Deep Learning (TensorFlow, Keras) with ResNet50: Image Binary Classifier

In this project, a model is trained to perform binary classifiaction for cats and dogs pictures. The pretrained model ResNet50 is used. This document is the first part of the whole training precess.

```
# (height, width, channels)
input_shape = (224, 224, 3)
batch_size = 8
learning_rate = 1e-4
neurons = 128
path_dataset = '../dataset_cat_dogs'
folder_cat = 'Cat'
folder_dog = 'Dog'
folder_models = '../models'

# Path in Google Colab
# path_dataset = '/content/drive/MyDrive/Colab
Notebooks/dataset_cat_dogs'

# Mount Google Drive if using Google Colab
# from google.colab import drive
# drive.mount('/content/drive/')

import pandas as pd
import matplotlib.pyplot as plt
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping,
ReduceLROnPlateau, ModelCheckpoint

# Find how many cats and dogs images exist
cat_imgs = os.listdir(os.path.join(path_dataset,folder_cat))
dog_imgs = os.listdir(os.path.join(path_dataset,folder_dog))
print(f'Cat images found: {len(cat_imgs)}')
print(f'Dog images found: {len(dog_imgs)}')
```

```
Cat images found: 12491
Dog images found: 12470
```

Classes are balanced.

Data augmentation

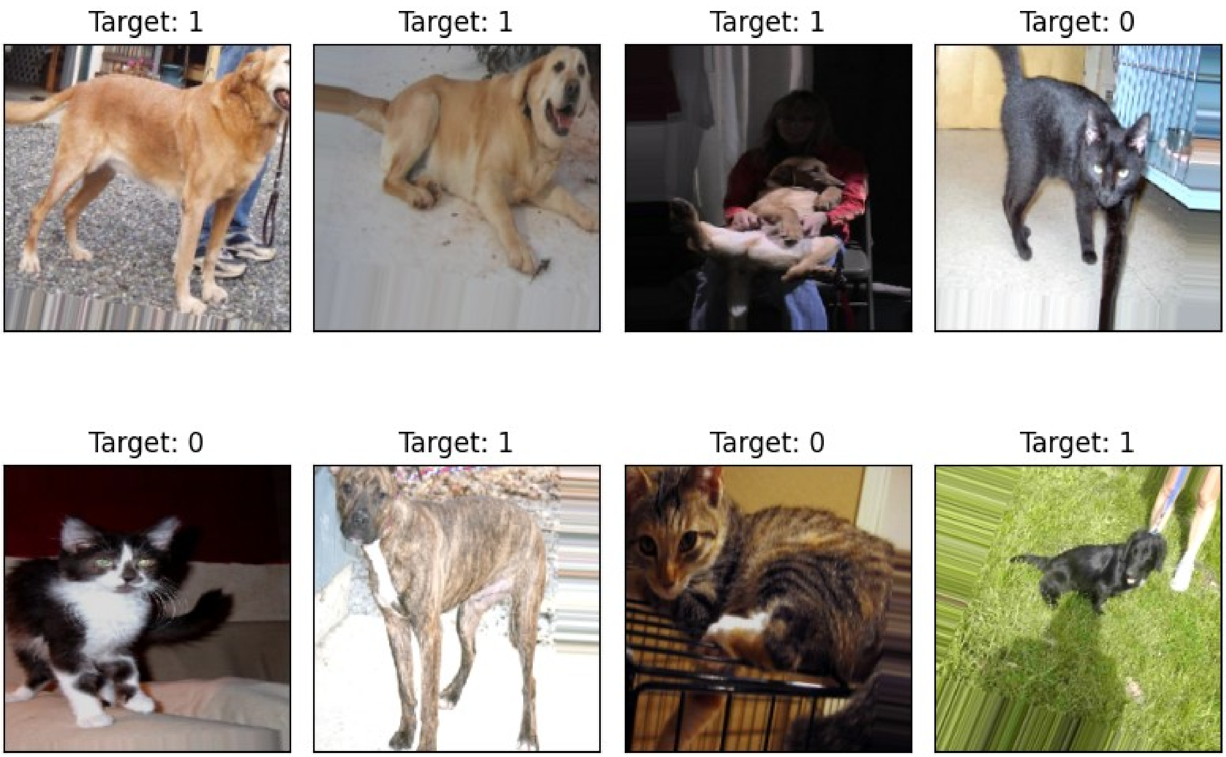
```python
def load_data(path, input_shape=input_shape, batch_size=batch_size,
seed=123, validation_split=0.2):
    """Function to create 2 ImageDataGenerators to split dataset into
train and validation datasets.
    Data augmentation is implemented for the validation dataset."""
    height, width = input_shape[:2]
    datagen = ImageDataGenerator(rescale=1.0/255, zoom_range=0.15,
        horizontal_flip=True, vertical_flip=False,
        height_shift_range=0.15, width_shift_range=0.15,
        brightness_range=(0.8, 1.2), rotation_range=20,
        validation_split=validation_split
    )
    train_data = datagen.flow_from_directory(path,
        target_size=(height, width), batch_size=batch_size,
        class_mode='binary', subset='training', seed=seed
    )
    val_datagen = ImageDataGenerator(rescale=1.0/255,
        validation_split=validation_split
    )
    val_data = val_datagen.flow_from_directory(path,
        target_size=(height, width), batch_size=batch_size,
        class_mode='binary', subset='validation', seed=seed
    )
    return train_data, val_data

# Split training and validation datasets
train, val = load_data(path_dataset)
```

```
Found 19968 images belonging to 2 classes.
Found 4991 images belonging to 2 classes.
```

```python
print(f"Classes found: {train.class_indices}")
print(f"Training images: {train.samples}")
print(f"Validation images: {val.samples}")
```

```
Classes found: {'Cat': 0, 'Dog': 1}
Training images: 19968
Validation images: 4991
```

```python
# Obtain images and target
images, labels = next(train)

# Show 8 training images (batch_size=8)
figure, axes = plt.subplots(nrows=2,ncols=4, figsize=(8, 6))
for item in zip(axes.ravel(), images, labels):
    axes, image, target = item
    axes.imshow(image)
    axes.set_title(f'Target: {target:.0f}')
    axes.set_xticks([])
    axes.set_yticks([])
plt.tight_layout()
plt.show()

# Images dimentions
print(images.shape)
```



```
(8, 224, 224, 3)
```

Iteration 1: Model creation and training with data augmentation (no fine-tuning yet)

```
def create_resnet_model(input_shape=input_shape, neurons=neurons,
                        learning_rate=learning_rate):
    """Function to create the model using the pretrained model
'ResNet50'
    and adding some final layers. The backbone is 'ResNet50',
    but it is freezed (not trained) in this iteration."""

    backbone = ResNet50(weights='imagenet', input_shape=input_shape,
                        include_top=False)

    # Freeze ResNet50 without the top
    backbone.trainable = False
    model = Sequential()
    model.add(backbone)
    model.add(GlobalAveragePooling2D())
    model.add(Dense(neurons, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    optimizer = Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer,
                  loss='binary_crossentropy', metrics=['accuracy'])
    return model

def train_model(model, train_data, val_data, epochs, version_model):
    file_name = f'binary_model_v{version_model}.h5'
    callbacks = [
        EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True, verbose=0),
        ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3,
min_lr=1e-6, verbose=0),
        ModelCheckpoint(file_name, monitor='val_accuracy',
save_best_only=True, verbose=1)
    ]

    history = model.fit(train_data, validation_data=val_data,
              epochs=epochs, callbacks=callbacks, verbose=2)

    return model, history

epochs = 25
version_model = 1
print(f"Parameters: batch_size = {batch_size}, learning_rate =
{learning_rate}, neurons = {neurons}, epochs = {epochs}")
```

```
Parameters: batch_size = 8, learning_rate = 0.0001, neurons = 128,
epochs = 25
```

```
# Create and train the model v1
model = create_resnet_model()
model, history_stage1 = train_model(model, train, val, epochs=epochs,
version_model=version_model)
```

```
Epoch 1/25
```

```
/home/ant/TensorFlow-Keras-ResNet50-InceptionV3/env/lib/python3.8/
site-packages/PIL/TiffImagePlugin.py:900: UserWarning: Truncated File
Read
  warnings.warn(str(msg))
```

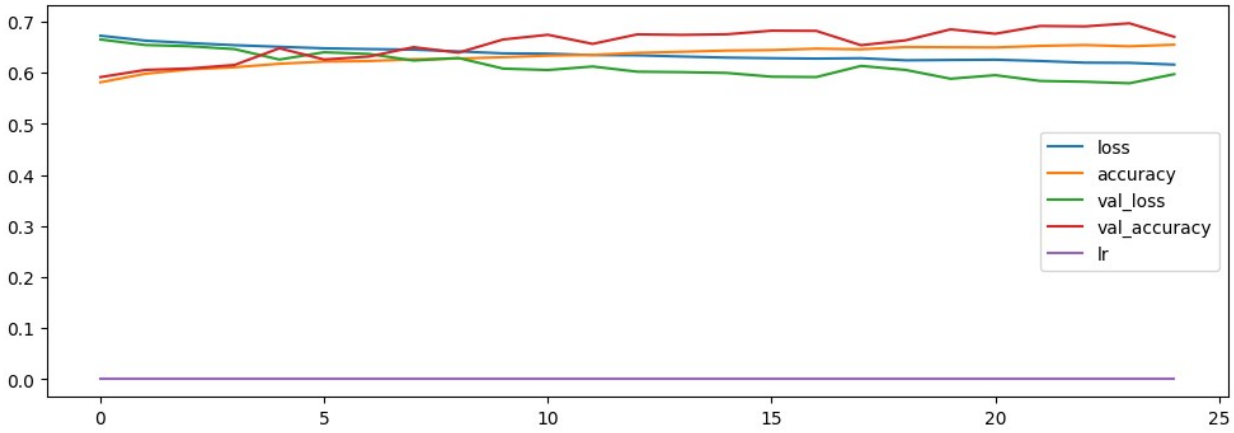
```
Epoch 1: val_accuracy improved from -inf to 0.59166, saving model to
binary_model_v1.h5
```

```
/home/ant/TensorFlow-Keras-ResNet50-InceptionV3/env/lib/python3.8/
site-packages/keras/src/engine/training.py:3000: UserWarning: You are
```

```
saving your model as an HDF5 file via `model.save()`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(

2496/2496 - 2267s - loss: 0.6728 - accuracy: 0.5815 - val_loss: 0.6654
- val_accuracy: 0.5917 - lr: 1.0000e-04 - 2267s/epoch - 908ms/step
Epoch 2/25

Epoch 2: val_accuracy improved from 0.59166 to 0.60569, saving model
to binary_model_v1.h5
2496/2496 - 2508s - loss: 0.6632 - accuracy: 0.5983 - val_loss: 0.6546
- val_accuracy: 0.6057 - lr: 1.0000e-04 - 2508s/epoch - 1s/step
Epoch 3/25

Epoch 3: val_accuracy improved from 0.60569 to 0.60870, saving model
to binary_model_v1.h5
2496/2496 - 2465s - loss: 0.6583 - accuracy: 0.6068 - val_loss: 0.6522
- val_accuracy: 0.6087 - lr: 1.0000e-04 - 2465s/epoch - 988ms/step
Epoch 4/25

Epoch 4: val_accuracy improved from 0.60870 to 0.61551, saving model
to binary_model_v1.h5
2496/2496 - 2427s - loss: 0.6542 - accuracy: 0.6110 - val_loss: 0.6465
- val_accuracy: 0.6155 - lr: 1.0000e-04 - 2427s/epoch - 972ms/step
Epoch 5/25

Epoch 5: val_accuracy improved from 0.61551 to 0.64817, saving model
to binary_model_v1.h5
2496/2496 - 2314s - loss: 0.6511 - accuracy: 0.6178 - val_loss: 0.6261
- val_accuracy: 0.6482 - lr: 1.0000e-04 - 2314s/epoch - 927ms/step
Epoch 6/25

Epoch 6: val_accuracy did not improve from 0.64817
2496/2496 - 2329s - loss: 0.6481 - accuracy: 0.6221 - val_loss: 0.6404
- val_accuracy: 0.6261 - lr: 1.0000e-04 - 2329s/epoch - 933ms/step
Epoch 7/25

Epoch 7: val_accuracy did not improve from 0.64817
2496/2496 - 40866s - loss: 0.6466 - accuracy: 0.6231 - val_loss:
0.6370 - val_accuracy: 0.6319 - lr: 1.0000e-04 - 40866s/epoch -
16s/step
Epoch 8/25

Epoch 8: val_accuracy improved from 0.64817 to 0.65037, saving model
to binary_model_v1.h5
2496/2496 - 2150s - loss: 0.6455 - accuracy: 0.6267 - val_loss: 0.6242
- val_accuracy: 0.6504 - lr: 1.0000e-04 - 2150s/epoch - 861ms/step
Epoch 9/25
Epoch 9: val_accuracy did not improve from 0.65037
2496/2496 - 2484s - loss: 0.6420 - accuracy: 0.6280 - val_loss: 0.6292
- val_accuracy: 0.6396 - lr: 1.0000e-04 - 2484s/epoch - 995ms/step
Epoch 10/25

Epoch 10: val_accuracy improved from 0.65037 to 0.66540, saving model
to binary_model_v1.h5
2496/2496 - 2513s - loss: 0.6382 - accuracy: 0.6308 - val_loss: 0.6084
- val_accuracy: 0.6654 - lr: 1.0000e-04 - 2513s/epoch - 1s/step
Epoch 11/25

Epoch 11: val_accuracy improved from 0.66540 to 0.67461, saving model
to binary_model_v1.h5
2496/2496 - 3152s - loss: 0.6376 - accuracy: 0.6337 - val_loss: 0.6056
- val_accuracy: 0.6746 - lr: 1.0000e-04 - 3152s/epoch - 1s/step
Epoch 12/25

Epoch 12: val_accuracy did not improve from 0.67461
2496/2496 - 2301s - loss: 0.6349 - accuracy: 0.6352 - val_loss: 0.6125
- val_accuracy: 0.6568 - lr: 1.0000e-04 - 2301s/epoch - 922ms/step
Epoch 13/25

Epoch 13: val_accuracy improved from 0.67461 to 0.67542, saving model
to binary_model_v1.h5
2496/2496 - 2261s - loss: 0.6343 - accuracy: 0.6392 - val_loss: 0.6023
- val_accuracy: 0.6754 - lr: 1.0000e-04 - 2261s/epoch - 906ms/step
Epoch 14/25

Epoch 14: val_accuracy did not improve from 0.67542
2496/2496 - 2272s - loss: 0.6320 - accuracy: 0.6413 - val_loss: 0.6015
- val_accuracy: 0.6744 - lr: 1.0000e-04 - 2272s/epoch - 910ms/step
Epoch 15/25

Epoch 15: val_accuracy improved from 0.67542 to 0.67562, saving model
to binary_model_v1.h5
2496/2496 - 2305s - loss: 0.6299 - accuracy: 0.6437 - val_loss: 0.5998
- val_accuracy: 0.6756 - lr: 1.0000e-04 - 2305s/epoch - 923ms/step
Epoch 16/25

Epoch 16: val_accuracy improved from 0.67562 to 0.68263, saving model
to binary_model_v1.h5
2496/2496 - 2299s - loss: 0.6288 - accuracy: 0.6446 - val_loss: 0.5925
- val_accuracy: 0.6826 - lr: 1.0000e-04 - 2299s/epoch - 921ms/step
Epoch 17/25

Epoch 17: val_accuracy did not improve from 0.68263
2496/2496 - 2334s - loss: 0.6279 - accuracy: 0.6474 - val_loss: 0.5918
- val_accuracy: 0.6822 - lr: 1.0000e-04 - 2334s/epoch - 935ms/step
Epoch 18/25
Epoch 18: val_accuracy did not improve from 0.68263
2496/2496 - 2426s - loss: 0.6287 - accuracy: 0.6461 - val_loss: 0.6137
- val_accuracy: 0.6542 - lr: 1.0000e-04 - 2426s/epoch - 972ms/step
Epoch 19/25

Epoch 19: val_accuracy did not improve from 0.68263
2496/2496 - 2375s - loss: 0.6246 - accuracy: 0.6506 - val_loss: 0.6057
- val_accuracy: 0.6638 - lr: 1.0000e-04 - 2375s/epoch - 951ms/step
Epoch 20/25

Epoch 20: val_accuracy improved from 0.68263 to 0.68523, saving model
to binary_model_v1.h5
2496/2496 - 2342s - loss: 0.6253 - accuracy: 0.6502 - val_loss: 0.5885
- val_accuracy: 0.6852 - lr: 1.0000e-04 - 2342s/epoch - 938ms/step
Epoch 21/25

Epoch 21: val_accuracy did not improve from 0.68523
2496/2496 - 2472s - loss: 0.6258 - accuracy: 0.6498 - val_loss: 0.5955
- val_accuracy: 0.6766 - lr: 1.0000e-04 - 2472s/epoch - 990ms/step
Epoch 22/25

Epoch 22: val_accuracy improved from 0.68523 to 0.69185, saving model
to binary_model_v1.h5
2496/2496 - 3582s - loss: 0.6233 - accuracy: 0.6528 - val_loss: 0.5842
- val_accuracy: 0.6918 - lr: 1.0000e-04 - 3582s/epoch - 1s/step
Epoch 23/25

Epoch 23: val_accuracy did not improve from 0.69185
2496/2496 - 3072s - loss: 0.6200 - accuracy: 0.6545 - val_loss: 0.5826
- val_accuracy: 0.6910 - lr: 1.0000e-04 - 3072s/epoch - 1s/step
Epoch 24/25

Epoch 24: val_accuracy improved from 0.69185 to 0.69705, saving model
to binary_model_v1.h5
2496/2496 - 4016s - loss: 0.6196 - accuracy: 0.6520 - val_loss: 0.5796
- val_accuracy: 0.6971 - lr: 1.0000e-04 - 4016s/epoch - 2s/step
Epoch 25/25

Epoch 25: val_accuracy did not improve from 0.69705
2496/2496 - 3274s - loss: 0.6162 - accuracy: 0.6551 - val_loss: 0.5974
- val_accuracy: 0.6706 - lr: 1.0000e-04 - 3274s/epoch - 1s/step
```

Result 1: val_accuracy=69%.

```
pd.DataFrame(history_stage1.history).plot(figsize=(12, 4))

<Axes: >
```



```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 128)               262272

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 23850113 (90.98 MB)
Trainable params: 262401 (1.00 MB)
Non-trainable params: 23587712 (89.98 MB)
_________________________________________________________________

# Save model
#
model.save(os.path.join(folder_models,f'binary_model_v{version_model}.
keras'))

import tensorflow as tf
print(f"TensorFlow Version: {tf.__version__}")

TensorFlow Version: 2.13.1
```