Confidential version 3

Deep Learning (TensorFlow, Keras) with ResNet50: Binary Classifier (part 3)

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 3: Fine-tuning 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
```

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_v3 =
load_model(os.path.join(folder_models,'binary_model_v2.h5'))
model_v3.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 = 3
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_v3.layers[0].layers[-20:]:
    layer.trainable = True

# Recompile
model_v3.compile(optimizer=Adam(learning_rate=learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])

# Retrain
model_v3, history_stage3 = train_model(model_v3, train, val,
epochs=epochs,
                                       version_model=version_model,
folder_models=folder_models)
```

```
Epoch 1/20
```

```
2025-09-26 16:57:38.524852: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
26615808 exceeds 10% of free system memory.
2025-09-26 16:57:39.745795: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
26615808 exceeds 10% of free system memory.
2025-09-26 16:57:40.605342: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
26615808 exceeds 10% of free system memory.
2025-09-26 16:57:41.569064: W
```

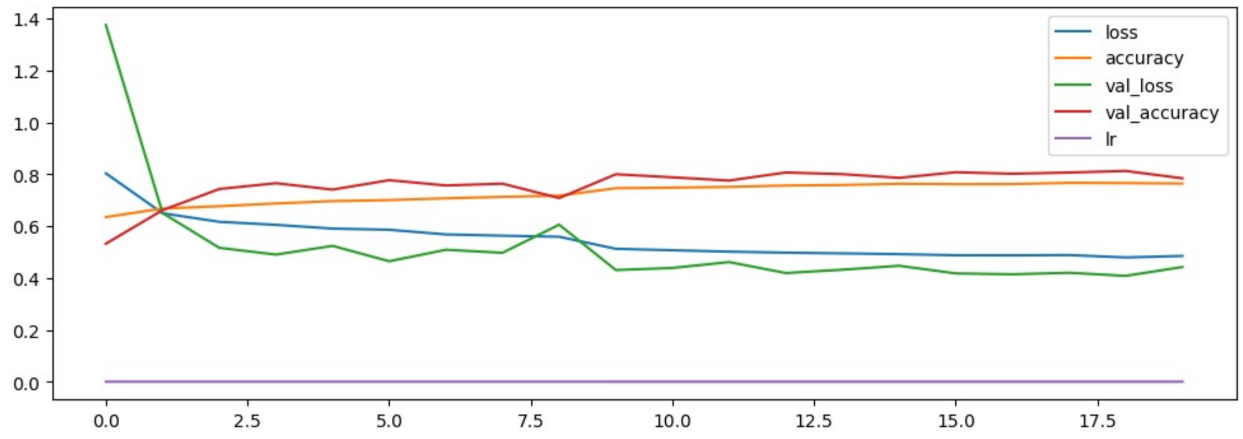
```
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
26615808 exceeds 10% of free system memory.
2025-09-26 16:57:42.509270: W
tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
26615808 exceeds 10% of free system memory.
/home/ant/tensorflow3/env/lib/python3.8/site-packages/PIL/TiffImagePlu
gin.py:900: UserWarning: Truncated File Read
  warnings.warn(str(msg))


Epoch 1: val_loss improved from inf to 1.37511, saving model to
models/binary_model_v3.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 - 2387s - loss: 0.8034 - accuracy: 0.6349 - val_loss: 1.3751
- val_accuracy: 0.5324 - lr: 1.0000e-05 - 2387s/epoch - 956ms/step
Epoch 2/20

Epoch 2: val_loss improved from 1.37511 to 0.65051, saving model to
models/binary_model_v3.h5
2496/2496 - 2453s - loss: 0.6496 - accuracy: 0.6678 - val_loss: 0.6505
- val_accuracy: 0.6616 - lr: 1.0000e-05 - 2453s/epoch - 983ms/step
Epoch 3/20

Epoch 3: val_loss improved from 0.65051 to 0.51640, saving model to
models/binary_model_v3.h5
2496/2496 - 2336s - loss: 0.6167 - accuracy: 0.6768 - val_loss: 0.5164
- val_accuracy: 0.7431 - lr: 1.0000e-05 - 2336s/epoch - 936ms/step
Epoch 4/20

Epoch 4: val_loss improved from 0.51640 to 0.49030, saving model to
models/binary_model_v3.h5
2496/2496 - 2315s - loss: 0.6050 - accuracy: 0.6874 - val_loss: 0.4903
- val_accuracy: 0.7656 - lr: 1.0000e-05 - 2315s/epoch - 927ms/step
Epoch 5/20

Epoch 5: val_loss did not improve from 0.49030
2496/2496 - 3165s - loss: 0.5905 - accuracy: 0.6965 - val_loss: 0.5239
- val_accuracy: 0.7409 - lr: 1.0000e-05 - 3165s/epoch - 1s/step
Epoch 6/20

Epoch 6: val_loss improved from 0.49030 to 0.46457, saving model to
models/binary_model_v3.h5
2496/2496 - 3215s - loss: 0.5860 - accuracy: 0.7005 - val_loss: 0.4646
- val_accuracy: 0.7772 - lr: 1.0000e-05 - 3215s/epoch - 1s/step
```

```
Epoch 7/20

Epoch 7: val_loss did not improve from 0.46457
2496/2496 - 2295s - loss: 0.5681 - accuracy: 0.7073 - val_loss: 0.5088
- val_accuracy: 0.7570 - lr: 1.0000e-05 - 2295s/epoch - 919ms/step
Epoch 8/20

Epoch 8: val_loss did not improve from 0.46457
2496/2496 - 2410s - loss: 0.5632 - accuracy: 0.7129 - val_loss: 0.4973
- val_accuracy: 0.7638 - lr: 1.0000e-05 - 2410s/epoch - 966ms/step
Epoch 9/20

Epoch 9: val_loss did not improve from 0.46457
2496/2496 - 2302s - loss: 0.5590 - accuracy: 0.7175 - val_loss: 0.6057
- val_accuracy: 0.7083 - lr: 1.0000e-05 - 2302s/epoch - 922ms/step
Epoch 10/20

Epoch 10: val_loss improved from 0.46457 to 0.43066, saving model to
models/binary_model_v3.h5
2496/2496 - 2272s - loss: 0.5125 - accuracy: 0.7461 - val_loss: 0.4307
- val_accuracy: 0.8000 - lr: 2.0000e-06 - 2272s/epoch - 910ms/step
Epoch 11/20

Epoch 11: val_loss did not improve from 0.43066
2496/2496 - 2258s - loss: 0.5070 - accuracy: 0.7484 - val_loss: 0.4387
- val_accuracy: 0.7882 - lr: 2.0000e-06 - 2258s/epoch - 905ms/step
Epoch 12/20

Epoch 12: val_loss did not improve from 0.43066
2496/2496 - 2251s - loss: 0.5016 - accuracy: 0.7514 - val_loss: 0.4614
- val_accuracy: 0.7758 - lr: 2.0000e-06 - 2251s/epoch - 902ms/step
Epoch 13/20

Epoch 13: val_loss improved from 0.43066 to 0.41910, saving model to
models/binary_model_v3.h5
2496/2496 - 2263s - loss: 0.4977 - accuracy: 0.7566 - val_loss: 0.4191
- val_accuracy: 0.8069 - lr: 2.0000e-06 - 2263s/epoch - 907ms/step
Epoch 14/20

Epoch 14: val_loss did not improve from 0.41910
2496/2496 - 2225s - loss: 0.4950 - accuracy: 0.7586 - val_loss: 0.4320
- val_accuracy: 0.8004 - lr: 2.0000e-06 - 2225s/epoch - 891ms/step
Epoch 15/20

Epoch 15: val_loss did not improve from 0.41910
2496/2496 - 2227s - loss: 0.4919 - accuracy: 0.7634 - val_loss: 0.4471
- val_accuracy: 0.7862 - lr: 2.0000e-06 - 2227s/epoch - 892ms/step
Epoch 16/20

Epoch 16: val_loss improved from 0.41910 to 0.41775, saving model to
```

```
models/binary_model_v3.h5
2496/2496 - 2219s - loss: 0.4877 - accuracy: 0.7620 - val_loss: 0.4178
- val_accuracy: 0.8079 - lr: 2.0000e-06 - 2219s/epoch - 889ms/step
Epoch 17/20

Epoch 17: val_loss improved from 0.41775 to 0.41431, saving model to
models/binary_model_v3.h5
2496/2496 - 2228s - loss: 0.4873 - accuracy: 0.7622 - val_loss: 0.4143
- val_accuracy: 0.8020 - lr: 2.0000e-06 - 2228s/epoch - 893ms/step
Epoch 18/20

Epoch 18: val_loss did not improve from 0.41431
2496/2496 - 2221s - loss: 0.4884 - accuracy: 0.7670 - val_loss: 0.4202
- val_accuracy: 0.8067 - lr: 2.0000e-06 - 2221s/epoch - 890ms/step
Epoch 19/20

Epoch 19: val_loss improved from 0.41431 to 0.40829, saving model to
models/binary_model_v3.h5
2496/2496 - 2219s - loss: 0.4793 - accuracy: 0.7661 - val_loss: 0.4083
- val_accuracy: 0.8129 - lr: 2.0000e-06 - 2219s/epoch - 889ms/step
Epoch 20/20

Epoch 20: val_loss did not improve from 0.40829
2496/2496 - 2206s - loss: 0.4850 - accuracy: 0.7642 - val_loss: 0.4423
- val_accuracy: 0.7844 - lr: 2.0000e-06 - 2206s/epoch - 884ms/step

model_v3.save(f'binary_model_v{version_model}.keras')
```

Result 3: val_acc = 81%.

```
pd.DataFrame(history_stage3.history).plot(figsize=(12, 4))

<Axes: >
```



Make predictions

```
import numpy as np
import os
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model

def prediction(path: 'str', model) -> None:
    test_img = image.load_img(path, target_size=input_shape)
    test_img = image.img_to_array(test_img) / 255.0
    test_img = np.expand_dims(test_img, axis=0)

    prob = model.predict(test_img)[0][0]

    print(f"Probability to be Dog: {prob:.4f}")
    print(" □ Dog\n" if prob >= 0.5 else " 😺 Cat\n")

model_v3 =
load_model(os.path.join(folder_models,'binary_model_v3.h5'))

# Make predictions
prediction('./test/cat_or_dog_1.jpg', model_v3)
prediction('./test/cat_or_dog_2.jpg', model_v3)
prediction('./test/cat_or_dog_3.jpg', model_v3)
```

```
1/1 [==============================] - 1s 1s/step
Probability to be Dog: 0.5615
 □ Dog

1/1 [==============================] - 0s 103ms/step
Probability to be Dog: 0.2824
 😺 Cat

1/1 [==============================] - 0s 134ms/step
Probability to be Dog: 0.1678
 😺 Cat
```