```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import zipfile
from google.colab import drive
import random
```

Double-click (or enter) to edit

Loading dataset

```
drive.mount('/content/drive')

zip_path = "/content/drive/MyDrive/animal_data.zip"
extract_path = "/content/animal_data"

with zipfile.ZipFile(zip_path, 'r') as z:
    z.extractall(extract_path)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
!find /content/animal_data -name "._*" -delete
!find /content/animal_data -type f -not -iname "*.jpg" -not -iname "*.jpeg" -not -iname "*.png" -delete
!rm -rf /content/animal_data/animal_data/animal_data
!rm -rf /content/animal_data/animal_data/__MACOSX
```

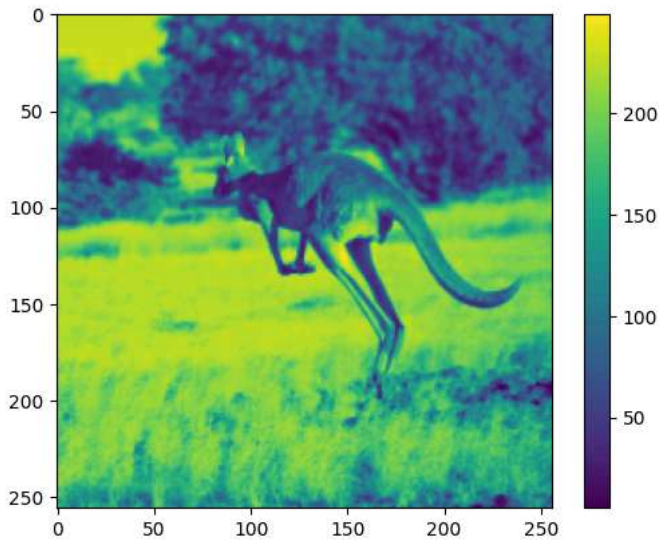
```
train_data = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/animal_data/animal_data",
    validation_split=0.2,
    subset="training",
    seed=42,
    image_size=(256, 256),
    batch_size=32,
    color_mode="rgb"
)

test_data = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/animal_data/animal_data",
    validation_split=0.2,
    subset="validation",
    seed=42,
    image_size=(256, 256),
    batch_size=32,
    color_mode="rgb"
)

class_names = train_data.class_names
# class_names = class_names[0:14]
print(class_names)
```

```
Found 1944 files belonging to 15 classes.
Using 1556 files for training.
Found 1944 files belonging to 15 classes.
Using 388 files for validation.
['Bear', 'Bird', 'Cat', 'Cow', 'Deer', 'Dog', 'Dolphin', 'Elephant', 'Giraffe', 'Horse', 'Kangaroo', 'Lion', 'Panda', 'Tiger', '
```

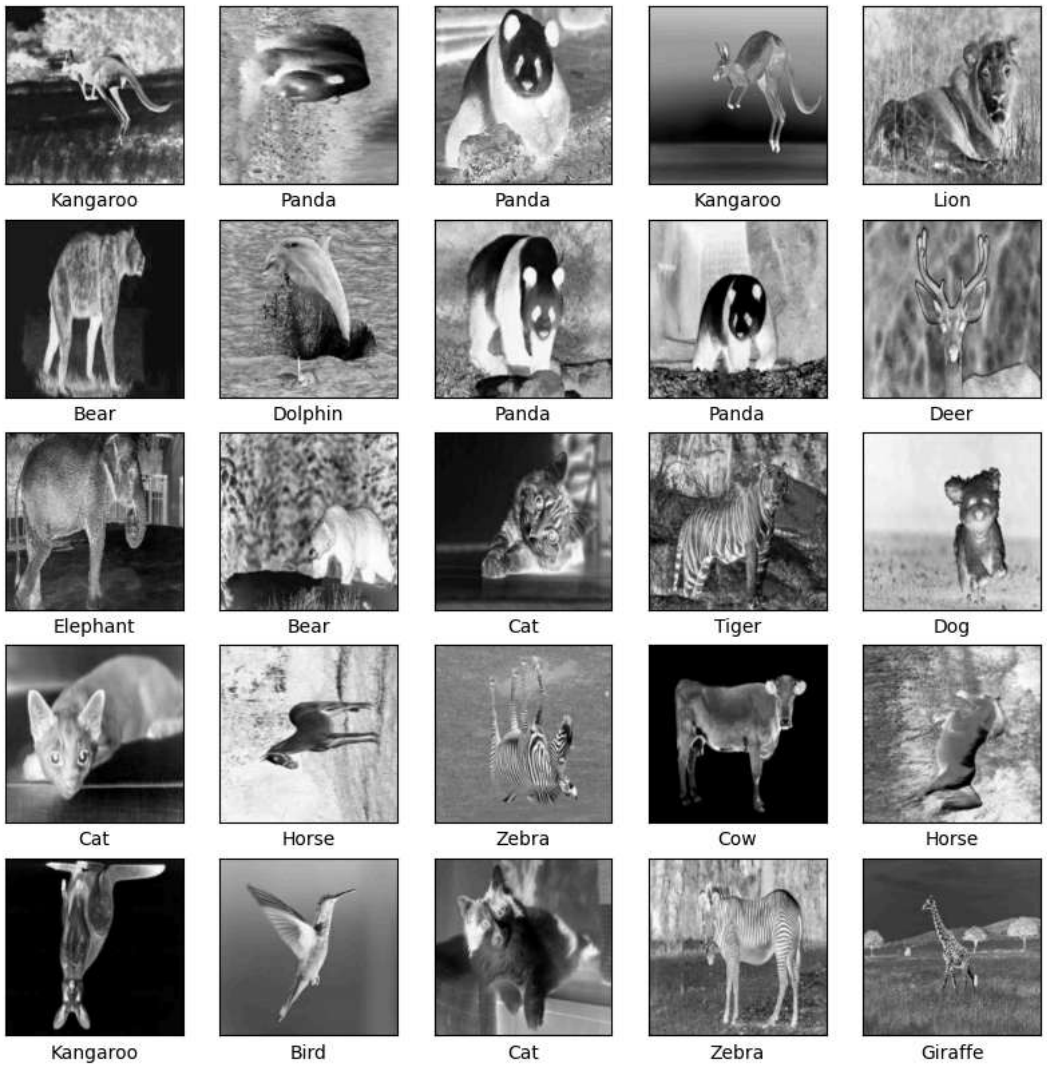
```
def unpack(data):
    images = []
    labels = []

    data = data.unbatch()

    for img, label in ds:
        images.append(img.numpy())
        labels.append(label.numpy())
```

```
    return np.array(images), np.array(labels)

train_images, train_labels = unpack(train_data)
test_images, test_labels = unpack(test_data)

train_images = tf.image.rgb_to_grayscale(train_images).numpy()
test_images  = tf.image.rgb_to_grayscale(test_images).numpy()
```

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```




```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```




```
model = tf.keras.Sequential([
    tf.keras.layers.Rescaling(1./255),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(256, 256, 1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(input_shape=(256, 256, 1)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(15)
])
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

```
model.fit(train_data, validation_data=test_data, epochs=10)
```

```
Epoch 1/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 153s 3s/step - accuracy: 0.1146 - loss: 31.9311 - val_accuracy: 0.2448 - val_loss: 2.5177
Epoch 2/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 143s 3s/step - accuracy: 0.3538 - loss: 2.2268 - val_accuracy: 0.4124 - val_loss: 1.9994
Epoch 3/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 198s 3s/step - accuracy: 0.5559 - loss: 1.6538 - val_accuracy: 0.5515 - val_loss: 1.5905
Epoch 4/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 142s 3s/step - accuracy: 0.7524 - loss: 1.0278 - val_accuracy: 0.6443 - val_loss: 1.3838
Epoch 5/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 152s 3s/step - accuracy: 0.8931 - loss: 0.5261 - val_accuracy: 0.6804 - val_loss: 1.2867
Epoch 6/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 192s 3s/step - accuracy: 0.9421 - loss: 0.3268 - val_accuracy: 0.6624 - val_loss: 1.3166
Epoch 7/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 139s 3s/step - accuracy: 0.9662 - loss: 0.2044 - val_accuracy: 0.6778 - val_loss: 1.2577
Epoch 8/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 145s 3s/step - accuracy: 0.9927 - loss: 0.0972 - val_accuracy: 0.6804 - val_loss: 1.4084
Epoch 9/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 153s 3s/step - accuracy: 0.9874 - loss: 0.1012 - val_accuracy: 0.6753 - val_loss: 1.2735
Epoch 10/10
```

```
49/49 ━━━━━━━━━━━━━━━━━━━━ 190s 3s/step - accuracy: 0.9980 - loss: 0.0387 - val_accuracy: 0.6649 - val_loss: 1.3429
<keras.src.callbacks.history.History at 0x7b7bf3d4f4a0>
```

```
test_loss, test_acc = model.evaluate(test_data, verbose=2)

print('\nTest accuracy:', test_acc)
```

```
13/13 - 8s - 595ms/step - accuracy: 0.6649 - loss: 1.3429

Test accuracy: 0.6649484634399414
```

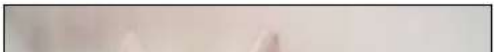
```
def conv(data):
    images, labels = [], []
    for img, lbl in data.unbatch():
        images.append(img.numpy())
        labels.append(lbl.numpy())
    return np.array(images), np.array(labels)

probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
test_images_np, test_labels_np = conv(test_data)
predictions = probability_model.predict(test_images_np)
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 9s 624ms/step
```

```
predictions[0]
```

```
def plot_image(i, predictions_array, true_label, img):
  true_label, img = true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)
```

```
for test_images_batch, test_labels_batch in test_data.take(1):
    predictions_batch = probability_model.predict(test_images_batch, verbose=0)

    i = 0
    plt.figure(figsize=(8, 4))
    plt.subplot(1, 2, 1)
    plot_image(i, predictions_batch[i], test_labels_batch.numpy(), test_images_batch.numpy()/255)

    # plt.subplot(1, 2, 2)
    # plot_value_array(i, predictions_batch[i], test_labels_batch.numpy())

    plt.tight_layout()
    plt.show()

    break
```



```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import zipfile
from google.colab import drive
import random
```

Double-click (or enter) to edit

Loading dataset

```
drive.mount('/content/drive')

zip_path = "/content/drive/MyDrive/animal_data.zip"
extract_path = "/content/animal_data"

with zipfile.ZipFile(zip_path, 'r') as z:
    z.extractall(extract_path)
```

```
Mounted at /content/drive
```

```
!find /content/animal_data -name "._*" -delete
!find /content/animal_data -type f -not -iname "*.jpg" -not -iname "*.jpeg" -not -iname "*.png" -delete
!rm -rf /content/animal_data/animal_data/animal_data
!rm -rf /content/animal_data/animal_data/__MACOSX
```

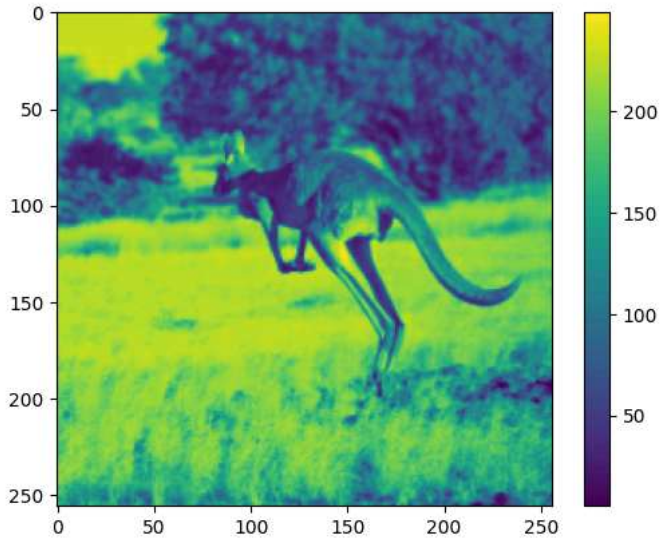
```
train_data = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/animal_data/animal_data",
    validation_split=0.2,
    subset="training",
    seed=42,
    image_size=(256, 256),
    batch_size=32,
    color_mode="rgb"
)

test_data = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/animal_data/animal_data",
    validation_split=0.2,
    subset="validation",
    seed=42,
    image_size=(256, 256),
    batch_size=32,
    color_mode="rgb"
)

class_names = train_data.class_names
# class_names = class_names[0:14]
print(class_names)
```

```
Found 1944 files belonging to 15 classes.
Using 1556 files for training.
Found 1944 files belonging to 15 classes.
Using 388 files for validation.
['Bear', 'Bird', 'Cat', 'Cow', 'Deer', 'Dog', 'Dolphin', 'Elephant', 'Giraffe', 'Horse', 'Kangaroo', 'Lion', 'Panda', 'Tiger', '
```

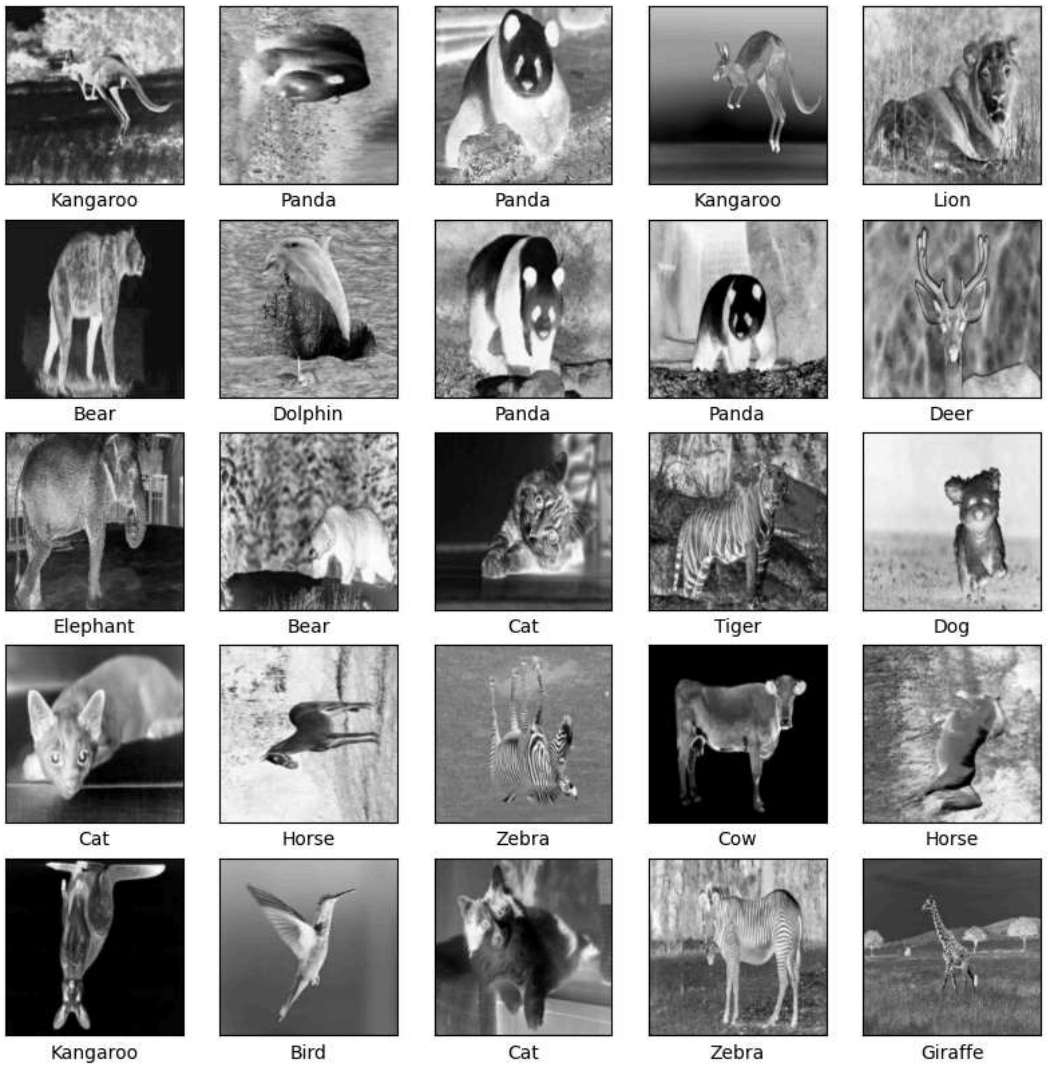
```
def unpack(data):
    images = []
    labels = []

    data = data.unbatch()

    for img, label in ds:
        images.append(img.numpy())
        labels.append(label.numpy())
```

```
    return np.array(images), np.array(labels)

train_images, train_labels = unpack(train_data)
test_images, test_labels = unpack(test_data)

train_images = tf.image.rgb_to_grayscale(train_images).numpy()
test_images  = tf.image.rgb_to_grayscale(test_images).numpy()
```

