0qgcye1lo

April 30, 2024

[1]:
```
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import os
import cv2
```

1 File paths for Images

[2]:
```
train_data_path = 'C://Users//gaura//Documents//GitHub//Scaler-Projects//
 ↪Computer Vision//Project//data//ninjacart_data//ninjacart_data//train'
test_data_path = 'C://Users//gaura//Documents//GitHub//Scaler-Projects//
 ↪Computer Vision//Project//data//ninjacart_data//ninjacart_data//test'
```

[3]:
```
os.listdir(train_data_path)
```

[3]:
```
['indian market', 'onion', 'potato', 'tomato']
```

2 Training image data count check

[4]:
```
train_image_count_check = {}
for value in os.listdir(train_data_path):
    tmp_path = train_data_path+"//"+str(value)
    train_image_count_check[str(value)] = len(os.listdir(tmp_path))
train_image_count_check
```

[4]:
```
{'indian market': 599, 'onion': 849, 'potato': 898, 'tomato': 789}
```

3 Testing image data count check

[5]:
```
test_image_count_check = {}
for value in os.listdir(test_data_path):
    tmp_path = test_data_path+"//"+str(value)
```

```
    test_image_count_check[str(value)] = len(os.listdir(tmp_path))
test_image_count_check
```

[5]:
```
{'indian market': 81, 'onion': 83, 'potato': 81, 'tomato': 106}
```

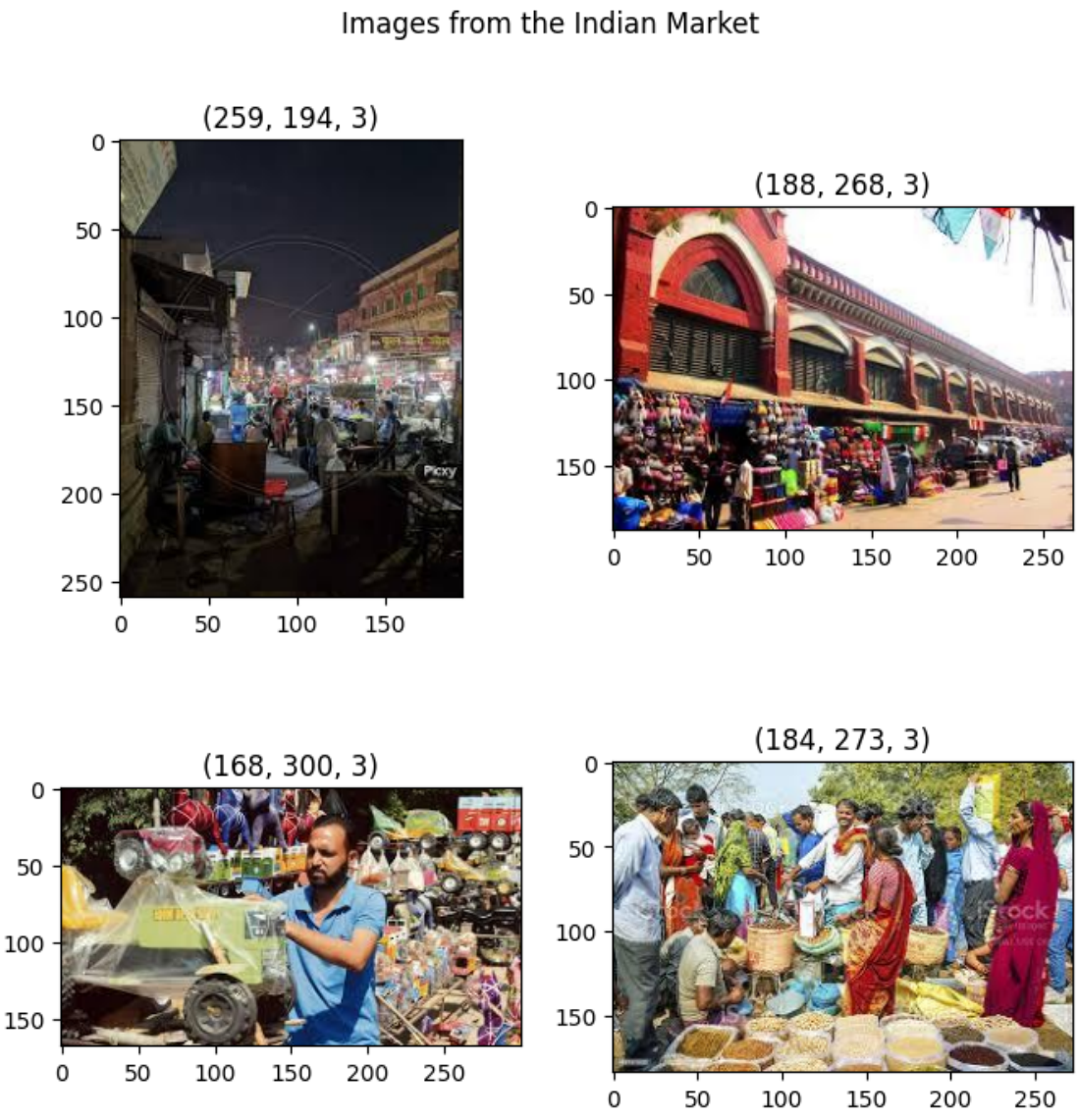
4 Ploting Images from Training Folder

[6]:
```
def plot_images(path,label,title):
    data = path+"//"+label
    first_four_images = os.listdir(data)[:4]
    fig, axs = plt.subplots(2,2, figsize=(8,8))
    for img,ax in enumerate(axs.flat):
        img_path = data+"//"+first_four_images[img]
        image = plt.imread(img_path)
        ax.set_title(image.shape)
        ax.imshow(image)
    fig.suptitle(title)
    plt.show()
```

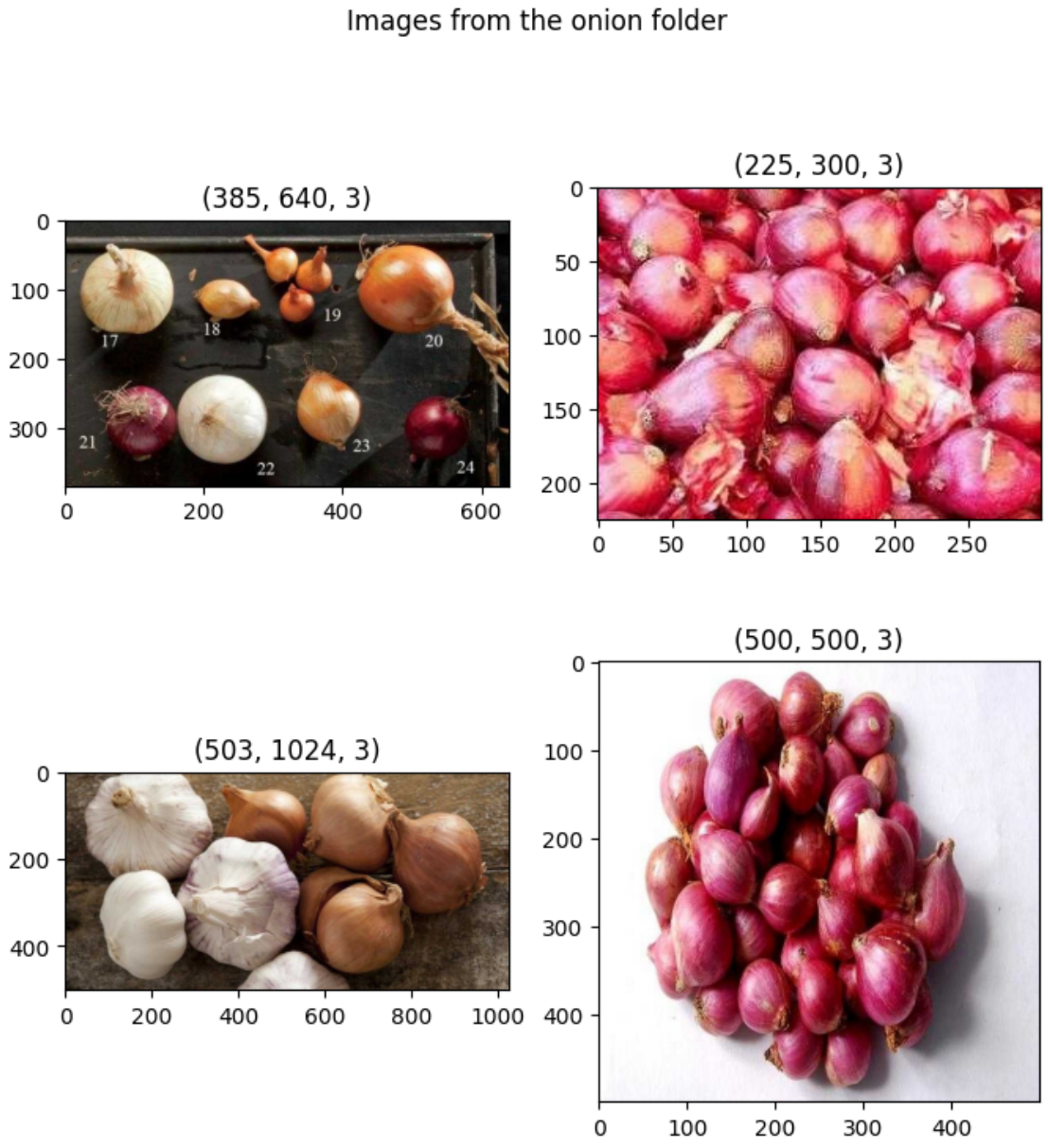
[7]:
```
os.listdir(train_data_path+"//"+'indian market')[-1:-4:-1]
```

[7]:
```
['village-market-827122__340.jpg', 'vendor-4049916__340.jpg', 'uyg.jpeg']
```

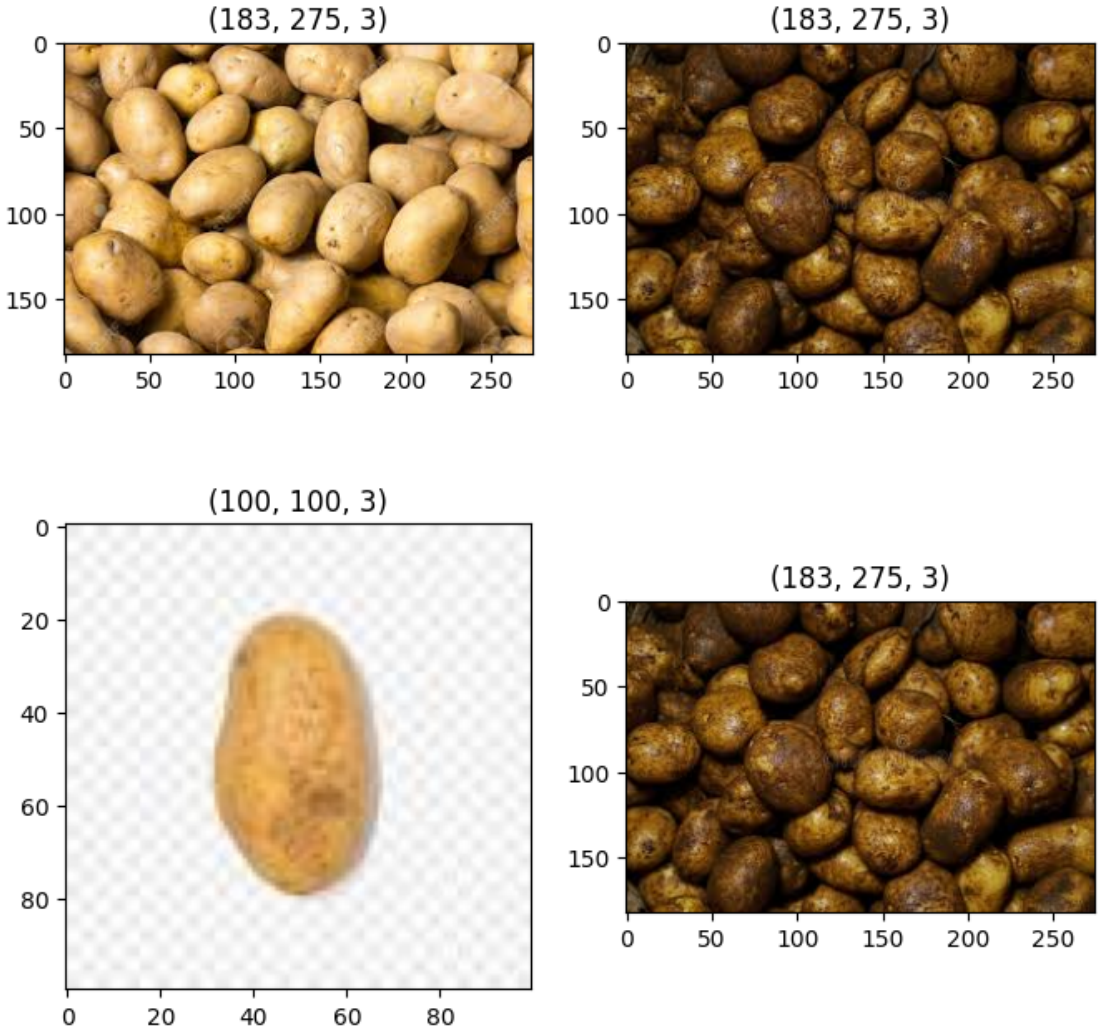
[8]:
```
plot_images(train_data_path,"indian market","Images from the Indian Market")
```



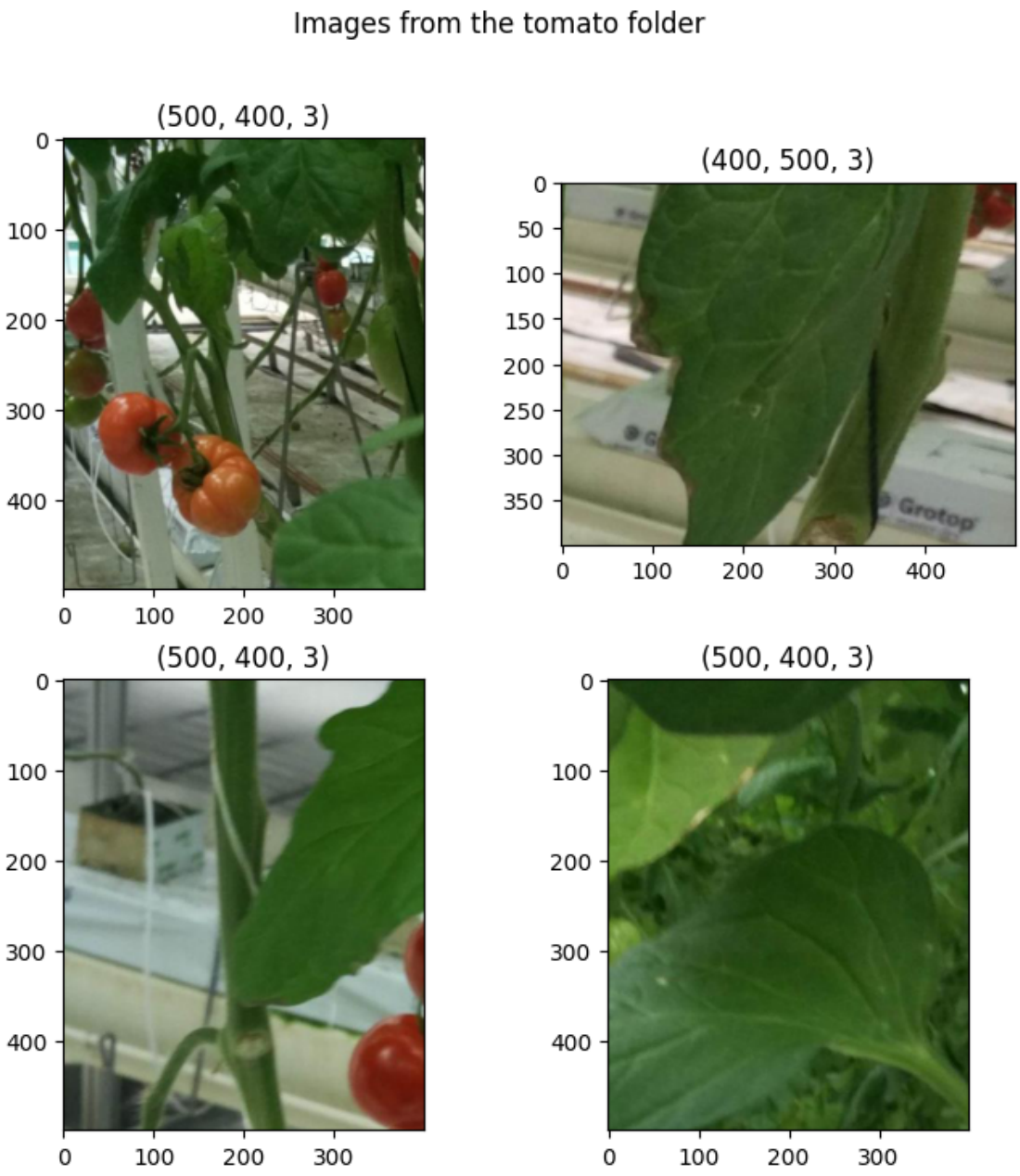
[9]:
```
plot_images(train_data_path,"onion","Images from the onion folder")
```



[10]:
```
plot_images(train_data_path,"potato","Images from the potato folder")
```

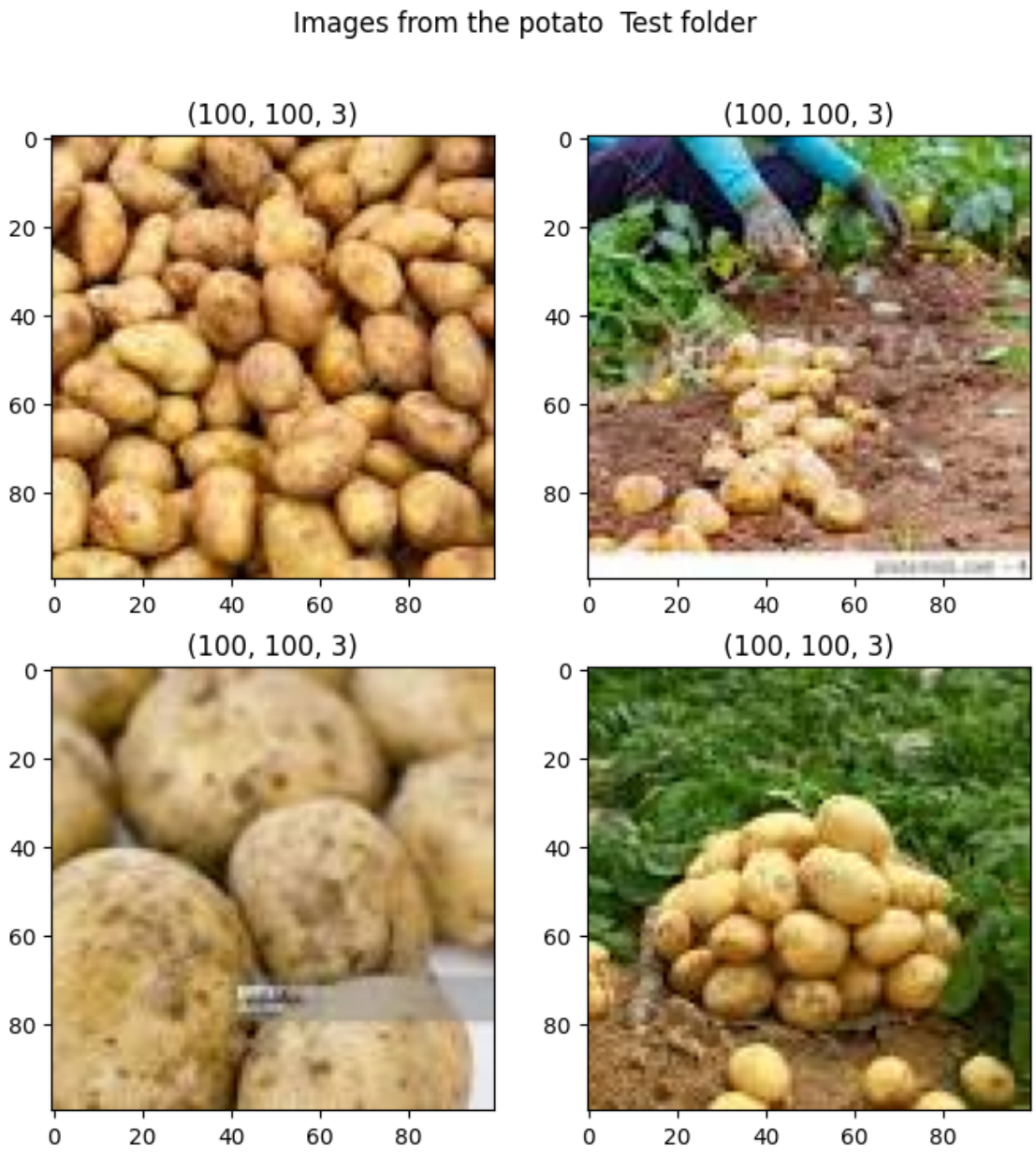


[11]:
```
plot_images(train_data_path,"tomato","Images from the tomato folder")
```

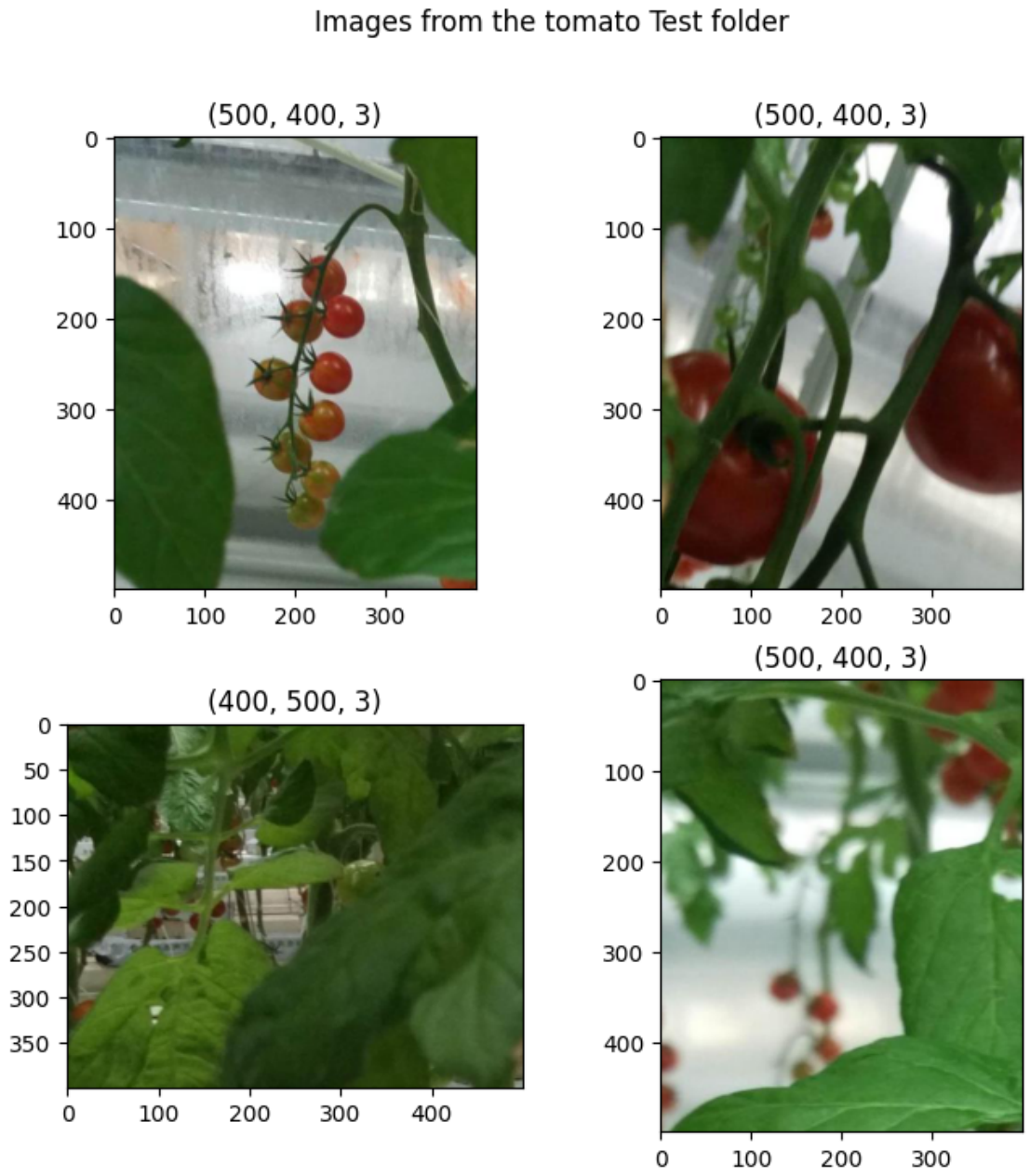


5 Ploting Images from Test Folder

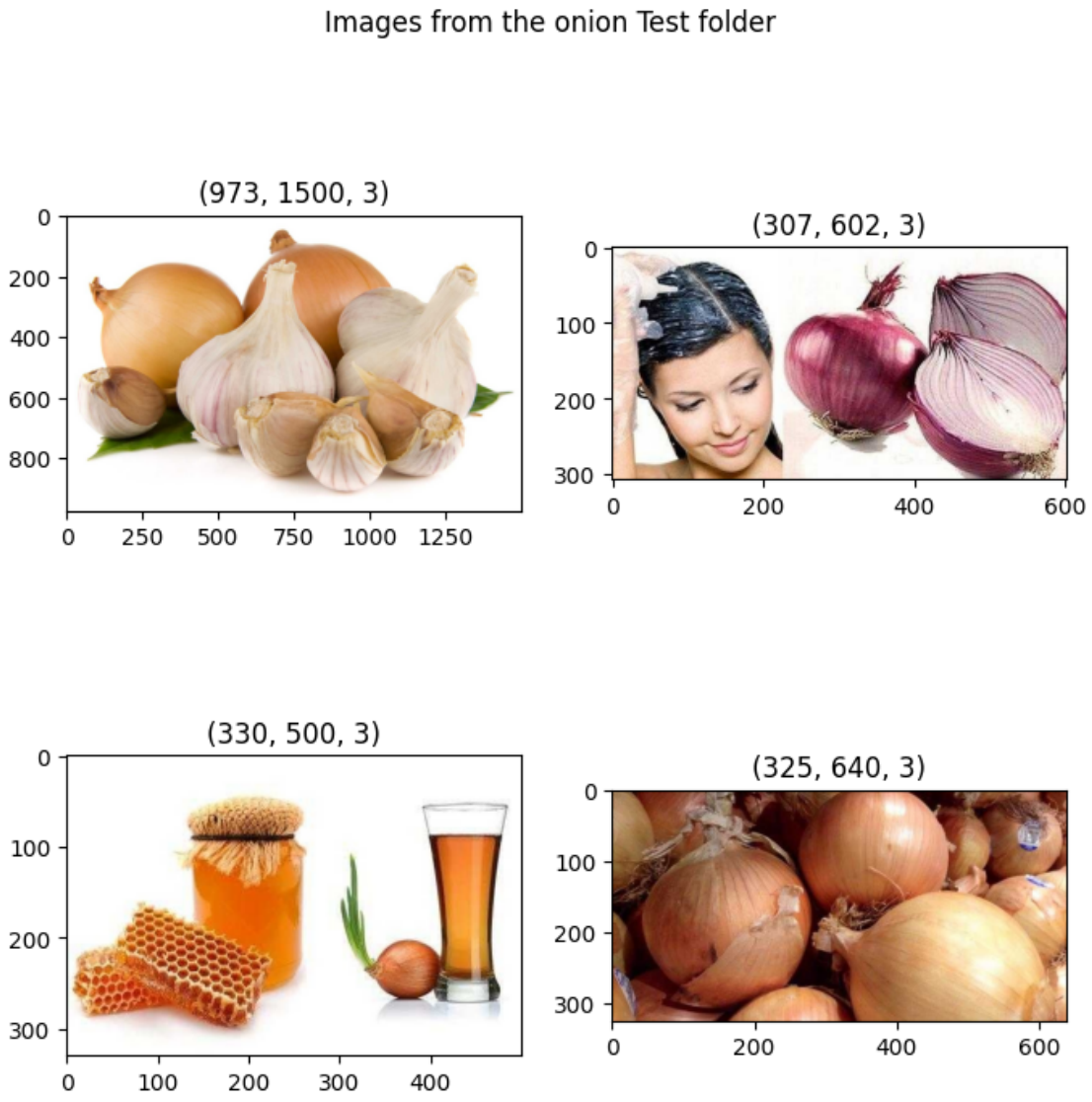
[12]:
```
plot_images(test_data_path,"potato","Images from the potato  Test folder")
```



[13]:
```
plot_images(test_data_path,"tomato","Images from the tomato Test folder")
```



[14]:
```
plot_images(test_data_path,"onion","Images from the onion Test folder")
```



[15]:
```
plot_images(test_data_path,"indian market","Images from the indian market Test␣
↪folder")
```

6 Loading Training and Test Data

[16]:
```
input_shape = (128,128)
train_data = tf.keras.utils.
  ↪image_dataset_from_directory(train_data_path,image_size=input_shape,batch_size=64,)
test_data = tf.keras.utils.
  ↪image_dataset_from_directory(test_data_path,image_size=input_shape,batch_size=64)
```

```
Found 3135 files belonging to 4 classes.
```

```
Found 351 files belonging to 4 classes.
```

[17]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import␣
↪Rescaling,RandomFlip,RandomTranslation,RandomRotation,RandomZoom,RandomContrast,RandomBrigh
```

7 Data Preprocessing and Data augumentation

[18]:
```
preproces_image = Sequential([
    Rescaling(scale=1./255)
])
```

[19]:
```
train_data = train_data.map(lambda x,y :␣
↪(preproces_image(x,training=True),y),num_parallel_calls = tf.data.AUTOTUNE)
test_data = test_data.map(lambda x,y :␣
↪(preproces_image(x,training=True),y),num_parallel_calls = tf.data.AUTOTUNE)
```

[20]:
```
len(train_data)
```

[20]:
```
49
```

[21]:
```
# image_augumentation = Sequential([
#     RandomFlip("horizontal"),
#     RandomFlip("vertical"),
#     RandomRotation(factor=0.5),
#     RandomZoom(height_factor=-0.1,width_factor=-0.1)
# ])
```

[22]:
```
# train_ds = train_data.map(lambda x,y :␣
↪(image_augumentation(x,training=True),y),num_parallel_calls =tf.data.
↪AUTOTUNE)
```

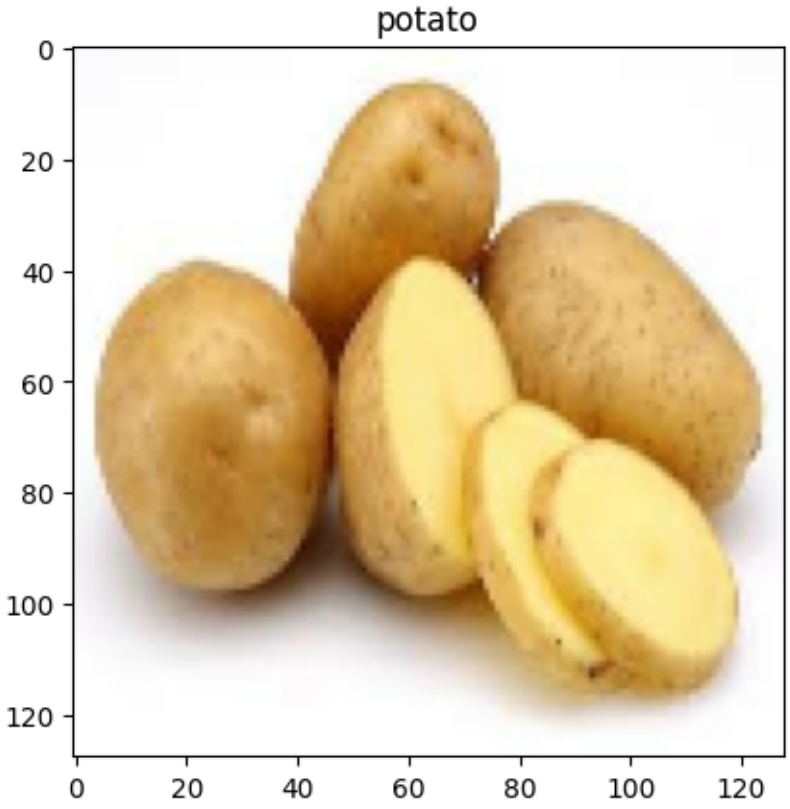
