IST 652 Advanced Topic Presentation

KERAS

GROUP 8
Bastin| Anish Kumar | Sri Venkata Namana

Introduction

This presentation will cover:

- Fundamentals of Keras
- Image classification using Keras
- Advantages and Disadvantages

What is Keras ?

“Keras is a high-level neural networks API, designed to enable fast experimentation with deep neural networks”

Background : Originally developed by François Chollet, Keras was created with the goal of providing a user-friendly interface for building and training neural networks.

It is not a Python Standard Library

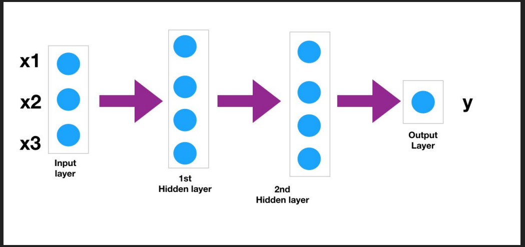
Keras offers compatibility with various backend engines: Keras can run on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK), offering flexibility and compatibility with different computational backends.

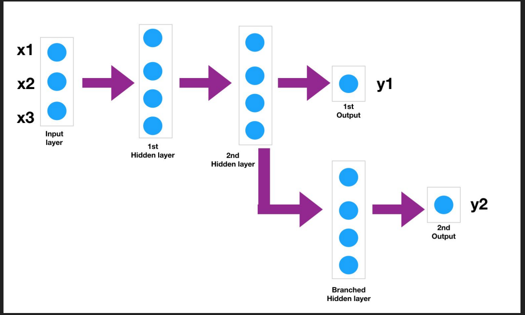
Keras API’s

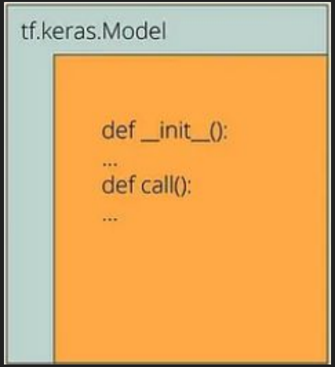
1. Sequential API: Allows users to create models layer-by-layer in a linear stack. Ideal for building simple models where each layer has exactly one input tensor and one output tensor. (Single input, single output models)

2. Functional API: Allows for the creation of complex models with multiple inputs, multiple outputs, shared layers, and branching architectures.

3. Subclassing API: Enables dynamic architecture creation and modification at runtime, making it suitable for research and experimentation.

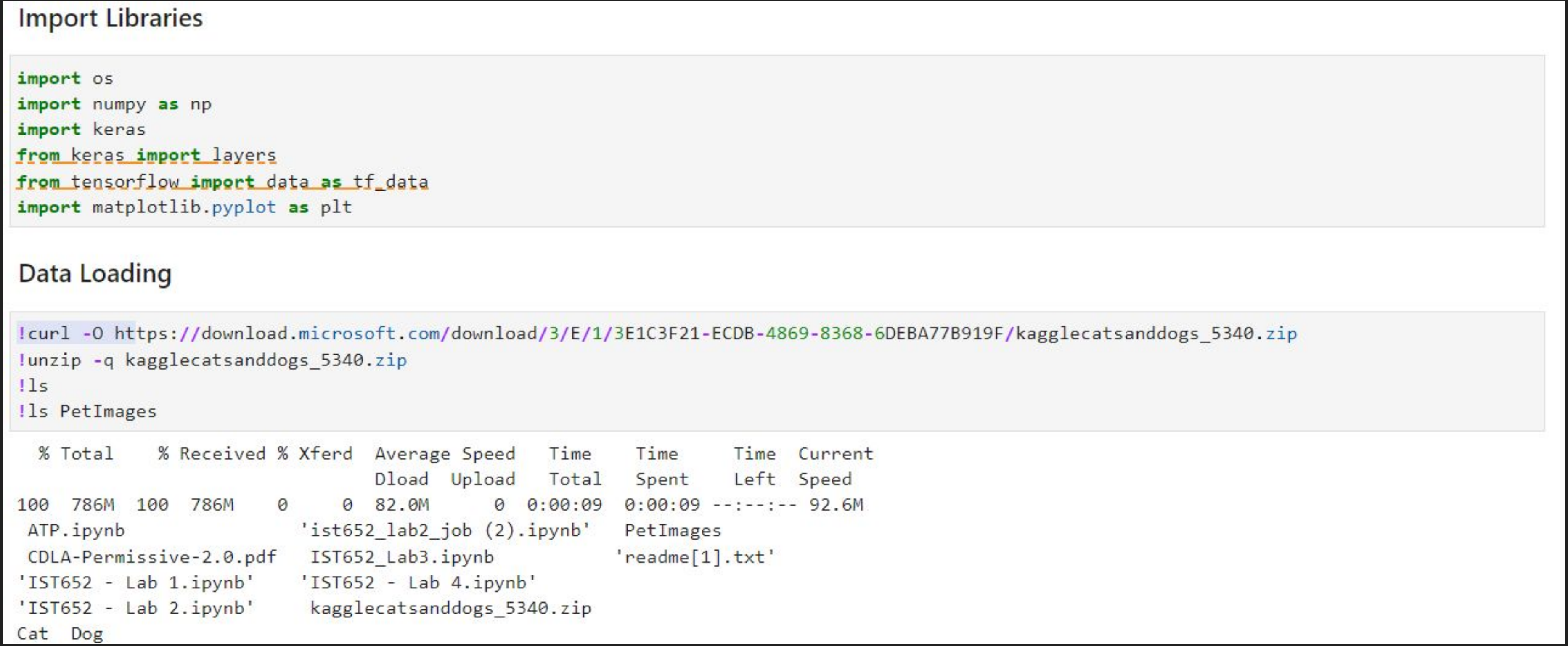






```
def __init__():
...
def call():
...
```

Image Classification Using Keras



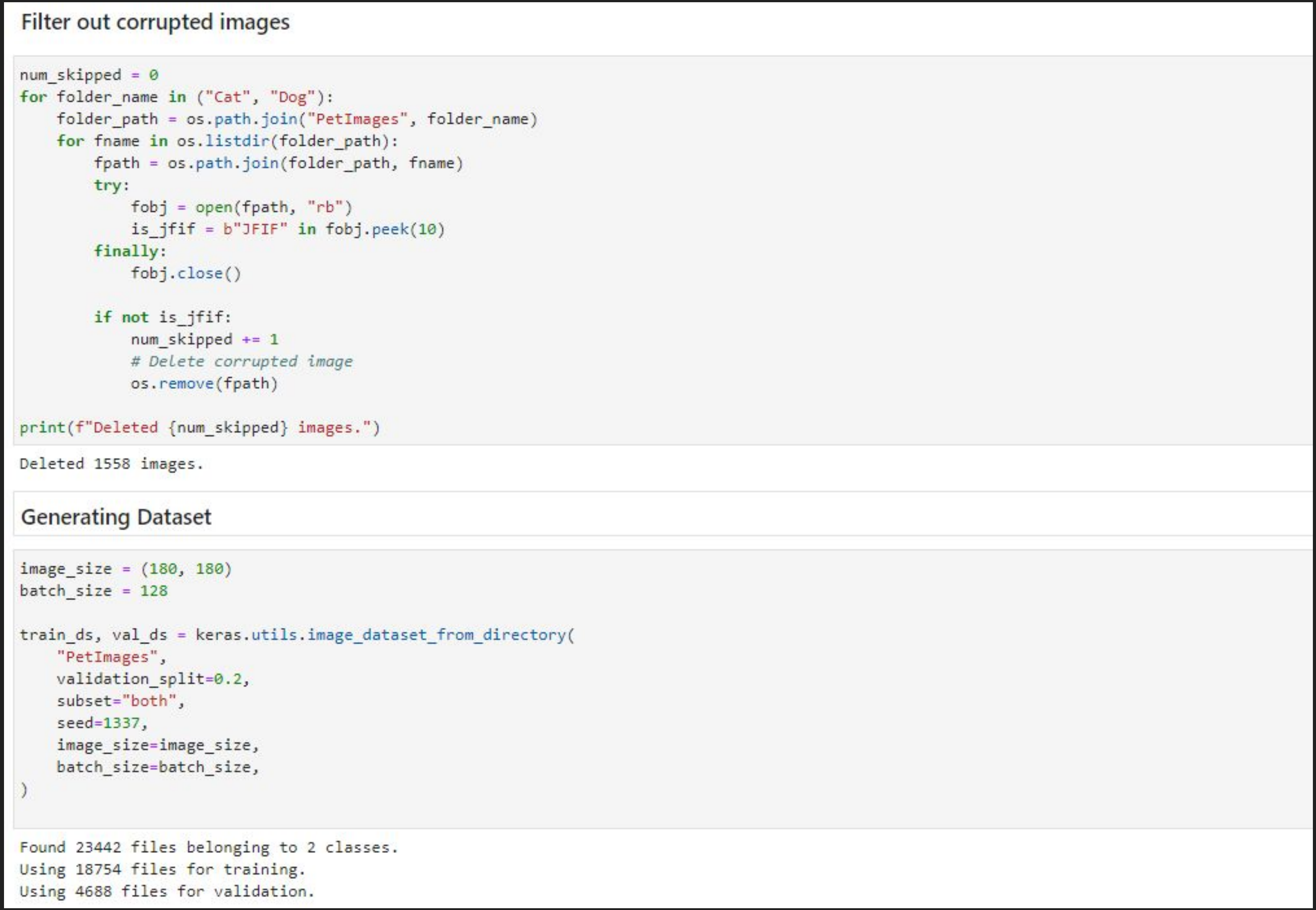
Import Libraries

```
import os
import numpy as np
import keras
from keras import layers
from tensorflow import data as tf_data
import matplotlib.pyplot as plt
```

Data Loading

```
!curl -O https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-4869-8368-6DEBA77B919F/kagglecatsanddogs_5340.zip
!unzip -q kagglecatsanddogs_5340.zip
!ls
!ls PetImages
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  786M  100  786M    0     0  82.0M      0  0:00:09  0:00:09 --:--:-- 92.6M
 ATP.ipynb                'ist652_lab2_job (2).ipynb'   PetImages
 CDLA-Permissive-2.0.pdf   IST652_Lab3.ipynb           'readme[1].txt'
'IST652 - Lab 1.ipynb'    'IST652 - Lab 4.ipynb'
'IST652 - Lab 2.ipynb'     kagglecatsanddogs_5340.zip
Cat  Dog
```



Filter out corrupted images

```python
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 = b"JFIF" in fobj.peek(10)
        finally:
            fobj.close()

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

print(f"Deleted {num_skipped} images.")
```

```
Deleted 1558 images.
```

Generating Dataset

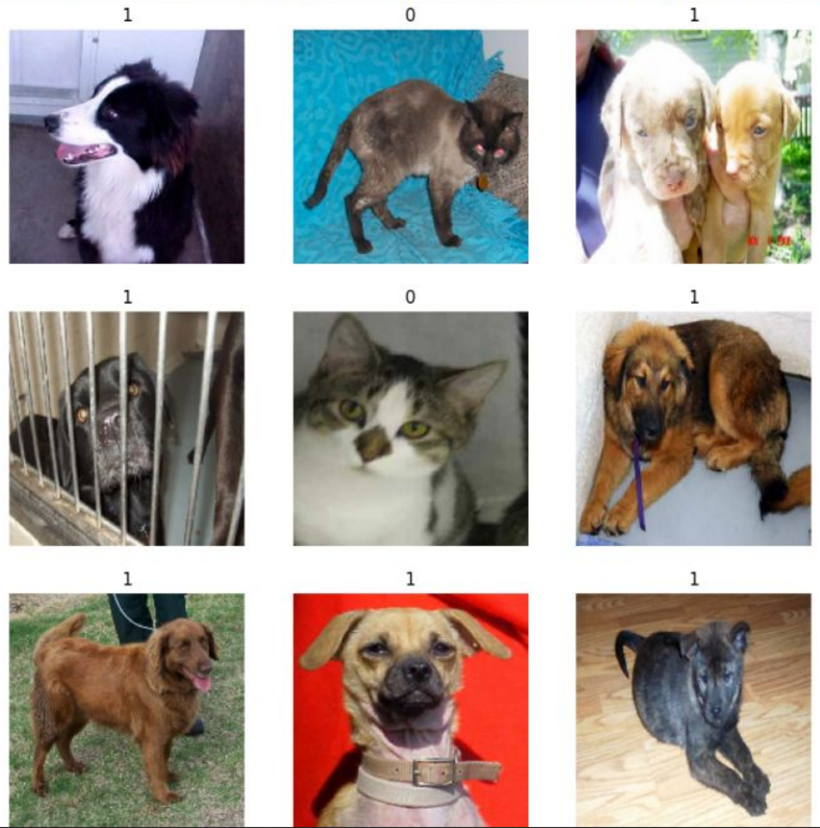
```python
image_size = (180, 180)
batch_size = 128

train_ds, val_ds = keras.utils.image_dataset_from_directory(
    "PetImages",
    validation_split=0.2,
    subset="both",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
)
```

```
Found 23442 files belonging to 2 classes.
Using 18754 files for training.
Using 4688 files for validation.
```

Visualize the data

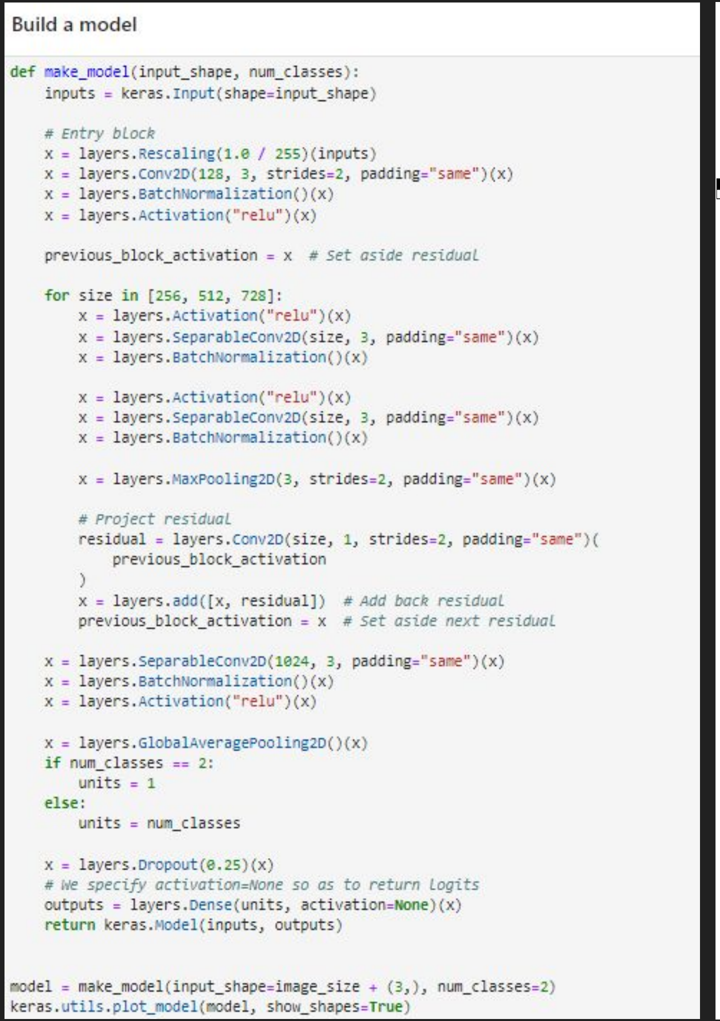
```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(np.array(images[i]).astype("uint8"))
        plt.title(int(labels[i]))
        plt.axis("off")
```

```
2024-03-29 11:20:17.615900: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```



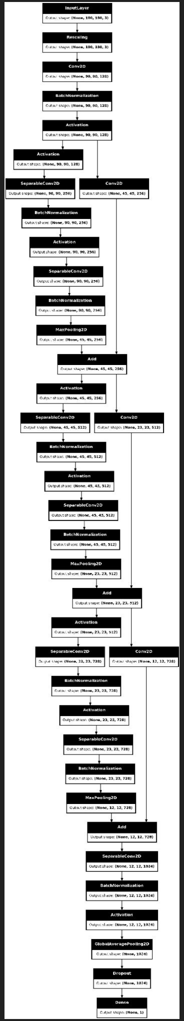
Scaled down version of Xception CNN network

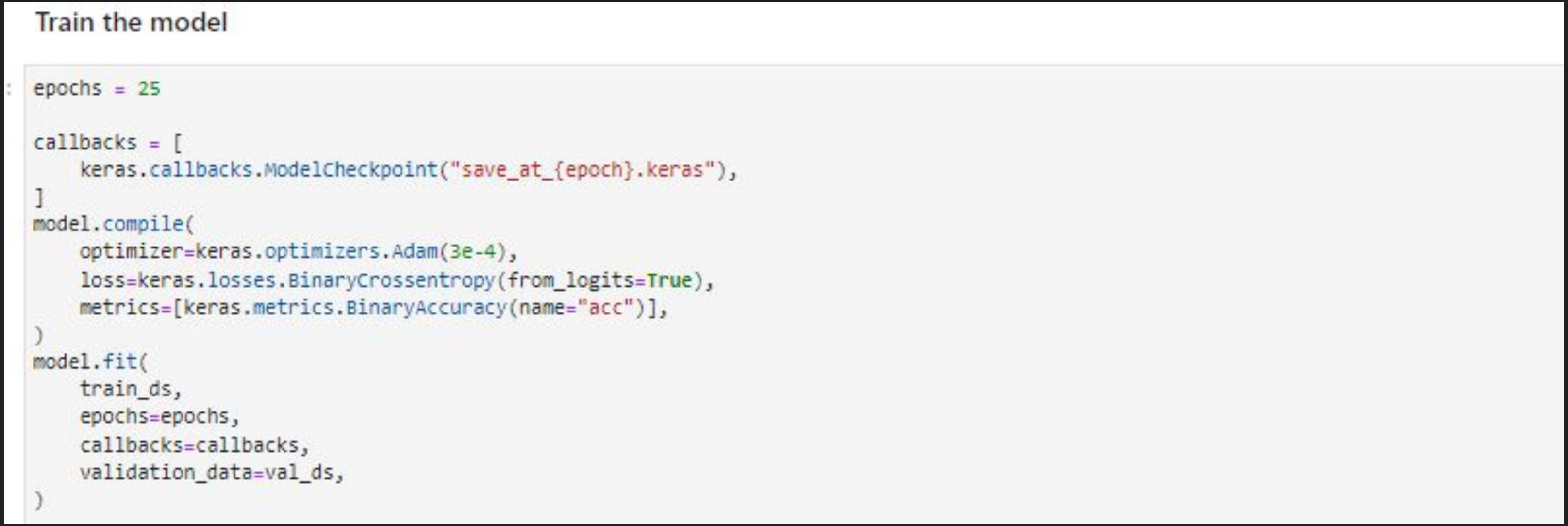
- Fully convolutional
- Parallel convolutional branches
- Depthwise separable convolution



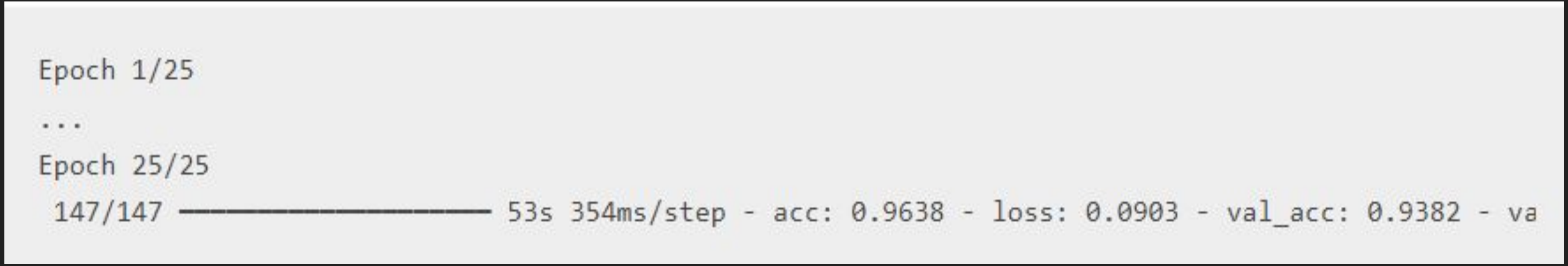
Build a model

```
def make_model(input_shape, num_classes):
    inputs = keras.Input(shape=input_shape)

    # Entry block
    x = layers.Rescaling(1.0 / 255)(inputs)
    x = layers.Conv2D(128, 3, strides=2, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    previous_block_activation = x  # Set aside residual

    for size in [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:
        units = 1
    else:
        units = num_classes

    x = layers.Dropout(0.25)(x)
    # We specify activation=None so as to return logits
    outputs = layers.Dense(units, activation=None)(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)
```

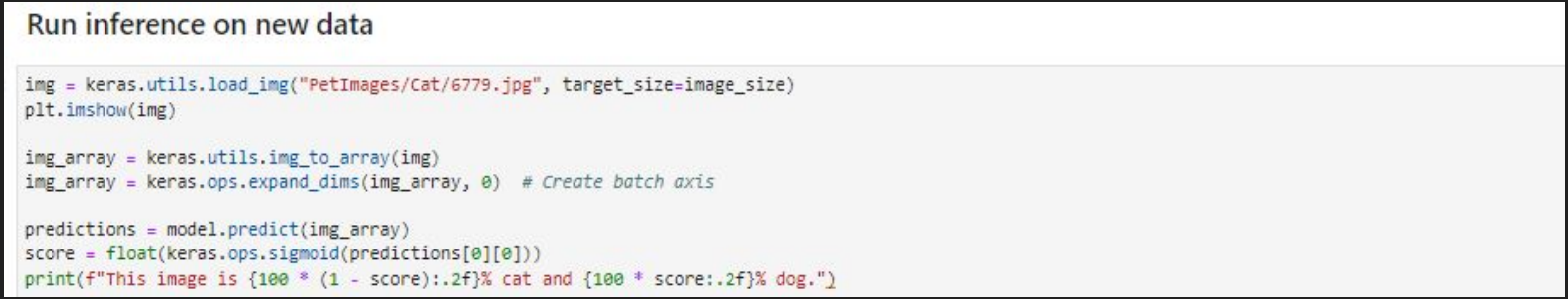



Train the model

```
epochs = 25

callbacks = [
    keras.callbacks.ModelCheckpoint("save_at_{epoch}.keras"),
]
model.compile(
    optimizer=keras.optimizers.Adam(3e-4),
    loss=keras.losses.BinaryCrossentropy(from_logits=True),
    metrics=[keras.metrics.BinaryAccuracy(name="acc")],
)
model.fit(
    train_ds,
    epochs=epochs,
    callbacks=callbacks,
    validation_data=val_ds,
)
```



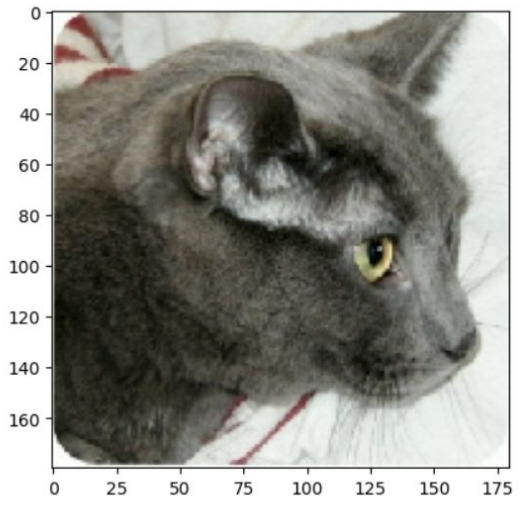
```
Epoch 1/25
...
Epoch 25/25
 147/147 ━━━━━━━━━━━━━━━━━━━━ 53s 354ms/step - acc: 0.9638 - loss: 0.0903 - val_acc: 0.9382 - va
```



Run inference on new data

```
img = keras.utils.load_img("PetImages/Cat/6779.jpg", target_size=image_size)
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = keras.ops.expand_dims(img_array, 0)  # Create batch axis

predictions = model.predict(img_array)
score = float(keras.ops.sigmoid(predictions[0][0]))
print(f"This image is {100 * (1 - score):.2f}% cat and {100 * score:.2f}% dog.")
```

```
 1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
This image is 94.30% cat and 5.70% dog.
```



Advantages of Keras

- **Simplicity**: Keras provides a simple and intuitive interface, making it easy to design, build, and train deep learning models.
- **Modularity**: Keras offers a modular approach to building neural networks, allowing users to easily create complex architectures by stacking layers and defining connections between them.
- **Flexibility**: Keras supports multiple backend engines, including TensorFlow, Theano, and Microsoft Cognitive Toolkit (CNTK), providing users with flexibility in choosing the backend that best suits their needs.
- **Extensibility**: Keras allows for easy customization and extension through subclassing and functional API, enabling users to define custom layers, loss functions, and metrics.
- **Community Support**: Keras benefits from a large and active community of users, researchers, and developers who contribute to its development, documentation, and ecosystem.

Disadvantages of Keras

- **Abstraction Overhead**: While Keras' high-level API simplifies model development, it also abstracts away some low-level details, which may limit fine-grained control and customization for advanced users.
- **Performance Overhead:** Keras' abstraction layer may introduce some performance overhead compared to using the backend frameworks directly. While Keras strives to optimize performance, users may experience slightly slower execution times for certain operations or models compared to native implementations in TensorFlow or other backends.
- **Backend Dependency**: Although Keras supports multiple backend engines, it's primarily developed and optimized for TensorFlow.
- **Limited Research Focus**: While Keras excels in rapid prototyping and production-level deployment, it may not be the first choice for research-oriented projects that require extensive experimentation and customization.
- **Documentation Fragmentation**: Keras' documentation is comprehensive and well-maintained, but it can be fragmented due to the integration with multiple backend engines.

Conclusion

Overall, Keras offers a powerful and user-friendly platform for deep learning development, with a balance of simplicity, flexibility, and community support. While it may not be the best fit for every use case, its ease of use and broad adoption make it a popular choice for beginners, practitioners, and production-level applications.