly explain what the `to_categorical` function does, and the types of problems in which it would be useful.

The to_cateogrical function converts vectors of integers that represent classes into a matrix with rows that use 0 1 coding (binary) to indicate the classes in the vector. This would be useful when you're modeling a qualitative response that is represented by values but those values don't actually indicate the order of the data.

Problem 2: Understanding the Math

Part a (Computation and Explanation: 2 pts)

Consider two predictors, x_1 and x_2, feeding into a two-unit hidden layer. Suppose that the hidden layer uses ReLU activation functions with w_{kj} given in Equation (10.6) of the book and the output layer uses a linear activation function with β_j given in equation (10.6).

Find the activation functions $A_1(x_1, x_2)$ and $A_2(x_1, x_2)$. Then, find $f(x_1, x_2)$ explicitly in terms of x_1 and x_2. Note: $f(x_1, x_2)$ should be a piecewise function.

```
knitr::include_graphics("Prob2a.jpg")
```

Lab 11 Problem 2

(a) Predictors x_1 and x_2 in 2 unit hidden layer.

ReLU activation functions

$$g(z) = (z)_+ = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise} \end{cases}$$

Find activation functions

$$A_k = h_k(x) = g(w_{k0} + \sum_{j=1}^{2} w_{kj} x_j)$$

$$A_1(x_1, x_2) = g(w_{10} + \sum_{j=1}^{2} w_{1j} x_j)$$
$$A_2(x_1, x_2) = g(w_{20} + \sum_{j=1}^{2} w_{2j} x_j)$$

$\beta_0 = 0, \quad \beta_1 = \frac{1}{4}, \quad \beta_2 = -\frac{1}{4}$
$w_{10} = 0, \quad w_{11} = 1, \quad w_{12} = 1$
$w_{20} = 0, \quad w_{21} = 1, \quad w_{22} = -1$
} from 10.6

$$A_1 = h_1(x) = g(0 + x_1 + x_2) \Rightarrow A_1(x_1, x_2) = g(x_1 + x_2)$$
$$A_2 = h_2(x) = g(0 + x_1 - x_2) \quad A_2(x_1, x_2) = g(x_1 - x_2)$$

Output layer uses a linear activation function

$$f(x) = \beta_0 + \sum_{k=1}^{2} \beta_k A_k$$

$$f(x_1, x_2) = 0 + \frac{1}{4} g(x_1 + x_2) - \frac{1}{4} g(x_1 - x_2)$$

$$f(x_1, x_2) = \begin{cases} 0 & x_1 + x_2 < 0 \\ \frac{1}{4}(x_1 + x_2) & \text{otherwise} \end{cases} + \begin{cases} 0 & x_1 - x_2 < 0 \\ -\frac{1}{4}(x_1 - x_2) & \text{otherwise} \end{cases}$$

$$\Rightarrow \frac{1}{4}[(x_1 + x_2) - (x_1 - x_2)]$$

$$\Rightarrow f(x_1, x_2) = \begin{cases} 0 & x_2 < 0 \\ \frac{1}{2} x_2 & \text{otherwise} \end{cases}$$

Part b (Computation and Explanation: 2 pts)

Consider the following 5x5 matrix:

```
A <- matrix(c(5, 5, 5, 5, 5,
              0, 4, 4, 4, 4,
              0, 0, 3, 3, 3,
              0, 0, 0, 2, 2,
              0, 0, 0, 0, 1),
            nrow = 5, byrow = T
            )
print(A)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    5    5    5    5    5
## [2,]    0    4    4    4    4
## [3,]    0    0    3    3    3
## [4,]    0    0    0    2    2
## [5,]    0    0    0    0    1
```

Without running anything in R, find the output of convolving A with the 2x2 convolution filter

```
conv <- matrix(c(1, 0, 0, 1), nrow = 2)
print(conv)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

followed by a 2×2 max pool.

To find the output of convolving A with the 2x2 convolution filter, we apply the filter to every 2×2 submatrix of the original 5x5 matrix. The result will be a 4x4 matrix. We get the entries as follows:

For the first submatrix, $5(1) + 5(0) + 0(0) + 4(1) = 9$ which is our first entry in the convolved matrix. Similary, the (1,2) entry is $5(1) + 5(0) + 4(0) + 4(1) = 9$. We repeat this for every 2x2 submatrix, giving us the final matrix below.

```
conv_A <- matrix(c(9, 9, 9, 9,
                   0, 7, 7, 7,
                   0, 0, 5, 5,
                   0, 0, 0, 3),
                 nrow = 4, byrow = T
                 )
print(conv_A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    9    9    9    9
## [2,]    0    7    7    7
## [3,]    0    0    5    5
## [4,]    0    0    0    3
```

Using this matrix, we perform max pooling by choosing the maximum value in each non-overlapping 2x2 submatrix. For example, the first 2x2 submatrix

```
example <- matrix(c(9, 0, 9, 7), nrow = 2)
print(example)
```

```
##      [,1] [,2]
## [1,]    9    9
```

```
## [2,]    0    7
```

has maximum value 9 which is the first entry in our final matrix.

Our final matrix is below.

```
final_A <- matrix(c(9, 0, 9, 5), nrow = 2)
print(final_A)
```

```
##      [,1] [,2]
## [1,]    9    9
## [2,]    0    5
```