In [6]:
```
#!curl -O https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-4869-8368-6DEBA77B919F/kagglecatsanddogs_3367a.zip
```

In [6]:
```
#pip install tensorflow==2.7.0
```

```
 Volume in drive C is OS
 Volume Serial Number is 16A5-7111

 Directory of C:\Users\dtukel\Desktop\jnotebook

12/20/2021  01:55 PM       824,894,548 kagglecatsanddogs_3367a.zip
               1 File(s)    824,894,548 bytes
               0 Dir(s)  23,279,734,528 bytes free
```

In [11]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

In [23]:
```
import os

num_skipped = 0
for folder_name in ("Cat", "Dog"):
    folder_path = os.path.join("PetImages", folder_name)
    for fname in os.listdir(folder_path):
        fpath = os.path.join(folder_path, fname)
        try:
            fobj = open(fpath, "rb")
            is_jfif = tf.compat.as_bytes("JFIF") in fobj.peek(10)
        finally:
            fobj.close()

        if not is_jfif:
            num_skipped += 1
            # Delete corrupted image
            os.remove(fpath)

print("Deleted %d images" % num_skipped)
```

```
Deleted 0 images
```

In [33]:
```
image_size = (180, 180)
batch_size = 32

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "PetImages",
    validation_split=0.2,
    subset="training",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
)
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "PetImages",
    validation_split=0.2,
    subset="validation",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
)
```

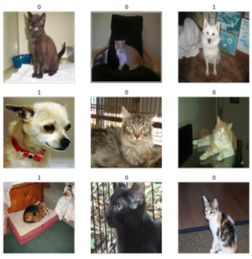
```
Found 23422 files belonging to 2 classes.
Using 18738 files for training.
Found 23422 files belonging to 2 classes.
Using 4684 files for validation.
```

In [4]:
```
train_ds.take(1)
```

Out[4]:
```
<TakeDataset shapes: ((None, 180, 180, 3), (None,)), types: (tf.float32, tf.int32)>
```

In [5]:
```
import matplotlib.pyplot as plt

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



In [6]:
```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
    ]
)
```

In [7]:
```
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")
```

In [8]:
```
augmented_train_ds = train_ds.map(
  lambda x, y: (data_augmentation(x, training=True), y))
```

In [9]:
```
train_ds = train_ds.prefetch(buffer_size=32)
val_ds = val_ds.prefetch(buffer_size=32)
```

In [10]:
```
def make_model(input_shape, num_classes):
    inputs = keras.Input(shape=input_shape)
    # Image augmentation block
    x = data_augmentation(inputs)

    # Entry block
    x = layers.Rescaling(1.0 / 255)(x)
    x = layers.Conv2D(32, 3, strides=2, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    x = layers.Conv2D(64, 3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    previous_block_activation = x  # Set aside residual

    for size in [128, 256, 512, 728]:
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.MaxPooling2D(3, strides=2, padding="same")(x)

        # Project residual
        residual = layers.Conv2D(size, 1, strides=2, padding="same")(
            previous_block_activation
        )
        x = layers.add([x, residual])  # Add back residual
        previous_block_activation = x  # Set aside next residual

    x = layers.SeparableConv2D(1024, 3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    x = layers.GlobalAveragePooling2D()(x)
    if num_classes == 2:
        activation = "sigmoid"
        units = 1
    else:
        activation = "softmax"
        units = num_classes

    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(units, activation=activation)(x)
    return keras.Model(inputs, outputs)


model = make_model(input_shape=image_size + (3,), num_classes=2)
keras.utils.plot_model(model, show_shapes=True)
```

```
('You must install pydot (`pip install pydot`) and install graphviz (see instructions at https://graphviz.gitlab.io/download/) ', 'for plot_model/model_to_dot to work.')
```

In [11]:
```
#pip install pydot
```

```
Collecting pydot
  Downloading pydot-1.4.2-py2.py3-none-any.whl (21 KB)
Requirement already satisfied: pyparsing>=2.1.4 in c:\users\dtukel\anaconda3\lib\site-packages (from pydot) (2.4.7)
Installing collected packages: pydot
Successfully installed pydot-1.4.2
Note: you may need to restart the kernel to use updated packages.
```

In [12]:
```
#pip install graphviz
```

```
Collecting graphviz
  Downloading graphviz-0.19.1-py3-none-any.whl (46 KB)
Installing collected packages: graphviz
Successfully installed graphviz-0.19.1
Note: you may need to restart the kernel to use updated packages.
```

In [12]:
```
epochs = 1

callbacks = [
    keras.callbacks.ModelCheckpoint("save_at_{epoch}.h5"),
]
model.compile(
    optimizer=keras.optimizers.Adam(1e-3),
    loss="binary_crossentropy",
    metrics=["accuracy"],
)
model.fit(
    train_ds, epochs=epochs, callbacks=callbacks, validation_data=val_ds,
)
```

```
586/586 [==============================] - ETA: 0s - loss: 0.3114 - accuracy: 0.8638
```

```
C:\Users\dtukel\Anaconda3\lib\site-packages\keras\engine\functional.py:1410: CustomMaskWarning: Custom mask layers require a config and must override get_config. When loading, the custom mask layer must be passed to the custom_objects argument.
  layer_config = serialize_layer_fn(layer)
```

```
586/586 [==============================] - 1656s 3s/step - loss: 0.3114 - accuracy: 0.8638 - val_loss: 0.4251 - val_accuracy: 0.8245
```

Out[12]:
```
<keras.callbacks.History at 0x1429bdd2250>
```

In [16]:
```
model.save('my_model_catdog.h5')
```

```
C:\Users\dtukel\Anaconda3\lib\site-packages\keras\engine\functional.py:1410: CustomMaskWarning: Custom mask layers require a config and must override get_config. When loading, the custom mask layer must be passed to the custom_objects argument.
  layer_config = serialize_layer_fn(layer)
```

In [13]:
```
img = keras.preprocessing.image.load_img(
    "PetImages/Cat/6779.jpg", target_size=image_size
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)  # Create batch axis

predictions = model.predict(img_array)
score = predictions[0]
print(
    "This image is %.2f percent cat and %.2f percent dog."
    % (100 * (1 - score), 100 * score)
)
```

This image is 50.01 percent cat and 49.99 percent dog.

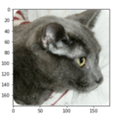
In [14]:
```
new_model = tf.keras.models.load_model('my_model_catdog.h5')
```

In [15]:
```
new_predictions = new_model.predict(img_array)
new_score = new_predictions[0]
print(
    "This image is %.2f percent cat and %.2f percent dog."
    % (100 * (1 - new_score), 100 * new_score)
)
```

This image is 99.53 percent cat and 0.47 percent dog.

In [16]:
```
plt.imshow(img)
```

Out[16]:
```
<matplotlib.image.AxesImage at 0x1de1a1a3b20>
```



In [17]:
```
img = keras.preprocessing.image.load_img(
    "PetImages/Dog/67.jpg", target_size=image_size
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)  # Create batch axis

predictions = model.predict(img_array)
score = predictions[0]
print(
    "This image is %.2f percent cat and %.2f percent dog."
    % (100 * (1 - score), 100 * score)
)
```

This image is 50.00 percent cat and 50.00 percent dog.

In [18]:
```
plt.imshow(img)
```

Out[18]:
```
<matplotlib.image.AxesImage at 0x1de1a703c10>
```

In [ ]: