

```
from google.colab import files
files.upload()
```



Choose files kaggle.json

- **kaggle.json**(application/json) - 66 bytes, last modified: 20/10/2024 - 100% done

Saving kaggle.json to kaggle.json

```
66 bytes, last modified: 20/10/2024 - 100% done
```

```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!cd ~/.kaggle/
```

```
!kaggle competitions download -c dogs-vs-cats
```



Downloading dogs-vs-cats.zip to /content

99% 804M/812M [00:08<00:00, 176MB/s]

100% 812M/812M [00:08<00:00, 105MB/s]

```
!unzip -qq dogs-vs-cats.zip
!unzip -qq train.zip
```

Copying Images to Train, Test and Validation Folders

```
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small")

def make_subset(subset_name, start_index, end_index):
    for category in ("cat", "dog"):
        dir = new_base_dir / subset_name / category
        os.makedirs(dir)
        fnames = [f"{category}.{i}.jpg" for i in range(start_index, end_index)]
        for fname in fnames:
            shutil.copyfile(src=original_dir / fname,
                            dst=dir / fname)
make_subset("validation", start_index=0, end_index=500)
make_subset("test", start_index=500, end_index=1000)
make_subset("train", start_index=1000, end_index=2000)
```

Data Preprocessing

```
from tensorflow.keras.utils import image_dataset_from_directory

train_dataset = image_dataset_from_directory(
    new_base_dir / "train",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    new_base_dir / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    new_base_dir / "test",
    image_size=(180, 180),
    batch_size=32)
```



Found 2000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.

Model Building

```

from tensorflow import keras
from tensorflow.keras import layers
from keras import regularizers
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: 981,941 (3.78 MB)

```

from keras.optimizers import Adam
model.compile(loss="binary_crossentropy",
              optimizer="adam",
              metrics=["accuracy"])

for data_batch, labels_batch in train_dataset:
    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,)

```

callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)

```

```

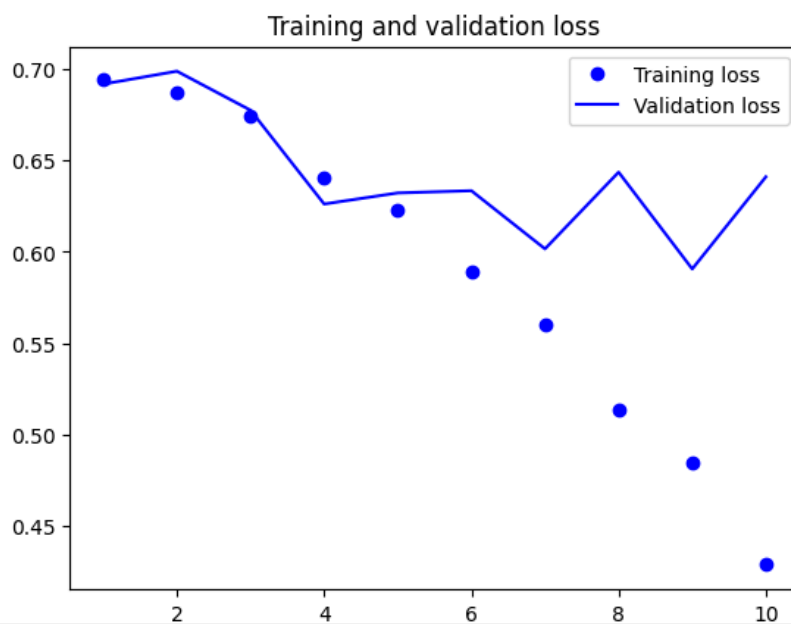
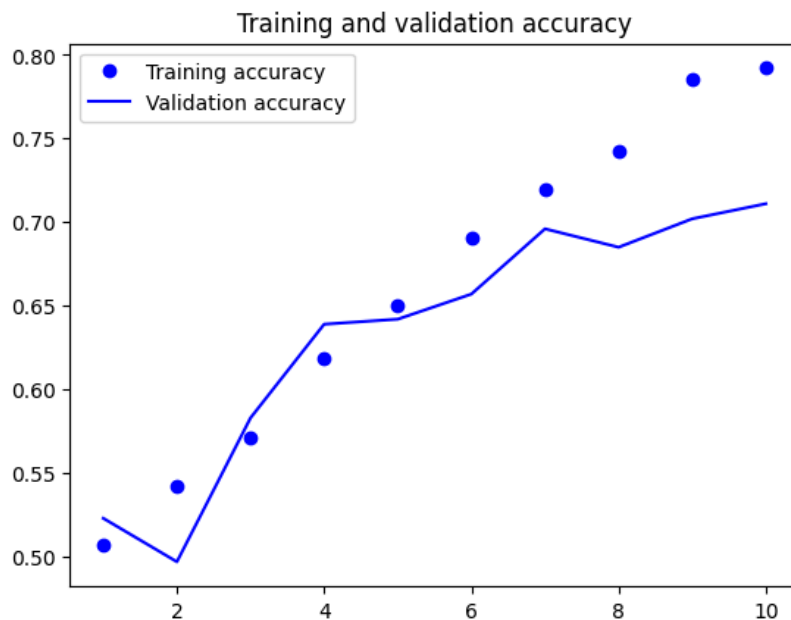
Epoch 1/10
63/63 ————— 198s 3s/step - accuracy: 0.4875 - loss: 0.6985 - val_accuracy: 0.5230 - val_loss: 0.6919
Epoch 2/10
63/63 ————— 205s 3s/step - accuracy: 0.5423 - loss: 0.6891 - val_accuracy: 0.4970 - val_loss: 0.6990
Epoch 3/10
63/63 ————— 195s 3s/step - accuracy: 0.5598 - loss: 0.6802 - val_accuracy: 0.5830 - val_loss: 0.6776
Epoch 4/10
63/63 ————— 205s 3s/step - accuracy: 0.6039 - loss: 0.6492 - val_accuracy: 0.6390 - val_loss: 0.6262
Epoch 5/10
63/63 ————— 214s 3s/step - accuracy: 0.6481 - loss: 0.6228 - val_accuracy: 0.6420 - val_loss: 0.6323
Epoch 6/10
63/63 ————— 246s 3s/step - accuracy: 0.6941 - loss: 0.5789 - val_accuracy: 0.6570 - val_loss: 0.6335
Epoch 7/10
63/63 ————— 204s 3s/step - accuracy: 0.7102 - loss: 0.5613 - val_accuracy: 0.6960 - val_loss: 0.6016
Epoch 8/10
63/63 ————— 215s 3s/step - accuracy: 0.7449 - loss: 0.5165 - val_accuracy: 0.6850 - val_loss: 0.6437
Epoch 9/10
63/63 ————— 238s 3s/step - accuracy: 0.7805 - loss: 0.4916 - val_accuracy: 0.7020 - val_loss: 0.5906
Epoch 10/10
63/63 ————— 204s 3s/step - accuracy: 0.7834 - loss: 0.4377 - val_accuracy: 0.7110 - val_loss: 0.6411

```

```

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()

