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 3 of the trainig process.

Iteration 2: Fine-tuning, data augmentation and learning_rate=1e-5 (smaller)

```
# (height, width, channels)
input_shape = (224, 224, 3)
batch_size = 8
learning_rate = 1e-5
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
import tensorflow as tf

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

# Ensure GPU is available
physical_devices = tf.config.list_physical_devices('GPU')
if len(physical_devices) > 0:
    tf.config.experimental.set_memory_growth(physical_devices[0],
True)
    print("GPU is available and memory growth is enabled.")
else:
    print("GPU not available, training will be on CPU.")
```

```
TensorFlow Version: 2.13.1
GPU not available, training will be on CPU.
```

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
```

Model retraining with fine-tuning

```
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

# Load model v2
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)
_________________________________________________________________
```

```
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 = 1e-05, neurons = 128,
epochs = 20
```

```
# last 20 layers
for layer in model_v2.layers[0].layers[-20:]:
    layer.trainable = True

# 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

2025-09-28 23:22:23.053430: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
25690112 exceeds 10% of free system memory.
2025-09-28 23:22:23.786025: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
25690112 exceeds 10% of free system memory.
2025-09-28 23:22:23.817266: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
26615808 exceeds 10% of free system memory.
2025-09-28 23:22:23.837559: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
25690112 exceeds 10% of free system memory.
2025-09-28 23:22:23.852126: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
25690112 exceeds 10% of free system memory.
/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_loss improved from inf to 0.53716, saving model to
../models/binary_model_v2.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 - 3103s - loss: 0.6324 - accuracy: 0.6555 - val_loss: 0.5372
- val_accuracy: 0.7273 - lr: 1.0000e-05 - 3103s/epoch - 1s/step
Epoch 2/20

Epoch 2: val_loss did not improve from 0.53716
2496/2496 - 2299s - loss: 0.5888 - accuracy: 0.6864 - val_loss: 0.5459
- val_accuracy: 0.7219 - lr: 1.0000e-05 - 2299s/epoch - 921ms/step
Epoch 3/20

Epoch 3: val_loss improved from 0.53716 to 0.50976, saving model
to ../models/binary_model_v2.h5
2496/2496 - 2281s - loss: 0.5722 - accuracy: 0.7014 - val_loss: 0.5098
- val_accuracy: 0.7441 - lr: 1.0000e-05 - 2281s/epoch - 914ms/step
Epoch 4/20
```

```
Epoch 4: val_loss did not improve from 0.50976
2496/2496 - 2284s - loss: 0.5611 - accuracy: 0.7113 - val_loss: 0.5772
- val_accuracy: 0.6904 - lr: 1.0000e-05 - 2284s/epoch - 915ms/step
Epoch 5/20

Epoch 5: val_loss did not improve from 0.50976
2496/2496 - 2299s - loss: 0.5532 - accuracy: 0.7151 - val_loss: 0.5524
- val_accuracy: 0.7219 - lr: 1.0000e-05 - 2299s/epoch - 921ms/step
Epoch 6/20

Epoch 6: val_loss did not improve from 0.50976
2496/2496 - 2264s - loss: 0.5438 - accuracy: 0.7211 - val_loss: 0.5101
- val_accuracy: 0.7578 - lr: 1.0000e-05 - 2264s/epoch - 907ms/step
Epoch 7/20

Epoch 7: val_loss improved from 0.50976 to 0.46507, saving model
to ../models/binary_model_v2.h5
2496/2496 - 2249s - loss: 0.5179 - accuracy: 0.7404 - val_loss: 0.4651
- val_accuracy: 0.7812 - lr: 2.0000e-06 - 2249s/epoch - 901ms/step
Epoch 8/20

Epoch 8: val_loss improved from 0.46507 to 0.42941, saving model
to ../models/binary_model_v2.h5
2496/2496 - 2232s - loss: 0.5095 - accuracy: 0.7485 - val_loss: 0.4294
- val_accuracy: 0.7980 - lr: 2.0000e-06 - 2232s/epoch - 894ms/step
Epoch 9/20

Epoch 9: val_loss improved from 0.42941 to 0.42757, saving model
to ../models/binary_model_v2.h5
2496/2496 - 2230s - loss: 0.5023 - accuracy: 0.7479 - val_loss: 0.4276
- val_accuracy: 0.7962 - lr: 2.0000e-06 - 2230s/epoch - 893ms/step
Epoch 10/20

Epoch 10: val_loss did not improve from 0.42757
2496/2496 - 2223s - loss: 0.5007 - accuracy: 0.7516 - val_loss: 0.4337
- val_accuracy: 0.7994 - lr: 2.0000e-06 - 2223s/epoch - 891ms/step
Epoch 11/20

Epoch 11: val_loss improved from 0.42757 to 0.42463, saving model
to ../models/binary_model_v2.h5
2496/2496 - 2262s - loss: 0.4952 - accuracy: 0.7567 - val_loss: 0.4246
- val_accuracy: 0.7994 - lr: 2.0000e-06 - 2262s/epoch - 906ms/step
Epoch 12/20

Epoch 12: val_loss did not improve from 0.42463
2496/2496 - 2271s - loss: 0.4912 - accuracy: 0.7572 - val_loss: 0.4542
- val_accuracy: 0.7802 - lr: 2.0000e-06 - 2271s/epoch - 910ms/step
Epoch 13/20

Epoch 13: val_loss did not improve from 0.42463
```

```
2496/2496 - 2156s - loss: 0.4949 - accuracy: 0.7589 - val_loss: 0.4509
- val_accuracy: 0.7802 - lr: 2.0000e-06 - 2156s/epoch - 864ms/step
Epoch 14/20

Epoch 14: val_loss did not improve from 0.42463
2496/2496 - 2835s - loss: 0.4869 - accuracy: 0.7619 - val_loss: 0.6179
- val_accuracy: 0.7035 - lr: 2.0000e-06 - 2835s/epoch - 1s/step
Epoch 15/20

Epoch 15: val_loss did not improve from 0.42463
2496/2496 - 2759s - loss: 0.4814 - accuracy: 0.7649 - val_loss: 0.4386
- val_accuracy: 0.7986 - lr: 1.0000e-06 - 2759s/epoch - 1s/step
Epoch 16/20

Epoch 16: val_loss did not improve from 0.42463
2496/2496 - 3071s - loss: 0.4755 - accuracy: 0.7688 - val_loss: 0.4253
- val_accuracy: 0.8071 - lr: 1.0000e-06 - 3071s/epoch - 1s/step
```

Result 2: The training was stopped automtcally after 5 epochs without decreasing the 'val_loss'. This corresponds to the following line:

```
EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True, verbose=0)
```

Finally, the val_accuracy = 80%. One last iteration will be needed.

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