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```




```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```




```
model = tf.keras.Sequential([
    tf.keras.layers.Rescaling(1./255),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(15)
])
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`inp
  super().__init__(**kwargs)
```

```
model.fit(train_data, validation_data=test_data, epochs=10)
```

```
Epoch 1/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 127s 2s/step - accuracy: 0.1057 - loss: 20.8707 - val_accuracy: 0.2036 - val_loss: 2.4091
Epoch 2/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 121s 2s/step - accuracy: 0.3042 - loss: 2.2422 - val_accuracy: 0.3995 - val_loss: 1.9038
Epoch 3/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 140s 2s/step - accuracy: 0.5861 - loss: 1.4692 - val_accuracy: 0.6057 - val_loss: 1.4258
Epoch 4/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 123s 3s/step - accuracy: 0.8265 - loss: 0.7887 - val_accuracy: 0.6701 - val_loss: 1.2541
Epoch 5/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 134s 3s/step - accuracy: 0.9521 - loss: 0.3767 - val_accuracy: 0.6985 - val_loss: 1.2006
Epoch 6/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 124s 3s/step - accuracy: 0.9799 - loss: 0.2107 - val_accuracy: 0.6624 - val_loss: 1.3413
Epoch 7/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 142s 3s/step - accuracy: 0.9902 - loss: 0.1234 - val_accuracy: 0.6778 - val_loss: 1.2735
Epoch 8/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 139s 2s/step - accuracy: 0.9983 - loss: 0.0842 - val_accuracy: 0.7113 - val_loss: 1.2956
```

```
Epoch 9/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 124s 3s/step - accuracy: 0.9971 - loss: 0.0524 - val_accuracy: 0.6727 - val_loss: 1.3799
Epoch 10/10
49/49 ━━━━━━━━━━━━━━━━━━━━ 153s 3s/step - accuracy: 0.9997 - loss: 0.0312 - val_accuracy: 0.6907 - val_loss: 1.4604
<keras.src.callbacks.history.History at 0x7a554053d3d0>
```

```
test_loss, test_acc = model.evaluate(test_data, verbose=2)

print('\nTest accuracy:', test_acc)
```

```
13/13 - 18s - 1s/step - accuracy: 0.6907 - loss: 1.4604

Test accuracy: 0.6907216310501099
```

```
def conv(data):
    images, labels = [], []
    for img, lbl in data.unbatch():
        images.append(img.numpy())
        labels.append(lbl.numpy())
    return np.array(images), np.array(labels)

probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
test_images_np, test_labels_np = conv(test_data)
predictions = probability_model.predict(test_images_np)
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 7s 495ms/step
```

```
predictions[0]
```

```
array([1.5143600e-06, 9.3322067e-04, 7.9320256e-07, 7.3728975e-06,
       1.9821343e-04, 2.8497134e-05, 7.4051436e-06, 8.4713599e-07,
       3.1591876e-04, 1.5952288e-05, 9.9843407e-01, 1.9132598e-05,
       2.8812432e-05, 8.3556324e-06, 2.4968784e-08], dtype=float32)
```

```
def plot_image(i, predictions_array, true_label, img):
  true_label, img = true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  true_label = true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

```
probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])

for test_images_batch, test_labels_batch in test_data.take(1):
    predictions_batch = probability_model.predict(test_images_batch, verbose=0)

    i = 0
    plt.figure(figsize=(8, 4))
    plt.subplot(1, 2, 1)
    plot_image(i, predictions_batch[i], test_labels_batch.numpy(), test_images_batch.numpy()/255)
```

```
        # plt.subplot(1, 2, 2)
        # plot_value_array(i, predictions_batch[i], test_labels_batch.numpy())

        plt.tight_layout()
        plt.show()

        break
```