[23]:
```
val_data = train_data.take(15)
train_ds = train_data.skip(15)
print(tf.data.experimental.cardinality(train_ds))
```

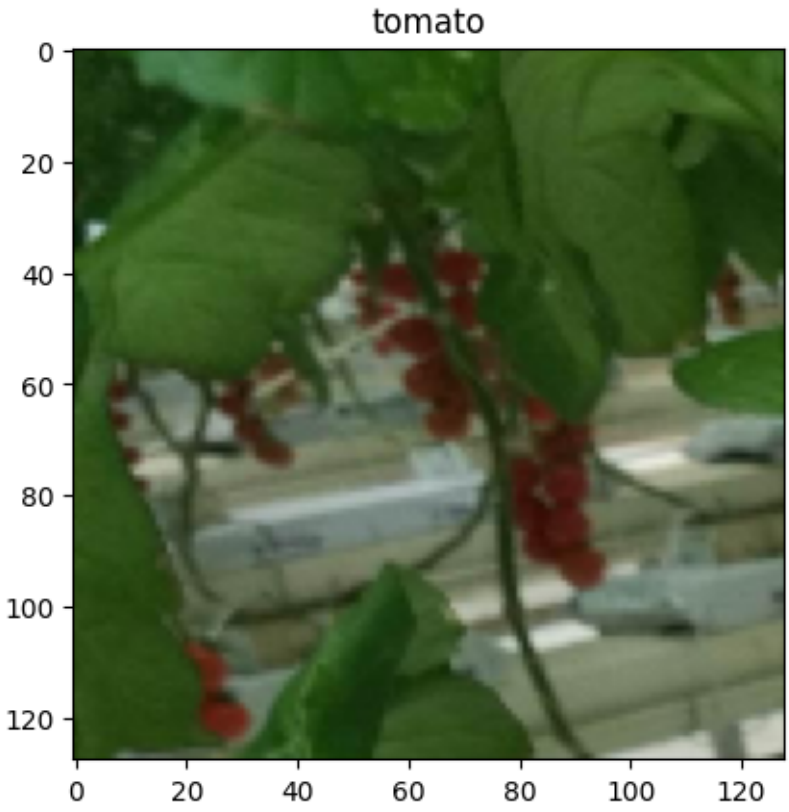
```
tf.Tensor(34, shape=(), dtype=int64)
```

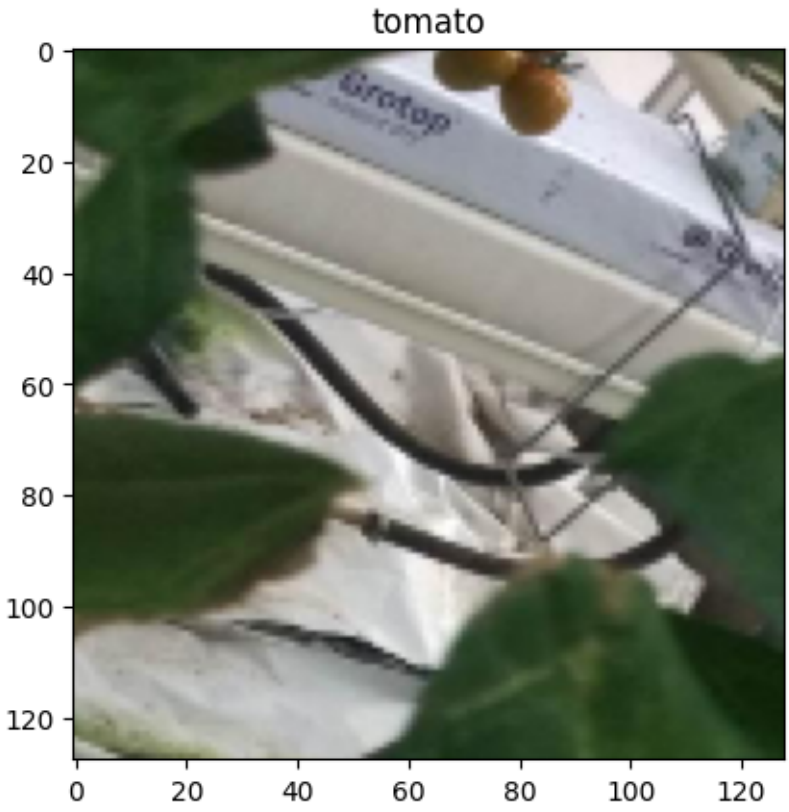
[24]:
```
images,label = next(iter(train_ds))
```

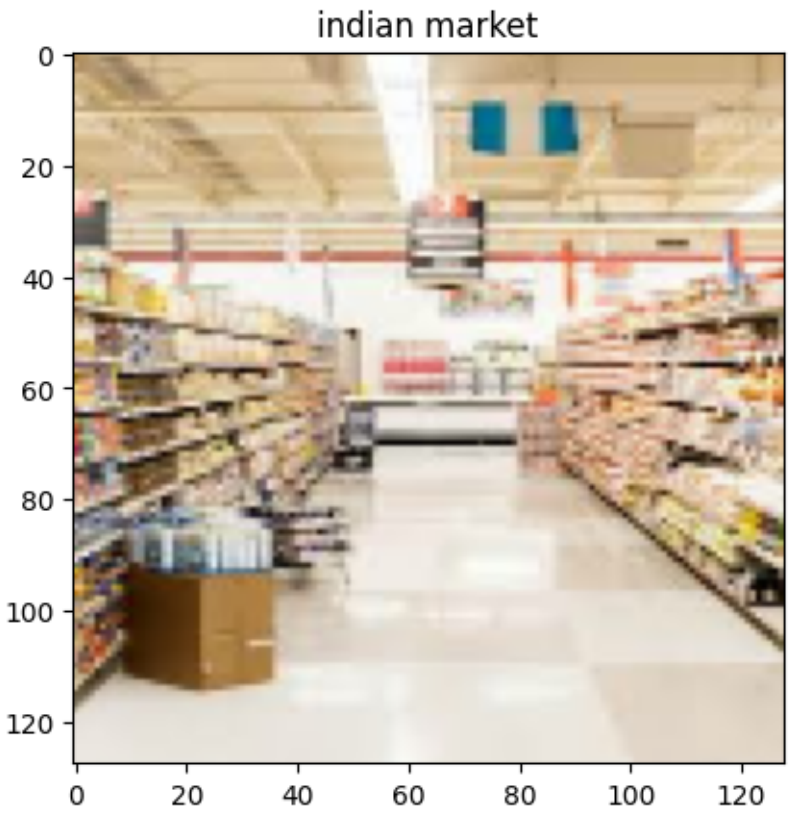
[25]:
```
reference_mapping_label = {
    2:'potato',
    3:'tomato',
    0:'indian market',
    1:'onion'
}
```

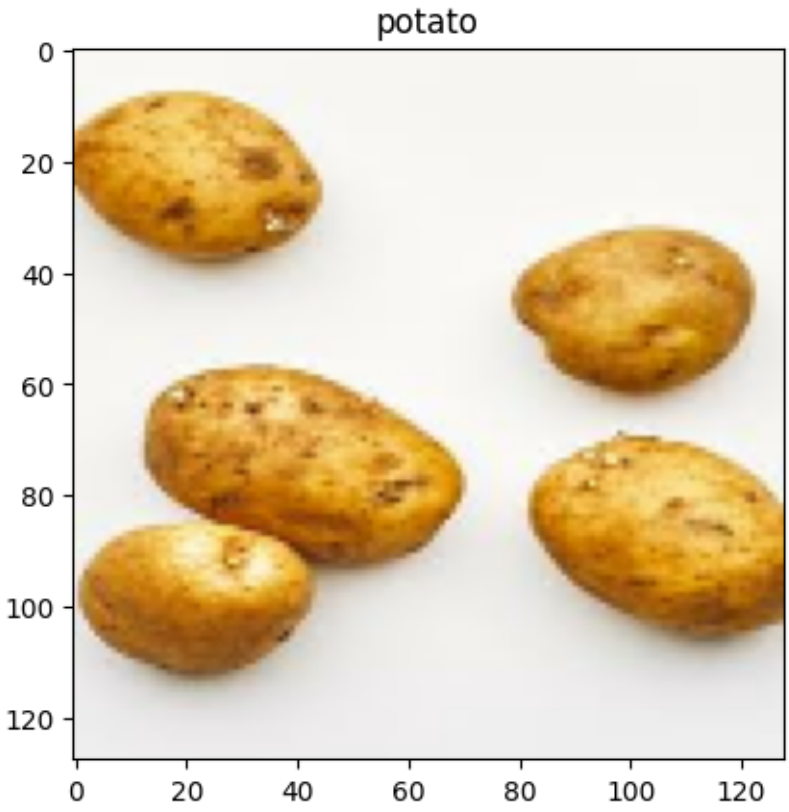
[26]:
```
for img,label in zip(images.numpy()[:10],label.numpy()[:10]):
    plt.title(reference_mapping_label[label])
   # plt.title(label)
    plt.imshow(img)
    plt.show()
```

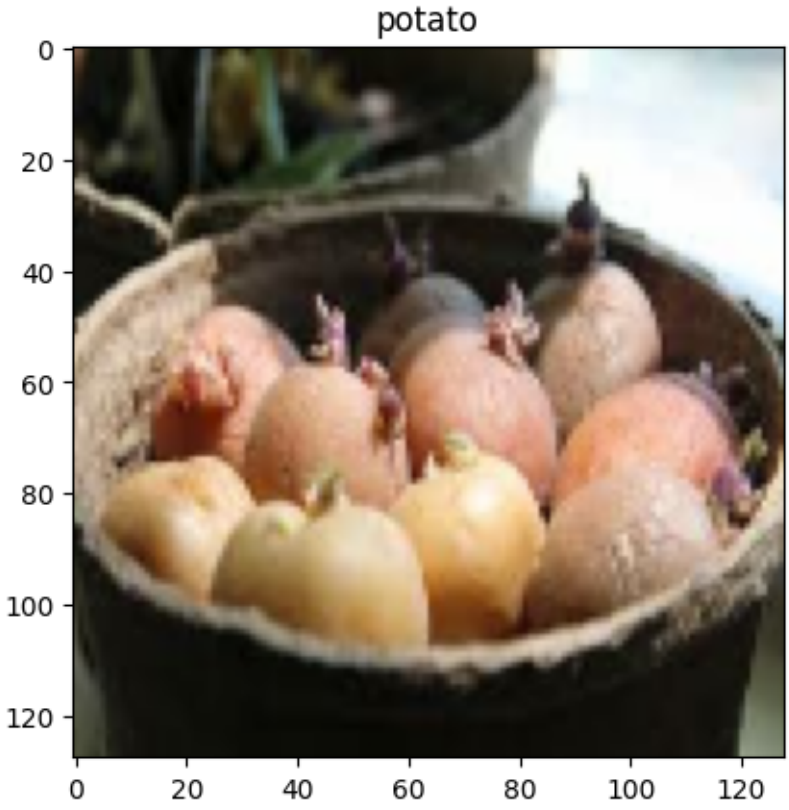


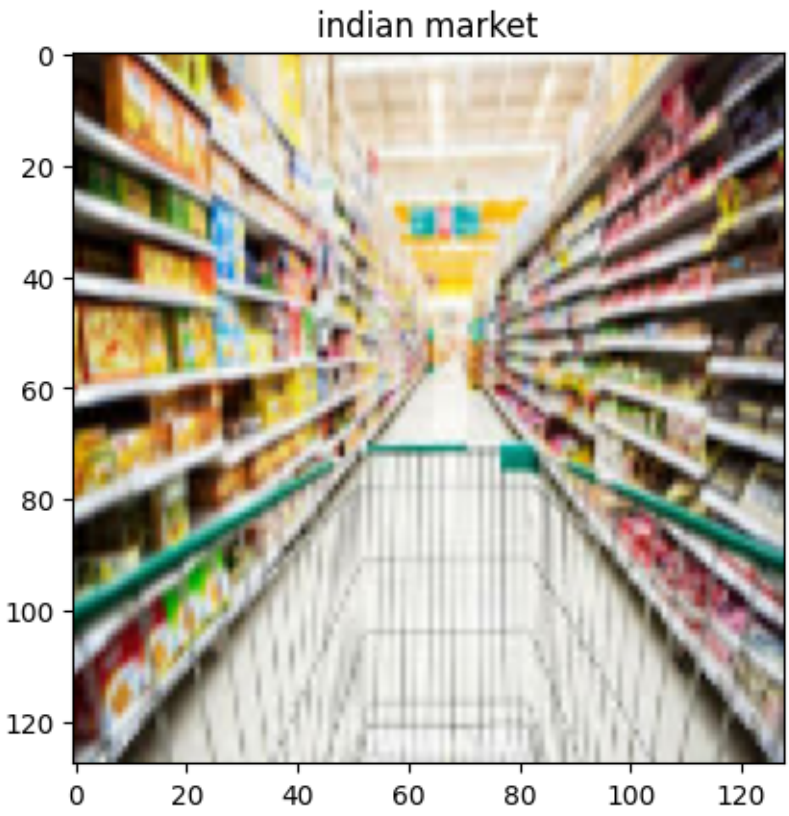

tomato
0
20
40
60
80
100
120
0
20
40
60
80
100
120

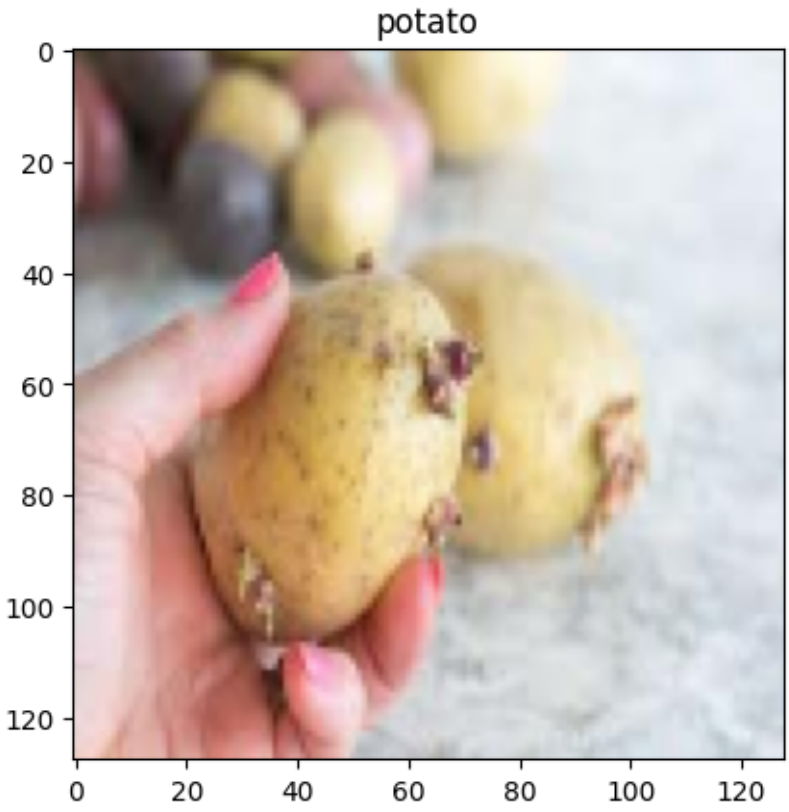

tomato
0
20
40
60
80
100
120
0
20
40
60
80
100
120

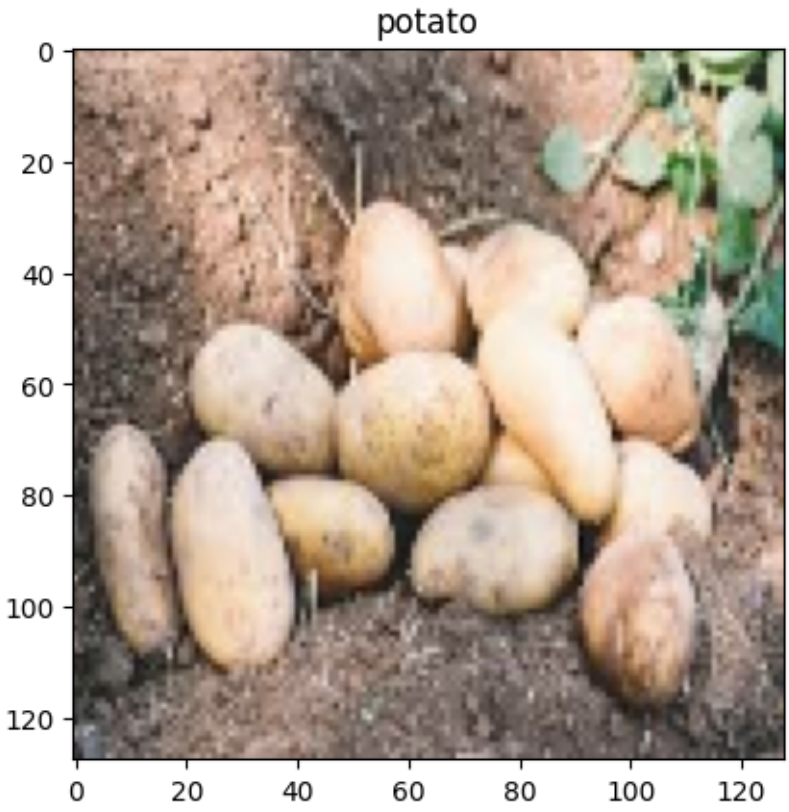

