```
import numpy as np
import keras
from keras import layers
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt
```

```
tfds.disable_progress_bar()

train_ds, validation_ds, test_ds = tfds.load(
    "cats_vs_dogs",
    # Reserve 10% for validation and 10% for test
    split=["train[:40%]", "train[40%:50%]", "train[50%:60%]"],
    as_supervised=True,  # Include labels
)

print(f"Number of training samples: {train_ds.cardinality()}")
print(f"Number of validation samples: {validation_ds.cardinality()}")
print(f"Number of test samples: {test_ds.cardinality()}")
```

```
Downloading and preparing dataset 786.67 MiB (download: 786.67 MiB, generated: 1.04 GiB, total: 1.81 GiB) to /root/tensorflow
WARNING:absl:1738 images were corrupted and were skipped
Dataset cats_vs_dogs downloaded and prepared to /root/tensorflow_datasets/cats_vs_dogs/4.0.1. Subsequent calls will reuse thi
Number of training samples: 9305
Number of validation samples: 2326
Number of test samples: 2326
```

```
plt.figure(figsize=(10, 10))
for i, (image, label) in enumerate(train_ds.take(9)):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(image)
    plt.title(int(label))
    plt.axis("off")
```

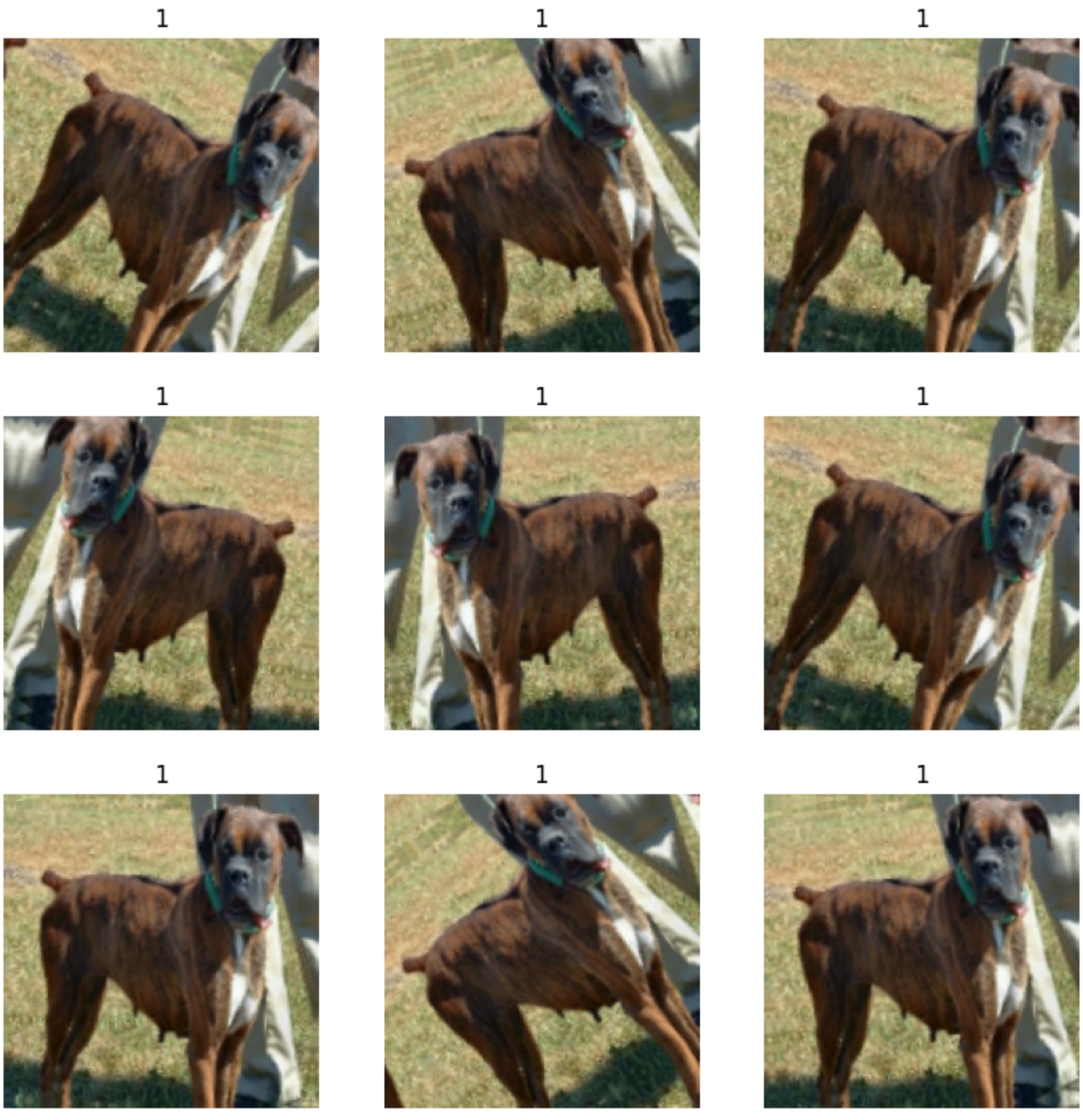


```
resize_fn = keras.layers.Resizing(150, 150)

train_ds = train_ds.map(lambda x, y: (resize_fn(x), y))
validation_ds = validation_ds.map(lambda x, y: (resize_fn(x), y))
test_ds = test_ds.map(lambda x, y: (resize_fn(x), y))
```

```python
augmentation_layers = [
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
]


def data_augmentation(x):
    for layer in augmentation_layers:
        x = layer(x)
    return x


train_ds = train_ds.map(lambda x, y: (data_augmentation(x), y))
```

```python
from tensorflow import data as tf_data

batch_size = 64

train_ds = train_ds.batch(batch_size).prefetch(tf_data.AUTOTUNE).cache()
validation_ds = validation_ds.batch(batch_size).prefetch(tf_data.AUTOTUNE).cache()
test_ds = test_ds.batch(batch_size).prefetch(tf_data.AUTOTUNE).cache()
```

```python
for images, labels in train_ds.take(1):
    plt.figure(figsize=(10, 10))
    first_image = images[0]
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        augmented_image = data_augmentation(np.expand_dims(first_image, 0))
        plt.imshow(np.array(augmented_image[0]).astype("int32"))
        plt.title(int(labels[0]))
        plt.axis("off")
```




```python
base_model = keras.applications.Xception(
    weights="imagenet",  # Load weights pre-trained on ImageNet.
    input_shape=(150, 150, 3),
    include_top=False,
)  # Do not include the ImageNet classifier at the top.

# Freeze the base_model
base_model.trainable = False
```

```
# Create new model on top
inputs = keras.Input(shape=(150, 150, 3))

# Pre-trained Xception weights requires that input be scaled
# from (0, 255) to a range of (-1., +1.), the rescaling layer
# outputs: `(inputs * scale) + offset`
scale_layer = keras.layers.Rescaling(scale=1 / 127.5, offset=-1)
x = scale_layer(inputs)

# The base model contains batchnorm layers. We want to keep them in inference moc
# when we unfreeze the base model for fine-tuning, so we make sure that the
# base_model is running in inference mode here.
x = base_model(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.2)(x)  # Regularize with dropout
outputs = keras.layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

model.summary(show_trainable=True)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_
83683744/83683744 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Model: "functional"
```

Layer (type)	Output Shape	Param #	Traina…
input_layer_1 (InputLayer)	(None, 150, 150, 3)	0	-
rescaling (Rescaling)	(None, 150, 150, 3)	0	-
xception (Functional)	(None, 5, 5, 2048)	20,861,480	N
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0	-
dropout (Dropout)	(None, 2048)	0	-
dense (Dense)	(None, 1)	2,049	Y

```
Total params: 20,863,529 (79.59 MB)
Trainable params: 2,049 (8.00 KB)
Non-trainable params: 20,861,480 (79.58 MB)
```

```
model.compile(
    optimizer=keras.optimizers.Adam(),
    loss=keras.losses.BinaryCrossentropy(from_logits=True),
    metrics=[keras.metrics.BinaryAccuracy()],
)

epochs = 2
print("Fitting the top layer of the model")
model.fit(train_ds, epochs=epochs, validation_data=validation_ds)
```

```
Fitting the top layer of the model
Epoch 1/2
146/146 ━━━━━━━━━━━━━━━━━━━━ 100s 568ms/step - binary_accuracy: 0.8537 - loss: 0.3007 - val_binary_accuracy: 0.9699 - val_los
Epoch 2/2
146/146 ━━━━━━━━━━━━━━━━━━━━ 18s 126ms/step - binary_accuracy: 0.9422 - loss: 0.1322 - val_binary_accuracy: 0.9716 - val_loss
<keras.src.callbacks.history.History at 0x7a2f291cbfd0>
```

```
# Unfreeze the base_model. Note that it keeps running in inference mode
# since we passed `training=False` when calling it. This means that
# the batchnorm layers will not update their batch statistics.
# This prevents the batchnorm layers from undoing all the training
# we've done so far.
base_model.trainable = True
model.summary(show_trainable=True)

model.compile(
    optimizer=keras.optimizers.Adam(1e-5),  # Low learning rate
    loss=keras.losses.BinaryCrossentropy(from_logits=True),
    metrics=[keras.metrics.BinaryAccuracy()],
)

epochs = 1
print("Fitting the end-to-end model")
model.fit(train_ds, epochs=epochs, validation_data=validation_ds)
```

Model: "functional"

Layer (type)	Output Shape	Param #	Traina…
input_layer_1 (InputLayer)	(None, 150, 150, 3)	0	-
rescaling (Rescaling)	(None, 150, 150, 3)	0	-
xception (Functional)	(None, 5, 5, 2048)	20,861,480	Y
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0	-
dropout (Dropout)	(None, 2048)	0	-
dense (Dense)	(None, 1)	2,049	Y

Total params: 20,867,629 (79.60 MB)
Trainable params: 20,809,001 (79.38 MB)
Non-trainable params: 54,528 (213.00 KB)
Optimizer params: 4,100 (16.02 KB)

```
Fitting the end-to-end model
146/146 ━━━━━━━━━━━━━━━━━━━━ 137s 577ms/step - binary_accuracy: 0.8508 - loss: 0.3771 - val_binary_accuracy: 0.9643 - val_los
<keras.src.callbacks.history.History at 0x7a2fbec9f040>
```

```
print("Test dataset evaluation")
model.evaluate(test_ds)
```

```
Test dataset evaluation
37/37 ━━━━━━━━━━━━━━━━━━━━ 4s 106ms/step - binary_accuracy: 0.9614 - loss: 0.0964
[0.1073535904288292, 0.9582974910736084]
```

```
Start coding or generate with AI.
```