Assignment-2

*Convolution *

** Uploading the data**

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Transferring images into training, validation, and testing folders

```
import os
import shutil
import pathlib

# Define original folders separately
source_train_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/train")
source_validation_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/validation")
source_test_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/test")

# Define destination base directory
target_dataset_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced")

def create_dataset_subset(subset_name, original_folder, start_index, end_index):

    print(f"\nCreating '{subset_name}' from {original_folder} indices {start_index} to {end_index}")

    for category in ["cats", "dogs"]:
        src_dir = original_folder / category
        dest_dir = target_dataset_dir / subset_name / category

        os.makedirs(dest_dir, exist_ok=True)

        files = sorted(os.listdir(src_dir))


        if end_index > len(files):
            print(f"WARNING: end_index {end_index} exceeds available files ({len(files)}). Adjusting.")
            end_index = len(files)

        subset_files = files[start_index:end_index]
        print(f"Copying {len(subset_files)} '{category}' images to '{dest_dir}'...")

        for fname in subset_files:
            src_file = src_dir / fname
            dst_file = dest_dir / fname
            shutil.copyfile(src_file, dst_file)

    print(f" {subset_name} created from {original_folder}.")

def main():



    create_dataset_subset("train", source_train_dir, 0, 500)
    create_dataset_subset("validation", source_validation_dir, 0, 250)
    create_dataset_subset("test",   source_test_dir, 0, 250)

if __name__ == "__main__":
    main()
```

```
Creating 'train' from /content/drive/MyDrive/cats_vs_dogs_small/train indices 0 to 500
Copying 500 'cats' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/train/cats'...
Copying 500 'dogs' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/train/dogs'...
 train created from /content/drive/MyDrive/cats_vs_dogs_small/train.

Creating 'validation' from /content/drive/MyDrive/cats_vs_dogs_small/validation indices 0 to 250
Copying 250 'cats' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/validation/cats'...
Copying 250 'dogs' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/validation/dogs'...
 validation created from /content/drive/MyDrive/cats_vs_dogs_small/validation.
```

```
    Creating 'test' from /content/drive/MyDrive/cats_vs_dogs_small/test indices 0 to 250
    Copying 250 'cats' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/test/cats'...
    Copying 250 'dogs' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/test/dogs'...
     test created from /content/drive/MyDrive/cats_vs_dogs_small/test.
```

```
create_dataset_subset("train", source_train_dir, 0, 500)
create_dataset_subset("validation", source_validation_dir, 0, 250)
create_dataset_subset("test",   source_test_dir, 0, 250)
```

```
    Creating 'train' from /content/drive/MyDrive/cats_vs_dogs_small/train indices 0 to 500
    Copying 500 'cats' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/train/cats'...
    Copying 500 'dogs' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/train/dogs'...
     train created from /content/drive/MyDrive/cats_vs_dogs_small/train.

    Creating 'validation' from /content/drive/MyDrive/cats_vs_dogs_small/validation indices 0 to 250
    Copying 250 'cats' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/validation/cats'...
    Copying 250 'dogs' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/validation/dogs'...
     validation created from /content/drive/MyDrive/cats_vs_dogs_small/validation.

    Creating 'test' from /content/drive/MyDrive/cats_vs_dogs_small/test indices 0 to 250
    Copying 250 'cats' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/test/cats'...
    Copying 250 'dogs' images to '/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced/test/dogs'...
     test created from /content/drive/MyDrive/cats_vs_dogs_small/test.
```

```
from tensorflow.keras.utils import image_dataset_from_directory

train_ds = image_dataset_from_directory(
    target_dataset_dir / "train",
    image_size=(180, 180),
    batch_size=32
)

val_ds = image_dataset_from_directory(
    target_dataset_dir / "validation",
    image_size=(180, 180),
    batch_size=32
)

test_ds = image_dataset_from_directory(
    target_dataset_dir / "test",
    image_size=(180, 180),
    batch_size=32
)
```

```
Found 1000 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
```

Training a convent neural network from the beginning

Model 1:The dataset consists of 1000 samples for training, 500 for validation, and 500 for testing.

```
from tensorflow import keras
from tensorflow.keras import layers
```

Building a simple ConvNet to distinguish between dogs and cats.

```
inputs = keras.Input(shape=(180, 180, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

```
model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 180, 180, 3)	0
rescaling (Rescaling)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d (MaxPooling2D)	(None, 89, 89, 32)	0
conv2d_1 (Conv2D)	(None, 87, 87, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 43, 43, 64)	0
conv2d_2 (Conv2D)	(None, 41, 41, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 20, 20, 128)	0
conv2d_3 (Conv2D)	(None, 18, 18, 256)	295,168
max_pooling2d_3 (MaxPooling2D)	(None, 9, 9, 256)	0
conv2d_4 (Conv2D)	(None, 7, 7, 256)	590,080
flatten (Flatten)	(None, 12544)	0
dense (Dense)	(None, 1)	12,545

Total params: 991,041 (3.78 MB)
Trainable params: 991,041 (3.78 MB)
Non-trainable params: 0 (0.00 B)

Setting up the model for training

```
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

Data preparation

```
from tensorflow.keras.utils import image_dataset_from_directory

# Base directory for the subsets
balanced_subset_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/small_dataset_balanced")

# Load datasets
train_ds = image_dataset_from_directory(
    balanced_subset_dir / "train",
    image_size=(180, 180),
    batch_size=32
)

val_ds = image_dataset_from_directory(
    balanced_subset_dir / "validation",
    image_size=(180, 180),
    batch_size=32
)

test_ds = image_dataset_from_directory(
    balanced_subset_dir / "test",
    image_size=(180, 180),
    batch_size=32
)
```

```
Found 1000 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
```

```
import numpy as np
import tensorflow as tf
random_numbers = np.random.normal(size=(1000, 16))
dataset = tf.data.Dataset.from_tensor_slices(random_numbers)


for i, element in enumerate(dataset):
    print(element.shape)
    if i >= 2:
        break
```

```
(16,)
(16,)
(16,)
```

```
batched_dataset = dataset.batch(32)
for i, element in enumerate(batched_dataset):
    print(element.shape)
    if i >= 2:
        break
```

```
(32, 16)
(32, 16)
(32, 16)
```

```
reshaped_dataset = dataset.map(lambda x: tf.reshape(x, (4, 4)))
for i, element in enumerate(reshaped_dataset):
    print(element.shape)
    if i >= 2:
        break
```

```
(4, 4)
(4, 4)
(4, 4)
```

Showing the dimensions of the data and labels returned by the Dataset

```
for data_batch, labels_batch in train_ds:
    print("data batch shape:", data_batch.shape)
    print("labels batch shape:", labels_batch.shape)
    break
```

```
data batch shape: (32, 180, 180, 3)
labels batch shape: (32,)
```

Training the model with a dataset

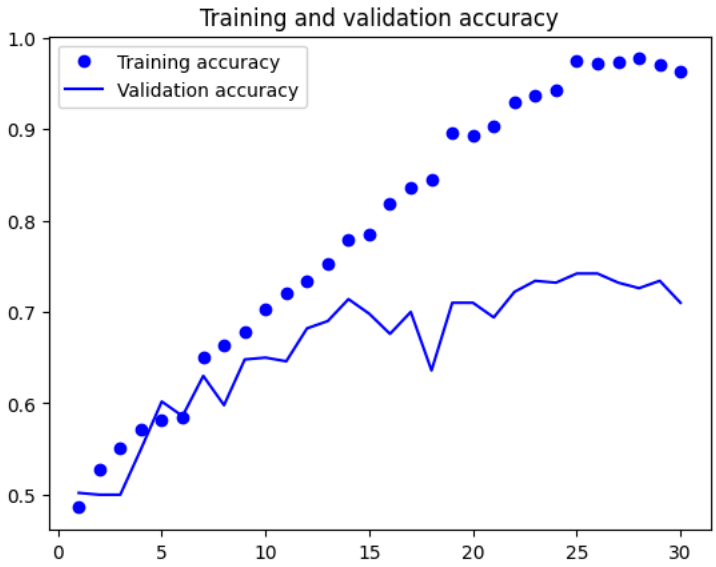
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_ds,
    epochs=30,
    validation_data=val_ds,
    callbacks=callbacks)
```

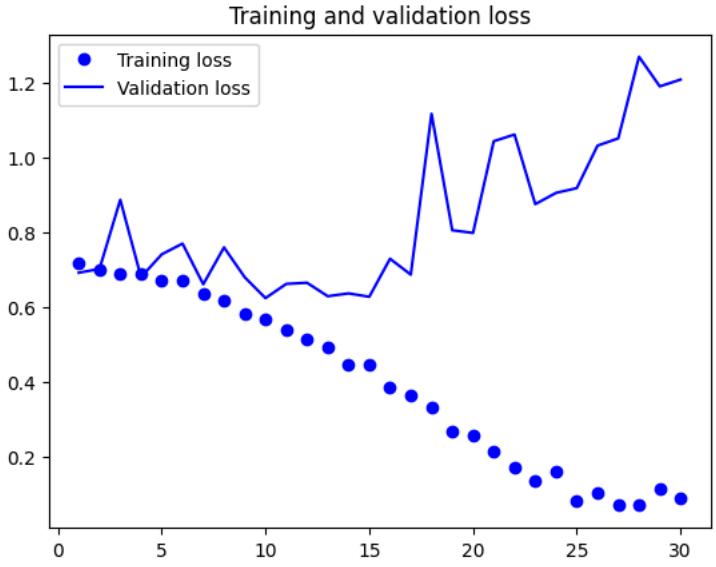
```
Epoch 1/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 16s 309ms/step - accuracy: 0.4607 - loss: 0.7651 - val_accuracy: 0.5020 - val_loss: 0.6917
Epoch 2/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 182ms/step - accuracy: 0.5038 - loss: 0.6964 - val_accuracy: 0.5000 - val_loss: 0.7021
Epoch 3/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 204ms/step - accuracy: 0.5436 - loss: 0.6923 - val_accuracy: 0.5000 - val_loss: 0.8871
Epoch 4/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 184ms/step - accuracy: 0.5528 - loss: 0.7245 - val_accuracy: 0.5500 - val_loss: 0.6801
Epoch 5/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 182ms/step - accuracy: 0.5543 - loss: 0.6774 - val_accuracy: 0.6020 - val_loss: 0.7410
Epoch 6/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 183ms/step - accuracy: 0.5525 - loss: 0.6972 - val_accuracy: 0.5860 - val_loss: 0.7698
Epoch 7/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 177ms/step - accuracy: 0.6323 - loss: 0.6582 - val_accuracy: 0.6300 - val_loss: 0.6608
Epoch 8/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 187ms/step - accuracy: 0.6423 - loss: 0.6288 - val_accuracy: 0.5980 - val_loss: 0.7596
Epoch 9/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 188ms/step - accuracy: 0.6656 - loss: 0.5885 - val_accuracy: 0.6480 - val_loss: 0.6795
Epoch 10/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 203ms/step - accuracy: 0.6988 - loss: 0.5713 - val_accuracy: 0.6500 - val_loss: 0.6238
Epoch 11/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 183ms/step - accuracy: 0.7000 - loss: 0.5378 - val_accuracy: 0.6460 - val_loss: 0.6618
Epoch 12/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 153ms/step - accuracy: 0.7137 - loss: 0.5255 - val_accuracy: 0.6820 - val_loss: 0.6649
Epoch 13/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 188ms/step - accuracy: 0.7402 - loss: 0.4898 - val_accuracy: 0.6900 - val_loss: 0.6288
Epoch 14/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 146ms/step - accuracy: 0.7799 - loss: 0.4510 - val_accuracy: 0.7140 - val_loss: 0.6365
Epoch 15/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 222ms/step - accuracy: 0.7760 - loss: 0.4689 - val_accuracy: 0.6980 - val_loss: 0.6273
Epoch 16/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 188ms/step - accuracy: 0.8183 - loss: 0.3982 - val_accuracy: 0.6760 - val_loss: 0.7291
Epoch 17/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 188ms/step - accuracy: 0.8180 - loss: 0.4011 - val_accuracy: 0.7000 - val_loss: 0.6867
```

```
Epoch 18/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 218ms/step - accuracy: 0.8374 - loss: 0.3272 - val_accuracy: 0.6360 - val_loss: 1.1173
Epoch 19/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 183ms/step - accuracy: 0.8714 - loss: 0.3248 - val_accuracy: 0.7100 - val_loss: 0.8055
Epoch 20/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 191ms/step - accuracy: 0.8828 - loss: 0.2814 - val_accuracy: 0.7100 - val_loss: 0.7983
Epoch 21/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 187ms/step - accuracy: 0.8796 - loss: 0.2720 - val_accuracy: 0.6940 - val_loss: 1.0440
Epoch 22/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 148ms/step - accuracy: 0.9369 - loss: 0.1726 - val_accuracy: 0.7220 - val_loss: 1.0618
Epoch 23/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 178ms/step - accuracy: 0.9388 - loss: 0.1440 - val_accuracy: 0.7340 - val_loss: 0.8754
Epoch 24/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 185ms/step - accuracy: 0.9547 - loss: 0.1362 - val_accuracy: 0.7320 - val_loss: 0.9056
Epoch 25/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 180ms/step - accuracy: 0.9747 - loss: 0.0762 - val_accuracy: 0.7420 - val_loss: 0.9183
Epoch 26/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 166ms/step - accuracy: 0.9702 - loss: 0.1141 - val_accuracy: 0.7420 - val_loss: 1.0321
Epoch 27/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 179ms/step - accuracy: 0.9813 - loss: 0.0508 - val_accuracy: 0.7320 - val_loss: 1.0516
Epoch 28/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 152ms/step - accuracy: 0.9713 - loss: 0.0826 - val_accuracy: 0.7260 - val_loss: 1.2701
Epoch 29/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 157ms/step - accuracy: 0.9774 - loss: 0.0862 - val_accuracy: 0.7340 - val_loss: 1.1908
```

Visualizing the loss and accuracy trends throughout training

```
import matplotlib.pyplot as plt
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```





Assessing the model's performance on the test set

```
test_model = keras.models.load_model("convnet_from_scratch.keras")
test_loss, test_acc = test_model.evaluate(test_ds)
print(f"Test accuracy: {test_acc:.3f}")
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 3s 139ms/step - accuracy: 0.6020 - loss: 0.6741
Test accuracy: 0.608
```

Since the validation and the tes accuracy of the model is very low that is 68%

To improve performance in developing a network that we trained from scratch, we will train our model on following techniques.

Model 1a: Applying Data Augmentation

```
from tensorflow import keras
from tensorflow.keras import layers
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)


inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
```

```
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])


callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_ds,
    epochs=30,
    validation_data=val_ds,
     callbacks=callbacks)
```

```
Epoch 1/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 227ms/step - accuracy: 0.4829 - loss: 0.6984 - val_accuracy: 0.5000 - val_loss: 0.7005
Epoch 2/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 181ms/step - accuracy: 0.5042 - loss: 0.6965 - val_accuracy: 0.5000 - val_loss: 0.8050
Epoch 3/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 196ms/step - accuracy: 0.5372 - loss: 0.6985 - val_accuracy: 0.6000 - val_loss: 0.6774
Epoch 4/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 180ms/step - accuracy: 0.5754 - loss: 0.6863 - val_accuracy: 0.5020 - val_loss: 0.6856
Epoch 5/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 157ms/step - accuracy: 0.5655 - loss: 0.6862 - val_accuracy: 0.5660 - val_loss: 0.6853
Epoch 6/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 151ms/step - accuracy: 0.5898 - loss: 0.6753 - val_accuracy: 0.5000 - val_loss: 1.1144
Epoch 7/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 220ms/step - accuracy: 0.5984 - loss: 0.6854 - val_accuracy: 0.6520 - val_loss: 0.6308
Epoch 8/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 177ms/step - accuracy: 0.6298 - loss: 0.6545 - val_accuracy: 0.5140 - val_loss: 0.7281
Epoch 9/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 140ms/step - accuracy: 0.6641 - loss: 0.6179 - val_accuracy: 0.5080 - val_loss: 1.1961
Epoch 10/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 183ms/step - accuracy: 0.6456 - loss: 0.7072 - val_accuracy: 0.6500 - val_loss: 0.6105
Epoch 11/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 178ms/step - accuracy: 0.6977 - loss: 0.5841 - val_accuracy: 0.5440 - val_loss: 0.7247
Epoch 12/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 180ms/step - accuracy: 0.6479 - loss: 0.6317 - val_accuracy: 0.6500 - val_loss: 0.6274
Epoch 13/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 147ms/step - accuracy: 0.6718 - loss: 0.6008 - val_accuracy: 0.6200 - val_loss: 0.6521
Epoch 14/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 213ms/step - accuracy: 0.6885 - loss: 0.5956 - val_accuracy: 0.6940 - val_loss: 0.5890
Epoch 15/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 178ms/step - accuracy: 0.6954 - loss: 0.5901 - val_accuracy: 0.6180 - val_loss: 0.7597
Epoch 16/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 143ms/step - accuracy: 0.6994 - loss: 0.5756 - val_accuracy: 0.7100 - val_loss: 0.5524
Epoch 17/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 211ms/step - accuracy: 0.7282 - loss: 0.5613 - val_accuracy: 0.6760 - val_loss: 0.6054
Epoch 18/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 141ms/step - accuracy: 0.6929 - loss: 0.6061 - val_accuracy: 0.7020 - val_loss: 0.5674
Epoch 19/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 162ms/step - accuracy: 0.7370 - loss: 0.5281 - val_accuracy: 0.6000 - val_loss: 0.8617
Epoch 20/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 179ms/step - accuracy: 0.7318 - loss: 0.5856 - val_accuracy: 0.7220 - val_loss: 0.5306
Epoch 21/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 214ms/step - accuracy: 0.7323 - loss: 0.5450 - val_accuracy: 0.7160 - val_loss: 0.5529
Epoch 22/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 177ms/step - accuracy: 0.7743 - loss: 0.5131 - val_accuracy: 0.7020 - val_loss: 0.5753
Epoch 23/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 177ms/step - accuracy: 0.7537 - loss: 0.5165 - val_accuracy: 0.7040 - val_loss: 0.6327
Epoch 24/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 180ms/step - accuracy: 0.7584 - loss: 0.5203 - val_accuracy: 0.6840 - val_loss: 0.7408
Epoch 25/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 182ms/step - accuracy: 0.7478 - loss: 0.5532 - val_accuracy: 0.6860 - val_loss: 0.6610
Epoch 26/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 149ms/step - accuracy: 0.7576 - loss: 0.5022 - val_accuracy: 0.7540 - val_loss: 0.5232
Epoch 27/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 182ms/step - accuracy: 0.7753 - loss: 0.4707 - val_accuracy: 0.7300 - val_loss: 0.7040
Epoch 28/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 144ms/step - accuracy: 0.7831 - loss: 0.4740 - val_accuracy: 0.7340 - val_loss: 0.6189
Epoch 29/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 217ms/step - accuracy: 0.7696 - loss: 0.4685 - val_accuracy: 0.7220 - val_loss: 0.6213
```

