In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version us
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/shark-species/sharks/basking/00000030.jpg
/kaggle/input/shark-species/sharks/basking/00000042.jpg
/kaggle/input/shark-species/sharks/basking/00000027.jpg
/kaggle/input/shark-species/sharks/basking/00000095.jpg
/kaggle/input/shark-species/sharks/basking/00000087.jpg
/kaggle/input/shark-species/sharks/basking/00000089.jpg
/kaggle/input/shark-species/sharks/basking/00000016.jpg
/kaggle/input/shark-species/sharks/basking/00000129.jpg
/kaggle/input/shark-species/sharks/basking/00000094.jpg
/kaggle/input/shark-species/sharks/basking/00000028.jpg
/kaggle/input/shark-species/sharks/basking/00000141.jpg
/kaggle/input/shark-species/sharks/basking/00000022.jpg
/kaggle/input/shark-species/sharks/basking/00000050.jpg
/kaggle/input/shark-species/sharks/basking/00000002.jpg
/kaggle/input/shark-species/sharks/basking/00000004.jpg
/kaggle/input/shark-species/sharks/basking/00000079.jpg
/kaggle/input/shark-species/sharks/basking/00000008.jpg
/kaggle/input/shark-species/sharks/basking/00000000.jpg
/kaggle/input/shark-species/sharks/basking/00000098.jpg
```

In [24]:
```
import numpy as np
import pandas as pd
from pathlib import Path
import os.path
import matplotlib.pyplot as plt
from IPython.display import Image, display
import matplotlib.cm as cm
from sklearn.model_selection import train_test_split
import tensorflow as tf
```

In [25]:
```
from fastai.vision.all import *
from fastai.imports import *
from fastai.vision.data import *
from fastai import *
```

In [26]:
```
image_dir = Path("/kaggle/input/shark-species/sharks")
# Get filepaths and labels
filepaths = list(image_dir.glob(r'*/*'))
labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths))

filepaths = pd.Series(filepaths, name='Filepath').astype(str)
labels = pd.Series(labels, name='Label')

# Concatenate filepaths and labels
image_df= pd.concat([filepaths, labels], axis=1)
```

In [34]:
```
image_df = image_df[image_df['Label'].isin(['whitetip','hammerhead','basking','tiger','white'])]
```

In [ ]:
```
#lst = []
#for l in image_df['Label'].unique():
#    lst.append(image_df[image_df['Label'] == l] .sample(50, random_state = 0))
# Concatenate the DataFrames
#image_df = pd.concat(lst)
```

In [35]:
```
image_df.describe()
```

Out[35]:

	Filepath	Label
count	577	577
unique	577	5
top	/kaggle/input/shark-species/sharks/basking/00000030.jpg	white
freq	1	135

In [36]:
```
image_dir1 = Path("/kaggle/input/crocodile-gharial-classification-fastai")
# Get filepaths and labels
filepaths = list(image_dir1.glob(r'**/*.jpg'))
labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths))

filepaths = pd.Series(filepaths, name='Filepath').astype(str)
labels = pd.Series(labels, name='Label')

# Concatenate filepaths and labels
image_df1= pd.concat([filepaths, labels], axis=1)
```

In [37]:
```
image_df1 = image_df1[image_df1['Label']=='alligator']
```

In [38]:
```
image_df1.describe()
```

Out[38]:

	Filepath	Label
count	238	238
unique	238	1
top	/kaggle/input/crocodile-gharial-classification-fastai/alligator/AmericanAlligatorHdWallpaper2013.jpg	alligator
freq	1	238

In [39]:
```
im_df = image_df.append(image_df1, ignore_index=True)
im_df = im_df.sample(frac=1).reset_index(drop = True)
im_df.head(3)
```

Out[39]:

	Filepath	Label
0	/kaggle/input/shark-species/sharks/tiger/00000036.jpg	tiger
1	/kaggle/input/shark-species/sharks/hammerhead/00000092.jpg	hammerhead
2	/kaggle/input/shark-species/sharks/hammerhead/00000064.jpg	hammerhead

In [40]:
```
im_df['Label'].unique()
```

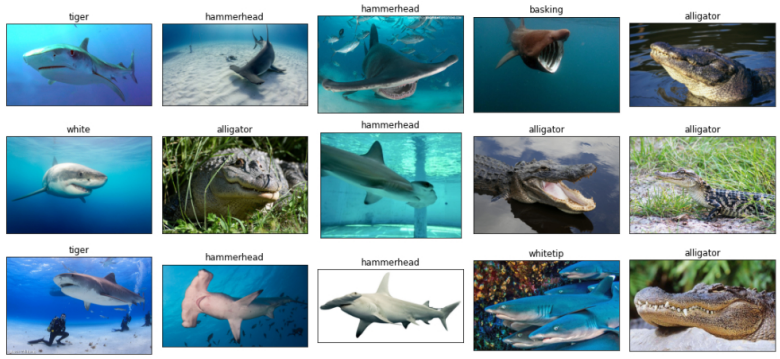
Out[40]:
```
array(['tiger', 'hammerhead', 'basking', 'alligator', 'white', 'whitetip'],
      dtype=object)
```

In [41]:
```
len(im_df['Label'].unique())
```

Out[41]:
```
6
```

In [42]:
```
fig, axes = plt.subplots(nrows=3, ncols=5, figsize=(15, 7),
                         subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    ax.imshow(plt.imread(im_df.Filepath[i]))
    ax.set_title(im_df.Label[i])
```

```
plt.tight_layout()
plt.show()
```



In [43]:
```
train_df, test_df = train_test_split(im_df, train_size=0.9, shuffle=True, random_state=1)
test_df.describe()
```

Out[43]:

	Filepath	Label
count	82	82
unique	82	6
top	/kaggle/input/shark-species/sharks/tiger/00000076.jpg	alligator
freq	1	21

In [44]:
```
train_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    preprocessing_function=tf.keras.applications.mobilenet_v2.preprocess_input,
    validation_split=0.2
)

test_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    preprocessing_function=tf.keras.applications.mobilenet_v2.preprocess_input
)
```

In [46]:
```
train_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='categorical',
    batch_size=32,
    shuffle=True,
    seed=42,
    subset='training'
)

val_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='categorical',
    batch_size=32,
    shuffle=True,
    seed=42,
    subset='validation'
)
test_images = test_generator.flow_from_dataframe(
    dataframe=test_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='categorical',
    batch_size=32,
    shuffle=False
)
```

```
Found 586 validated image filenames belonging to 6 classes.
Found 146 validated image filenames belonging to 6 classes.
Found 82 validated image filenames belonging to 6 classes.
```

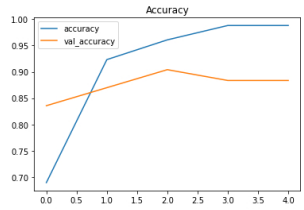
In [48]:
```
pretrained_model = tf.keras.applications.mobilenet_v2.MobileNetV2(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet',
    pooling='avg'
)

pretrained_model.trainable = False
```

In [49]:
```
inputs = pretrained_model.input

x = tf.keras.layers.Dense(128, activation='relu')(pretrained_model.output)
x = tf.keras.layers.Dense(128, activation='relu')(x)

outputs = tf.keras.layers.Dense(6, activation='softmax')(x)

model = tf.keras.Model(inputs=inputs, outputs=outputs)

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

history = model.fit(
    train_images,
    validation_data=val_images,
    epochs=50,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=1,
            restore_best_weights=True
        )
    ]
)
```

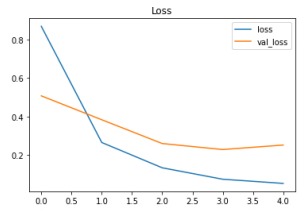
```
Epoch 1/50
19/19 [==============================] - 39s 2s/step - loss: 0.8705 - accuracy: 0.6894 - val_loss: 0.5076 - val_accuracy: 0.835
6
Epoch 2/50
19/19 [==============================] - 25s 1s/step - loss: 0.2642 - accuracy: 0.9232 - val_loss: 0.3828 - val_accuracy: 0.869
9
Epoch 3/50
19/19 [==============================] - 25s 1s/step - loss: 0.1322 - accuracy: 0.9608 - val_loss: 0.2588 - val_accuracy: 0.904
1
Epoch 4/50
19/19 [==============================] - 25s 1s/step - loss: 0.0726 - accuracy: 0.9881 - val_loss: 0.2275 - val_accuracy: 0.883
6
Epoch 5/50
19/19 [==============================] - 24s 1s/step - loss: 0.0508 - accuracy: 0.9881 - val_loss: 0.2510 - val_accuracy: 0.883
6
```

In [50]:
```
pd.DataFrame(history.history)[['accuracy','val_accuracy']].plot()
plt.title("Accuracy")
```

```
plt.show()
```



In [51]:
```
pd.DataFrame(history.history)[['loss','val_loss']].plot()
plt.title("Loss")
plt.show()
```



In [52]:
```
results = model.evaluate(test_images, verbose=0)

print("    Test Loss:",(results[0]))
print("Test Accuracy:",(results[1] * 100),"%")
```

```
    Test Loss: 0.11360262334346771
Test Accuracy: 96.34146094322205 %
```

In [ ]: