Assignment_02

Code By: Xan Lamoreux

Advanced Machine Learning

Import Keras Library and load cat vs dog dataset

```
library(keras)

original_dataset_dir <- "~/Downloads/kaggle_original_data"

base_dir <- "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
4/cat vs dog - dataset/sm/"

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

train_cats_dir <- file.path(train_dir, "cats")
train_dogs_dir <- file.path(train_dir, "dogs")

validation_cats_dir <- file.path(validation_dir, "cats")
validation_dogs_dir <- file.path(validation_dir, "dogs")

test_cats_dir <- file.path(test_dir, "cats")
test_dogs_dir <- file.path(test_dir, "dogs")
```

Number of cats and dogs images in each directory

```
cat("total training cat images:", length(list.files(train_cats_dir)), "\n")

## total training cat images: 1000

cat("total training dog images:", length(list.files(train_dogs_dir)), "\n")

## total training dog images: 1000

cat("total validation cat images:", length(list.files(validation_cats_dir)),
"\n")

## total validation cat images: 500

cat("total validation dog images:", length(list.files(validation_dogs_dir)),
"\n")

## total validation dog images: 500
```

```
cat("total test cat images:", length(list.files(test_cats_dir)), "\n")

## total test cat images: 500

cat("total test dog images:", length(list.files(test_dogs_dir)), "\n")

## total test dog images: 500
```

Building the model

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

## Loaded Tensorflow version 2.6.0

summary(model)

## Model: "sequential"
## _____
_____
## Layer (type)                        Output Shape                     Param
#
## =====================================================================
======
## conv2d_3 (Conv2D)                   (None, 148, 148, 32)             896
## _____
_____
## max_pooling2d_3 (MaxPooling2D)      (None, 74, 74, 32)               0
## _____
_____
## conv2d_2 (Conv2D)                   (None, 72, 72, 64)               18496
## _____
_____
## max_pooling2d_2 (MaxPooling2D)      (None, 36, 36, 64)               0
## _____
_____
## conv2d_1 (Conv2D)                   (None, 34, 34, 128)              73856
## _____
_____
```

```
## max_pooling2d_1 (MaxPooling2D)      (None, 17, 17, 128)             0
## _________________________________________________________________
_____
## conv2d (Conv2D)                     (None, 15, 15, 128)             147584
## _________________________________________________________________
_____
## max_pooling2d (MaxPooling2D)        (None, 7, 7, 128)               0
## _________________________________________________________________
_____
## flatten (Flatten)                   (None, 6272)                    0
## _________________________________________________________________
_____
## dense_1 (Dense)                     (None, 512)                     321177
6
## _________________________________________________________________
_____
## dense (Dense)                       (None, 1)                       513
## =================================================================
======
## Total params: 3,453,121
## Trainable params: 3,453,121
## Non-trainable params: 0
## _________________________________________________________________
_____
```

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

## Warning in backcompat_fix_rename_lr_to_learning_rate(...): the `lr` argume
nt has
## been renamed to `learning_rate`.
```

Creating Data Generator

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

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

validation_generator <- flow_images_from_directory(
  validation_dir,
  validation_datagen,
```

```
  target_size = c(150, 150),
  batch_size = 20,
  class_mode = "binary"
)

batch <- generator_next(train_generator)
str(batch)

## List of 2
##  $ : num [1:20, 1:150, 1:150, 1:3] 0.424 0.718 0.471 0.435 0.314 ...
##  $ : num [1:20(1d)] 1 0 1 1 0 0 1 1 1 1 ...
```

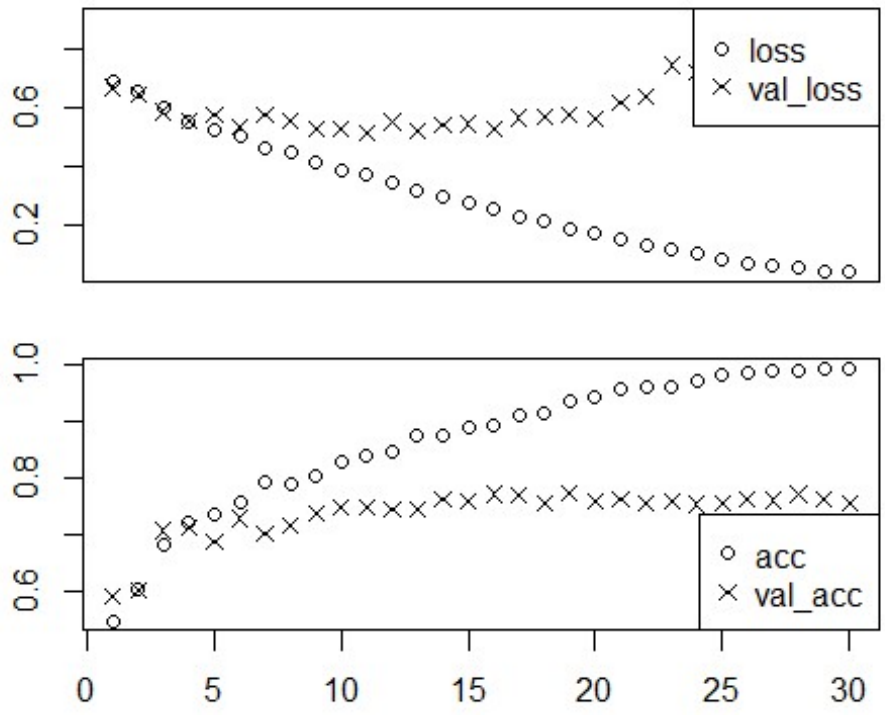
Train the model

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

## Warning in fit_generator(., train_generator, steps_per_epoch = 100, epochs
## = 30, : `fit_generator` is deprecated. Use `fit` instead, it now accept
## generators.
```

Plot the history

```
plot(history)
```



Save the Model

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

==========

————————————-TASK # 01—————————————-

==========

```
library(keras)

original_dataset_dir <- "~/Downloads/kaggle_original_data"

base_dir <- "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 04/cat vs dog - dataset/sm/"

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

train_cats_dir <- file.path(train_dir, "cats")
train_dogs_dir <- file.path(train_dir, "dogs")

validation_cats_dir <- file.path(validation_dir, "cats")
validation_dogs_dir <- file.path(validation_dir, "dogs")

test_cats_dir <- file.path(test_dir, "cats")
test_dogs_dir <- file.path(test_dir, "dogs")
```