```



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



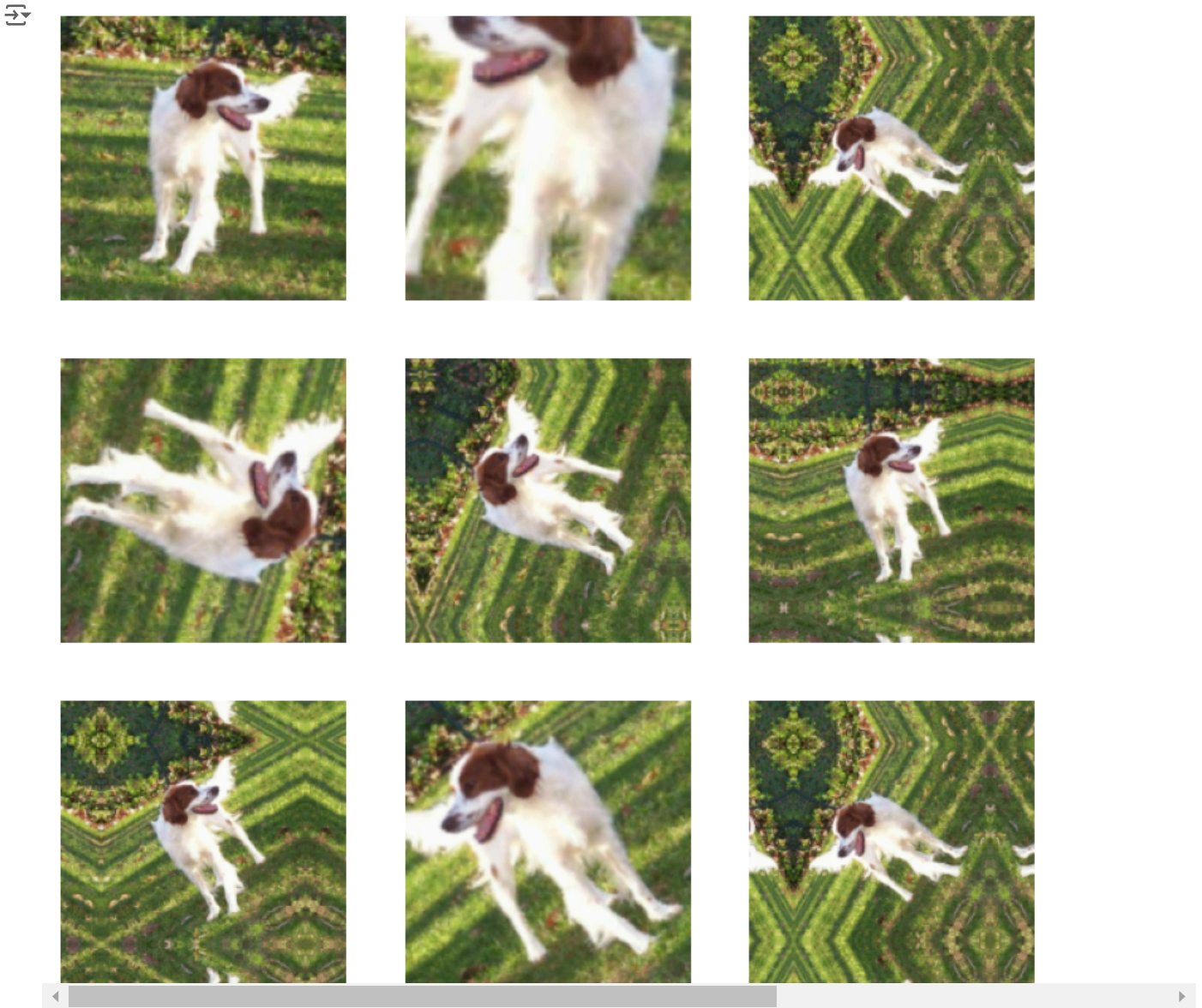
32/32 ————— 29s 892ms/step - accuracy: 0.6978 - loss: 0.6143
Test accuracy: 0.666

Adding Data Augmentation

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.3),
        layers.RandomZoom(0.6),
    ]
)

import warnings
warnings.filterwarnings("ignore")
plt.figure(figsize=(10, 10))
for images, _ in train_dataset.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")
```



```
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.Conv2D(filters=512, strides=2, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.0001))(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.summary()
from keras.optimizers import Adam
model.compile(loss="binary_crossentropy",
              optimizer=Adam(learning_rate=0.0001),
              metrics=["accuracy"])
```

Model: "functional_2"

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 180, 180, 3)	0
rescaling_1 (Rescaling)	(None, 180, 180, 3)	0
conv2d_5 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_4 (MaxPooling2D)	(None, 89, 89, 32)	0
conv2d_6 (Conv2D)	(None, 87, 87, 64)	18,496
max_pooling2d_5 (MaxPooling2D)	(None, 43, 43, 64)	0
conv2d_7 (Conv2D)	(None, 41, 41, 128)	73,856
max_pooling2d_6 (MaxPooling2D)	(None, 20, 20, 128)	0
conv2d_8 (Conv2D)	(None, 18, 18, 256)	295,168
max_pooling2d_7 (MaxPooling2D)	(None, 9, 9, 256)	0
conv2d_9 (Conv2D)	(None, 7, 7, 256)	590,080
flatten_1 (Flatten)	(None, 12544)	0
dense_1 (Dense)	(None, 512)	6,423,040
dropout (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 1)	513

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

Epoch 1/10
63/63 ————— 204s 3s/step - accuracy: 0.4771 - loss: 0.7878 - val_accuracy: 0.5000 - val_loss: 0.7559
Epoch 2/10
63/63 ————— 207s 3s/step - accuracy: 0.5605 - loss: 0.7446 - val_accuracy: 0.6350 - val_loss: 0.7099
Epoch 3/10
63/63 ————— 272s 3s/step - accuracy: 0.6265 - loss: 0.6987 - val_accuracy: 0.6560 - val_loss: 0.6599
Epoch 4/10
63/63 ————— 250s 3s/step - accuracy: 0.6572 - loss: 0.6590 - val_accuracy: 0.6710 - val_loss: 0.6402
Epoch 5/10
63/63 ————— 216s 3s/step - accuracy: 0.6688 - loss: 0.6279 - val_accuracy: 0.6870 - val_loss: 0.6220
Epoch 6/10
63/63 ————— 257s 3s/step - accuracy: 0.7266 - loss: 0.5947 - val_accuracy: 0.6850 - val_loss: 0.6166
Epoch 7/10
63/63 ————— 254s 3s/step - accuracy: 0.7155 - loss: 0.5860 - val_accuracy: 0.7230 - val_loss: 0.5835
Epoch 8/10
63/63 ————— 267s 3s/step - accuracy: 0.7587 - loss: 0.5499 - val_accuracy: 0.7000 - val_loss: 0.5931
Epoch 9/10
63/63 ————— 273s 4s/step - accuracy: 0.7529 - loss: 0.5362 - val_accuracy: 0.7560 - val_loss: 0.5416
Epoch 10/10
63/63 ————— 244s 3s/step - accuracy: 0.7799 - loss: 0.4940 - val_accuracy: 0.7450 - val_loss: 0.5666

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

32/32 ————— 30s 927ms/step - accuracy: 0.7357 - loss: 0.5772
Test accuracy: 0.728

```
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small_1")

def make_subset(subset_name, start_index, end_index):
    for category in ("cat", "dog"):
        dir = new_base_dir / subset_name / category
        os.makedirs(dir)
        fnames = [f"{category}.{i}.jpg" for i in range(start_index, end_index)]
        for fname in fnames:
            shutil.copyfile(src=original_dir / fname,
                            dst=dir / fname)
make_subset("validation", start_index=0, end_index=500)
make_subset("test", start_index=500, end_index=1000)
make_subset("train", start_index=1000, end_index=3000)
```

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

```
train_dataset = image_dataset_from_directory(
    new_base_dir / "train",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    new_base_dir / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    new_base_dir / "test",
    image_size=(180, 180),
    batch_size=32)
```

```
↳ Found 4000 files belonging to 2 classes.
   Found 1000 files belonging to 2 classes.
   Found 1000 files belonging to 2 classes.
```

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation_2000.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
↳ Epoch 1/10
125/125 ————— 380s 3s/step - accuracy: 0.7579 - loss: 0.5348 - val_accuracy: 0.7890 - val_loss: 0.4980
Epoch 2/10
125/125 ————— 384s 3s/step - accuracy: 0.7801 - loss: 0.4908 - val_accuracy: 0.7910 - val_loss: 0.4814
Epoch 3/10
125/125 ————— 381s 3s/step - accuracy: 0.8112 - loss: 0.4415 - val_accuracy: 0.8020 - val_loss: 0.4611
Epoch 4/10
125/125 ————— 381s 3s/step - accuracy: 0.8292 - loss: 0.4155 - val_accuracy: 0.7930 - val_loss: 0.4742
Epoch 5/10
125/125 ————— 383s 3s/step - accuracy: 0.8462 - loss: 0.3742 - val_accuracy: 0.8120 - val_loss: 0.4568
Epoch 6/10
125/125 ————— 390s 3s/step - accuracy: 0.8682 - loss: 0.3354 - val_accuracy: 0.7880 - val_loss: 0.5159
Epoch 7/10
125/125 ————— 387s 3s/step - accuracy: 0.8749 - loss: 0.3144 - val_accuracy: 0.8040 - val_loss: 0.4568
Epoch 8/10
125/125 ————— 441s 3s/step - accuracy: 0.9045 - loss: 0.2660 - val_accuracy: 0.7990 - val_loss: 0.4778
Epoch 9/10
125/125 ————— 385s 3s/step - accuracy: 0.9146 - loss: 0.2437 - val_accuracy: 0.8160 - val_loss: 0.4750
Epoch 10/10
125/125 ————— 441s 3s/step - accuracy: 0.9382 - loss: 0.1997 - val_accuracy: 0.8020 - val_loss: 0.5120
```

```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation_2000.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```


32/32 ————— 28s 863ms/step - accuracy: 0.8005 - loss: 0.4763
 Test accuracy: 0.798

Choosing a Random train sample size

Train Sample 2000

