Confidential version 2

# Deep Learning (TensorFlow, Keras) with ResNet50: Binary Classifier (part 2)

In this document, a model is trained to perform binary classifiaction for cats and dogs pictures by using the pretrained model ResNet50. This is part 2 of trainig.

```
# (height, width, channels)
input_shape = (224, 224, 3)
batch_size = 8
learning_rate = 0.001
neurons = 128
path_dataset = 'dataset_cat_dogs'
folder_cat = 'Cat'
folder_dog = 'Dog'
folder_models = 'models'

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
from tensorflow.keras.models import load_model
```

```
2025-09-22 21:06:09.905212: I tensorflow/tsl/cuda/cudart_stub.cc:28]
Could not find cuda drivers on your machine, GPU will not be used.
2025-09-22 21:06:14.762866: I tensorflow/tsl/cuda/cudart_stub.cc:28]
Could not find cuda drivers on your machine, GPU will not be used.
2025-09-22 21:06:14.782090: I
tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow
binary is optimized to use available CPU instructions in performance-
critical operations.
To enable the following instructions: AVX2 FMA, in other operations,
rebuild TensorFlow with the appropriate compiler flags.
2025-09-22 21:06:22.436290: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning:
Could not find TensorRT
```

## Data augmentation

```
def load_data(path, input_shape=input_shape, batch_size=batch_size,
              seed=123, validation_split=0.2):
    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)

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

```
Found 19968 images belonging to 2 classes.
Found 4991 images belonging to 2 classes.
Classes found: {'Cat': 0, 'Dog': 1}
Training images: 19968
Validation images: 4991
```

```
# 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)
```

```
def train_model(model, train_data, val_data, epochs, version_model,
folder_models=folder_models):
    file_name =
os.path.join(folder_models,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_loss',
save_best_only=True, verbose=1)
    ]

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

    return model, history
```

Iteration 2: Data augmentation, learning_rate = 1e-3, without fine-tunning

```
epochs = 20
version_model = 2
print(f"Parameters: batch_size = {batch_size}, learning_rate =
{learning_rate}, neurons = {neurons}, epochs = {epochs}")
```

```
Parameters: batch_size = 8, learning_rate = 0.001, neurons = 128,
epochs = 20
```

```
# Load model v1
model_v2 =
load_model(os.path.join(folder_models,'binary_model_v1.h5'))
model_v2.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)
_________________________________________________________________
```

```
# Recompile
model_v2.compile(optimizer=Adam(learning_rate=learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])

# Retrain
model_v2, history_stage2 = train_model(model_v2, train, val,
epochs=epochs,
                                       version_model=version_model,
folder_models=folder_models)
```

```
Epoch 1/20

2025-09-22 21:10:33.966510: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
25690112 exceeds 10% of free system memory.
2025-09-22 21:10:34.643891: W
```

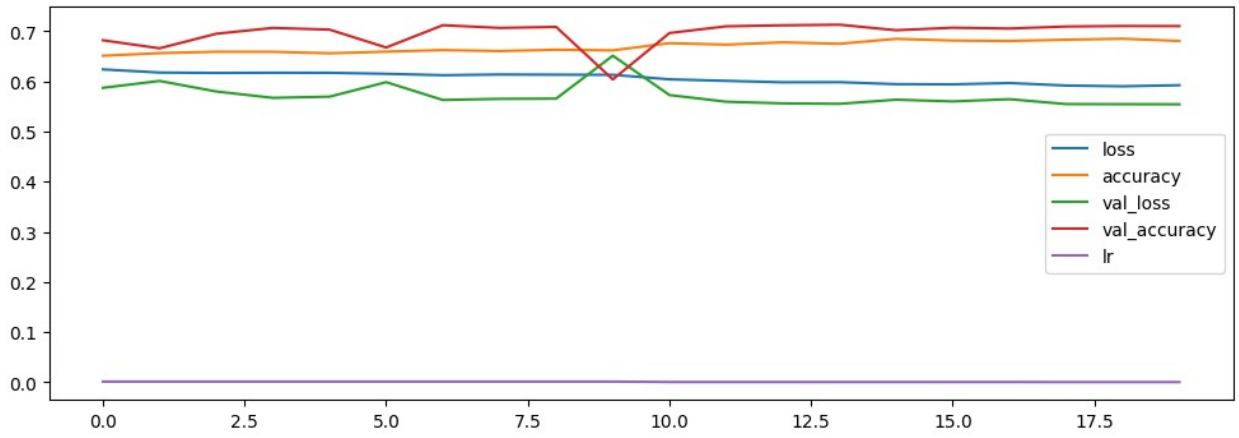
```
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
25690112 exceeds 10% of free system memory.
2025-09-22 21:10:34.680975: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
26615808 exceeds 10% of free system memory.
2025-09-22 21:10:34.706337: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
25690112 exceeds 10% of free system memory.
2025-09-22 21:10:34.717892: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
25690112 exceeds 10% of free system memory.
/home/ant/tensorflow3/env/lib/python3.8/site-packages/PIL/TiffImagePlu
gin.py:900: UserWarning: Truncated File Read
  warnings.warn(str(msg))


Epoch 1: val_loss improved from inf to 0.58728, saving model to
models/binary_model_v2.h5

/home/ant/tensorflow3/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 - 1844s - loss: 0.6241 - accuracy: 0.6513 - val_loss: 0.5873
- val_accuracy: 0.6822 - lr: 0.0010 - 1844s/epoch - 739ms/step
Epoch 2/20

Epoch 2: val_loss did not improve from 0.58728
2496/2496 - 1966s - loss: 0.6178 - accuracy: 0.6565 - val_loss: 0.6011
- val_accuracy: 0.6662 - lr: 0.0010 - 1966s/epoch - 788ms/step
Epoch 3/20

Epoch 3: val_loss improved from 0.58728 to 0.58000, saving model to
models/binary_model_v2.h5
2496/2496 - 1933s - loss: 0.6170 - accuracy: 0.6594 - val_loss: 0.5800
- val_accuracy: 0.6953 - lr: 0.0010 - 1933s/epoch - 774ms/step
Epoch 4/20

Epoch 4: val_loss improved from 0.58000 to 0.56732, saving model to
models/binary_model_v2.h5
2496/2496 - 3212s - loss: 0.6174 - accuracy: 0.6593 - val_loss: 0.5673
- val_accuracy: 0.7069 - lr: 0.0010 - 3212s/epoch - 1s/step
Epoch 5/20

Epoch 5: val_loss did not improve from 0.56732
2496/2496 - 1940s - loss: 0.6173 - accuracy: 0.6562 - val_loss: 0.5697
- val_accuracy: 0.7035 - lr: 0.0010 - 1940s/epoch - 777ms/step
```

```
Epoch 6/20

Epoch 6: val_loss did not improve from 0.56732
2496/2496 - 1800s - loss: 0.6153 - accuracy: 0.6595 - val_loss: 0.5986
- val_accuracy: 0.6678 - lr: 0.0010 - 1800s/epoch - 721ms/step
Epoch 7/20

Epoch 7: val_loss improved from 0.56732 to 0.56308, saving model to
models/binary_model_v2.h5
2496/2496 - 1780s - loss: 0.6125 - accuracy: 0.6627 - val_loss: 0.5631
- val_accuracy: 0.7123 - lr: 0.0010 - 1780s/epoch - 713ms/step
Epoch 8/20

Epoch 8: val_loss did not improve from 0.56308
2496/2496 - 1773s - loss: 0.6139 - accuracy: 0.6607 - val_loss: 0.5653
- val_accuracy: 0.7069 - lr: 0.0010 - 1773s/epoch - 710ms/step
Epoch 9/20

Epoch 9: val_loss did not improve from 0.56308
2496/2496 - 1771s - loss: 0.6135 - accuracy: 0.6634 - val_loss: 0.5658
- val_accuracy: 0.7089 - lr: 0.0010 - 1771s/epoch - 709ms/step
Epoch 10/20

Epoch 10: val_loss did not improve from 0.56308
2496/2496 - 1784s - loss: 0.6133 - accuracy: 0.6621 - val_loss: 0.6514
- val_accuracy: 0.6041 - lr: 0.0010 - 1784s/epoch - 715ms/step
Epoch 11/20

Epoch 11: val_loss did not improve from 0.56308
2496/2496 - 1767s - loss: 0.6043 - accuracy: 0.6764 - val_loss: 0.5729
- val_accuracy: 0.6967 - lr: 2.0000e-04 - 1767s/epoch - 708ms/step
Epoch 12/20

Epoch 12: val_loss improved from 0.56308 to 0.55957, saving model to
models/binary_model_v2.h5
2496/2496 - 1767s - loss: 0.6013 - accuracy: 0.6736 - val_loss: 0.5596
- val_accuracy: 0.7101 - lr: 2.0000e-04 - 1767s/epoch - 708ms/step
Epoch 13/20

Epoch 13: val_loss improved from 0.55957 to 0.55631, saving model to
models/binary_model_v2.h5
2496/2496 - 1754s - loss: 0.5984 - accuracy: 0.6781 - val_loss: 0.5563
- val_accuracy: 0.7121 - lr: 2.0000e-04 - 1754s/epoch - 703ms/step
Epoch 14/20

Epoch 14: val_loss improved from 0.55631 to 0.55535, saving model to
models/binary_model_v2.h5
2496/2496 - 1745s - loss: 0.5986 - accuracy: 0.6752 - val_loss: 0.5554
- val_accuracy: 0.7133 - lr: 2.0000e-04 - 1745s/epoch - 699ms/step
Epoch 15/20
```

```
Epoch 15: val_loss did not improve from 0.55535
2496/2496 - 1716s - loss: 0.5947 - accuracy: 0.6851 - val_loss: 0.5637
- val_accuracy: 0.7023 - lr: 2.0000e-04 - 1716s/epoch - 688ms/step
Epoch 16/20

Epoch 16: val_loss did not improve from 0.55535
2496/2496 - 1721s - loss: 0.5943 - accuracy: 0.6817 - val_loss: 0.5602
- val_accuracy: 0.7073 - lr: 2.0000e-04 - 1721s/epoch - 689ms/step
Epoch 17/20

Epoch 17: val_loss did not improve from 0.55535
2496/2496 - 1714s - loss: 0.5969 - accuracy: 0.6807 - val_loss: 0.5647
- val_accuracy: 0.7057 - lr: 2.0000e-04 - 1714s/epoch - 687ms/step
Epoch 18/20

Epoch 18: val_loss improved from 0.55535 to 0.55490, saving model to
models/binary_model_v2.h5
2496/2496 - 1702s - loss: 0.5919 - accuracy: 0.6833 - val_loss: 0.5549
- val_accuracy: 0.7097 - lr: 4.0000e-05 - 1702s/epoch - 682ms/step
Epoch 19/20

Epoch 19: val_loss improved from 0.55490 to 0.55467, saving model to
models/binary_model_v2.h5
2496/2496 - 1715s - loss: 0.5903 - accuracy: 0.6853 - val_loss: 0.5547
- val_accuracy: 0.7107 - lr: 4.0000e-05 - 1715s/epoch - 687ms/step
Epoch 20/20

Epoch 20: val_loss improved from 0.55467 to 0.55443, saving model to
models/binary_model_v2.h5
2496/2496 - 1705s - loss: 0.5926 - accuracy: 0.6807 - val_loss: 0.5544
- val_accuracy: 0.7105 - lr: 4.0000e-05 - 1705s/epoch - 683ms/step

model_v2.save(f'binary_model_v{version_model}.keras')
```

**Result 2:** val_acc = 71%.

```
pd.DataFrame(history_stage2.history).plot(figsize=(12, 4))

<Axes: >
```




```
# last 20 layers
for layer in model.layers[-20:]:
    layer.trainable = True

# recompile with learning _rate smaller
model.compile(optimizer=Adam(1e-5), loss='binary_crossentropy',
metrics=['accuracy'])

# entrenar fine-tuning
model, history_stage2 = train_model(model, train, val, epochs=5,
version_model=2)
model.save('final_resnet50_v2.keras')
```