```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation.keras")
test_loss, test_acc = test_model.evaluate(test_ds)
print(f"Test accuracy: {test_acc:.3f}")
```

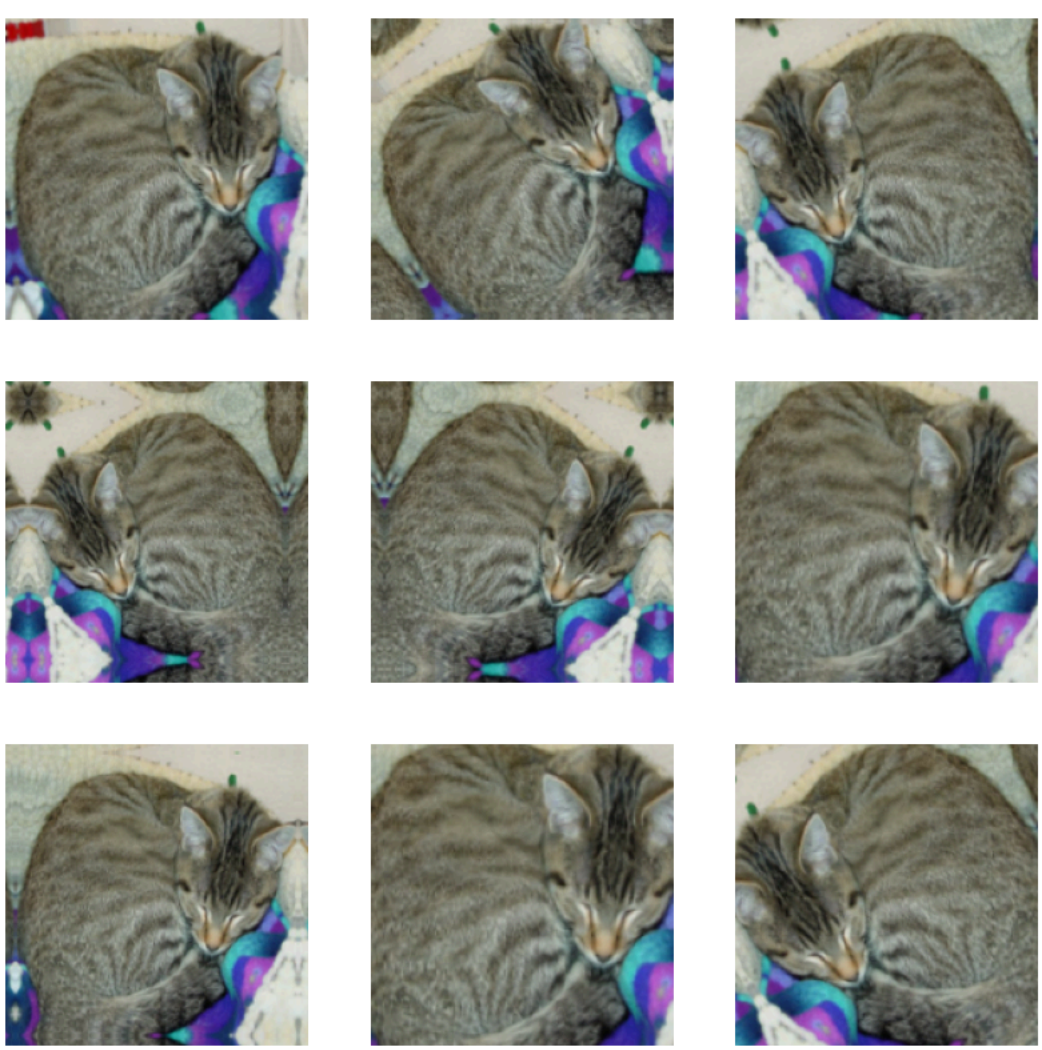
```
16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 90ms/step - accuracy: 0.7121 - loss: 0.5589
Test accuracy: 0.720
```

Creating a data augmentation step for an image model

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

Showing a selection of randomly augmented training images

```
plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_images[0].numpy().astype("uint8"))
        plt.axis("off")
```



Model 1b: Applying the Dropout Technique

```
inputs = keras.Input(shape=(180, 180, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
```

```
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])


callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_dropout.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_ds,
    epochs=30,
    validation_data=val_ds,
    callbacks=callbacks)
```

```
Epoch 1/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 246ms/step - accuracy: 0.4900 - loss: 0.7026 - val_accuracy: 0.5000 - val_loss: 0.6925
Epoch 2/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 141ms/step - accuracy: 0.5098 - loss: 0.6926 - val_accuracy: 0.5320 - val_loss: 0.6911
Epoch 3/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 182ms/step - accuracy: 0.5546 - loss: 0.7005 - val_accuracy: 0.5000 - val_loss: 2.7615
Epoch 4/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 185ms/step - accuracy: 0.5684 - loss: 0.9235 - val_accuracy: 0.6140 - val_loss: 0.6525
Epoch 5/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 147ms/step - accuracy: 0.5729 - loss: 0.6785 - val_accuracy: 0.5040 - val_loss: 0.7390
Epoch 6/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 167ms/step - accuracy: 0.5996 - loss: 0.6562 - val_accuracy: 0.5060 - val_loss: 0.9531
Epoch 7/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 143ms/step - accuracy: 0.5937 - loss: 0.7093 - val_accuracy: 0.6200 - val_loss: 0.6487
Epoch 8/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 183ms/step - accuracy: 0.6725 - loss: 0.6189 - val_accuracy: 0.6340 - val_loss: 0.6186
Epoch 9/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 180ms/step - accuracy: 0.6776 - loss: 0.6023 - val_accuracy: 0.6580 - val_loss: 0.6204
Epoch 10/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 164ms/step - accuracy: 0.6771 - loss: 0.6063 - val_accuracy: 0.6620 - val_loss: 0.6364
Epoch 11/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 142ms/step - accuracy: 0.6984 - loss: 0.5843 - val_accuracy: 0.6800 - val_loss: 0.5910
Epoch 12/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 176ms/step - accuracy: 0.6931 - loss: 0.5553 - val_accuracy: 0.6740 - val_loss: 0.5981
Epoch 13/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 176ms/step - accuracy: 0.7194 - loss: 0.5433 - val_accuracy: 0.6360 - val_loss: 0.7535
Epoch 14/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 138ms/step - accuracy: 0.7933 - loss: 0.4825 - val_accuracy: 0.6340 - val_loss: 0.8202
Epoch 15/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 175ms/step - accuracy: 0.7426 - loss: 0.5536 - val_accuracy: 0.6840 - val_loss: 0.5818
Epoch 16/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 179ms/step - accuracy: 0.7683 - loss: 0.4677 - val_accuracy: 0.6960 - val_loss: 0.5767
Epoch 17/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 178ms/step - accuracy: 0.8128 - loss: 0.3979 - val_accuracy: 0.6800 - val_loss: 0.7149
Epoch 18/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 141ms/step - accuracy: 0.8134 - loss: 0.3984 - val_accuracy: 0.7340 - val_loss: 0.5816
Epoch 19/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 178ms/step - accuracy: 0.8452 - loss: 0.3639 - val_accuracy: 0.7160 - val_loss: 0.7291
Epoch 20/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 175ms/step - accuracy: 0.8395 - loss: 0.3536 - val_accuracy: 0.6960 - val_loss: 0.6361
Epoch 21/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 140ms/step - accuracy: 0.8602 - loss: 0.3057 - val_accuracy: 0.7120 - val_loss: 0.6471
Epoch 22/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 176ms/step - accuracy: 0.8795 - loss: 0.2729 - val_accuracy: 0.7060 - val_loss: 0.5993
Epoch 23/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 177ms/step - accuracy: 0.8907 - loss: 0.2451 - val_accuracy: 0.7400 - val_loss: 0.6909
Epoch 24/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 174ms/step - accuracy: 0.9221 - loss: 0.1946 - val_accuracy: 0.7360 - val_loss: 0.7292
Epoch 25/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 139ms/step - accuracy: 0.9353 - loss: 0.1626 - val_accuracy: 0.6460 - val_loss: 1.3702
Epoch 26/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 174ms/step - accuracy: 0.9167 - loss: 0.1838 - val_accuracy: 0.7460 - val_loss: 0.7082
Epoch 27/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 183ms/step - accuracy: 0.9602 - loss: 0.1348 - val_accuracy: 0.7480 - val_loss: 0.7764
Epoch 28/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 174ms/step - accuracy: 0.9502 - loss: 0.1248 - val_accuracy: 0.7360 - val_loss: 1.0705
Epoch 29/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 202ms/step - accuracy: 0.9595 - loss: 0.1094 - val_accuracy: 0.7340 - val_loss: 1.0496
```

```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_dropout.keras")
test_loss, test_acc = test_model.evaluate(test_ds)
print(f"Test accuracy: {test_acc:.3f}")
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 90ms/step - accuracy: 0.7166 - loss: 0.5733
Test accuracy: 0.702
```

Model 1c: Applying both Image Augmentation and Dropout techniques

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)


inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])


callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation_dropout.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_ds,
    epochs=30,
    validation_data=val_ds,
    callbacks=callbacks)
```

```
Epoch 1/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 188ms/step - accuracy: 0.4749 - loss: 0.7409 - val_accuracy: 0.5000 - val_loss: 0.6917
Epoch 2/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 204ms/step - accuracy: 0.5095 - loss: 0.6943 - val_accuracy: 0.5000 - val_loss: 0.6917
Epoch 3/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 200ms/step - accuracy: 0.5360 - loss: 0.6913 - val_accuracy: 0.5000 - val_loss: 0.7703
Epoch 4/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 184ms/step - accuracy: 0.5129 - loss: 0.7007 - val_accuracy: 0.5300 - val_loss: 0.6899
Epoch 5/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 147ms/step - accuracy: 0.5501 - loss: 0.6877 - val_accuracy: 0.5920 - val_loss: 0.6728
Epoch 6/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 202ms/step - accuracy: 0.5994 - loss: 0.6742 - val_accuracy: 0.6080 - val_loss: 0.6679
Epoch 7/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 143ms/step - accuracy: 0.6253 - loss: 0.6538 - val_accuracy: 0.5140 - val_loss: 0.6824
Epoch 8/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 157ms/step - accuracy: 0.6019 - loss: 0.6607 - val_accuracy: 0.6000 - val_loss: 0.6549
Epoch 9/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 169ms/step - accuracy: 0.6635 - loss: 0.6358 - val_accuracy: 0.6660 - val_loss: 0.6274
Epoch 10/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 144ms/step - accuracy: 0.6491 - loss: 0.6604 - val_accuracy: 0.6500 - val_loss: 0.6237
Epoch 11/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 199ms/step - accuracy: 0.6235 - loss: 0.6289 - val_accuracy: 0.5400 - val_loss: 1.1512
Epoch 12/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 145ms/step - accuracy: 0.6455 - loss: 0.6645 - val_accuracy: 0.6680 - val_loss: 0.6206
Epoch 13/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 217ms/step - accuracy: 0.6509 - loss: 0.6060 - val_accuracy: 0.6320 - val_loss: 0.6318
Epoch 14/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 167ms/step - accuracy: 0.7051 - loss: 0.5723 - val_accuracy: 0.7320 - val_loss: 0.5628
Epoch 15/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 165ms/step - accuracy: 0.7170 - loss: 0.5479 - val_accuracy: 0.6160 - val_loss: 0.7264
Epoch 16/30
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 182ms/step - accuracy: 0.6857 - loss: 0.6132 - val_accuracy: 0.6940 - val_loss: 0.5701
Epoch 17/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 221ms/step - accuracy: 0.7105 - loss: 0.5672 - val_accuracy: 0.7220 - val_loss: 0.5603
Epoch 18/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 189ms/step - accuracy: 0.7426 - loss: 0.5301 - val_accuracy: 0.7280 - val_loss: 0.5577
Epoch 19/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 9s 152ms/step - accuracy: 0.7082 - loss: 0.5574 - val_accuracy: 0.6300 - val_loss: 0.6141
Epoch 20/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 182ms/step - accuracy: 0.7229 - loss: 0.5492 - val_accuracy: 0.6920 - val_loss: 0.5894
Epoch 21/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 191ms/step - accuracy: 0.7486 - loss: 0.5334 - val_accuracy: 0.6900 - val_loss: 0.5905
Epoch 22/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 181ms/step - accuracy: 0.7140 - loss: 0.5483 - val_accuracy: 0.7180 - val_loss: 0.5731
Epoch 23/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 180ms/step - accuracy: 0.7489 - loss: 0.5130 - val_accuracy: 0.7400 - val_loss: 0.6040
Epoch 24/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 180ms/step - accuracy: 0.7326 - loss: 0.5161 - val_accuracy: 0.7360 - val_loss: 0.5756
Epoch 25/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 10s 177ms/step - accuracy: 0.7813 - loss: 0.4760 - val_accuracy: 0.6920 - val_loss: 0.7286
Epoch 26/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 182ms/step - accuracy: 0.7636 - loss: 0.5264 - val_accuracy: 0.7580 - val_loss: 0.5162
Epoch 27/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 166ms/step - accuracy: 0.7686 - loss: 0.4866 - val_accuracy: 0.7280 - val_loss: 0.5587
Epoch 28/30
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 215ms/step - accuracy: 0.7630 - loss: 0.4742 - val_accuracy: 0.7580 - val_loss: 0.5101
Epoch 29/30
```

```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation_dropout.keras")
test_loss, test_acc = test_model.evaluate(test_ds)
print(f"Test accuracy: {test_acc:.3f}")
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 92ms/step - accuracy: 0.7054 - loss: 0.5452
Test accuracy: 0.718
```

Model 2) Increased the training sample size to 1500 while incorporating MaxPooling, Data Augmentation, and applying Dropout regularization with a rate of 0.5.

```
import os
import shutil
import pathlib

# Define original folders (your dataset path)
source_train_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/train")
source_validation_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/validation")
source_test_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/test")

# Destination for balanced dataset subsets
target_dataset_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset")

def create_dataset_subset(subset_name, original_folder, start_index, end_index):
    print(f"\nCreating '{subset_name}' subset from {original_folder} [{start_index}:{end_index}]")

    for category in ["cats", "dogs"]:
        src_dir = original_folder / category
        dest_dir = target_dataset_dir / subset_name / category

        # Clean old files if they exist
        if dest_dir.exists():
            shutil.rmtree(dest_dir)

        os.makedirs(dest_dir, exist_ok=True)

        files = sorted(os.listdir(src_dir))

        # Validate end_index
        if end_index > len(files):
            print(f" WARNING: end_index {end_index} exceeds available files ({len(files)}). Adjusting.")
            end_index = len(files)

        subset_files = files[start_index:end_index]
        print(f"Copying {len(subset_files)} '{category}' images to '{dest_dir}'...")

        for fname in subset_files:
            src_file = src_dir / fname
            dst_file = dest_dir / fname
            shutil.copyfile(src_file, dst_file)

    print(f" Subset '{subset_name}' created.")

def main():
```

```
    create_dataset_subset("train", source_train_dir, 0, 750)
    create_dataset_subset("validation", source_validation_dir, 0, 300)
    create_dataset_subset("test",   source_test_dir, 0, 300)

if __name__ == "__main__":
    main()
```

```
Creating 'train' subset from /content/drive/MyDrive/cats_vs_dogs_small/train [0:750]
Copying 750 'cats' images to '/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/train/cats'...
Copying 750 'dogs' images to '/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/train/dogs'...
 Subset 'train' created.

Creating 'validation' subset from /content/drive/MyDrive/cats_vs_dogs_small/validation [0:300]
Copying 300 'cats' images to '/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/validation/cats'...
Copying 300 'dogs' images to '/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/validation/dogs'...
 Subset 'validation' created.

Creating 'test' subset from /content/drive/MyDrive/cats_vs_dogs_small/test [0:300]
Copying 300 'cats' images to '/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/test/cats'...
Copying 300 'dogs' images to '/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/test/dogs'...
 Subset 'test' created.
```

