In [ ]:

In [3]:
```
##IMPORTS##
import tensorflow as tf
import numpy as np
import pandas as pd
import pathlib
import matplotlib.pyplot as plt
```

In [4]:
```
##Setting directory paths to the training data & test data image sets
data_dir=pathlib.Path('C:\\Users\\Pickle Mustard\\Documents\\Machine Learning Proje
test_dir=pathlib.Path('C:\\Users\\Pickle Mustard\\Documents\\Machine Learning Proje
```

In [5]:
```
batch_size = 128
num_classes = 10
epochs = 20
img_height=180
img_width=180
```

In [6]:
```
##Creating the datasets with the images
train_ds = tf.keras.utils.image_dataset_from_directory(
data_dir,
validation_split=0.2,
subset="validation",
seed=6461,
image_size=(img_height,img_width),)
```

```
Found 7328 files belonging to 15 classes.
Using 1465 files for validation.
```

In [7]:
```
##Test dataset
test_ds = tf.keras.utils.image_dataset_from_directory(
test_dir,
validation_split=0.2,
subset="validation",
seed=6461,
image_size=(img_height,img_width),)
```

```
Found 1841 files belonging to 15 classes.
Using 368 files for validation.
```

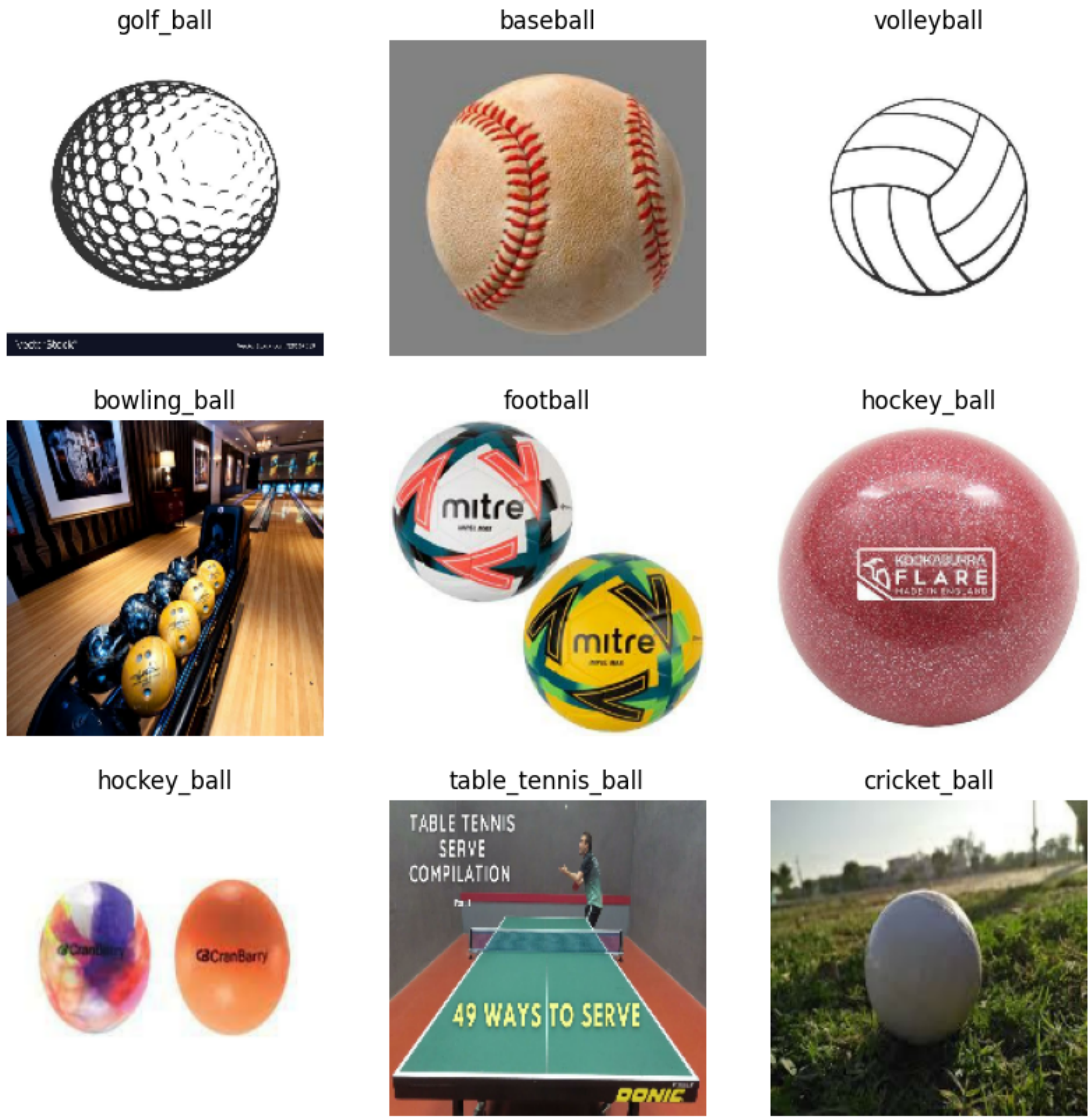
In [8]:
```
##Little bit of validation
for image_batch, labels_batch in train_ds:
    print(image_batch.shape)
    print(labels_batch.shape)
```

```
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
```

```
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(32, 180, 180, 3)
(32,)
(25, 180, 180, 3)
(25,)
```

In [9]:

```
##Little more validation
class_names = train_ds.class_names
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(class_names[labels[i]])
        plt.axis("off")
```



The images in the dataset range in how they are presented. It's not all just images of the ball from perfect lighting and taken at slightly different angles. Even here, there are images with text, different numbers of the identifying image, different lighting conditions, different background conditions, and the worst is the table tennis image which only has a few pixels relating to the identifier. So let's see how well this will perform. I'm not expecting a lot out of this.

In [16]:
```
##Referencing # of identifiers
normalization_layer = tf.keras.layers.Rescaling(1./255)
print(class_names)
```

```
['american_football', 'baseball', 'basketball', 'billiard_ball', 'bowling_ball', 'cr
icket_ball', 'football', 'golf_ball', 'hockey_ball', 'hockey_puck', 'rugby_ball', 's
huttlecock', 'table_tennis_ball', 'tennis_ball', 'volleyball']
```

In [17]:
```
normalized_train_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
normalized_test_ds = test_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_train_ds))
first_image = image_batch[0]
print(np.min(first_image), np.max(first_image))
```

```
0.027902687 1.0
```

In [18]:
```
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

In [19]:
```
num_classes = 15
```

In [16]:
```
##Starting with a dense model. This took a long time to train...
dense_model = tf.keras.models.Sequential([
    tf.keras.layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
    #tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(num_classes),
])
#dense_model.summary()
```

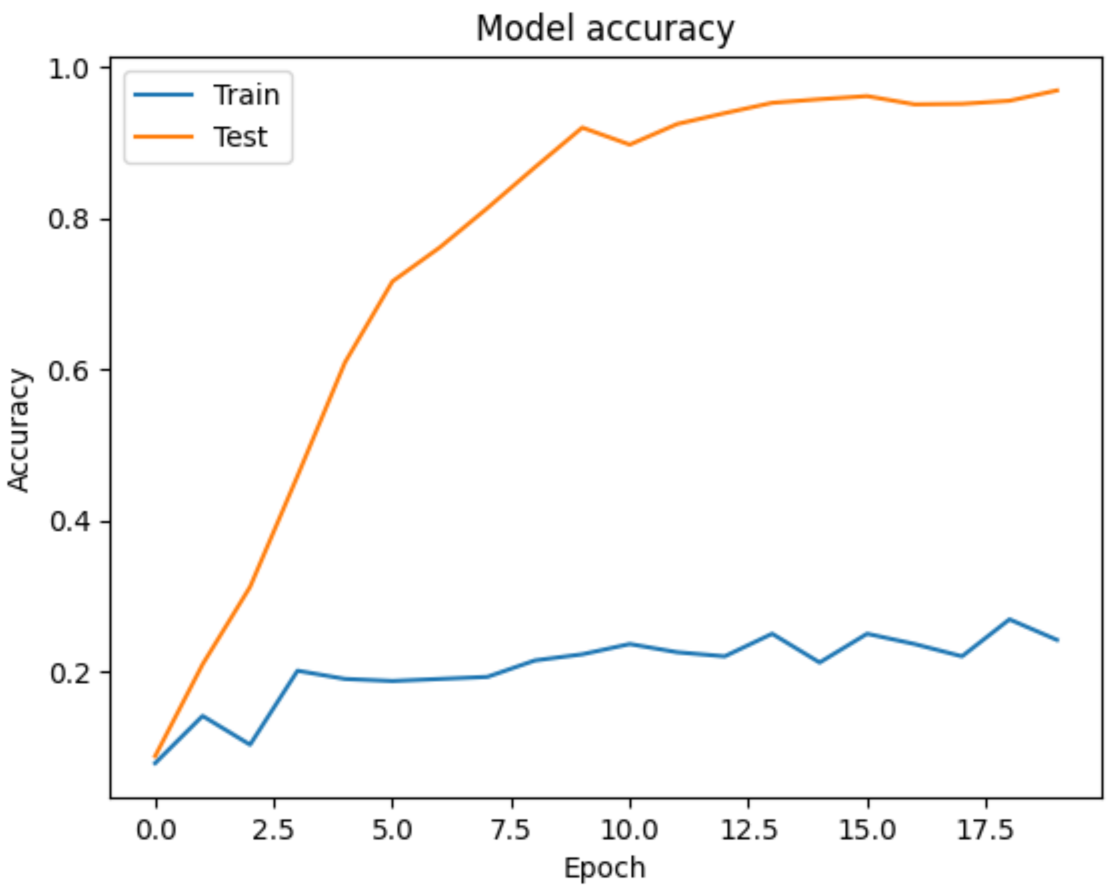
In [17]:
```
dense_model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=
                   optimizer='adam',
                   metrics=['accuracy'])
```

In [18]:
```
history = dense_model.fit(train_ds, batch_size=batch_size, epochs=20, verbose=1, va
```

```
Epoch 1/20
46/46 [==============================] - 147s 3s/step - loss: 66.9512 - accuracy: 0.
0881 - val_loss: 2.6981 - val_accuracy: 0.0788
Epoch 2/20
46/46 [==============================] - 144s 3s/step - loss: 2.5129 - accuracy: 0.2
096 - val_loss: 2.7578 - val_accuracy: 0.1413
Epoch 3/20
46/46 [==============================] - 145s 3s/step - loss: 2.2243 - accuracy: 0.3
119 - val_loss: 3.0507 - val_accuracy: 0.1033
Epoch 4/20
46/46 [==============================] - 150s 3s/step - loss: 1.8567 - accuracy: 0.4
587 - val_loss: 3.3114 - val_accuracy: 0.2011
Epoch 5/20
46/46 [==============================] - 148s 3s/step - loss: 1.4254 - accuracy: 0.6
089 - val_loss: 3.8457 - val_accuracy: 0.1902
Epoch 6/20
46/46 [==============================] - 147s 3s/step - loss: 1.0666 - accuracy: 0.7
160 - val_loss: 4.2640 - val_accuracy: 0.1875
Epoch 7/20
46/46 [==============================] - 153s 3s/step - loss: 1.0214 - accuracy: 0.7
611 - val_loss: 4.1538 - val_accuracy: 0.1902
Epoch 8/20
46/46 [==============================] - 144s 3s/step - loss: 0.8453 - accuracy: 0.8
130 - val_loss: 4.7311 - val_accuracy: 0.1929
Epoch 9/20
46/46 [==============================] - 144s 3s/step - loss: 0.5940 - accuracy: 0.8
669 - val_loss: 4.8416 - val_accuracy: 0.2147
Epoch 10/20
46/46 [==============================] - 144s 3s/step - loss: 0.3931 - accuracy: 0.9
195 - val_loss: 5.4984 - val_accuracy: 0.2228
Epoch 11/20
46/46 [==============================] - 144s 3s/step - loss: 0.4290 - accuracy: 0.8
969 - val_loss: 5.4957 - val_accuracy: 0.2364
Epoch 12/20
46/46 [==============================] - 144s 3s/step - loss: 0.3270 - accuracy: 0.9
242 - val_loss: 6.5493 - val_accuracy: 0.2255
Epoch 13/20
46/46 [==============================] - 144s 3s/step - loss: 0.3437 - accuracy: 0.9
386 - val_loss: 5.9260 - val_accuracy: 0.2201
Epoch 14/20
46/46 [==============================] - 144s 3s/step - loss: 0.2230 - accuracy: 0.9
522 - val_loss: 6.4134 - val_accuracy: 0.2500
Epoch 15/20
46/46 [==============================] - 144s 3s/step - loss: 0.2121 - accuracy: 0.9
570 - val_loss: 6.7575 - val_accuracy: 0.2120
Epoch 16/20
46/46 [==============================] - 144s 3s/step - loss: 0.1816 - accuracy: 0.9
611 - val_loss: 6.9331 - val_accuracy: 0.2500
Epoch 17/20
46/46 [==============================] - 144s 3s/step - loss: 0.2849 - accuracy: 0.9
502 - val_loss: 7.8092 - val_accuracy: 0.2364
Epoch 18/20
46/46 [==============================] - 144s 3s/step - loss: 0.2721 - accuracy: 0.9
509 - val_loss: 7.1758 - val_accuracy: 0.2201
Epoch 19/20
46/46 [==============================] - 144s 3s/step - loss: 0.2080 - accuracy: 0.9
```

```
549 - val_loss: 6.8538 - val_accuracy: 0.2690
Epoch 20/20
46/46 [==============================] - 144s 3s/step - loss: 0.1671 - accuracy: 0.9
686 - val_loss: 7.7074 - val_accuracy: 0.2418
```

In [19]:
```
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



Ok, not great. Pretty bad if we're being honest. It was able to predict pretty well through the training set but once it started to attempt to validate, it didn't get higher than 27% correct. That's abyssmal. I tried different drop-out rates and densities but could not get it above around 30% correct during validation. Might be an indicator of the poor-quality of the image set but let's continue and find out how well a convolutional model does.

In [58]:
```
##Convolution Neural Network with increasing filters
model = tf.keras.models.Sequential([
    tf.keras.layers.Rescaling(1./255),
    tf.keras.layers.Conv2D(16, 3, padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(num_classes, name="outputs"),
])
```

In [59]:
```
model.compile(
optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy'])
```

In [60]:
```
history = model.fit(
    train_ds,
    validation_data=test_ds,
    epochs=20
)
```

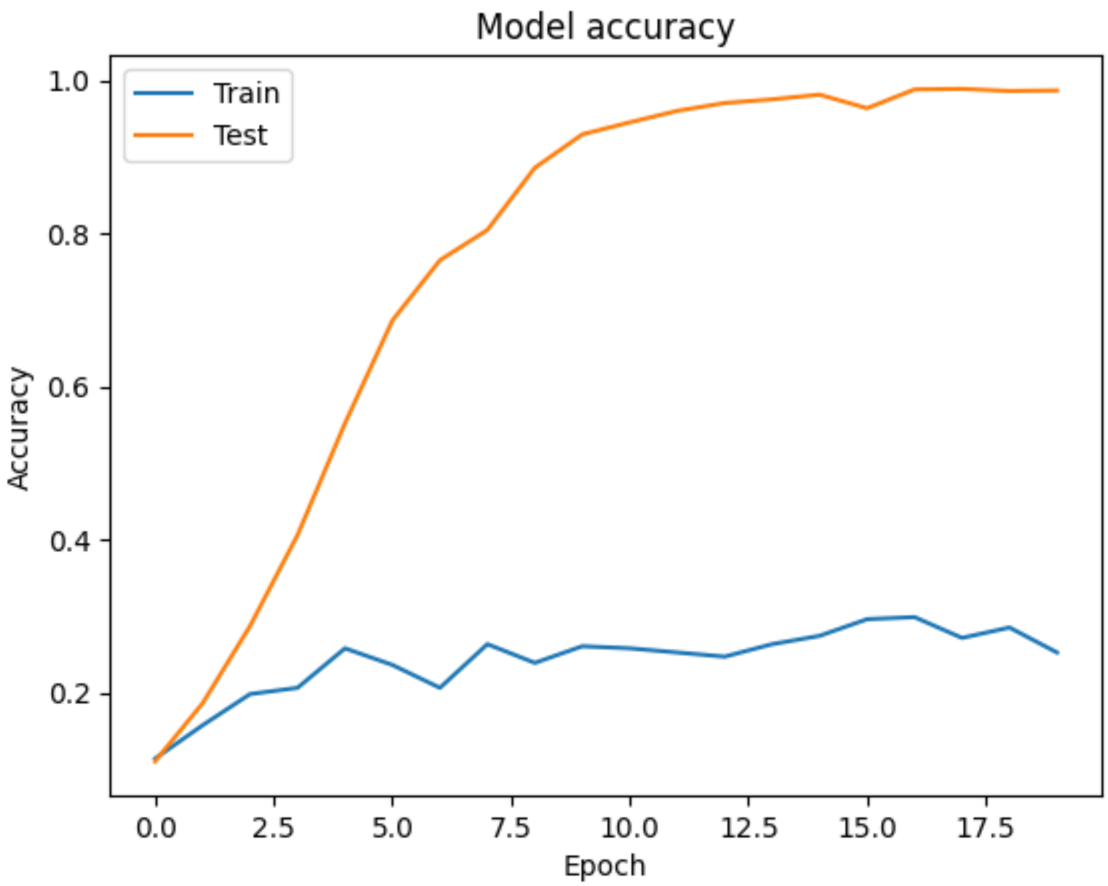
```
Epoch 1/20
46/46 [==============================] - 16s 334ms/step - loss: 2.6751 - accuracy:
0.1106 - val_loss: 2.6129 - val_accuracy: 0.1141
Epoch 2/20
46/46 [==============================] - 15s 317ms/step - loss: 2.5003 - accuracy:
0.1863 - val_loss: 2.5447 - val_accuracy: 0.1576
Epoch 3/20
46/46 [==============================] - 14s 315ms/step - loss: 2.2701 - accuracy:
0.2874 - val_loss: 2.4961 - val_accuracy: 0.1984
Epoch 4/20
46/46 [==============================] - 15s 319ms/step - loss: 1.9280 - accuracy:
0.4061 - val_loss: 2.5582 - val_accuracy: 0.2065
Epoch 5/20
46/46 [==============================] - 14s 315ms/step - loss: 1.4835 - accuracy:
0.5522 - val_loss: 2.6851 - val_accuracy: 0.2582
Epoch 6/20
46/46 [==============================] - 15s 317ms/step - loss: 1.0730 - accuracy:
0.6867 - val_loss: 2.9474 - val_accuracy: 0.2364
Epoch 7/20
46/46 [==============================] - 15s 318ms/step - loss: 0.7675 - accuracy:
0.7652 - val_loss: 3.8902 - val_accuracy: 0.2065
Epoch 8/20
46/46 [==============================] - 15s 316ms/step - loss: 0.6344 - accuracy:
0.8048 - val_loss: 3.7601 - val_accuracy: 0.2636
Epoch 9/20
46/46 [==============================] - 15s 316ms/step - loss: 0.4055 - accuracy:
0.8860 - val_loss: 5.2231 - val_accuracy: 0.2391
Epoch 10/20
46/46 [==============================] - 16s 351ms/step - loss: 0.2543 - accuracy:
0.9297 - val_loss: 5.2680 - val_accuracy: 0.2609
Epoch 11/20
46/46 [==============================] - 16s 358ms/step - loss: 0.1913 - accuracy:
0.9454 - val_loss: 6.0018 - val_accuracy: 0.2582
Epoch 12/20
46/46 [==============================] - 19s 413ms/step - loss: 0.1532 - accuracy:
0.9604 - val_loss: 6.7585 - val_accuracy: 0.2527
Epoch 13/20
46/46 [==============================] - 21s 465ms/step - loss: 0.1292 - accuracy:
0.9706 - val_loss: 7.0060 - val_accuracy: 0.2473
Epoch 14/20
46/46 [==============================] - 22s 475ms/step - loss: 0.1035 - accuracy:
0.9754 - val_loss: 6.7564 - val_accuracy: 0.2636
Epoch 15/20
46/46 [==============================] - 18s 385ms/step - loss: 0.0724 - accuracy:
0.9816 - val_loss: 6.7645 - val_accuracy: 0.2745
Epoch 16/20
46/46 [==============================] - 19s 421ms/step - loss: 0.1250 - accuracy:
0.9638 - val_loss: 6.7818 - val_accuracy: 0.2962
Epoch 17/20
46/46 [==============================] - 18s 388ms/step - loss: 0.0625 - accuracy:
0.9884 - val_loss: 7.4075 - val_accuracy: 0.2989
Epoch 18/20
46/46 [==============================] - 18s 396ms/step - loss: 0.0531 - accuracy:
0.9891 - val_loss: 7.7675 - val_accuracy: 0.2717
Epoch 19/20
46/46 [==============================] - 18s 391ms/step - loss: 0.0515 - accuracy:
```

```
0.9863 - val_loss: 7.7974 - val_accuracy: 0.2853
Epoch 20/20
46/46 [==============================] - 18s 386ms/step - loss: 0.0670 - accuracy:
0.9870 - val_loss: 7.7638 - val_accuracy: 0.2527
```

In [61]:
```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_3 (Rescaling)     (None, 180, 180, 3)       0

 conv2d_3 (Conv2D)           (None, 180, 180, 16)      448

 max_pooling2d_3 (MaxPooling  (None, 90, 90, 16)       0
 2D)

 conv2d_4 (Conv2D)           (None, 90, 90, 32)        4640

 max_pooling2d_4 (MaxPooling  (None, 45, 45, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 45, 45, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 22, 22, 64)       0
 2D)

 dropout_8 (Dropout)         (None, 22, 22, 64)        0

 flatten_2 (Flatten)         (None, 30976)             0

 dense_5 (Dense)             (None, 128)               3965056

 outputs (Dense)             (None, 15)                1935

=================================================================
Total params: 3,990,575
Trainable params: 3,990,575
Non-trainable params: 0
_________________________________________________________________
```

In [62]:
```
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



That's slightly better... though not really. It seems more consistent in its predictions lying around the 27% range correct. That's just not good but a little more consistent than the dense NN. Let's try with a consistent size for the filters.

In [20]:
```
##Convolution Neural Network with increasing filters
second_cnn_model = tf.keras.models.Sequential([
    tf.keras.layers.Rescaling(1./255),
    tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(num_classes, name="outputs"),
])
```

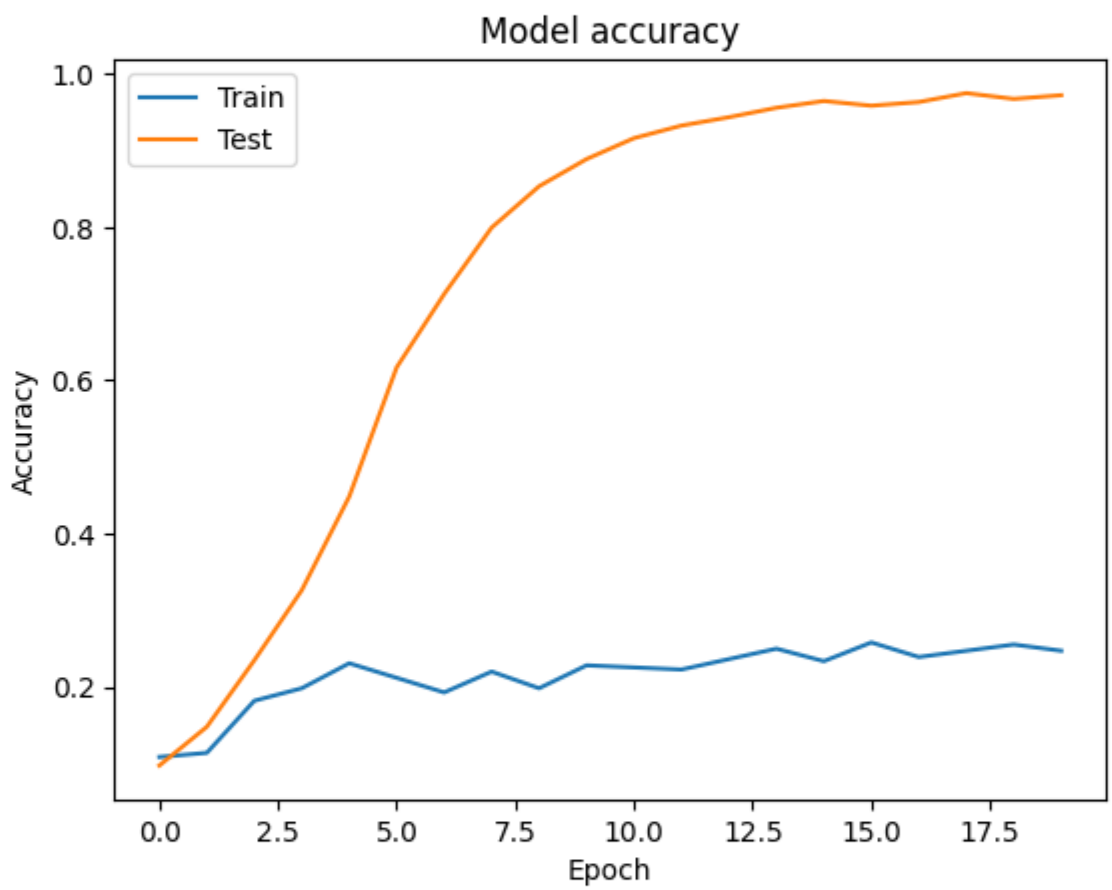
In [21]:
```
second_cnn_model.compile(
optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy'])
```

In [22]:
```
history = second_cnn_model.fit(
    train_ds,
    validation_data=test_ds,
    epochs=20
)
```

```
Epoch 1/20
46/46 [==============================] - 26s 536ms/step - loss: 2.6516 - accuracy:
0.1065 - val_loss: 2.5312 - val_accuracy: 0.1685
Epoch 2/20
46/46 [==============================] - 21s 461ms/step - loss: 2.4183 - accuracy:
0.2225 - val_loss: 2.4682 - val_accuracy: 0.2283
Epoch 3/20
46/46 [==============================] - 20s 438ms/step - loss: 2.1482 - accuracy:
0.3222 - val_loss: 2.5832 - val_accuracy: 0.2092
Epoch 4/20
46/46 [==============================] - 21s 454ms/step - loss: 1.7895 - accuracy:
0.4355 - val_loss: 2.5838 - val_accuracy: 0.2446
Epoch 5/20
46/46 [==============================] - 21s 453ms/step - loss: 1.3691 - accuracy:
0.5768 - val_loss: 2.8847 - val_accuracy: 0.2636
Epoch 6/20
46/46 [==============================] - 20s 445ms/step - loss: 1.0060 - accuracy:
0.7003 - val_loss: 3.1939 - val_accuracy: 0.2717
Epoch 7/20
46/46 [==============================] - 20s 447ms/step - loss: 0.7830 - accuracy:
0.7611 - val_loss: 3.3039 - val_accuracy: 0.2663
Epoch 8/20
46/46 [==============================] - 21s 454ms/step - loss: 0.5408 - accuracy:
0.8348 - val_loss: 4.0279 - val_accuracy: 0.2717
Epoch 9/20
46/46 [==============================] - 21s 454ms/step - loss: 0.4161 - accuracy:
0.8737 - val_loss: 4.7060 - val_accuracy: 0.2989
Epoch 10/20
46/46 [==============================] - 21s 448ms/step - loss: 0.2463 - accuracy:
0.9365 - val_loss: 5.2302 - val_accuracy: 0.2826
Epoch 11/20
46/46 [==============================] - 20s 434ms/step - loss: 0.1903 - accuracy:
0.9447 - val_loss: 5.6797 - val_accuracy: 0.2473
Epoch 12/20
46/46 [==============================] - 20s 434ms/step - loss: 0.1695 - accuracy:
0.9549 - val_loss: 5.5471 - val_accuracy: 0.2772
Epoch 13/20
46/46 [==============================] - 21s 458ms/step - loss: 0.0937 - accuracy:
0.9741 - val_loss: 5.6087 - val_accuracy: 0.2935
Epoch 14/20
46/46 [==============================] - 21s 462ms/step - loss: 0.0838 - accuracy:
0.9782 - val_loss: 6.5201 - val_accuracy: 0.2527
Epoch 15/20
46/46 [==============================] - 21s 452ms/step - loss: 0.0889 - accuracy:
0.9747 - val_loss: 6.1544 - val_accuracy: 0.2826
Epoch 16/20
46/46 [==============================] - 21s 463ms/step - loss: 0.0847 - accuracy:
0.9795 - val_loss: 6.3391 - val_accuracy: 0.2745
Epoch 17/20
46/46 [==============================] - 21s 455ms/step - loss: 0.0655 - accuracy:
0.9802 - val_loss: 6.5923 - val_accuracy: 0.2908
Epoch 18/20
46/46 [==============================] - 21s 451ms/step - loss: 0.0640 - accuracy:
0.9829 - val_loss: 7.1884 - val_accuracy: 0.2935
Epoch 19/20
46/46 [==============================] - 21s 464ms/step - loss: 0.0556 - accuracy:
```

```
0.9843 - val_loss: 7.2917 - val_accuracy: 0.3043
Epoch 20/20
46/46 [==============================] - 21s 455ms/step - loss: 0.0675 - accuracy:
0.9829 - val_loss: 6.8372 - val_accuracy: 0.2826
```

In [69]:
```
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



Not better. Seems like the CNN is very minorly better at classifying the images than the dense nn but both get failing grades. Next up is a pre-trained model.

In [47]:
```
##Trying a Pre-trained model
base_model = tf.keras.applications.MobileNetV2(input_shape=(img_height, img_width,
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [9
6, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the def
ault.
```

In [48]:
```
feature_batch = base_model(image_batch)
print(feature_batch.shape)
```

```
(32, 6, 6, 1280)
```

In [49]:
```
base_model.trainable=False
base_model.summary()
```

```
Model: "mobilenetv2_1.00_224"
_________________________________________________________________________________________
______________
 Layer (type)                   Output Shape         Param #     Connected to
=========================================================================================
==============
 input_1 (InputLayer)           [(None, 180, 180, 3  0           []
                                )]

 Conv1 (Conv2D)                 (None, 90, 90, 32)   864         ['input_1[0][0]']

 bn_Conv1 (BatchNormalization)  (None, 90, 90, 32)   128         ['Conv1[0][0]']

 Conv1_relu (ReLU)              (None, 90, 90, 32)   0           ['bn_Conv1[0][0]']

 expanded_conv_depthwise (Depth  (None, 90, 90, 32)  288         ['Conv1_relu
[0][0]']
 wiseConv2D)

 expanded_conv_depthwise_BN (Ba  (None, 90, 90, 32)  128         ['expanded_conv_dep
thwise[0][0]']
 tchNormalization)

 expanded_conv_depthwise_relu (  (None, 90, 90, 32)  0           ['expanded_conv_dep
thwise_BN[0][0
 ReLU)                                                           ]']

 expanded_conv_project (Conv2D)  (None, 90, 90, 16)  512         ['expanded_conv_dep
thwise_relu[0]
                                                                 [0]']

 expanded_conv_project_BN (Batc  (None, 90, 90, 16)  64          ['expanded_conv_pro
ject[0][0]']
 hNormalization)

 block_1_expand (Conv2D)        (None, 90, 90, 96)   1536        ['expanded_conv_pro
ject_BN[0][0]'
                                                                 ]

 block_1_expand_BN (BatchNormal  (None, 90, 90, 96)  384         ['block_1_expand
[0][0]']
 ization)

 block_1_expand_relu (ReLU)     (None, 90, 90, 96)   0           ['block_1_expand_BN
[0][0]']

 block_1_pad (ZeroPadding2D)    (None, 91, 91, 96)   0           ['block_1_expand_re
lu[0][0]']

 block_1_depthwise (DepthwiseCo  (None, 45, 45, 96)  864         ['block_1_pad
[0][0]']
 nv2D)

 block_1_depthwise_BN (BatchNor  (None, 45, 45, 96)  384         ['block_1_depthwise
[0][0]']
 malization)
```

```
 block_1_depthwise_relu (ReLU)  (None, 45, 45, 96)   0           ['block_1_depthwise
_BN[0][0]']

 block_1_project (Conv2D)       (None, 45, 45, 24)   2304        ['block_1_depthwise
_relu[0][0]']

 block_1_project_BN (BatchNorma  (None, 45, 45, 24)  96          ['block_1_project
[0][0]']
 lization)

 block_2_expand (Conv2D)        (None, 45, 45, 144)  3456        ['block_1_project_B
N[0][0]']

 block_2_expand_BN (BatchNormal  (None, 45, 45, 144)  576        ['block_2_expand
[0][0]']
 ization)

 block_2_expand_relu (ReLU)     (None, 45, 45, 144)  0           ['block_2_expand_BN
[0][0]']

 block_2_depthwise (DepthwiseCo  (None, 45, 45, 144)  1296       ['block_2_expand_re
lu[0][0]']
 nv2D)

 block_2_depthwise_BN (BatchNor  (None, 45, 45, 144)  576        ['block_2_depthwise
[0][0]']
 malization)

 block_2_depthwise_relu (ReLU)  (None, 45, 45, 144)  0           ['block_2_depthwise
_BN[0][0]']

 block_2_project (Conv2D)       (None, 45, 45, 24)   3456        ['block_2_depthwise
_relu[0][0]']

 block_2_project_BN (BatchNorma  (None, 45, 45, 24)  96          ['block_2_project
[0][0]']
 lization)

 block_2_add (Add)              (None, 45, 45, 24)   0           ['block_1_project_B
N[0][0]',
                                                                  'block_2_project_B
N[0][0]']

 block_3_expand (Conv2D)        (None, 45, 45, 144)  3456        ['block_2_add
[0][0]']

 block_3_expand_BN (BatchNormal  (None, 45, 45, 144)  576        ['block_3_expand
[0][0]']
 ization)

 block_3_expand_relu (ReLU)     (None, 45, 45, 144)  0           ['block_3_expand_BN
[0][0]']

 block_3_pad (ZeroPadding2D)    (None, 47, 47, 144)  0           ['block_3_expand_re
lu[0][0]']
```

```
 block_3_depthwise (DepthwiseCo  (None, 23, 23, 144)  1296       ['block_3_pad
[0][0]']
 nv2D)

 block_3_depthwise_BN (BatchNor  (None, 23, 23, 144)  576        ['block_3_depthwise
[0][0]']
 malization)

 block_3_depthwise_relu (ReLU)  (None, 23, 23, 144)  0          ['block_3_depthwise
_BN[0][0]']

 block_3_project (Conv2D)       (None, 23, 23, 32)   4608       ['block_3_depthwise
_relu[0][0]']

 block_3_project_BN (BatchNorma  (None, 23, 23, 32)  128        ['block_3_project
[0][0]']
 lization)

 block_4_expand (Conv2D)        (None, 23, 23, 192)  6144       ['block_3_project_B
N[0][0]']

 block_4_expand_BN (BatchNormal  (None, 23, 23, 192)  768        ['block_4_expand
[0][0]']
 ization)

 block_4_expand_relu (ReLU)     (None, 23, 23, 192)  0          ['block_4_expand_BN
[0][0]']

 block_4_depthwise (DepthwiseCo  (None, 23, 23, 192)  1728       ['block_4_expand_re
lu[0][0]']
 nv2D)

 block_4_depthwise_BN (BatchNor  (None, 23, 23, 192)  768        ['block_4_depthwise
[0][0]']
 malization)

 block_4_depthwise_relu (ReLU)  (None, 23, 23, 192)  0          ['block_4_depthwise
_BN[0][0]']

 block_4_project (Conv2D)       (None, 23, 23, 32)   6144       ['block_4_depthwise
_relu[0][0]']

 block_4_project_BN (BatchNorma  (None, 23, 23, 32)  128        ['block_4_project
[0][0]']
 lization)

 block_4_add (Add)              (None, 23, 23, 32)   0          ['block_3_project_B
N[0][0]',
                                                                 'block_4_project_B
N[0][0]']

 block_5_expand (Conv2D)        (None, 23, 23, 192)  6144       ['block_4_add
[0][0]']

 block_5_expand_BN (BatchNormal  (None, 23, 23, 192)  768        ['block_5_expand
```

```
[0][0]']
 ization)

 block_5_expand_relu (ReLU)     (None, 23, 23, 192)  0           ['block_5_expand_BN
[0][0]']

 block_5_depthwise (DepthwiseCo  (None, 23, 23, 192)  1728       ['block_5_expand_re
lu[0][0]']
 nv2D)

 block_5_depthwise_BN (BatchNor  (None, 23, 23, 192)  768        ['block_5_depthwise
[0][0]']
 malization)

 block_5_depthwise_relu (ReLU)  (None, 23, 23, 192)  0           ['block_5_depthwise
_BN[0][0]']

 block_5_project (Conv2D)       (None, 23, 23, 32)   6144        ['block_5_depthwise
_relu[0][0]']

 block_5_project_BN (BatchNorma  (None, 23, 23, 32)  128         ['block_5_project
[0][0]']
 lization)

 block_5_add (Add)              (None, 23, 23, 32)   0           ['block_4_add
[0][0]',
                                                                  'block_5_project_B
N[0][0]']

 block_6_expand (Conv2D)        (None, 23, 23, 192)  6144        ['block_5_add
[0][0]']

 block_6_expand_BN (BatchNormal  (None, 23, 23, 192)  768        ['block_6_expand
[0][0]']
 ization)

 block_6_expand_relu (ReLU)     (None, 23, 23, 192)  0           ['block_6_expand_BN
[0][0]']

 block_6_pad (ZeroPadding2D)    (None, 25, 25, 192)  0           ['block_6_expand_re
lu[0][0]']

 block_6_depthwise (DepthwiseCo  (None, 12, 12, 192)  1728       ['block_6_pad
[0][0]']
 nv2D)

 block_6_depthwise_BN (BatchNor  (None, 12, 12, 192)  768        ['block_6_depthwise
[0][0]']
 malization)

 block_6_depthwise_relu (ReLU)  (None, 12, 12, 192)  0           ['block_6_depthwise
_BN[0][0]']

 block_6_project (Conv2D)       (None, 12, 12, 64)   12288       ['block_6_depthwise
_relu[0][0]']
```

```
 block_6_project_BN (BatchNorma  (None, 12, 12, 64)  256         ['block_6_project
[0][0]']
 lization)

 block_7_expand (Conv2D)        (None, 12, 12, 384)  24576       ['block_6_project_B
N[0][0]']

 block_7_expand_BN (BatchNormal  (None, 12, 12, 384)  1536       ['block_7_expand
[0][0]']
 ization)

 block_7_expand_relu (ReLU)     (None, 12, 12, 384)  0           ['block_7_expand_BN
[0][0]']

 block_7_depthwise (DepthwiseCo  (None, 12, 12, 384)  3456       ['block_7_expand_re
lu[0][0]']
 nv2D)

 block_7_depthwise_BN (BatchNor  (None, 12, 12, 384)  1536       ['block_7_depthwise
[0][0]']
 malization)

 block_7_depthwise_relu (ReLU)  (None, 12, 12, 384)  0           ['block_7_depthwise
_BN[0][0]']

 block_7_project (Conv2D)       (None, 12, 12, 64)   24576       ['block_7_depthwise
_relu[0][0]']

 block_7_project_BN (BatchNorma  (None, 12, 12, 64)  256         ['block_7_project
[0][0]']
 lization)

 block_7_add (Add)              (None, 12, 12, 64)   0           ['block_6_project_B
N[0][0]',
                                                                  'block_7_project_B
N[0][0]']

 block_8_expand (Conv2D)        (None, 12, 12, 384)  24576       ['block_7_add
[0][0]']

 block_8_expand_BN (BatchNormal  (None, 12, 12, 384)  1536       ['block_8_expand
[0][0]']
 ization)

 block_8_expand_relu (ReLU)     (None, 12, 12, 384)  0           ['block_8_expand_BN
[0][0]']

 block_8_depthwise (DepthwiseCo  (None, 12, 12, 384)  3456       ['block_8_expand_re
lu[0][0]']
 nv2D)

 block_8_depthwise_BN (BatchNor  (None, 12, 12, 384)  1536       ['block_8_depthwise
[0][0]']
 malization)

 block_8_depthwise_relu (ReLU)  (None, 12, 12, 384)  0           ['block_8_depthwise
```

```
_BN[0][0]']

 block_8_project (Conv2D)       (None, 12, 12, 64)   24576       ['block_8_depthwise
_relu[0][0]']

 block_8_project_BN (BatchNorma  (None, 12, 12, 64)  256         ['block_8_project
[0][0]']
 lization)

 block_8_add (Add)              (None, 12, 12, 64)   0           ['block_7_add
[0][0]',
                                                                  'block_8_project_B
N[0][0]']

 block_9_expand (Conv2D)        (None, 12, 12, 384)  24576       ['block_8_add
[0][0]']

 block_9_expand_BN (BatchNormal  (None, 12, 12, 384)  1536       ['block_9_expand
[0][0]']
 ization)

 block_9_expand_relu (ReLU)     (None, 12, 12, 384)  0           ['block_9_expand_BN
[0][0]']

 block_9_depthwise (DepthwiseCo  (None, 12, 12, 384)  3456       ['block_9_expand_re
lu[0][0]']
 nv2D)

 block_9_depthwise_BN (BatchNor  (None, 12, 12, 384)  1536       ['block_9_depthwise
[0][0]']
 malization)

 block_9_depthwise_relu (ReLU)  (None, 12, 12, 384)  0           ['block_9_depthwise
_BN[0][0]']

 block_9_project (Conv2D)       (None, 12, 12, 64)   24576       ['block_9_depthwise
_relu[0][0]']

 block_9_project_BN (BatchNorma  (None, 12, 12, 64)  256         ['block_9_project
[0][0]']
 lization)

 block_9_add (Add)              (None, 12, 12, 64)   0           ['block_8_add
[0][0]',
                                                                  'block_9_project_B
N[0][0]']

 block_10_expand (Conv2D)       (None, 12, 12, 384)  24576       ['block_9_add
[0][0]']

 block_10_expand_BN (BatchNorma  (None, 12, 12, 384)  1536       ['block_10_expand
[0][0]']
 lization)

 block_10_expand_relu (ReLU)    (None, 12, 12, 384)  0           ['block_10_expand_B
N[0][0]']
```

```
 block_10_depthwise (DepthwiseC  (None, 12, 12, 384)  3456       ['block_10_expand_r
elu[0][0]']
 onv2D)

 block_10_depthwise_BN (BatchNo  (None, 12, 12, 384)  1536       ['block_10_depthwis
e[0][0]']
 rmalization)

 block_10_depthwise_relu (ReLU)  (None, 12, 12, 384)  0          ['block_10_depthwis
e_BN[0][0]']

 block_10_project (Conv2D)      (None, 12, 12, 96)   36864      ['block_10_depthwis
e_relu[0][0]']

 block_10_project_BN (BatchNorm  (None, 12, 12, 96)  384        ['block_10_project
[0][0]']
 alization)

 block_11_expand (Conv2D)       (None, 12, 12, 576)  55296      ['block_10_project_
BN[0][0]']

 block_11_expand_BN (BatchNorma  (None, 12, 12, 576)  2304       ['block_11_expand
[0][0]']
 lization)

 block_11_expand_relu (ReLU)    (None, 12, 12, 576)  0          ['block_11_expand_B
N[0][0]']

 block_11_depthwise (DepthwiseC  (None, 12, 12, 576)  5184       ['block_11_expand_r
elu[0][0]']
 onv2D)

 block_11_depthwise_BN (BatchNo  (None, 12, 12, 576)  2304       ['block_11_depthwis
e[0][0]']
 rmalization)

 block_11_depthwise_relu (ReLU)  (None, 12, 12, 576)  0          ['block_11_depthwis
e_BN[0][0]']

 block_11_project (Conv2D)      (None, 12, 12, 96)   55296      ['block_11_depthwis
e_relu[0][0]']

 block_11_project_BN (BatchNorm  (None, 12, 12, 96)  384        ['block_11_project
[0][0]']
 alization)

 block_11_add (Add)             (None, 12, 12, 96)   0          ['block_10_project_
BN[0][0]',
                                                                  'block_11_project_
BN[0][0]']

 block_12_expand (Conv2D)       (None, 12, 12, 576)  55296      ['block_11_add
[0][0]']

 block_12_expand_BN (BatchNorma  (None, 12, 12, 576)  2304       ['block_12_expand
```

```
[0][0]']
 lization)

 block_12_expand_relu (ReLU)    (None, 12, 12, 576)  0           ['block_12_expand_B
N[0][0]']

 block_12_depthwise (DepthwiseC  (None, 12, 12, 576)  5184       ['block_12_expand_r
elu[0][0]']
 onv2D)

 block_12_depthwise_BN (BatchNo  (None, 12, 12, 576)  2304       ['block_12_depthwis
e[0][0]']
 rmalization)

 block_12_depthwise_relu (ReLU)  (None, 12, 12, 576)  0          ['block_12_depthwis
e_BN[0][0]']

 block_12_project (Conv2D)      (None, 12, 12, 96)   55296       ['block_12_depthwis
e_relu[0][0]']

 block_12_project_BN (BatchNorm  (None, 12, 12, 96)  384         ['block_12_project
[0][0]']
 alization)

 block_12_add (Add)             (None, 12, 12, 96)   0           ['block_11_add
[0][0]',
                                                                  'block_12_project_
BN[0][0]']

 block_13_expand (Conv2D)       (None, 12, 12, 576)  55296       ['block_12_add
[0][0]']

 block_13_expand_BN (BatchNorma  (None, 12, 12, 576)  2304       ['block_13_expand
[0][0]']
 lization)

 block_13_expand_relu (ReLU)    (None, 12, 12, 576)  0           ['block_13_expand_B
N[0][0]']

 block_13_pad (ZeroPadding2D)   (None, 13, 13, 576)  0           ['block_13_expand_r
elu[0][0]']

 block_13_depthwise (DepthwiseC  (None, 6, 6, 576)   5184        ['block_13_pad
[0][0]']
 onv2D)

 block_13_depthwise_BN (BatchNo  (None, 6, 6, 576)   2304        ['block_13_depthwis
e[0][0]']
 rmalization)

 block_13_depthwise_relu (ReLU)  (None, 6, 6, 576)   0           ['block_13_depthwis
e_BN[0][0]']

 block_13_project (Conv2D)      (None, 6, 6, 160)    92160       ['block_13_depthwis
e_relu[0][0]']
```

```
 block_13_project_BN (BatchNorm  (None, 6, 6, 160)   640         ['block_13_project
[0][0]']
 alization)

 block_14_expand (Conv2D)       (None, 6, 6, 960)   153600      ['block_13_project_
BN[0][0]']

 block_14_expand_BN (BatchNorma  (None, 6, 6, 960)   3840        ['block_14_expand
[0][0]']
 lization)

 block_14_expand_relu (ReLU)    (None, 6, 6, 960)   0           ['block_14_expand_B
N[0][0]']

 block_14_depthwise (DepthwiseC  (None, 6, 6, 960)   8640        ['block_14_expand_r
elu[0][0]']
 onv2D)

 block_14_depthwise_BN (BatchNo  (None, 6, 6, 960)   3840        ['block_14_depthwis
e[0][0]']
 rmalization)

 block_14_depthwise_relu (ReLU)  (None, 6, 6, 960)   0           ['block_14_depthwis
e_BN[0][0]']

 block_14_project (Conv2D)      (None, 6, 6, 160)   153600      ['block_14_depthwis
e_relu[0][0]']

 block_14_project_BN (BatchNorm  (None, 6, 6, 160)   640         ['block_14_project
[0][0]']
 alization)

 block_14_add (Add)             (None, 6, 6, 160)   0           ['block_13_project_
BN[0][0]',
                                                                 'block_14_project_
BN[0][0]']

 block_15_expand (Conv2D)       (None, 6, 6, 960)   153600      ['block_14_add
[0][0]']

 block_15_expand_BN (BatchNorma  (None, 6, 6, 960)   3840        ['block_15_expand
[0][0]']
 lization)

 block_15_expand_relu (ReLU)    (None, 6, 6, 960)   0           ['block_15_expand_B
N[0][0]']

 block_15_depthwise (DepthwiseC  (None, 6, 6, 960)   8640        ['block_15_expand_r
elu[0][0]']
 onv2D)

 block_15_depthwise_BN (BatchNo  (None, 6, 6, 960)   3840        ['block_15_depthwis
e[0][0]']
 rmalization)

 block_15_depthwise_relu (ReLU)  (None, 6, 6, 960)   0           ['block_15_depthwis
```

```
e_BN[0][0]']

 block_15_project (Conv2D)      (None, 6, 6, 160)    153600      ['block_15_depthwis
e_relu[0][0]']

 block_15_project_BN (BatchNorm  (None, 6, 6, 160)   640         ['block_15_project
[0][0]']
 alization)

 block_15_add (Add)             (None, 6, 6, 160)    0           ['block_14_add
[0][0]',
                                                                  'block_15_project_
BN[0][0]']

 block_16_expand (Conv2D)       (None, 6, 6, 960)    153600      ['block_15_add
[0][0]']

 block_16_expand_BN (BatchNorma  (None, 6, 6, 960)   3840        ['block_16_expand
[0][0]']
 lization)

 block_16_expand_relu (ReLU)    (None, 6, 6, 960)    0           ['block_16_expand_B
N[0][0]']

 block_16_depthwise (DepthwiseC  (None, 6, 6, 960)   8640        ['block_16_expand_r
elu[0][0]']
 onv2D)

 block_16_depthwise_BN (BatchNo  (None, 6, 6, 960)   3840        ['block_16_depthwis
e[0][0]']
 rmalization)

 block_16_depthwise_relu (ReLU)  (None, 6, 6, 960)   0           ['block_16_depthwis
e_BN[0][0]']

 block_16_project (Conv2D)      (None, 6, 6, 320)    307200      ['block_16_depthwis
e_relu[0][0]']

 block_16_project_BN (BatchNorm  (None, 6, 6, 320)   1280        ['block_16_project
[0][0]']
 alization)

 Conv_1 (Conv2D)                (None, 6, 6, 1280)   409600      ['block_16_project_
BN[0][0]']

 Conv_1_bn (BatchNormalization)  (None, 6, 6, 1280)  5120        ['Conv_1[0][0]']

 out_relu (ReLU)                (None, 6, 6, 1280)   0           ['Conv_1_bn[0][0]']

==================================================================================
==============
Total params: 2,257,984
Trainable params: 0
Non-trainable params: 2,257,984
__________________________________________________________________________________
______________
```

In [50]:
```
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
feature_batch_average = global_average_layer(feature_batch)
print(feature_batch_average.shape)
```

```
(32, 1280)
```

In [51]:
```
prediction_layer = tf.keras.layers.Dense(15)
prediction_batch = prediction_layer(feature_batch_average)
print(prediction_batch.shape)
```

```
(32, 15)
```

In [52]:
```
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.RandomFlip('horizontal'),
  tf.keras.layers.RandomRotation(0.2),
])
```

In [53]:
```
inputs = tf.keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = tf.keras.applications.mobilenet_v2.preprocess_input(x)
x = base_model(x, training=False)
x = global_average_layer(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
```

In [54]:
```
base_learning_rate = 0.0001
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_2 (InputLayer)        [(None, 180, 180, 3)]     0         
                                                                 
 sequential_3 (Sequential)   (None, 180, 180, 3)       0         
                                                                 
 tf.math.truediv (TFOpLambda  (None, 180, 180, 3)      0         
 )                                                               
                                                                 
 tf.math.subtract (TFOpLambd  (None, 180, 180, 3)      0         
 a)                                                              
                                                                 
 mobilenetv2_1.00_224 (Funct  (None, 6, 6, 1280)       2257984   
 ional)                                                          
                                                                 
 global_average_pooling2d (G  (None, 1280)             0         
 lobalAveragePooling2D)                                          
                                                                 
 dropout_3 (Dropout)         (None, 1280)              0         
                                                                 
 dense_3 (Dense)             (None, 15)                19215     
                                                                 
=================================================================
Total params: 2,277,199
Trainable params: 19,215
Non-trainable params: 2,257,984
_________________________________________________________________
```

In [55]:
```
initial_epochs = 10

loss0, accuracy0 = model.evaluate(test_ds)
print("initial loss: {:.2f}".format(loss0))
print("initial accuracy: {:.2f}".format(accuracy0))
```

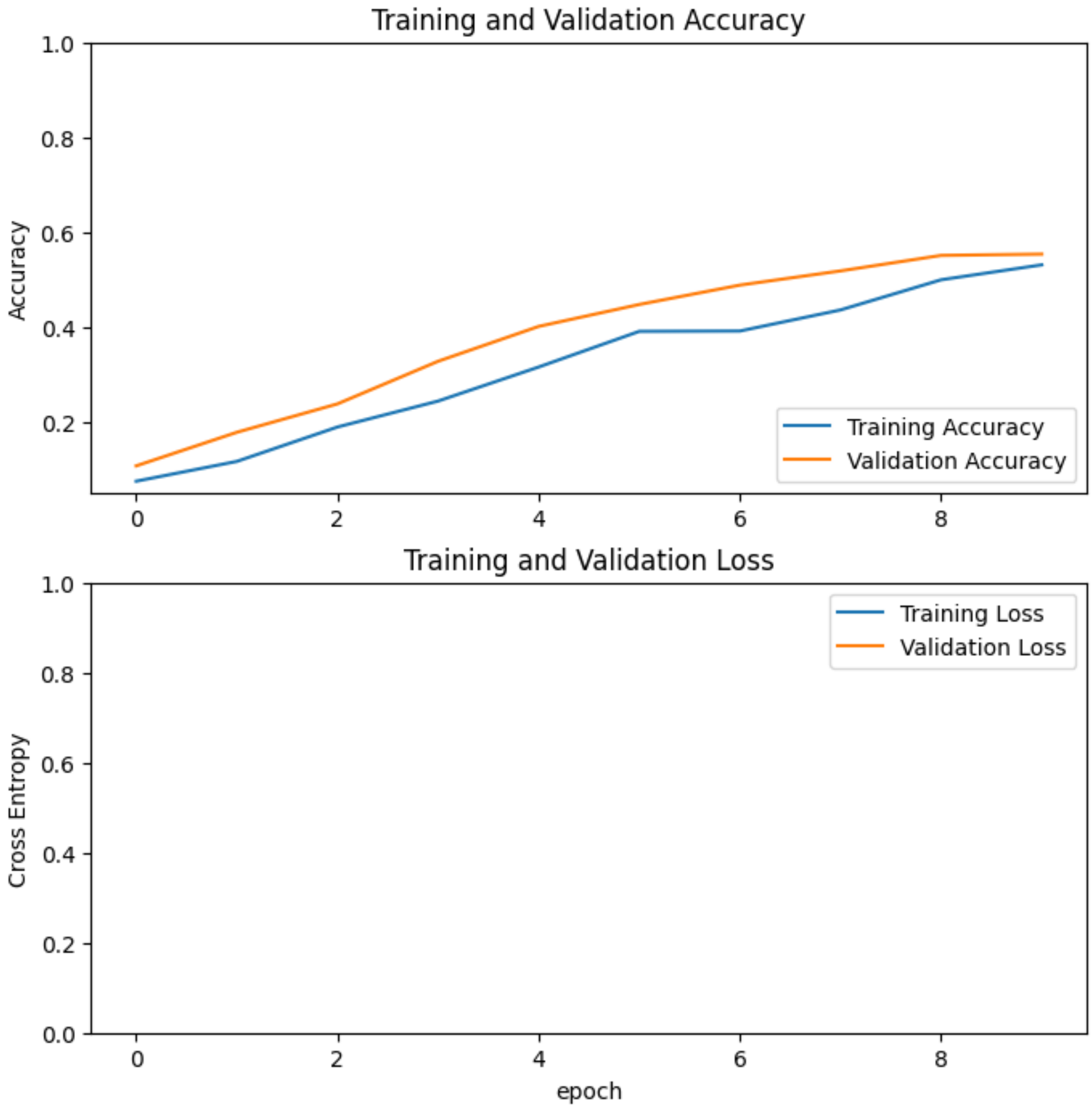
```
12/12 [==============================] - 5s 298ms/step - loss: 3.4748 - accuracy: 0.
0380
initial loss: 3.47
initial accuracy: 0.04
```

In [57]:
```
history = model.fit(train_ds,
                    epochs=20,
                    validation_data=test_ds)
```

```
Epoch 1/20
46/46 [==============================] - 21s 466ms/step - loss: 2.3879 - accuracy: 0.2430 - val_loss: 2.1833 - val_accuracy: 0.3179
Epoch 2/20
46/46 [==============================] - 22s 469ms/step - loss: 2.1727 - accuracy: 0.3181 - val_loss: 2.0124 - val_accuracy: 0.3696
Epoch 3/20
46/46 [==============================] - 21s 462ms/step - loss: 2.0468 - accuracy: 0.3570 - val_loss: 1.8680 - val_accuracy: 0.4212
Epoch 4/20
46/46 [==============================] - 22s 473ms/step - loss: 1.8687 - accuracy: 0.4082 - val_loss: 1.7505 - val_accuracy: 0.4837
Epoch 5/20
46/46 [==============================] - 22s 479ms/step - loss: 1.7765 - accuracy: 0.4546 - val_loss: 1.6481 - val_accuracy: 0.5217
Epoch 6/20
46/46 [==============================] - 20s 446ms/step - loss: 1.6857 - accuracy: 0.4812 - val_loss: 1.5628 - val_accuracy: 0.5435
Epoch 7/20
46/46 [==============================] - 21s 466ms/step - loss: 1.5709 - accuracy: 0.5133 - val_loss: 1.4949 - val_accuracy: 0.5652
Epoch 8/20
46/46 [==============================] - 21s 457ms/step - loss: 1.5094 - accuracy: 0.5311 - val_loss: 1.4294 - val_accuracy: 0.5842
Epoch 9/20
46/46 [==============================] - 21s 448ms/step - loss: 1.4233 - accuracy: 0.5672 - val_loss: 1.3776 - val_accuracy: 0.6005
Epoch 10/20
46/46 [==============================] - 21s 454ms/step - loss: 1.4008 - accuracy: 0.5754 - val_loss: 1.3326 - val_accuracy: 0.6060
Epoch 11/20
46/46 [==============================] - 21s 451ms/step - loss: 1.2951 - accuracy: 0.6082 - val_loss: 1.2932 - val_accuracy: 0.6386
Epoch 12/20
46/46 [==============================] - 21s 457ms/step - loss: 1.2970 - accuracy: 0.6020 - val_loss: 1.2565 - val_accuracy: 0.6495
Epoch 13/20
46/46 [==============================] - 24s 517ms/step - loss: 1.2435 - accuracy: 0.6150 - val_loss: 1.2261 - val_accuracy: 0.6576
Epoch 14/20
46/46 [==============================] - 22s 475ms/step - loss: 1.2073 - accuracy: 0.6334 - val_loss: 1.1985 - val_accuracy: 0.6603
Epoch 15/20
46/46 [==============================] - 20s 441ms/step - loss: 1.1852 - accuracy: 0.6355 - val_loss: 1.1732 - val_accuracy: 0.6630
Epoch 16/20
46/46 [==============================] - 21s 462ms/step - loss: 1.1431 - accuracy: 0.6485 - val_loss: 1.1494 - val_accuracy: 0.6658
Epoch 17/20
46/46 [==============================] - 21s 466ms/step - loss: 1.1038 - accuracy: 0.6621 - val_loss: 1.1256 - val_accuracy: 0.6685
Epoch 18/20
46/46 [==============================] - 21s 464ms/step - loss: 1.0847 - accuracy: 0.6635 - val_loss: 1.1050 - val_accuracy: 0.6821
Epoch 19/20
46/46 [==============================] - 23s 513ms/step - loss: 1.0388 - accuracy:
```

```
0.6867 - val_loss: 1.0898 - val_accuracy: 0.6848
Epoch 20/20
46/46 [==============================] - 25s 542ms/step - loss: 1.0323 - accuracy:
0.6840 - val_loss: 1.0740 - val_accuracy: 0.6902
```

In [57]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```


Training and Validation Accuracy
Accuracy
Training Accuracy
Validation Accuracy
Training and Validation Loss
Cross Entropy
Training Loss
Validation Loss
epoch

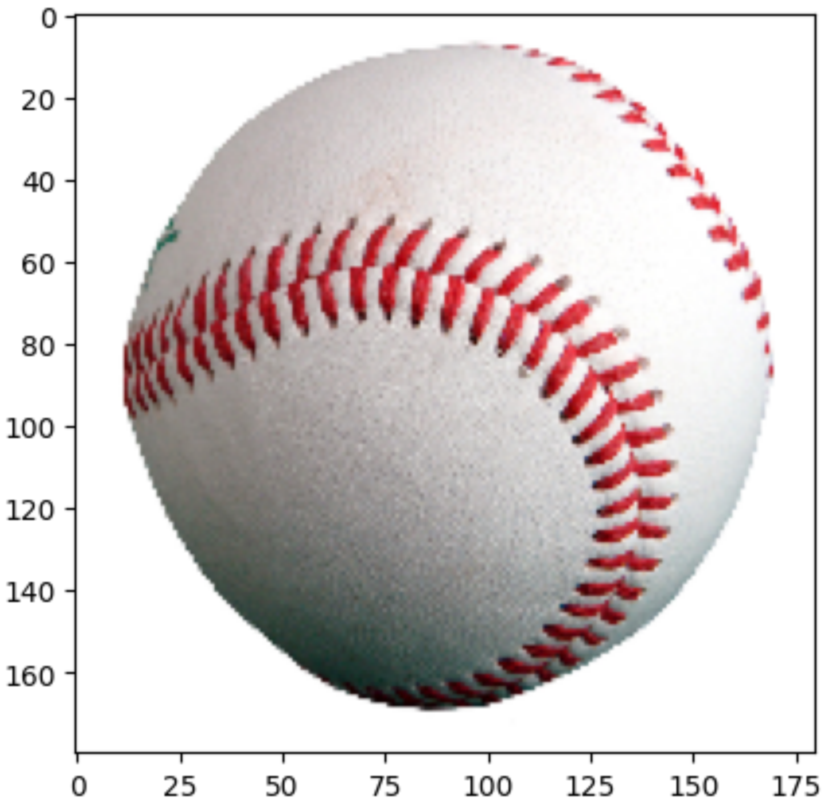
This is much better than the others. Around a 50% accuracy rate on validation is far more acceptable than the 1/4 accuracy rate on the prior models.

So in total, the order in which I would use these are the pretrained -> increasing filter CNN -> static filter CNN -> Dense NN. The pre-trained model has an accuracy rate that is fine for the use of describing items from images. It's not that important so the failure rate is good enough. The model it was trained off of has a much larger dataset it was trained from so it makes sense why it would have a greater success rate than this rather miniscule one that I used to train from. It was also relatively quick in training, taking around the same time as it took to train the CNNs. I would avoid using the Dense models for these types of image training tasks. The training time took a while and was still inaccurate. The CNNs could have more use if the dataset of images was better. They train quickly so iterations can take place. But a lot of tweaking would have to be done with both the image dataset and the layers to reduce the overfitting it seemed to do when working with such a small dataset.

Human vs Machine

Time to test the class and see how well you do in comparison to the prediction abilities of the computer.

