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

```
!unzip cats_vs_dogs_small_dataset.zip
```

```
!mv cats_vs_dogs_small_dataset.zip cat_dog_small
```

```
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("train", start_index=0, end_index=1000)
make_subset("validation", start_index=1000, end_index=1500)
make_subset("test", start_index=1500, end_index=2000)
```

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

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.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

```
model.summary()
```

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

```
!pip install tensorflow_datasets

import tensorflow_datasets as tfds

cat_dog_small = tfds.load(
    'cats_vs_dogs',
    split='train[:40%]+train[:30%]',
    as_supervised=True,
    shuffle_files=True,
)

train_dataset = cats_dogs_small.split(
    (new_base_dir / "train"),
    image_size=(180, 180),
    batch_size=32)
validation_dataset = cats_dogs_small.split(
    new_base_dir / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = cats_dogs_small.subsplit(
    0.2,
    seed=1234,
    new_base_dir=(
        new_base_dir / "test"),
    image_size=(180, 180),
    batch_size=32)


train_data = train_dataset.flow_from_directory(
    train_dir,
    target_size = (150,150),
    batch_size = 20,
    class_mode = 'binary'
)
validation_data = validation_dataset.flow_from_directory(
    validation_dir,
    target_size = (150,150),
    batch_size = 20,
    class_mode = 'binary'
)


history = model.fit_generator(
    train_data,
    steps_per_epoch = 100,
    epochs = 30,
    validationdata = validation_data,
    validation_steps = 50
)


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=512, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=1024, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model2 = keras.Model(inputs=inputs, outputs=outputs)


model2.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

By adding more layers, the model was reducing overfitting and performance was improved.

```
dataaug = ImageDataGenerator(
    rotation_range = 40,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True,
    fill_mode = 'nearest'
)


cat_dog_small = tfds.load(
    'dataaug',
    split='train[:40%]+train[:30%]',
    as_supervised=True,
    shuffle_files=True,
)

train_dataset = cats_dogs_small.split(
    (new_base_dir / "train"),
    image_size=(180, 180),
    batch_size=32)
validation_dataset = cats_dogs_small.split(
    new_base_dir / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = cats_dogs_small.subsplit(
    0.2,
    seed=1234,
    new_base_dir=(
        new_base_dir / "test"),
    image_size=(180, 180),
    batch_size=32)


train_data = train_dataset.flow_from_directory(
    train_dir,
    target_size = (150,150),
    batch_size = 20,
    class_mode = 'binary'
)
validation_data = validation_dataset.flow_from_directory(
    validation_dir,
    target_size = (150,150),
    batch_size = 20,
    class_mode = 'binary'
)


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.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model3 = keras.Model(inputs=inputs, outputs=outputs)


model3.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

Data Augmentation was used to increase training sample size, and validation increased.

```
dataaug = ImageDataGenerator(
    rotation_range = 80,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True,
    fill_mode = 'nearest'
```

```
)


cat_dog_small = tfds.load(
    'dataaug',
    split='train[:40%]+train[:30%]',
    as_supervised=True,
    shuffle_files=True,
)

train_dataset = cats_dogs_small.split(
    (new_base_dir / "train"),
    image_size=(180, 180),
    batch_size=32)
validation_dataset = cats_dogs_small.split(
    new_base_dir / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = cats_dogs_small.subsplit(
    0.2,
    seed=1234,
    new_base_dir=(
        new_base_dir / "test"),
    image_size=(180, 180),
    batch_size=32)


train_data = train_dataset.flow_from_directory(
    train_dir,
    target_size = (150,150),
    batch_size = 20,
    class_mode = 'binary'
)
validation_data = validation_dataset.flow_from_directory(
    validation_dir,
    target_size = (150,150),
    batch_size = 20,
    class_mode = 'binary'
)


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.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model4 = keras.Model(inputs=inputs, outputs=outputs)


model4.compile(loss="binary_crossentropy",
               optimizer="rmsprop",
               metrics=["accuracy"])
```

The training sample was increased to create a better perform and increase validity and accuracy.

```
Start coding or generate with AI.


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

```
def get_features_and_labels(dataset):
    all_features = []
    all_labels = []
    for images, labels in dataset:
        preprocessed_images = keras.applications.vgg16.preprocess_input(cat_dog_small)
        features = conv_base.predict(preprocessed_images)
        all_features.append(features)
        all_labels.append(labels)
    return np.concatenate(all_features), np.concatenate(all_labels)

train_features, train_labels =  get_features_and_labels(train_data)
val_features, val_labels =  get_features_and_labels(validation_data)
test_features, test_labels =  get_features_and_labels(test_data)


train_features.shape


This was the standard pretrained model. Below is a model with data augmentation for better fit, validation, and accuracy.


dataaug = ImageDataGenerator(
    rotation_range = 80,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True,
    fill_mode = 'nearest'
)


inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = keras.applications.vgg16.preprocess_input(x)
```