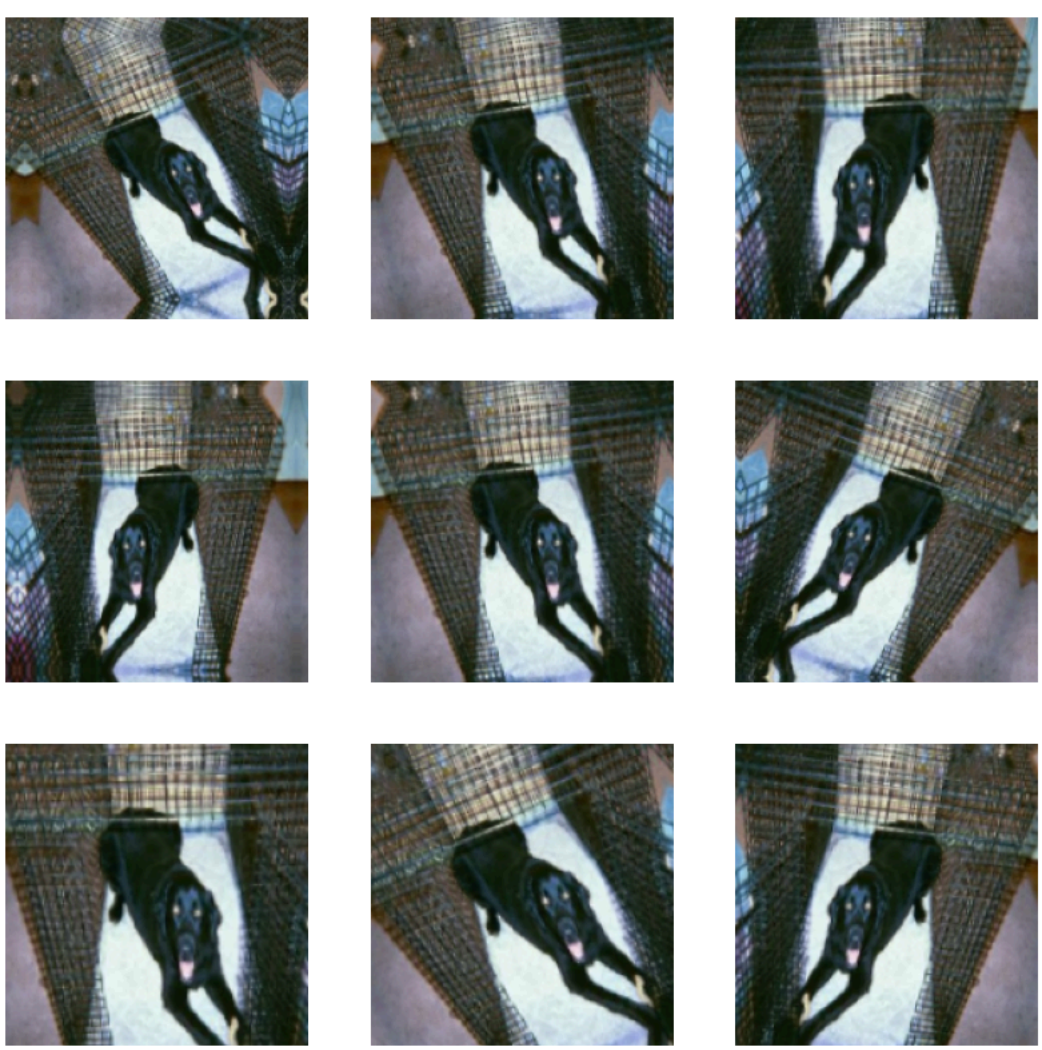
```
from tensorflow.keras.utils import image_dataset_from_directory
from pathlib import Path

target_dataset_dir = Path("/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset")

train_ds = image_dataset_from_directory(
    target_dataset_dir / "train",
    image_size=(180, 180),
    batch_size=32
)

val_ds = image_dataset_from_directory(
    target_dataset_dir / "validation",
    image_size=(180, 180),
    batch_size=32
)

test_ds = image_dataset_from_directory(
    target_dataset_dir / "test",
    image_size=(180, 180),
    batch_size=32
)

# Prefetch for speed
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.prefetch(buffer_size=AUTOTUNE)
```

```
Found 1500 files belonging to 2 classes.
Found 600 files belonging to 2 classes.
Found 600 files belonging to 2 classes.
```

Designing a new convolutional neural network with image augmentation and dropout layers

```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

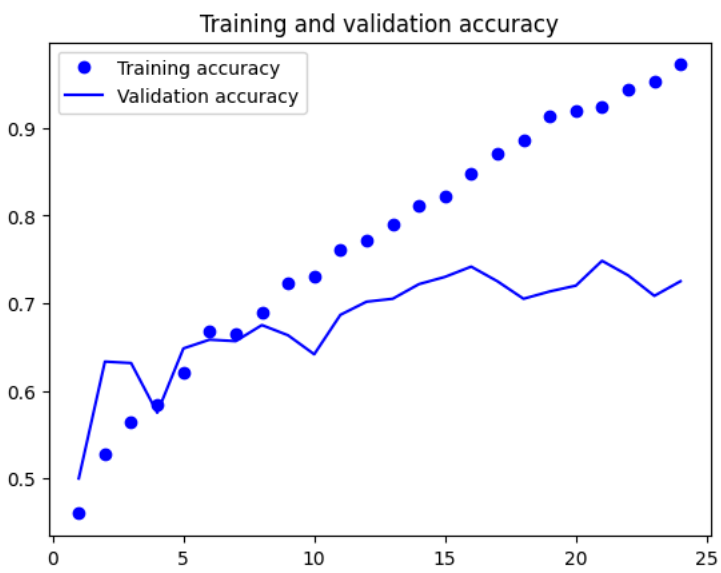
```
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
from keras.callbacks import EarlyStopping
from keras import regularizers

# used early stopping to stop optimization when it isn't helping any more.
early_stopping_monitor = EarlyStopping(patience=10)


data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)



plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_images[0].numpy().astype("uint8"))
        plt.axis("off")
```

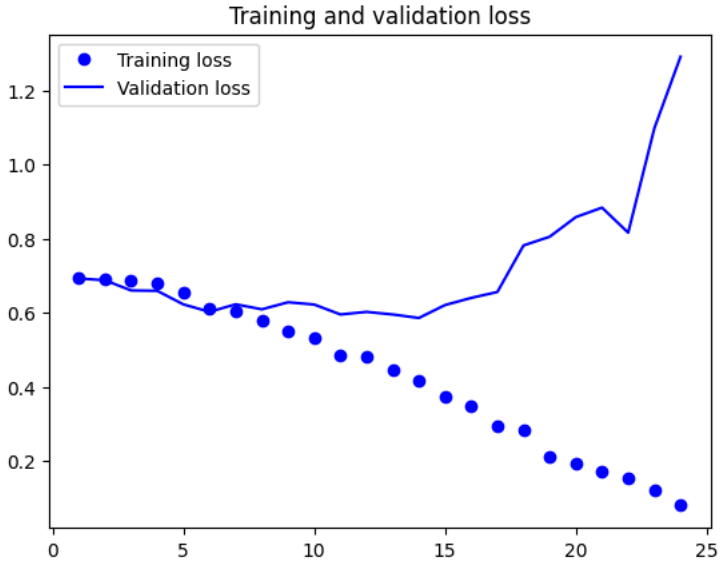


```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch.keras",
        save_best_only=True,
        monitor="val_loss"), early_stopping_monitor
]
history = model.fit(
    train_ds,
    epochs=30,
    validation_data=val_ds,
    callbacks=callbacks)
```

```
Epoch 1/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 17s 311ms/step - accuracy: 0.4481 - loss: 0.6971 - val_accuracy: 0.5000 - val_loss: 0.6928
Epoch 2/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 12s 159ms/step - accuracy: 0.5278 - loss: 0.6926 - val_accuracy: 0.6333 - val_loss: 0.6880
Epoch 3/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 172ms/step - accuracy: 0.5687 - loss: 0.6919 - val_accuracy: 0.6317 - val_loss: 0.6606
Epoch 4/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 7s 159ms/step - accuracy: 0.5563 - loss: 0.6871 - val_accuracy: 0.5750 - val_loss: 0.6594
Epoch 5/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 165ms/step - accuracy: 0.6050 - loss: 0.6625 - val_accuracy: 0.6483 - val_loss: 0.6229
Epoch 6/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 188ms/step - accuracy: 0.6514 - loss: 0.6195 - val_accuracy: 0.6583 - val_loss: 0.6029
Epoch 7/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 12s 216ms/step - accuracy: 0.6586 - loss: 0.6131 - val_accuracy: 0.6567 - val_loss: 0.6232
Epoch 8/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 160ms/step - accuracy: 0.6673 - loss: 0.6035 - val_accuracy: 0.6750 - val_loss: 0.6096
Epoch 9/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 141ms/step - accuracy: 0.7127 - loss: 0.5598 - val_accuracy: 0.6633 - val_loss: 0.6286
Epoch 10/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 169ms/step - accuracy: 0.6925 - loss: 0.5598 - val_accuracy: 0.6417 - val_loss: 0.6225
Epoch 11/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 157ms/step - accuracy: 0.7372 - loss: 0.5072 - val_accuracy: 0.6867 - val_loss: 0.5957
Epoch 12/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 7s 157ms/step - accuracy: 0.7453 - loss: 0.5108 - val_accuracy: 0.7017 - val_loss: 0.6027
Epoch 13/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 12s 193ms/step - accuracy: 0.7832 - loss: 0.4648 - val_accuracy: 0.7050 - val_loss: 0.5957
Epoch 14/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 7s 143ms/step - accuracy: 0.7953 - loss: 0.4324 - val_accuracy: 0.7217 - val_loss: 0.5862
Epoch 15/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 154ms/step - accuracy: 0.8211 - loss: 0.3829 - val_accuracy: 0.7300 - val_loss: 0.6211
Epoch 16/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 154ms/step - accuracy: 0.8364 - loss: 0.3654 - val_accuracy: 0.7417 - val_loss: 0.6403
Epoch 17/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 161ms/step - accuracy: 0.8563 - loss: 0.3236 - val_accuracy: 0.7250 - val_loss: 0.6563
Epoch 18/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 176ms/step - accuracy: 0.8823 - loss: 0.2973 - val_accuracy: 0.7050 - val_loss: 0.7819
Epoch 19/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 152ms/step - accuracy: 0.9174 - loss: 0.2193 - val_accuracy: 0.7133 - val_loss: 0.8054
Epoch 20/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 160ms/step - accuracy: 0.9152 - loss: 0.2039 - val_accuracy: 0.7200 - val_loss: 0.8584
Epoch 21/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 146ms/step - accuracy: 0.9053 - loss: 0.2084 - val_accuracy: 0.7483 - val_loss: 0.8839
Epoch 22/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 166ms/step - accuracy: 0.9427 - loss: 0.1438 - val_accuracy: 0.7317 - val_loss: 0.8164
Epoch 23/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 7s 155ms/step - accuracy: 0.9528 - loss: 0.1191 - val_accuracy: 0.7083 - val_loss: 1.0980
Epoch 24/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 176ms/step - accuracy: 0.9613 - loss: 0.1030 - val_accuracy: 0.7250 - val_loss: 1.2910
```

```
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```






```
test_model = keras.models.load_model("convnet_from_scratch.keras")
test_loss, test_acc = test_model.evaluate(test_ds)
print(f"Test accuracy: {test_acc:.3f}")
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 3s 85ms/step - accuracy: 0.7144 - loss: 0.6254
Test accuracy: 0.720
```

Model 3: Increasing the Training sample size to 1700

```
import os
import shutil
import pathlib
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.utils import image_dataset_from_directory

# 1. Function to Create Subsets
def make_subset(subset_name, original_dir, target_dataset_dir, start_index, end_index):
    print(f"\nCreating subset: {subset_name} from {original_dir} [{start_index}:{end_index}]")

    for category in ["cats", "dogs"]:
        src_dir = original_dir / category
        dest_dir = target_dataset_dir / subset_name / category

        if dest_dir.exists():
            shutil.rmtree(dest_dir)

        os.makedirs(dest_dir, exist_ok=True)

        files = sorted(os.listdir(src_dir))

        if end_index > len(files):
            print(f" end_index {end_index} exceeds available files ({len(files)}). Adjusting...")
```

```
            end_index = len(files)

        subset_files = files[start_index:end_index]

        print(f"Copying {len(subset_files)} files from {src_dir} to {dest_dir}...")

        for fname in subset_files:
            src_file = src_dir / fname
            dst_file = dest_dir / fname

            shutil.copyfile(src_file, dst_file)

    print(f"Subset '{subset_name}' created successfully!")

# 2. Paths Setup
source_train_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/train")
source_validation_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/validation")
source_test_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/test")

target_dataset_dir = pathlib.Path("/content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset")

# 3. Create Subsets
make_subset("train_4", source_train_dir, target_dataset_dir, 0, 850)
make_subset("validation_4", source_validation_dir, target_dataset_dir, 0, 250)
make_subset("test_4",   source_test_dir, target_dataset_dir, 0, 250)

# 4. Load the Subset Datasets
train_ds_4 = image_dataset_from_directory(
    target_dataset_dir / "train_4",
    image_size=(180, 180),
    batch_size=32
)

val_ds_4 = image_dataset_from_directory(
    target_dataset_dir / "validation_4",
    image_size=(180, 180),
    batch_size=32
)

test_ds_4 = image_dataset_from_directory(
    target_dataset_dir / "test_4",
    image_size=(180, 180),
    batch_size=32
)

# 5. Define the Callbacks
early_stopping_monitor = keras.callbacks.EarlyStopping(
    monitor="val_loss",
    patience=3,
    restore_best_weights=True
)

callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch.keras",
        save_best_only=True,
        monitor="val_loss"
    ),
    early_stopping_monitor
]

# 6. Build Your Model
model = keras.Sequential([
    keras.layers.Rescaling(1./255, input_shape=(180, 180, 3)),
    keras.layers.Conv2D(32, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(),
    keras.layers.Conv2D(128, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(),
    keras.layers.Flatten(),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

```
# 7. Train the Model
history = model.fit(
    train_ds_4,
    epochs=30,
    validation_data=val_ds_4,
    callbacks=callbacks
)
```

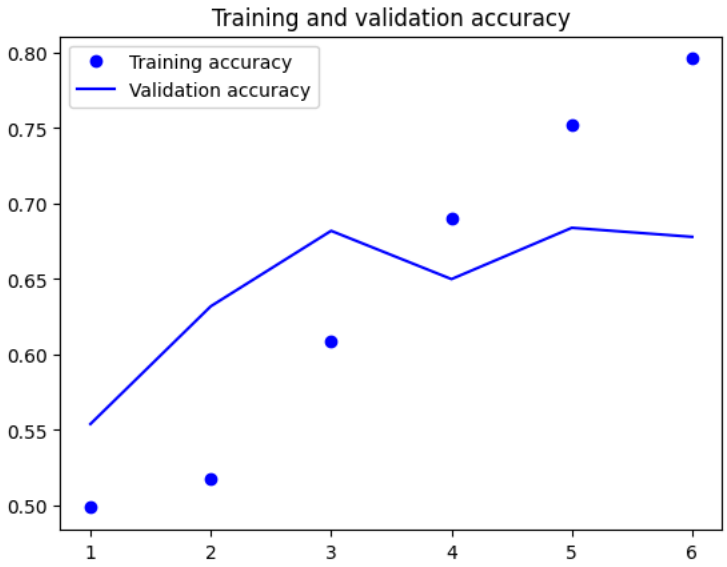
```
Creating subset: train_4 from /content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/train [0:850]
 end_index 850 exceeds available files (750). Adjusting...
Copying 750 files from /content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/train/cats to /content/drive/
Copying 750 files from /content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/train/dogs to /content/drive/
Subset 'train_4' created successfully!

Creating subset: validation_4 from /content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/validation [0:250
Copying 250 files from /content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/validation/cats to /content/d
Copying 250 files from /content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/validation/dogs to /content/d
Subset 'validation_4' created successfully!

Creating subset: test_4 from /content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/test [0:250]
Copying 250 files from /content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/test/cats to /content/drive/M
Copying 250 files from /content/drive/MyDrive/cats_vs_dogs_small/cats_vs_dogs_small_subset/test/dogs to /content/drive/M
Subset 'test_4' created successfully!
Found 1500 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
Epoch 1/30
/usr/local/lib/python3.11/dist-packages/keras/src/layers/preprocessing/tf_data_layer.py:19: UserWarning: Do not pass an
  super().__init__(**kwargs)
47/47 ━━━━━━━━━━━━━━━━━━━━ 27s 403ms/step - accuracy: 0.4801 - loss: 1.0577 - val_accuracy: 0.5540 - val_loss: 0.6909
Epoch 2/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 14s 309ms/step - accuracy: 0.4838 - loss: 0.6925 - val_accuracy: 0.6320 - val_loss: 0.6469
Epoch 3/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 232ms/step - accuracy: 0.5761 - loss: 0.6697 - val_accuracy: 0.6820 - val_loss: 0.6006
Epoch 4/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 167ms/step - accuracy: 0.6900 - loss: 0.5832 - val_accuracy: 0.6500 - val_loss: 0.6293
Epoch 5/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 168ms/step - accuracy: 0.7359 - loss: 0.5216 - val_accuracy: 0.6840 - val_loss: 0.6207
Epoch 6/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 7s 146ms/step - accuracy: 0.7873 - loss: 0.4434 - val_accuracy: 0.6780 - val_loss: 0.6669
```

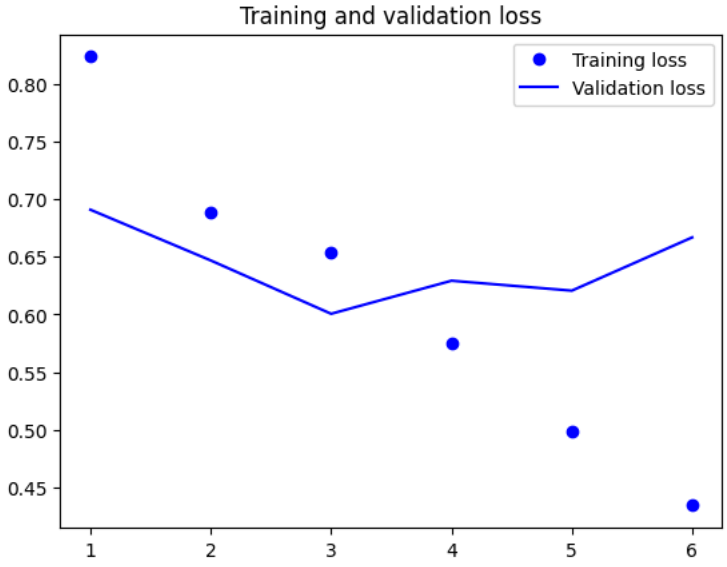
```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

```
plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_images[0].numpy().astype("uint8"))
        plt.axis("off")
```

```
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```






```
test_model = keras.models.load_model("convnet_from_scratch.keras")
test_loss, test_acc = test_model.evaluate(test_ds_4)
print(f"Test accuracy: {test_acc:.3f}")
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 3s 139ms/step - accuracy: 0.6496 - loss: 0.6409
Test accuracy: 0.642
```

Feature extraction enhanced through data augmentation

Creating and locking the VGG16 convolutional base

Pre-Trained Model - 1000 Training samples

```
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False,
    input_shape=(180, 180, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_t
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
conv_base.summary()
```

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_layer_12 (InputLayer)	(None, 180, 180, 3)	0
block1_conv1 (Conv2D)	(None, 180, 180, 64)	1,792
block1_conv2 (Conv2D)	(None, 180, 180, 64)	36,928
block1_pool (MaxPooling2D)	(None, 90, 90, 64)	0
block2_conv1 (Conv2D)	(None, 90, 90, 128)	73,856
block2_conv2 (Conv2D)	(None, 90, 90, 128)	147,584
block2_pool (MaxPooling2D)	(None, 45, 45, 128)	0
block3_conv1 (Conv2D)	(None, 45, 45, 256)	295,168
block3_conv2 (Conv2D)	(None, 45, 45, 256)	590,080
block3_conv3 (Conv2D)	(None, 45, 45, 256)	590,080
block3_pool (MaxPooling2D)	(None, 22, 22, 256)	0
block4_conv1 (Conv2D)	(None, 22, 22, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 22, 22, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 22, 22, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 11, 11, 512)	0
block5_conv1 (Conv2D)	(None, 11, 11, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 11, 11, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 11, 11, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 5, 5, 512)	0

Total params: 14,714,688 (56.13 MB)
Trainable params: 14,714,688 (56.13 MB)
Non-trainable params: 0 (0.00 B)

```
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False)
```

```
conv_base.trainable = True
for layer in conv_base.layers[:-4]:
    layer.trainable = False
```

Adding a data augmentation stage and a classifier to the convolutional base

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

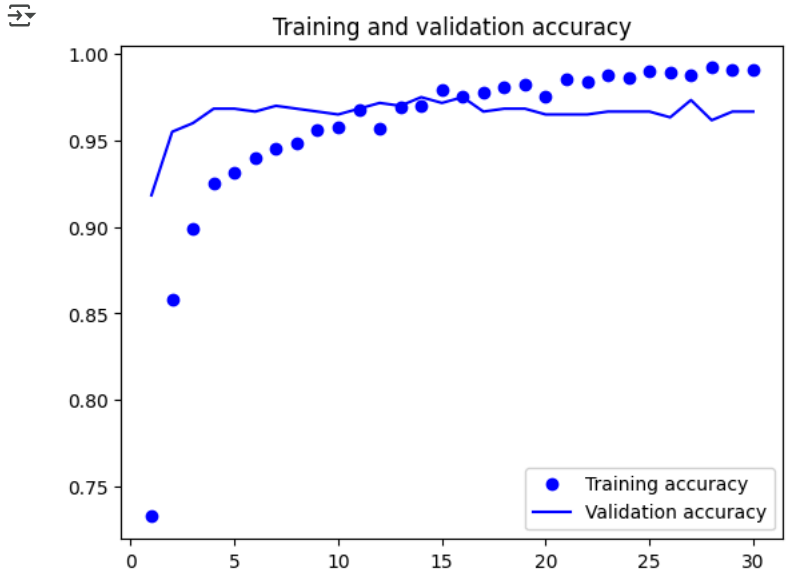
```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = keras.applications.vgg16.preprocess_input(x)
x = conv_base(x)
x = layers.Flatten()(x)
x = layers.Dense(256)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(loss="binary_crossentropy",
              optimizer=keras.optimizers.RMSprop(learning_rate=1e-5),
              metrics=["accuracy"])
```

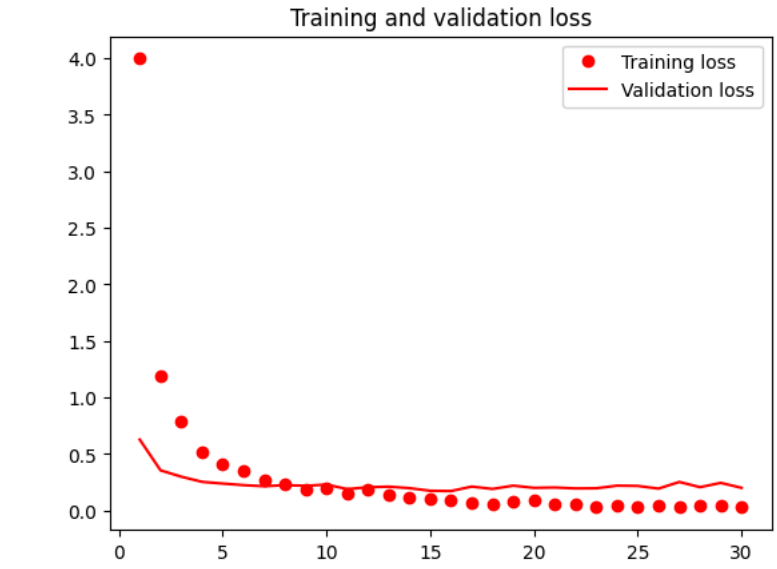
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="fine_tuning.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
```

```
    train_ds,
    epochs=30,
    validation_data=val_ds,
    callbacks=callbacks)
```

```
Epoch 1/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 18s 269ms/step - accuracy: 0.6610 - loss: 6.2363 - val_accuracy: 0.9183 - val_loss: 0.6277
Epoch 2/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 195ms/step - accuracy: 0.8583 - loss: 1.2681 - val_accuracy: 0.9550 - val_loss: 0.3561
Epoch 3/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 177ms/step - accuracy: 0.8970 - loss: 0.8774 - val_accuracy: 0.9600 - val_loss: 0.2995
Epoch 4/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 203ms/step - accuracy: 0.9254 - loss: 0.5290 - val_accuracy: 0.9683 - val_loss: 0.2541
Epoch 5/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 201ms/step - accuracy: 0.9401 - loss: 0.3808 - val_accuracy: 0.9683 - val_loss: 0.2389
Epoch 6/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 187ms/step - accuracy: 0.9404 - loss: 0.3779 - val_accuracy: 0.9667 - val_loss: 0.2249
Epoch 7/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 178ms/step - accuracy: 0.9414 - loss: 0.2563 - val_accuracy: 0.9700 - val_loss: 0.2151
Epoch 8/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 193ms/step - accuracy: 0.9505 - loss: 0.2225 - val_accuracy: 0.9683 - val_loss: 0.2247
Epoch 9/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 12s 225ms/step - accuracy: 0.9547 - loss: 0.1888 - val_accuracy: 0.9667 - val_loss: 0.2193
Epoch 10/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 17s 150ms/step - accuracy: 0.9625 - loss: 0.1426 - val_accuracy: 0.9650 - val_loss: 0.2300
Epoch 11/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 202ms/step - accuracy: 0.9686 - loss: 0.1710 - val_accuracy: 0.9683 - val_loss: 0.1898
Epoch 12/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 168ms/step - accuracy: 0.9555 - loss: 0.1784 - val_accuracy: 0.9717 - val_loss: 0.2065
Epoch 13/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 160ms/step - accuracy: 0.9726 - loss: 0.1305 - val_accuracy: 0.9700 - val_loss: 0.2116
Epoch 14/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 174ms/step - accuracy: 0.9701 - loss: 0.1005 - val_accuracy: 0.9750 - val_loss: 0.1982
Epoch 15/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 165ms/step - accuracy: 0.9809 - loss: 0.0877 - val_accuracy: 0.9717 - val_loss: 0.1749
Epoch 16/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 189ms/step - accuracy: 0.9813 - loss: 0.0618 - val_accuracy: 0.9750 - val_loss: 0.1727
Epoch 17/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 181ms/step - accuracy: 0.9801 - loss: 0.0685 - val_accuracy: 0.9667 - val_loss: 0.2121
Epoch 18/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 163ms/step - accuracy: 0.9816 - loss: 0.0507 - val_accuracy: 0.9683 - val_loss: 0.1913
Epoch 19/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 150ms/step - accuracy: 0.9837 - loss: 0.0868 - val_accuracy: 0.9683 - val_loss: 0.2204
Epoch 20/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 188ms/step - accuracy: 0.9831 - loss: 0.0911 - val_accuracy: 0.9650 - val_loss: 0.2008
Epoch 21/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 164ms/step - accuracy: 0.9864 - loss: 0.0511 - val_accuracy: 0.9650 - val_loss: 0.2039
Epoch 22/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 168ms/step - accuracy: 0.9873 - loss: 0.0409 - val_accuracy: 0.9650 - val_loss: 0.1959
Epoch 23/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 175ms/step - accuracy: 0.9897 - loss: 0.0402 - val_accuracy: 0.9667 - val_loss: 0.1970
Epoch 24/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 164ms/step - accuracy: 0.9829 - loss: 0.0527 - val_accuracy: 0.9667 - val_loss: 0.2203
Epoch 25/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 165ms/step - accuracy: 0.9906 - loss: 0.0298 - val_accuracy: 0.9667 - val_loss: 0.2179
Epoch 26/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 175ms/step - accuracy: 0.9921 - loss: 0.0289 - val_accuracy: 0.9633 - val_loss: 0.1941
Epoch 27/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 166ms/step - accuracy: 0.9856 - loss: 0.0359 - val_accuracy: 0.9733 - val_loss: 0.2531
Epoch 28/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 154ms/step - accuracy: 0.9921 - loss: 0.0540 - val_accuracy: 0.9617 - val_loss: 0.2070
Epoch 29/30
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 162ms/step - accuracy: 0.9895 - loss: 0.0429 - val_accuracy: 0.9667 - val_loss: 0.2449
```

```
import matplotlib.pyplot as plt
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "ro", label="Training loss")
plt.plot(epochs, val_loss, "r", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```






```
model = keras.models.load_model("fine_tuning.keras")
test_loss, test_acc = model.evaluate(test_ds)
print(f"Test accuracy: {test_acc:.3f}")
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 3s 104ms/step - accuracy: 0.9687 - loss: 0.2743
Test accuracy: 0.970
```

Pre-Trained Model Utilizing 1500 Training Samples

```
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False,
    input_shape=(180, 180, 3))


conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False)

conv_base.trainable = True
for layer in conv_base.layers[:-4]:
    layer.trainable = False


data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = keras.applications.vgg16.preprocess_input(x)
x = conv_base(x)
x = layers.Flatten()(x)
x = layers.Dense(256)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(loss="binary_crossentropy",
              optimizer=keras.optimizers.RMSprop(learning_rate=1e-5),
              metrics=["accuracy"])
```

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="fine_tuning2.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_ds,
    epochs=10,
    validation_data=val_ds,
    callbacks=callbacks)
```

```
Epoch 1/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 12s 204ms/step - accuracy: 0.6683 - loss: 6.3828 - val_accuracy: 0.9000 - val_loss: 0.8866
Epoch 2/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 12s 243ms/step - accuracy: 0.8437 - loss: 1.4751 - val_accuracy: 0.9400 - val_loss: 0.3731
Epoch 3/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 17s 182ms/step - accuracy: 0.9082 - loss: 0.8091 - val_accuracy: 0.9600 - val_loss: 0.3230
Epoch 4/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 202ms/step - accuracy: 0.9206 - loss: 0.7035 - val_accuracy: 0.9683 - val_loss: 0.2047
Epoch 5/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 176ms/step - accuracy: 0.9348 - loss: 0.4545 - val_accuracy: 0.9717 - val_loss: 0.2107
Epoch 6/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 180ms/step - accuracy: 0.9537 - loss: 0.2881 - val_accuracy: 0.9800 - val_loss: 0.1634
Epoch 7/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 195ms/step - accuracy: 0.9514 - loss: 0.2768 - val_accuracy: 0.9800 - val_loss: 0.1604
Epoch 8/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 185ms/step - accuracy: 0.9570 - loss: 0.2763 - val_accuracy: 0.9783 - val_loss: 0.1656
Epoch 9/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 8s 161ms/step - accuracy: 0.9591 - loss: 0.2314 - val_accuracy: 0.9683 - val_loss: 0.2151
Epoch 10/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 176ms/step - accuracy: 0.9618 - loss: 0.1491 - val_accuracy: 0.9750 - val_loss: 0.1386
```

```
model = keras.models.load_model("fine_tuning2.keras")
test_loss, test_acc = model.evaluate(test_ds)
print(f"Test accuracy: {test_acc:.3f}")
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 3s 114ms/step - accuracy: 0.9713 - loss: 0.2675
Test accuracy: 0.970
```

Pre-Trained Model Utilizing 1700 Training Samples

```
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False,
    input_shape=(180, 180, 3))
```

```
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False)

conv_base.trainable = True
for layer in conv_base.layers[:-4]:
    layer.trainable = False
```

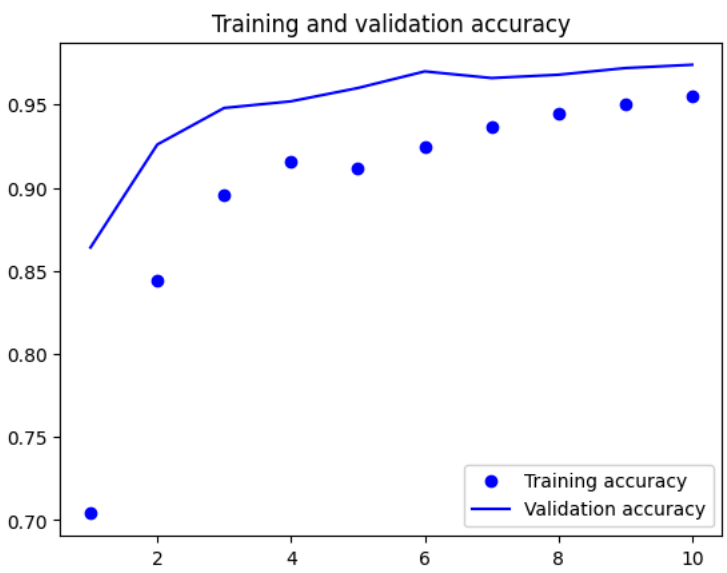
```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

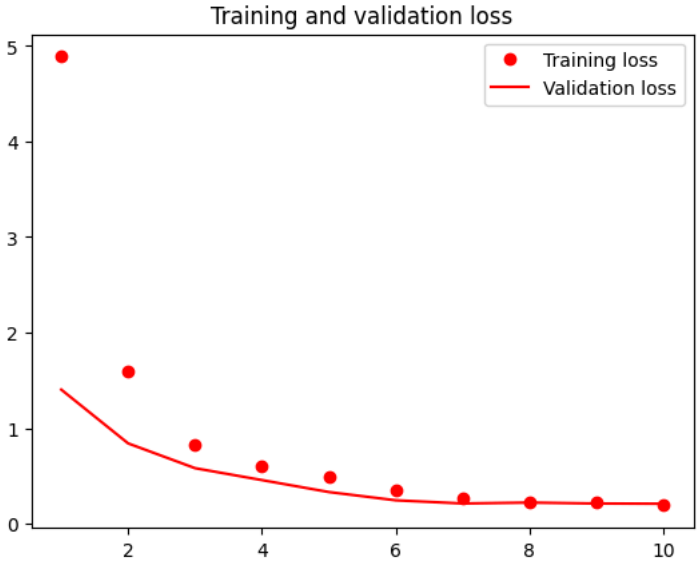
```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
```

```
x = keras.applications.vgg16.preprocess_input(x)
x = conv_base(x)
x = layers.Flatten()(x)
x = layers.Dense(256)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(loss="binary_crossentropy",
              optimizer=keras.optimizers.RMSprop(learning_rate=1e-5),
              metrics=["accuracy"])


callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="fine_tuning3.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_ds_4,
    epochs=10,
    validation_data=val_ds_4,
    callbacks=callbacks)
```

```
Epoch 1/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 15s 264ms/step - accuracy: 0.6315 - loss: 7.2205 - val_accuracy: 0.8640 - val_loss: 1.4027
Epoch 2/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 196ms/step - accuracy: 0.8256 - loss: 1.8493 - val_accuracy: 0.9260 - val_loss: 0.8396
Epoch 3/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 196ms/step - accuracy: 0.8807 - loss: 0.9664 - val_accuracy: 0.9480 - val_loss: 0.5804
Epoch 4/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 209ms/step - accuracy: 0.9222 - loss: 0.5705 - val_accuracy: 0.9520 - val_loss: 0.4556
Epoch 5/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 206ms/step - accuracy: 0.9153 - loss: 0.4765 - val_accuracy: 0.9600 - val_loss: 0.3305
Epoch 6/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 193ms/step - accuracy: 0.9342 - loss: 0.3163 - val_accuracy: 0.9700 - val_loss: 0.2439
Epoch 7/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 10s 191ms/step - accuracy: 0.9385 - loss: 0.2588 - val_accuracy: 0.9660 - val_loss: 0.2108
Epoch 8/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 192ms/step - accuracy: 0.9484 - loss: 0.1915 - val_accuracy: 0.9680 - val_loss: 0.2203
Epoch 9/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 9s 194ms/step - accuracy: 0.9547 - loss: 0.2414 - val_accuracy: 0.9720 - val_loss: 0.2105
Epoch 10/10
47/47 ━━━━━━━━━━━━━━━━━━━━ 11s 203ms/step - accuracy: 0.9531 - loss: 0.2379 - val_accuracy: 0.9740 - val_loss: 0.2081
```

```
import matplotlib.pyplot as plt
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "ro", label="Training loss")
plt.plot(epochs, val_loss, "r", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```






```
model = keras.models.load_model("fine_tuning3.keras")
test_loss, test_acc = model.evaluate(test_ds)
print(f"Test accuracy: {test_acc:.3f}")
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 3s 130ms/step - accuracy: 0.9571 - loss: 0.2577
Test accuracy: 0.962
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.