In [88]:
```
#Getting all the images in
beachball_image_path = "gametime/beachball_prediction.jpg"
baseball_image_path = "gametime/baseball_prediction.jpg"
kettle_image_path = "gametime/kettle_prediction.jpg"
wreckingball_image_path = "gametime/wreckingball_prediction.jpg"
cat_image_path = "gametime/cat_prediction.jpg"
beachball_image = tf.keras.preprocessing.image.load_img(beachball_image_path, targe
baseball_image = tf.keras.preprocessing.image.load_img(baseball_image_path, target_
kettle_image = tf.keras.preprocessing.image.load_img(kettle_image_path, target_size
wreckingball_image = tf.keras.preprocessing.image.load_img(wreckingball_image_path,
cat_image = tf.keras.preprocessing.image.load_img(cat_image_path, target_size=(224,
##Starting with a baseball, which is included in the set
plt.imshow(baseball_image)
plt.show()
```



Is the above image:

a: Baseball

b: Hockey Puck

In [77]:
```
img_array = tf.keras.preprocessing.image.img_to_array(baseball_image)
img_batch = np.expand_dims(img_array, axis=0)
img_preprocessed = tf.keras.applications.mobilenet_v2.preprocess_input(img_batch)
```

In [78]:
```
prediction = model.predict(img_preprocessed)
```

```
1/1 [==============================] - 0s 35ms/step
```

In [79]:
```
print(class_names)
#print(tf.keras.applications.mobilenet_v2.decode_predictions(prediction))
```

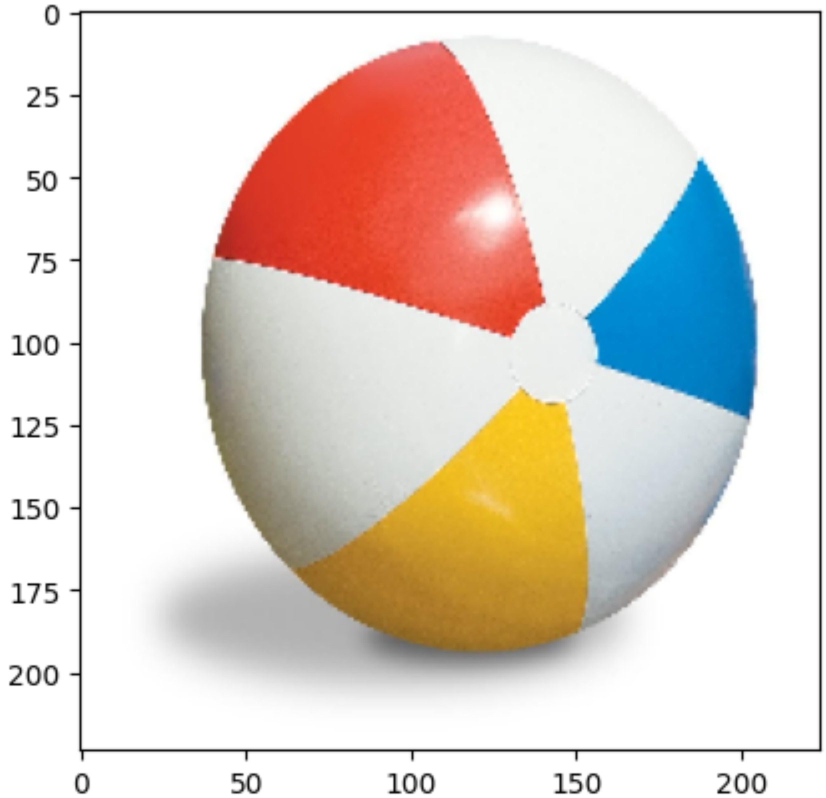
```
['american_football', 'baseball', 'basketball', 'billiard_ball', 'bowling_ball', 'cr
icket_ball', 'football', 'golf_ball', 'hockey_ball', 'hockey_puck', 'rugby_ball', 's
huttlecock', 'table_tennis_ball', 'tennis_ball', 'volleyball']
```

In [80]:
```
print(prediction)
```

```
[[-3.9127197  -1.4664302  -2.1226485  -0.98599964 -0.7799562  -0.83548254
  -2.6480684  -1.5858105  -2.0919716   0.22281113 -2.6008315  -1.6556082
  -0.31354156 -0.7221802  -0.9952256 ]]
```

That's right, a hockey puck

In [81]:
```
plt.imshow(beachball_image)
plt.show()
```



Is the above image:

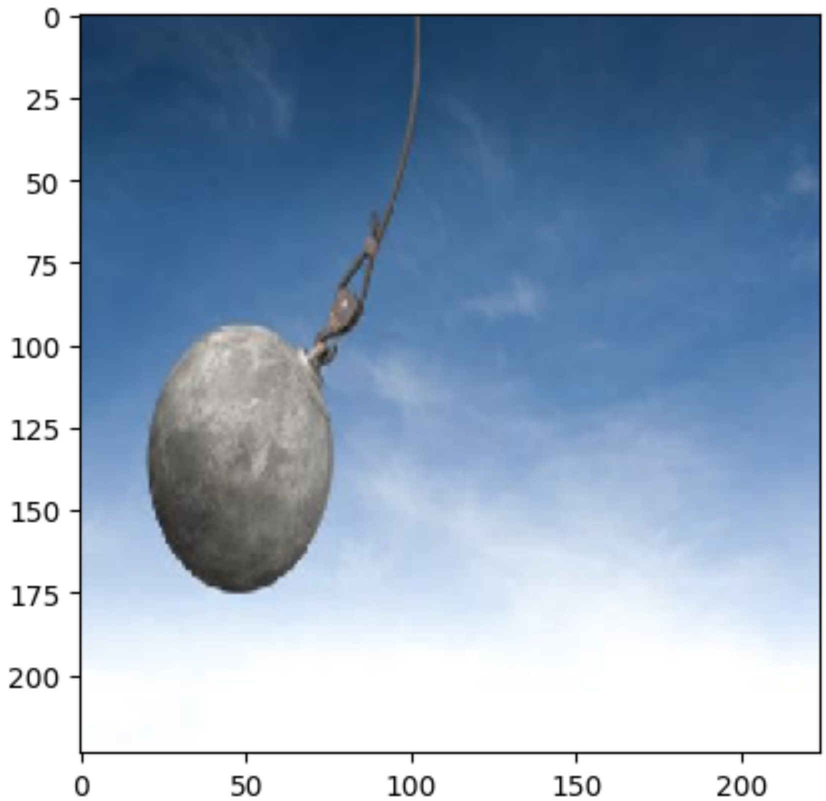
a: Table Tennis Ball

b: Volleyball

In [ ]:
```
img_array = tf.keras.preprocessing.image.img_to_array(beachball_image)
img_batch = np.expand_dims(img_array, axis=0)
img_preprocessed = tf.keras.applications.mobilenet_v2.preprocess_input(img_batch)
prediction = model.predict(img_preprocessed)
print(prediction)
```

That's right, a Table Tennis Ball

In [83]:
```
plt.imshow(wreckingball_image)
plt.show()
```



Is the above image:

a: Table Tennis Ball

b: Wrecking Ball

c: Bowling Ball

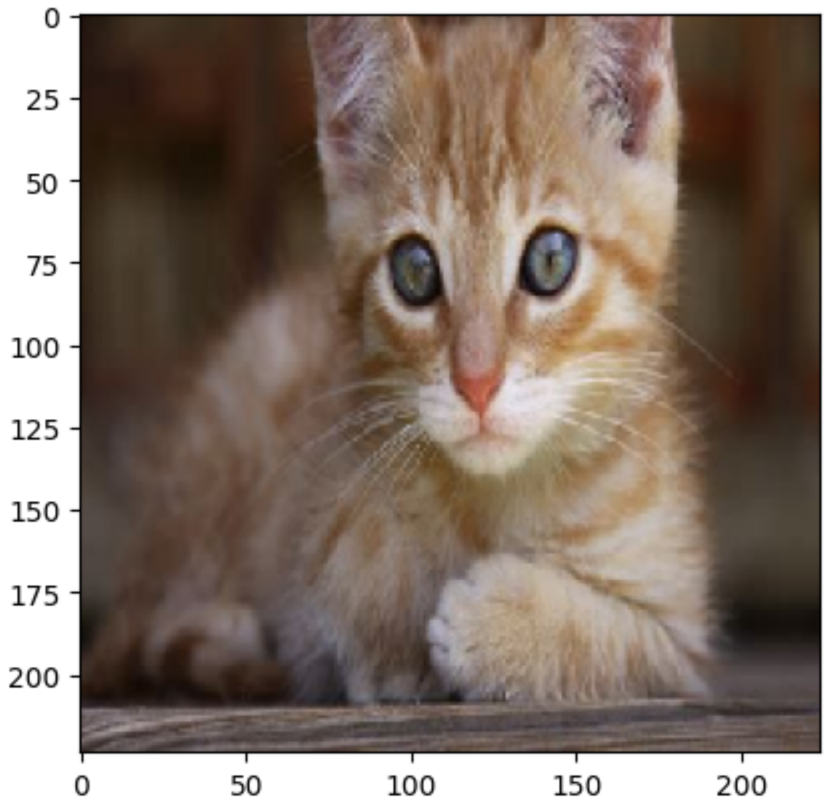
In [84]:
```
img_array = tf.keras.preprocessing.image.img_to_array(wreckingball_image)
img_batch = np.expand_dims(img_array, axis=0)
img_preprocessed = tf.keras.applications.mobilenet_v2.preprocess_input(img_batch)
prediction = model.predict(img_preprocessed)
print(prediction)
```

```
1/1 [==============================] - 0s 43ms/step
[[-3.4980984  -2.069184   -2.797303   -0.58421934 -1.4077848  -1.5072114
  -2.5720487  -1.52115    -2.077485    0.00589995 -3.1487935  -1.3932493
   0.45937595 -0.6425327  -0.35317883]]
```

That's right, a Table Tennis Ball

Final one

In [89]:
```
plt.imshow(cat_image)
plt.show()
```



Is the above image:

a: Table Tennis Ball

b: Baseball

c: Bowling Ball

d: Cat

In [90]:
```
img_array = tf.keras.preprocessing.image.img_to_array(cat_image)
img_batch = np.expand_dims(img_array, axis=0)
img_preprocessed = tf.keras.applications.mobilenet_v2.preprocess_input(img_batch)
prediction = model.predict(img_preprocessed)
print(prediction)
```

```
1/1 [==============================] - 0s 34ms/step
[[-3.435506   -1.9267582  -2.8801734  -0.57759637 -1.3328717  -1.3097317
  -2.690632   -1.6267952  -2.0840218   0.01304085 -3.297552   -1.391803
   0.42075038 -0.52846014 -0.38047805]]
```

That's Right, None of them. It's a hockey puck