Number of cats and dogs images in each directory

```
cat("total training cat images:", length(list.files(train_cats_dir)), "\n")

## total training cat images: 1000

cat("total training dog images:", length(list.files(train_dogs_dir)), "\n")

## total training dog images: 1000

cat("total validation cat images:", length(list.files(validation_cats_dir)), "\n")

## total validation cat images: 500

cat("total validation dog images:", length(list.files(validation_dogs_dir)), "\n")

## total validation dog images: 500
```

```
cat("total test cat images:", length(list.files(test_cats_dir)), "\n")

## total test cat images: 500

cat("total test dog images:", length(list.files(test_dogs_dir)), "\n")

## total test dog images: 500
```

Building the model (Using DropOut to reduce overfitting)

```
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3), activation = "relu",
                input_shape = c(150, 150, 3)) %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu") %>
%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu") %>
%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

## Loaded Tensorflow version 2.6.0

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

## Warning in backcompat_fix_rename_lr_to_learning_rate(...): the `lr` argume
nt has
## been renamed to `learning_rate`.

summary(model)

## Model: "sequential"
## _____
_____
## Layer (type)                        Output Shape                     Param
#
## =====================================================================
======
## conv2d_3 (Conv2D)                   (None, 148, 148, 32)             896
## _____
_____
## max_pooling2d_3 (MaxPooling2D)      (None, 74, 74, 32)               0
```

```
## _________________________________________________________________
_____
## conv2d_2 (Conv2D)                  (None, 72, 72, 64)              18496
## _________________________________________________________________
_____
## max_pooling2d_2 (MaxPooling2D)     (None, 36, 36, 64)              0
## _________________________________________________________________
_____
## conv2d_1 (Conv2D)                  (None, 34, 34, 128)             73856
## _________________________________________________________________
_____
## max_pooling2d_1 (MaxPooling2D)     (None, 17, 17, 128)             0
## _________________________________________________________________
_____
## conv2d (Conv2D)                    (None, 15, 15, 128)             147584
## _________________________________________________________________
_____
## max_pooling2d (MaxPooling2D)       (None, 7, 7, 128)               0
## _________________________________________________________________
_____
## flatten (Flatten)                  (None, 6272)                    0
## _________________________________________________________________
_____
## dropout (Dropout)                  (None, 6272)                    0
## _________________________________________________________________
_____
## dense_1 (Dense)                    (None, 512)                     321177
6
## _________________________________________________________________
_____
## dense (Dense)                      (None, 1)                       513
## =================================================================
======
## Total params: 3,453,121
## Trainable params: 3,453,121
## Non-trainable params: 0
## _________________________________________________________________
_____
```

Creating Data Generator (using Data Augmentation Technique to reduce overfitting)

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

```
test_datagen <- image_data_generator(rescale = 1/255)
train_generator <- flow_images_from_directory(
  train_dir,
  datagen,
  target_size = c(150, 150),
  batch_size = 20,
  class_mode = "binary"
)

validation_generator <- flow_images_from_directory(
  validation_dir,
  test_datagen,
  target_size = c(150, 150),
  batch_size = 20,
  class_mode = "binary"
)



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

```
## List of 2
##  $ : num [1:20, 1:150, 1:150, 1:3] 0.359 0.447 0.197 0.402 0.577 ...
##  $ : num [1:20(1d)] 1 0 0 0 1 0 1 1 0 0 ...
```

Train the model

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

```
## Warning in fit_generator(., train_generator, steps_per_epoch = 100, epochs
## = 100, : `fit_generator` is deprecated. Use `fit` instead, it now accept
## generators.
```

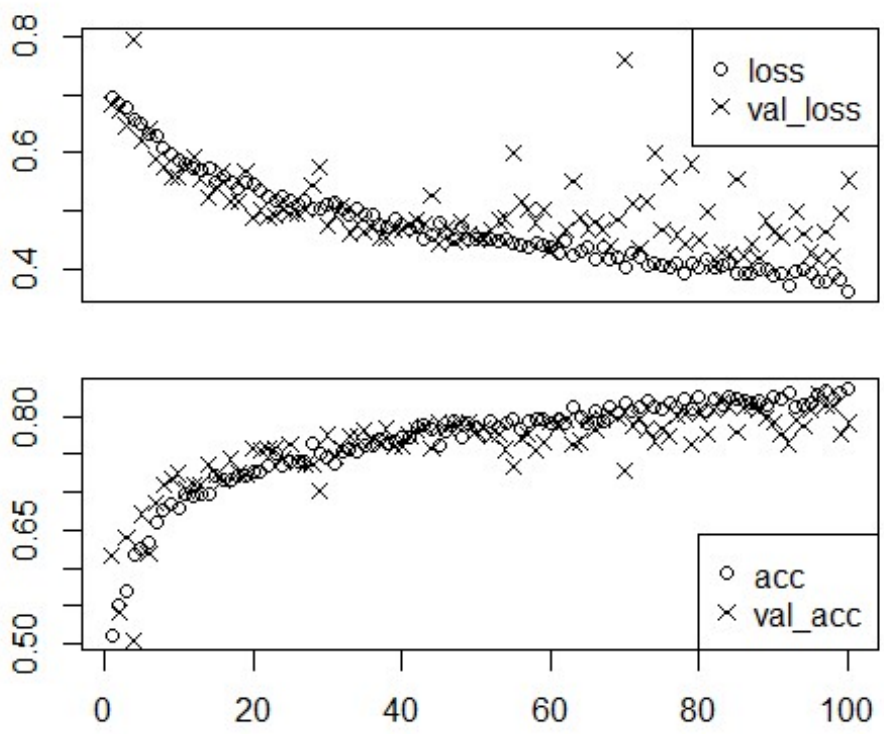
Plot the history

```
str(history)
```

```
## List of 2
##  $ params :List of 3
##   ..$ verbose: int 1
##   ..$ epochs : int 100
##   ..$ steps  : int 100
##  $ metrics:List of 4
##   ..$ loss    : num [1:100] 0.695 0.683 0.677 0.658 0.649 ...
##   ..$ acc     : num [1:100] 0.511 0.55 0.571 0.617 0.626 ...
```

```
##   ..$ val_loss: num [1:100] 0.683 0.675 0.647 0.796 0.619 ...
##   ..$ val_acc : num [1:100] 0.617 0.541 0.641 0.504 0.671 ...
##  - attr(*, "class")= chr "keras_training_history"

plot(history)
```




```
results <- model %>% evaluate(validation_generator)

results

##      loss       acc
## 0.5520354 0.7910000
```

Save the Model

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

==========

----------TASK # 02----------

==========

```
library(keras)

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

```
base_dir <- "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
4/cat vs dog - dataset/md/"

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

train_cats_dir <- file.path(train_dir, "cats")
train_dogs_dir <- file.path(train_dir, "dogs")

validation_cats_dir <- file.path(validation_dir, "cats")
validation_dogs_dir <- file.path(validation_dir, "dogs")

test_cats_dir <- file.path(test_dir, "cats")
test_dogs_dir <- file.path(test_dir, "dogs")
```

Number of cats and dogs images in each directory

USING MORE TRAINING DATA (2000 images for training for each category)

```
cat("total training cat images:", length(list.files(train_cats_dir)), "\n")

## total training cat images: 2000

cat("total training dog images:", length(list.files(train_dogs_dir)), "\n")

## total training dog images: 2000

cat("total validation cat images:", length(list.files(validation_cats_dir)),
"\n")

## total validation cat images: 500

cat("total validation dog images:", length(list.files(validation_dogs_dir)),
"\n")

## total validation dog images: 500

cat("total test cat images:", length(list.files(test_cats_dir)), "\n")

## total test cat images: 500

cat("total test dog images:", length(list.files(test_dogs_dir)), "\n")

## total test dog images: 500
```

Building the model (Using DropOut to reduce overfitting)

```
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3), activation = "relu",
                input_shape = c(150, 150, 3)) %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%
```

```
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu") %>
%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu") %>
%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

## Loaded Tensorflow version 2.6.0

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

## Warning in backcompat_fix_rename_lr_to_learning_rate(...): the `lr` argume
nt has
## been renamed to `learning_rate`.

summary(model)

## Model: "sequential"
## _____
_____
## Layer (type)                        Output Shape                     Param
#
## =====================================================================
======
## conv2d_3 (Conv2D)                   (None, 148, 148, 32)             896
## _____
_____
## max_pooling2d_3 (MaxPooling2D)      (None, 74, 74, 32)               0
## _____
_____
## conv2d_2 (Conv2D)                   (None, 72, 72, 64)               18496
## _____
_____
## max_pooling2d_2 (MaxPooling2D)      (None, 36, 36, 64)               0
## _____
_____
## conv2d_1 (Conv2D)                   (None, 34, 34, 128)              73856
## _____
_____
## max_pooling2d_1 (MaxPooling2D)      (None, 17, 17, 128)              0
## _____
_____
```

```
## conv2d (Conv2D)                        (None, 15, 15, 128)             147584
## _____
_____
## max_pooling2d (MaxPooling2D)           (None, 7, 7, 128)               0
## _____
_____
## flatten (Flatten)                      (None, 6272)                    0
## _____
_____
## dropout (Dropout)                      (None, 6272)                    0
## _____
_____
## dense_1 (Dense)                        (None, 512)                     321177
6
## _____
_____
## dense (Dense)                          (None, 1)                       513
## ================================================================================
======
## Total params: 3,453,121
## Trainable params: 3,453,121
## Non-trainable params: 0
## _____
_____
```

Creating Data Generator

using Data Augmentation Technique to reduce overfitting

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

test_datagen <- image_data_generator(rescale = 1/255)
train_generator <- flow_images_from_directory(
  train_dir,
  datagen,
  target_size = c(150, 150),
  batch_size = 20,
  class_mode = "binary"
)

validation_generator <- flow_images_from_directory(
  validation_dir,
  test_datagen,
```

```
  target_size = c(150, 150),
  batch_size = 20,
  class_mode = "binary"
)



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

```
## List of 2
##  $ : num [1:20, 1:150, 1:150, 1:3] 0.953 0.56 0.504 0.466 0.145 ...
##  $ : num [1:20(1d)] 1 0 1 0 1 1 1 1 0 1 ...
```

Train the model

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

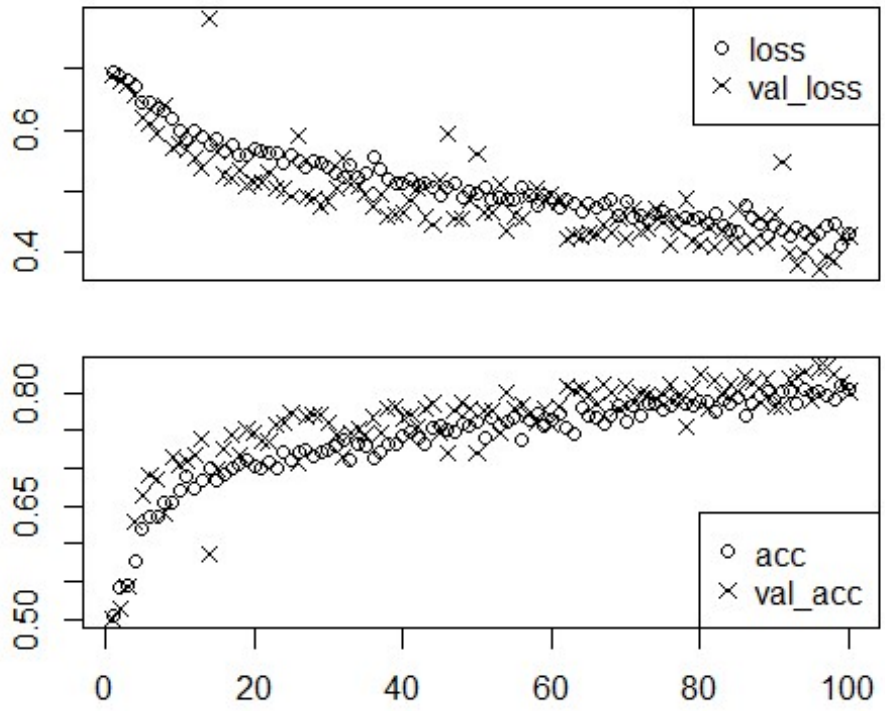
```
## Warning in fit_generator(., train_generator, steps_per_epoch = 100, epochs
## = 100, : `fit_generator` is deprecated. Use `fit` instead, it now accept
## generators.
```