```
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small_5")

def make_subset(subset_name, start_index, end_index):
    for category in ("cat", "dog"):
        dir = new_base_dir / subset_name / category
        os.makedirs(dir)
        fnames = [f"{category}.{i}.jpg" for i in range(start_index, end_index)]
        for fname in fnames:
            shutil.copyfile(src=original_dir / fname,
                            dst=dir / fname)
make_subset("validation", start_index=0, end_index=500)
make_subset("test", start_index=500, end_index=1000)
make_subset("train", start_index=2000, end_index=4000)

from tensorflow.keras.utils import image_dataset_from_directory

train_dataset = image_dataset_from_directory(
    new_base_dir / "train",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    new_base_dir / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    new_base_dir / "test",
    image_size=(180, 180),
    batch_size=32)

Found 4000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation_30000.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

Epoch 1/10
 125/125 ————— 387s 3s/step - accuracy: 0.8611 - loss: 0.3784 - val_accuracy: 0.8230 - val_loss: 0.4401
 Epoch 2/10
 125/125 ————— 436s 3s/step - accuracy: 0.8857 - loss: 0.3121 - val_accuracy: 0.8220 - val_loss: 0.4495
 Epoch 3/10
 125/125 ————— 462s 3s/step - accuracy: 0.9151 - loss: 0.2538 - val_accuracy: 0.8240 - val_loss: 0.4580
 Epoch 4/10
 125/125 ————— 421s 3s/step - accuracy: 0.9255 - loss: 0.2209 - val_accuracy: 0.8180 - val_loss: 0.4827
 Epoch 5/10
 125/125 ————— 384s 3s/step - accuracy: 0.9401 - loss: 0.1888 - val_accuracy: 0.8270 - val_loss: 0.4965
 Epoch 6/10
 125/125 ————— 453s 3s/step - accuracy: 0.9550 - loss: 0.1594 - val_accuracy: 0.8250 - val_loss: 0.5238
 Epoch 7/10
 125/125 ————— 430s 3s/step - accuracy: 0.9661 - loss: 0.1315 - val_accuracy: 0.8260 - val_loss: 0.5467
 Epoch 8/10
 125/125 ————— 383s 3s/step - accuracy: 0.9804 - loss: 0.0979 - val_accuracy: 0.8190 - val_loss: 0.5777
 Epoch 9/10
 125/125 ————— 455s 3s/step - accuracy: 0.9837 - loss: 0.0899 - val_accuracy: 0.8290 - val_loss: 0.6005
 Epoch 10/10

125/125 ————— 429s 3s/step - accuracy: 0.9851 - loss: 0.0837 - val_accuracy: 0.8120 - val_loss: 0.6848

```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation_30000.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

32/32 ————— 25s 749ms/step - accuracy: 0.8082 - loss: 0.4732
Test accuracy: 0.804

T B I < > ↺ ↻ 🖼️ 🔍 ⋮ ⋮ — Ψ 😊 ☰

Train Sample: 3000

Train Sample: 3000

```
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small_3")

def make_subset(subset_name, start_index, end_index):
    for category in ("cat", "dog"):
        dir = new_base_dir / subset_name / category
        os.makedirs(dir)
        fnames = [f"{category}.{i}.jpg" for i in range(start_index, end_index)]
        for fname in fnames:
            shutil.copyfile(src=original_dir / fname,
                            dst=dir / fname)
make_subset("validation", start_index=0, end_index=500)
make_subset("test", start_index=500, end_index=1000)
make_subset("train", start_index=1000, end_index=5000)
```

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

```
train_dataset = image_dataset_from_directory(
    new_base_dir / "train",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    new_base_dir / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    new_base_dir / "test",
    image_size=(180, 180),
    batch_size=32)
```

Found 8000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation_4000.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

Epoch 1/10
250/250 ————— 767s 3s/step - accuracy: 0.9043 - loss: 0.2766 - val_accuracy: 0.8350 - val_loss: 0.4255
Epoch 2/10
250/250 ————— 794s 3s/step - accuracy: 0.9428 - loss: 0.1886 - val_accuracy: 0.8410 - val_loss: 0.4564
Epoch 3/10
250/250 ————— 748s 3s/step - accuracy: 0.9620 - loss: 0.1450 - val_accuracy: 0.8390 - val_loss: 0.4747
Epoch 4/10
250/250 ————— 819s 3s/step - accuracy: 0.9736 - loss: 0.1194 - val_accuracy: 0.8500 - val_loss: 0.4954
Epoch 5/10

```

250/250 ————— 756s 3s/step - accuracy: 0.9797 - loss: 0.0980 - val_accuracy: 0.8410 - val_loss: 0.5422
Epoch 6/10
250/250 ————— 747s 3s/step - accuracy: 0.9880 - loss: 0.0799 - val_accuracy: 0.8410 - val_loss: 0.5479
Epoch 7/10
250/250 ————— 762s 3s/step - accuracy: 0.9900 - loss: 0.0720 - val_accuracy: 0.8320 - val_loss: 0.6330
Epoch 8/10
250/250 ————— 809s 3s/step - accuracy: 0.9932 - loss: 0.0616 - val_accuracy: 0.8170 - val_loss: 0.7698
Epoch 9/10
250/250 ————— 805s 3s/step - accuracy: 0.9845 - loss: 0.0795 - val_accuracy: 0.8320 - val_loss: 0.7296
Epoch 10/10
250/250 ————— 763s 3s/step - accuracy: 0.9936 - loss: 0.0572 - val_accuracy: 0.8260 - val_loss: 0.7521

```

```

test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation_4000.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")

```

```

32/32 ————— 26s 785ms/step - accuracy: 0.8300 - loss: 0.4552
Test accuracy: 0.831

```

```

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

```

```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16\_weights\_tf\_dim\_ordering\_58889256/58889256 0s 0us/step
Model: "vgg16"

```

Layer (type)	Output Shape	Param #
input_layer_3 (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

```

data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.3),
        layers.RandomZoom(0.6)
    ]
)

```


```

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

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)
from keras.optimizers import Adam
model.compile(loss="binary_crossentropy",
              optimizer=Adam(learning_rate=0.0001),
              metrics=["accuracy"])

callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="vgg16_with_data_augmentation.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=1,
    validation_data=validation_dataset,
    callbacks=callbacks)

```

 250/250 ————— 0s 15s/step - accuracy: 0.7751 - loss: 0.5539

KeyboardInterrupt Traceback (most recent call last)

[<ipython-input-42-b9c16b926119>](#) in <cell line: 7>()

5 monitor="val_loss")

6]

----> 7 history = model.fit(

8 train_dataset,

9 epochs=1,

⏮ 2 frames

[/usr/local/lib/python3.10/dist-packages/keras/src/utils/traceback_utils.py](#) in error_handler(*args, **kwargs)

115 filtered_tb = None

116 try:

--> 117 return fn(*args, **kwargs)

118 except Exception as e:

119 filtered_tb = _process_traceback_frames(e.__traceback__)

KeyboardInterrupt: