Importação dos dados

```
download.file('https://storage.googleapis.com/ventania10/Infiltration3.zip',
destfile = "Infiltration2.zip", mode = "wb", method = "auto", quiet=FALSE)
```

```
download.file('https://storage.googleapis.com/ventania10/Normal3.zip',
destfile = "Normal2.zip", mode = "wb", method = "auto", quiet=FALSE)
```

```
unzip("Normal2.zip")
```

```
unzip("Infiltration2.zip")
```

Referenciando pastas de teste, treino e validação

```
base_dir <- "Infiltration_Normal"
train_dir <- file.path(base_dir, "train")
test_dir <- file.path(base_dir, "test")
validation_dir <- file.path(base_dir, "validation")

train_Infiltration_dir <- file.path(train_dir, "Infiltration")
train_Normal_dir <- file.path(train_dir, "Normal")

test_Infiltration_dir <- file.path(test_dir, "Infiltration")
test_Normal_dir <- file.path(test_dir, "Normal")
```

```
validation_Infiltration_dir <- file.path(validation_dir, "Infiltration")
validation_Normal_dir <- file.path(validation_dir, "Normal")

dir.create(base_dir)
```

Analisando diretórios

```
cat("total train Infiltration images:", length(list.files(train_Infiltration_dir)), "\n")
cat("total train Normal images:", length(list.files(train_Normal_dir)), "\n")

cat("total validation Infiltration images:", length(list.files(validation_Infiltration_dir)), "\n")
cat("total validation Normal images:", length(list.files(validation_Normal_dir)), "\n")

cat("total test Infiltration images:", length(list.files(test_Infiltration_dir)), "\n")
cat("total test Normal images:", length(list.files(test_Normal_dir)), "\n")
```

```
total train Infiltration images: 7150
total train Normal images: 7510
total validation Infiltration images: 1785
total validation Normal images: 1880
total test Infiltration images: 612
total test Normal images: 610
```

Bibliotecas dependentes:

```
devtools::install_github("rstudio/keras")
library(tidyverse)
library(keras)
```

```
Skipping install of 'keras' from a github remote, the SHA1 (f29f7025) has not changed since last install.
```

```
  Use `force = TRUE` to force installation

Warning message in system("timedatectl", intern = TRUE):
“running command 'timedatectl' had status 1”
── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

✓ ggplot2 3.3.6     ✓ purrr   0.3.4
✓ tibble  3.1.7     ✓ dplyr   1.0.9
✓ tidyr   1.2.0     ✓ stringr 1.4.0
✓ readr   2.1.2     ✓ forcats 0.5.1

── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

```
system("sed -i -e 's,R_LD_LIBRARY_PATH}:${LD_LIBRARY,LD_LIBRARY_PATH}:${R_LD_LIBRARY,' /usr/lib/R/etc/ldpaths")
```

```
tensorflow::tf_gpu_configured('GPU')
```

```
Loaded Tensorflow version 2.8.2

TRUE
```

```
### leituras das imagens/normalização
validation_datagen <- image_data_generator(rescale = 1/255)

train_datagen <- image_data_generator(
  rescale = 1/255,
  rotation_range = 30,
  zoom_range = 0.2,
  horizontal_flip = F
)
```

```
train_generator <- flow_images_from_directory(
  train_dir,
  train_datagen,
  target_size = c(150, 150),
  batch_size = 50,
  class_mode = "binary"
)

test_generator <- flow_images_from_directory(
  test_dir,
  validation_datagen,
  target_size = c(150, 150),
  batch_size = 50,
  class_mode = "binary"
)

validation_generator <- flow_images_from_directory(
  validation_dir,
  validation_datagen,
  target_size = c(150, 150),
  batch_size = 50,
  class_mode = "binary"
)
```

Arquitetura da rede

```
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_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%
  layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%
  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_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu") %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu") %>%
  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 = 256, kernel_size = c(3, 3), activation = "relu") %>%
  layer_conv_2d(filters = 256, kernel_size = c(3, 3), activation = "relu") %>%
  layer_conv_2d(filters = 256, kernel_size = c(3, 3), activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2))

summary(model)
```

```
Model: "sequential"
_________________________________________________________________________________
 Layer (type)                        Output Shape                      Param #
=================================================================================
 conv2d_11 (Conv2D)                  (None, 148, 148, 32)              896
 max_pooling2d_3 (MaxPooling2D)      (None, 74, 74, 32)                0
 conv2d_10 (Conv2D)                  (None, 72, 72, 64)                18496
 conv2d_9 (Conv2D)                   (None, 70, 70, 64)                36928
 conv2d_8 (Conv2D)                   (None, 68, 68, 64)                36928
 conv2d_7 (Conv2D)                   (None, 66, 66, 64)                36928
 max_pooling2d_2 (MaxPooling2D)      (None, 33, 33, 64)                0
 conv2d_6 (Conv2D)                   (None, 31, 31, 128)               73856
 conv2d_5 (Conv2D)                   (None, 29, 29, 128)               147584
 conv2d_4 (Conv2D)                   (None, 27, 27, 128)               147584
 conv2d_3 (Conv2D)                   (None, 25, 25, 128)               147584
 max_pooling2d_1 (MaxPooling2D)      (None, 12, 12, 128)               0
 conv2d_2 (Conv2D)                   (None, 10, 10, 256)               295168
 conv2d_1 (Conv2D)                   (None, 8, 8, 256)                 590080
```

```
 conv2d (Conv2D)                    (None, 6, 6, 256)               590080
 max_pooling2d (MaxPooling2D)       (None, 3, 3, 256)               0
=================================================================
Total params: 2,122,112
Trainable params: 2,122,112
Non-trainable params: 0
_________________________________________________________________
```

```
model <- model %>%
  layer_flatten() %>%
  layer_dense(units = 512, activation = "relu") %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 256, activation = "relu") %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dropout(rate = 0.1) %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

  summary(model)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                       Output Shape                    Param #
=================================================================
 conv2d_11 (Conv2D)                 (None, 148, 148, 32)            896
 max_pooling2d_3 (MaxPooling2D)     (None, 74, 74, 32)              0
 conv2d_10 (Conv2D)                 (None, 72, 72, 64)              18496
 conv2d_9 (Conv2D)                  (None, 70, 70, 64)              36928
 conv2d_8 (Conv2D)                  (None, 68, 68, 64)              36928
 conv2d_7 (Conv2D)                  (None, 66, 66, 64)              36928
```

```
 max_pooling2d_2 (MaxPooling2D)     (None, 33, 33, 64)              0
 conv2d_6 (Conv2D)                  (None, 31, 31, 128)             73856
 conv2d_5 (Conv2D)                  (None, 29, 29, 128)             147584
 conv2d_4 (Conv2D)                  (None, 27, 27, 128)             147584
 conv2d_3 (Conv2D)                  (None, 25, 25, 128)             147584
 max_pooling2d_1 (MaxPooling2D)     (None, 12, 12, 128)             0
 conv2d_2 (Conv2D)                  (None, 10, 10, 256)             295168
 conv2d_1 (Conv2D)                  (None, 8, 8, 256)               590080
 conv2d (Conv2D)                    (None, 6, 6, 256)               590080
 max_pooling2d (MaxPooling2D)       (None, 3, 3, 256)               0
 flatten (Flatten)                  (None, 2304)                    0
 dense_6 (Dense)                    (None, 512)                     1180160
 dropout_2 (Dropout)                (None, 512)                     0
 dense_5 (Dense)                    (None, 256)                     131328
 dense_4 (Dense)                    (None, 128)                     32896
 dropout_1 (Dropout)                (None, 128)                     0
 dense_3 (Dense)                    (None, 64)                      8256
 dense_2 (Dense)                    (None, 32)                      2080
 dropout (Dropout)                  (None, 32)                      0
 dense_1 (Dense)                    (None, 16)                      528
 dense (Dense)                      (None, 1)                       17
================================================================================
Total params: 3,477,377
Trainable params: 3,477,377
Non-trainable params: 0
_________________________________________________________________________________
```

Definindo função de perda

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

```
)
```

Definindo callbacks

```
early_stop <- callback_early_stopping(
  monitor = "val_loss",
  min_delta = 1e-3,
  patience = 10,
  verbose = 1
)

checkpoint <- callback_model_checkpoint(
  "Infiltration_Normal_1.h5",
  monitor = "val_loss",
  verbose = 1,
  save_best_only = T,
  mode = "min",
  period = NULL
)
```

```
batch <- generator_next(train_generator)
str(batch)
```

```
     List of 2
      $ : num [1:50, 1:150, 1:150, 1:3] 0.02802 0.00392 0.01263 0 0.39608 ...
      $ : num [1:50(1d)] 1 0 1 0 0 1 1 0 0 1 ...
```

```
batch <- generator_next(validation_generator)
str(batch)
```

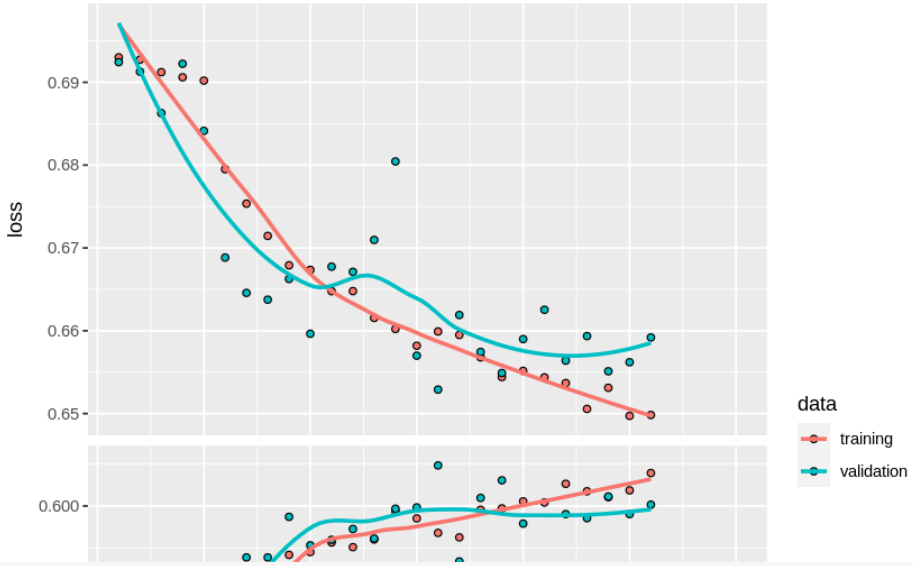
```
List of 2
 $ : num [1:50, 1:150, 1:150, 1:3] 0.482 0.459 0.804 0.98 0 ...
 $ : num [1:50(1d)] 0 1 1 0 1 1 0 0 1 0 ...
```

Treinamento da rede

```
history <- model %>% fit(
  train_generator,
  epochs = 30,
  steps_per_epoch = 290,
  validation_data = validation_generator,
  validation_steps = 50,
  callbacks = c(checkpoint,early_stop)
)
```

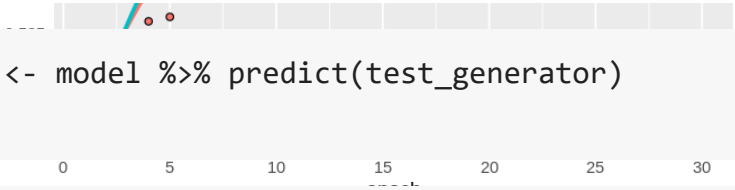
Resultados

```
plot(history)
```




```
model %>% evaluate(test_generator)
```

```
loss:  0.644891798496246 acc:  0.618657946586609
```




```
classes <- model %>% predict(test_generator)
```

```
table("real"=test_generator$labels,"predito"=round(classes))
(154+476)/(154+134+458+476)
476/(134+476)
154/(154+458)
```

```
          predito
```

```
save.image(file="modelo1.rdata")
```

```
        1 134 476
```

vendo o processamento do modelo

```
img_path <- '00000021_001.png'
img <- image_load(img_path, target_size = c(150, 150))
img_tensor <- image_to_array(img)
img_tensor <- array_reshape(img_tensor, c(1, 150, 150, 3))
img_tensor <- img_tensor / 255
```

```
     1 · 150 · 150 · 3
```

```
layer_outputs <- lapply(model$layers[1:8], function(layer) layer$output)
activation_model <- keras_model(inputs = model$input, outputs = layer_outputs)
```

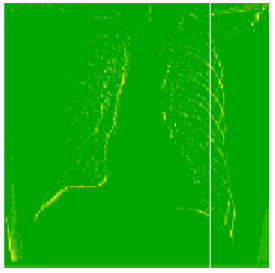
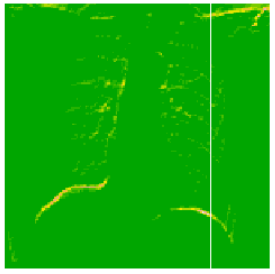
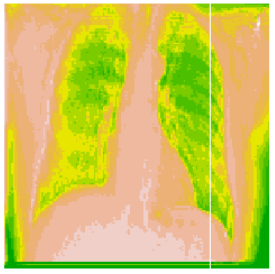
```
activations <- activation_model %>% predict(img_tensor)
```

```
first_layer_activation <- activations[[1]]
dim(first_layer_activation)
```

```
     1 · 148 · 148 · 32
```

```
plot_channel <- function(channel) {
 rotate <- function(x) t(apply(x, 2, rev))
 image(rotate(channel), axes = FALSE, asp = 1,
 col = terrain.colors(12))
}
```

```
par(mfrow = c(2,2))
plot(as.raster(img_tensor[1,,,]))
plot_channel(first_layer_activation[1,,,1])
plot_channel(first_layer_activation[1,,,2])
plot_channel(first_layer_activation[1,,,3])
```

✓ 0s conclusão: 21:04