Plot the history

```
str(history)
```

```
## List of 2
##  $ params :List of 3
##   ..$ verbose: int 1
##   ..$ epochs : int 100
##   ..$ steps  : int 100
##  $ metrics:List of 4
##   ..$ loss    : num [1:100] 0.696 0.688 0.683 0.671 0.646 ...
##   ..$ acc     : num [1:100] 0.503 0.543 0.545 0.576 0.618 ...
##   ..$ val_loss: num [1:100] 0.689 0.681 0.674 0.656 0.62 ...
##   ..$ val_acc : num [1:100] 0.5 0.514 0.542 0.629 0.664 ...
##  - attr(*, "class")= chr "keras_training_history"
```

```
plot(history)
```




```
results <- model %>% evaluate(validation_generator)

results

##      loss       acc
## 0.4275039 0.8020000
```

Save the Model

```
model %>% save_model_hdf5("cats_and_dogs_medium_01.h5")
```

==========

————————————-TASK # 03————————————-

==========

```
library(keras)

original_dataset_dir <- "~/Downloads/kaggle_original_data"

base_dir <- "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 04/cat vs dog - dataset/lg/"
```

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

train_cats_dir <- file.path(train_dir, "cats")
train_dogs_dir <- file.path(train_dir, "dogs")

validation_cats_dir <- file.path(validation_dir, "cats")
validation_dogs_dir <- file.path(validation_dir, "dogs")

test_cats_dir <- file.path(test_dir, "cats")
test_dogs_dir <- file.path(test_dir, "dogs")
```

Number of cats and dogs images in each directory

USING MORE TRAINING DATA (2000 images for training for each category)

```
cat("total training cat images:", length(list.files(train_cats_dir)), "\n")

## total training cat images: 5000

cat("total training dog images:", length(list.files(train_dogs_dir)), "\n")

## total training dog images: 5000

cat("total validation cat images:", length(list.files(validation_cats_dir)),
"\n")

## total validation cat images: 1000

cat("total validation dog images:", length(list.files(validation_dogs_dir)),
"\n")

## total validation dog images: 1000

cat("total test cat images:", length(list.files(test_cats_dir)), "\n")

## total test cat images: 1500

cat("total test dog images:", length(list.files(test_dogs_dir)), "\n")

## total test dog images: 1500
```

Building the model (Using DropOut to reduce overfitting)

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

```
  layer_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu") %>
%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

## Loaded Tensorflow version 2.6.0

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

## Warning in backcompat_fix_rename_lr_to_learning_rate(...): the `lr` argume
nt has
## been renamed to `learning_rate`.

summary(model)

## Model: "sequential"
## _____________________________________________________________________________
_____
## Layer (type)                        Output Shape                      Param
#
## =============================================================================
======
## conv2d_3 (Conv2D)                   (None, 148, 148, 32)              896
## _____________________________________________________________________________
_____
## max_pooling2d_3 (MaxPooling2D)      (None, 74, 74, 32)                0
## _____________________________________________________________________________
_____
## conv2d_2 (Conv2D)                   (None, 72, 72, 64)                18496
## _____________________________________________________________________________
_____
## max_pooling2d_2 (MaxPooling2D)      (None, 36, 36, 64)                0
## _____________________________________________________________________________
_____
## conv2d_1 (Conv2D)                   (None, 34, 34, 128)               73856
## _____________________________________________________________________________
_____
## max_pooling2d_1 (MaxPooling2D)      (None, 17, 17, 128)               0
## _____________________________________________________________________________
_____
## conv2d (Conv2D)                     (None, 15, 15, 128)               147584
## _____________________________________________________________________________
_____
## max_pooling2d (MaxPooling2D)        (None, 7, 7, 128)                 0
```

```
## ____________________________________________________________________________
_____
## flatten (Flatten)                   (None, 6272)                    0
## ____________________________________________________________________________
_____
## dropout (Dropout)                   (None, 6272)                    0
## ____________________________________________________________________________
_____
## dense_1 (Dense)                     (None, 512)                     321177
6
## ____________________________________________________________________________
_____
## dense (Dense)                       (None, 1)                       513
## ============================================================================
======
## Total params: 3,453,121
## Trainable params: 3,453,121
## Non-trainable params: 0
## ____________________________________________________________________________
_____
```

Creating Data Generator

using Data Augmentation Technique to reduce overfitting

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

test_datagen <- image_data_generator(rescale = 1/255)
train_generator <- flow_images_from_directory(
  train_dir,
  datagen,
  target_size = c(150, 150),
  batch_size = 20,
  class_mode = "binary"
)

validation_generator <- flow_images_from_directory(
  validation_dir,
  test_datagen,
  target_size = c(150, 150),
  batch_size = 20,
  class_mode = "binary"
)
```

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

```
## List of 2
##  $ : num [1:20, 1:150, 1:150, 1:3] 0.365 0.243 0.522 0.233 0.711 ...
##  $ : num [1:20(1d)] 0 1 0 1 1 0 1 0 1 0 ...
```

Train the model

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

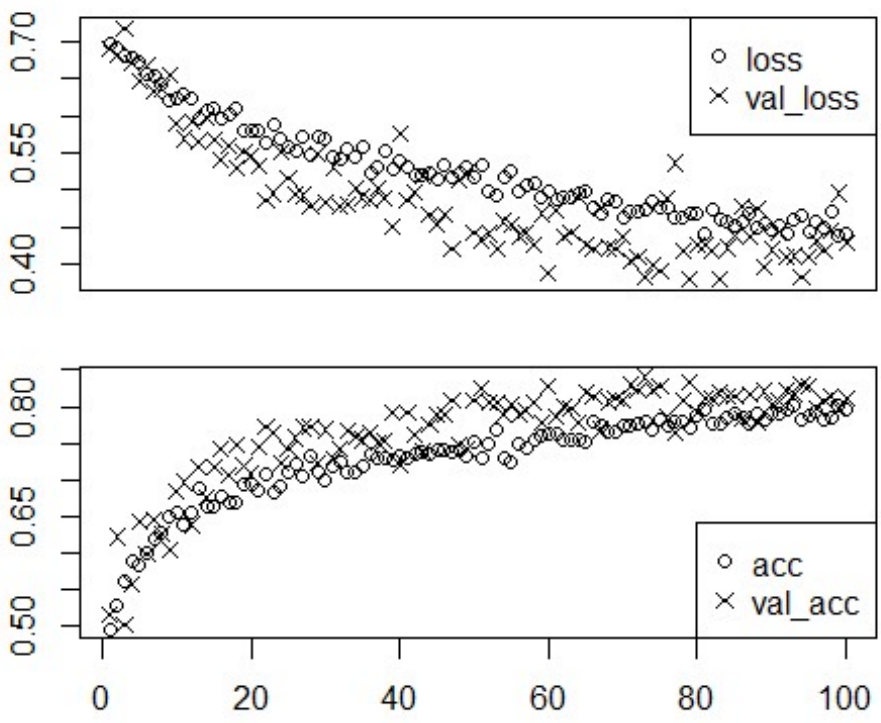
```
## Warning in fit_generator(., train_generator, steps_per_epoch = 100, epochs
## = 100, : `fit_generator` is deprecated. Use `fit` instead, it now accept
## generators.
```

Plot the history

```
str(history)
```

```
## List of 2
##  $ params :List of 3
##   ..$ verbose: int 1
##   ..$ epochs : int 100
##   ..$ steps  : int 100
##  $ metrics:List of 4
##   ..$ loss    : num [1:100] 0.697 0.691 0.68 0.677 0.673 ...
##   ..$ acc     : num [1:100] 0.495 0.527 0.562 0.589 0.582 ...
##   ..$ val_loss: num [1:100] 0.689 0.683 0.718 0.672 0.647 ...
##   ..$ val_acc : num [1:100] 0.515 0.621 0.5 0.557 0.643 ...
##  - attr(*, "class")= chr "keras_training_history"
```

```
plot(history)
```




```
results <- model %>% evaluate(validation_generator)

results

##      loss       acc
## 0.4071001 0.8175000
```

Save the Model

```
model %>% save_model_hdf5("cats_and_dogs_lg_01.h5")
```

==========

----------TASK # 04.1----------

==========

Base Model

```
library(keras)

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

```
## Loaded Tensorflow version 2.6.0

conv_base

## Model
## Model: "vgg16"
## _________________________________________________________________
_______
## Layer (type)                 Output Shape              Param
#
## =================================================================
======
## input_1 (InputLayer)         [(None, 150, 150, 3)]     0
## _________________________________________________________________
_______
## block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
## _________________________________________________________________
_______
## block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
## _________________________________________________________________
_______
## block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
## _________________________________________________________________
_______
## block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
## _________________________________________________________________
_______
## block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
## _________________________________________________________________
_______
## block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
## _________________________________________________________________
_______
## block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168
## _________________________________________________________________
_______
## block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080
## _________________________________________________________________
_______
## block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080
## _________________________________________________________________
_______
## block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0
## _________________________________________________________________
_______
## block4_conv1 (Conv2D)        (None, 18, 18, 512)       118016
0
## _________________________________________________________________
_______
## block4_conv2 (Conv2D)        (None, 18, 18, 512)       235980
```

```
8
## _________________________________________________________________
_____
## block4_conv3 (Conv2D)               (None, 18, 18, 512)             235980
8
## _________________________________________________________________
_____
## block4_pool (MaxPooling2D)          (None, 9, 9, 512)               0
## _________________________________________________________________
_____
## block5_conv1 (Conv2D)               (None, 9, 9, 512)               235980
8
## _________________________________________________________________
_____
## block5_conv2 (Conv2D)               (None, 9, 9, 512)               235980
8
## _________________________________________________________________
_____
## block5_conv3 (Conv2D)               (None, 9, 9, 512)               235980
8
## _________________________________________________________________
_____
## block5_pool (MaxPooling2D)          (None, 4, 4, 512)               0
## =================================================================
======
## Total params: 14,714,688
## Trainable params: 14,714,688
## Non-trainable params: 0
## _________________________________________________________________
_____
```

```
base_dir <- "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
4/cat vs dog - dataset/sm/"
train_dir <- file.path(base_dir, "train")
validation_dir <- file.path(base_dir, "validation")
test_dir <- file.path(base_dir, "test")

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

extract_features <- function(directory, sample_count) {
  features <- array(0, dim = c(sample_count, 4, 4, 512))
  labels <- array(0, dim = c(sample_count))
  generator <- flow_images_from_directory(
    directory = directory,
    generator = datagen,
    target_size = c(150, 150),
    batch_size = batch_size,
    class_mode = "binary"
  )
```

```
  i <- 0
  while(TRUE) {
    batch <- generator_next(generator)
    inputs_batch <- batch[[1]]
    labels_batch <- batch[[2]]
    features_batch <- conv_base %>% predict(inputs_batch)
    index_range <- ((i * batch_size)+1):((i + 1) * batch_size)
    features[index_range,,,] <- features_batch
    labels[index_range] <- labels_batch
    i <- i + 1
    if (i * batch_size >= sample_count)
      break
  }
  list(
    features = features,
    labels = labels
  )
}


train <- extract_features(train_dir, 1000)
validation <- extract_features(validation_dir, 500)
test <- extract_features(test_dir, 500)
```

Reshape the features

```
reshape_features <- function(features) {
  array_reshape(features, dim = c(nrow(features), 4 * 4 * 512))
}

train$features <- reshape_features(train$features)
validation$features <- reshape_features(validation$features)
test$features <- reshape_features(test$features)
```

creating the model

```
model <- keras_model_sequential() %>%
  layer_dense(units = 256, activation = "relu",
              input_shape = 4 * 4 * 512) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 1, activation = "sigmoid")

model %>% compile(
  optimizer = optimizer_rmsprop(lr = 2e-5),
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

## Warning in backcompat_fix_rename_lr_to_learning_rate(...): the `lr` argume
nt has
## been renamed to `learning_rate`.
```

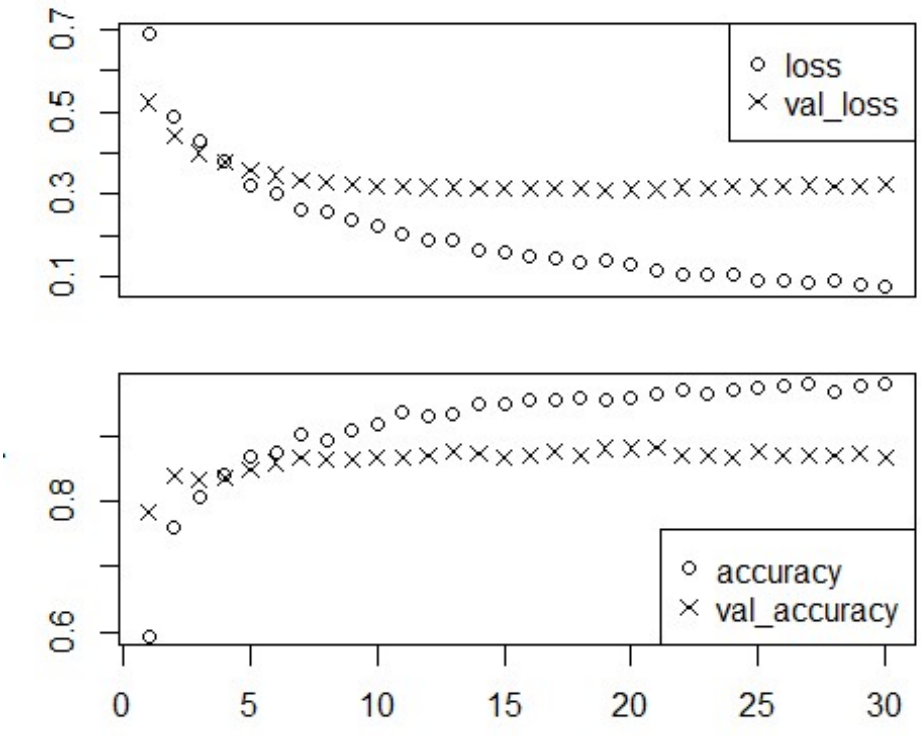
```
history <- model %>% fit(
  train$features, train$labels,
  epochs = 30,
  batch_size = 20,
  validation_data = list(validation$features, validation$labels)
)
```

Plot the history

```
str(history)
```

```
## List of 2
##  $ params :List of 3
##   ..$ verbose: int 1
##   ..$ epochs : int 30
##   ..$ steps  : int 50
##  $ metrics:List of 4
##   ..$ loss        : num [1:30] 0.688 0.488 0.429 0.38 0.32 ...
##   ..$ accuracy    : num [1:30] 0.594 0.761 0.807 0.842 0.87 ...
##   ..$ val_loss    : num [1:30] 0.522 0.441 0.399 0.377 0.359 ...
##   ..$ val_accuracy: num [1:30] 0.784 0.84 0.834 0.836 0.85 ...
##  - attr(*, "class")= chr "keras_training_history"
```

```
plot(history)
```



Save the Model

```
model %>% save_model_hdf5("cats_and_dogs_pretrained_04_1.h5")
```

==========

———————————-TASK # 04.2———————————

==========

Base Model

```
library(keras)

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

## Loaded Tensorflow version 2.6.0

conv_base

## Model
## Model: "vgg16"
## _________________________________________________________________
## Layer (type)                 Output Shape              Param
#
## =================================================================
## input_1 (InputLayer)         [(None, 150, 150, 3)]     0
## _________________________________________________________________
## block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
## _________________________________________________________________
## block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
## _________________________________________________________________
```

```
_____
## block1_pool (MaxPooling2D)          (None, 75, 75, 64)              0
## _____
_____
## block2_conv1 (Conv2D)               (None, 75, 75, 128)             73856
## _____
_____
## block2_conv2 (Conv2D)               (None, 75, 75, 128)             147584
## _____
_____
## block2_pool (MaxPooling2D)          (None, 37, 37, 128)             0
## _____
_____
## block3_conv1 (Conv2D)               (None, 37, 37, 256)             295168
## _____
_____
## block3_conv2 (Conv2D)               (None, 37, 37, 256)             590080
## _____
_____
## block3_conv3 (Conv2D)               (None, 37, 37, 256)             590080
## _____
_____
## block3_pool (MaxPooling2D)          (None, 18, 18, 256)             0
## _____
_____
## block4_conv1 (Conv2D)               (None, 18, 18, 512)             118016
0
## _____
_____
## block4_conv2 (Conv2D)               (None, 18, 18, 512)             235980
8
## _____
_____
## block4_conv3 (Conv2D)               (None, 18, 18, 512)             235980
8
## _____
_____
## block4_pool (MaxPooling2D)          (None, 9, 9, 512)               0
## _____
_____
## block5_conv1 (Conv2D)               (None, 9, 9, 512)               235980
8
## _____
_____
## block5_conv2 (Conv2D)               (None, 9, 9, 512)               235980
8
## _____
_____
## block5_conv3 (Conv2D)               (None, 9, 9, 512)               235980
8
```

```
## ____________________________________________________________________________
_____
## block5_pool (MaxPooling2D)          (None, 4, 4, 512)               0
## ============================================================================
======
## Total params: 14,714,688
## Trainable params: 14,714,688
## Non-trainable params: 0
## ____________________________________________________________________________
_____
```

```r
base_dir <- "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
4/cat vs dog - dataset/md/"
train_dir <- file.path(base_dir, "train")
validation_dir <- file.path(base_dir, "validation")
test_dir <- file.path(base_dir, "test")

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

extract_features <- function(directory, sample_count) {
  features <- array(0, dim = c(sample_count, 4, 4, 512))
  labels <- array(0, dim = c(sample_count))
  generator <- flow_images_from_directory(
    directory = directory,
    generator = datagen,
    target_size = c(150, 150),
    batch_size = batch_size,
    class_mode = "binary"
  )
  i <- 0
  while(TRUE) {
    batch <- generator_next(generator)
    inputs_batch <- batch[[1]]
    labels_batch <- batch[[2]]
    features_batch <- conv_base %>% predict(inputs_batch)
    index_range <- ((i * batch_size)+1):((i + 1) * batch_size)
    features[index_range,,,] <- features_batch
    labels[index_range] <- labels_batch
    i <- i + 1
    if (i * batch_size >= sample_count)
      break
  }
  list(
    features = features,
    labels = labels
  )
}
```

```
train <- extract_features(train_dir, 2000)
validation <- extract_features(validation_dir, 500)
test <- extract_features(test_dir, 500)
```

Reshape the features

```
reshape_features <- function(features) {
  array_reshape(features, dim = c(nrow(features), 4 * 4 * 512))
}

train$features <- reshape_features(train$features)
validation$features <- reshape_features(validation$features)
test$features <- reshape_features(test$features)
```

creating the model

```
model <- keras_model_sequential() %>%
  layer_dense(units = 256, activation = "relu",
              input_shape = 4 * 4 * 512) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 1, activation = "sigmoid")

model %>% compile(
  optimizer = optimizer_rmsprop(lr = 2e-5),
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

## Warning in backcompat_fix_rename_lr_to_learning_rate(...): the `lr` argume
nt has
## been renamed to `learning_rate`.

history <- model %>% fit(
  train$features, train$labels,
  epochs = 30,
  batch_size = 20,
  validation_data = list(validation$features, validation$labels)
)
```

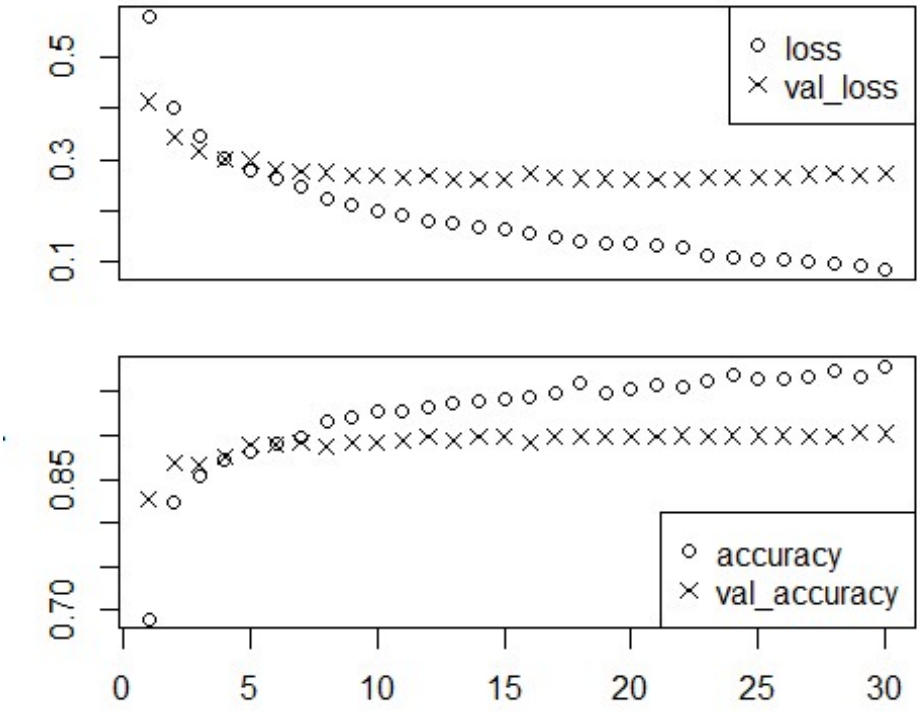
Plot the history

```
str(history)

## List of 2
##  $ params :List of 3
##   ..$ verbose: int 1
##   ..$ epochs : int 30
##   ..$ steps  : int 100
##  $ metrics:List of 4
##   ..$ loss        : num [1:30] 0.579 0.401 0.347 0.304 0.279 ...
##   ..$ accuracy    : num [1:30] 0.69 0.822 0.853 0.872 0.881 ...
##   ..$ val_loss    : num [1:30] 0.413 0.343 0.316 0.299 0.298 ...
```

```
##   ..$ val_accuracy: num [1:30] 0.826 0.868 0.866 0.876 0.89 ...
##  - attr(*, "class")= chr "keras_training_history"

plot(history)
```



Save the Model

```
model %>% save_model_hdf5("cats_and_dogs_pretrained_04_2.h5")
```

==========

———————————-TASK # 04.3———————————

==========

Base Model

```
library(keras)

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

## Loaded Tensorflow version 2.6.0

conv_base

## Model
## Model: "vgg16"
## _________________________________________________________________
## Layer (type)                 Output Shape              Param
#
## =================================================================
## input_1 (InputLayer)         [(None, 150, 150, 3)]     0
## _________________________________________________________________
## block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
## _________________________________________________________________
## block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
## _________________________________________________________________
## block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
## _________________________________________________________________
## block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
## _________________________________________________________________
## block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
## _________________________________________________________________
## block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
## _________________________________________________________________
```

```
_____
## block3_conv1 (Conv2D)               (None, 37, 37, 256)             295168
## _________________________________________________________________
_____
## block3_conv2 (Conv2D)               (None, 37, 37, 256)             590080
## _________________________________________________________________
_____
## block3_conv3 (Conv2D)               (None, 37, 37, 256)             590080
## _________________________________________________________________
_____
## block3_pool (MaxPooling2D)          (None, 18, 18, 256)             0
## _________________________________________________________________
_____
## block4_conv1 (Conv2D)               (None, 18, 18, 512)             118016
0
## _________________________________________________________________
_____
## block4_conv2 (Conv2D)               (None, 18, 18, 512)             235980
8
## _________________________________________________________________
_____
## block4_conv3 (Conv2D)               (None, 18, 18, 512)             235980
8
## _________________________________________________________________
_____
## block4_pool (MaxPooling2D)          (None, 9, 9, 512)               0
## _________________________________________________________________
_____
## block5_conv1 (Conv2D)               (None, 9, 9, 512)               235980
8
## _________________________________________________________________
_____
## block5_conv2 (Conv2D)               (None, 9, 9, 512)               235980
8
## _________________________________________________________________
_____
## block5_conv3 (Conv2D)               (None, 9, 9, 512)               235980
8
## _________________________________________________________________
_____
## block5_pool (MaxPooling2D)          (None, 4, 4, 512)               0
## =================================================================
======
## Total params: 14,714,688
## Trainable params: 14,714,688
## Non-trainable params: 0
## _________________________________________________________________
_____
```

```
base_dir <- "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
4/cat vs dog - dataset/lg/"
train_dir <- file.path(base_dir, "train")
validation_dir <- file.path(base_dir, "validation")
test_dir <- file.path(base_dir, "test")

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

extract_features <- function(directory, sample_count) {
  features <- array(0, dim = c(sample_count, 4, 4, 512))
  labels <- array(0, dim = c(sample_count))
  generator <- flow_images_from_directory(
    directory = directory,
    generator = datagen,
    target_size = c(150, 150),
    batch_size = batch_size,
    class_mode = "binary"
  )
  i <- 0
  while(TRUE) {
    batch <- generator_next(generator)
    inputs_batch <- batch[[1]]
    labels_batch <- batch[[2]]
    features_batch <- conv_base %>% predict(inputs_batch)
    index_range <- ((i * batch_size)+1):((i + 1) * batch_size)
    features[index_range,,,] <- features_batch
    labels[index_range] <- labels_batch
    i <- i + 1
    if (i * batch_size >= sample_count)
      break
  }
  list(
    features = features,
    labels = labels
  )
}


train <- extract_features(train_dir, 5000)
validation <- extract_features(validation_dir, 1000)
test <- extract_features(test_dir, 1500)
```

Reshape the features

```
reshape_features <- function(features) {
  array_reshape(features, dim = c(nrow(features), 4 * 4 * 512))
}

train$features <- reshape_features(train$features)
```

```
validation$features <- reshape_features(validation$features)
test$features <- reshape_features(test$features)
```

creating the model

```
model <- keras_model_sequential() %>%
  layer_dense(units = 256, activation = "relu",
              input_shape = 4 * 4 * 512) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 1, activation = "sigmoid")

model %>% compile(
  optimizer = optimizer_rmsprop(lr = 2e-5),
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)
```

```
## Warning in backcompat_fix_rename_lr_to_learning_rate(...): the `lr` argume
nt has
## been renamed to `learning_rate`.
```

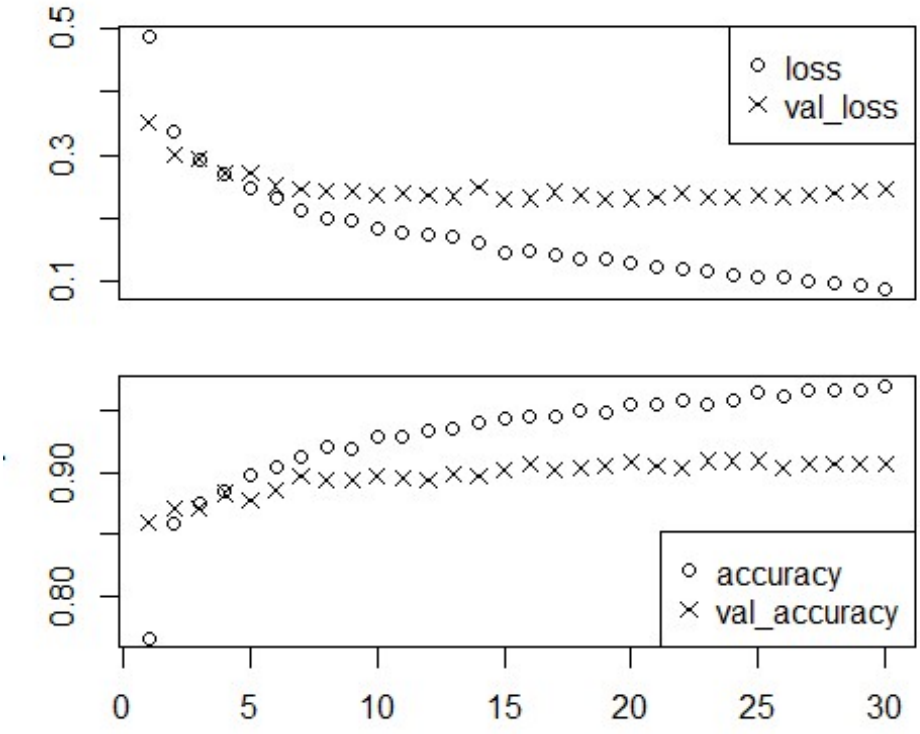
```
history <- model %>% fit(
  train$features, train$labels,
  epochs = 30,
  batch_size = 20,
  validation_data = list(validation$features, validation$labels)
)
```

Plot the history

```
str(history)
```

```
## List of 2
##  $ params :List of 3
##   ..$ verbose: int 1
##   ..$ epochs : int 30
##   ..$ steps  : int 250
##  $ metrics:List of 4
##   ..$ loss        : num [1:30] 0.486 0.336 0.293 0.27 0.247 ...
##   ..$ accuracy    : num [1:30] 0.766 0.858 0.875 0.885 0.898 ...
##   ..$ val_loss    : num [1:30] 0.35 0.301 0.295 0.272 0.271 ...
##   ..$ val_accuracy: num [1:30] 0.859 0.871 0.87 0.883 0.877 ...
##  - attr(*, "class")= chr "keras_training_history"
```

```
plot(history)
```



Save the Model

```
model %>% save_model_hdf5("cats_and_dogs_pretrained_04_3.h5")
```

SUMMARY

	Validation Accuracy	Validation Loss
TASK # 01 Training = 1000 Validation = 500 Test = 500	0.7910000	0.5520354
TASK # 02 Training = 2000 Validation = 500 Test = 500	0.8020000	0.4275039
TASK # 03 Training = 5000 Validation = 1000 Test = 1000	0.8175000	0.4071001
TASK # 04_1 (pre-trained) Training = 1000 Validation = 500 Test = 500	0.85 (approx.)	0.35 (approx.)
TASK # 04_2 (pre-trained) Training = 2000 Validation = 500 Test = 500	0.89 (approx.)	0.30 (approx.)
TASK # 04_3 (pre-trained) Training = 5000 Validation = 1000 Test = 1500	0.91 (approx.)	0.28 (approx.)

As we can see from the above table, the summary shows that as we increase the size of training data the performance increase gradually. As soon as we use the pre-trained network there is a huge performance difference while using the same data set.

Hence, we can say that increasing the amount of data set and using the pre-trained network will give us a better result.