indian market
0
20
40
60
80
100
120
0
20
40
60
80
100
120

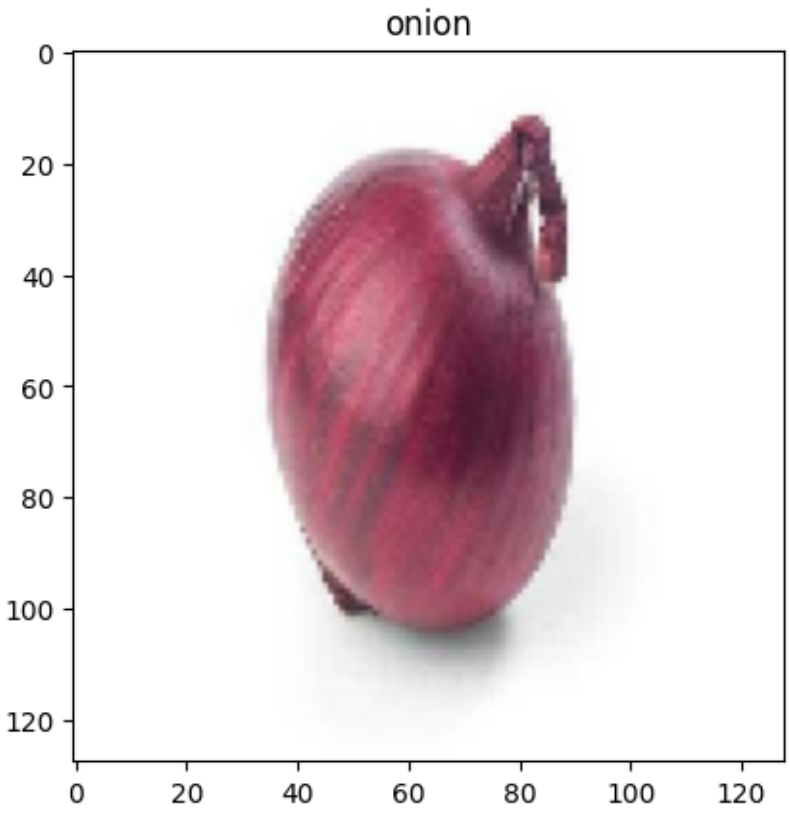

potato
0
20
40
60
80
100
120
0
20
40
60
80
100
120


potato
0
20
40
60
80
100
120
0
20
40
60
80
100
120


indian market
0
20
40
60
80
100
120
0
20
40
60
80
100
120


potato
0
20
40
60
80
100
120
0
20
40
60
80
100
120


potato
0
20
40
60
80
100
120
0
20
40
60
80
100
120



8 Scratch Implementation of CNN Model

[27]:
```
from tensorflow.keras.layers import␣
↪Conv2D,Dropout,BatchNormalization,MaxPooling2D,GlobalAveragePooling2D, Dense
from tensorflow.keras import regularizers
```

[28]:
```
input_shape = (128, 128, 3)
model = Sequential([

  ␣
↪Conv2D(filters=16,kernel_size=3,activation='relu',padding="same",input_shape=input_shape),
    Conv2D(filters=16,kernel_size=3,activation='relu'),
    Dropout(0.2),

  ␣
↪Conv2D(filters=32,kernel_size=3,activation='relu',padding="same",kernel_regularizer=regular
↪L2(1e-4)),
```

```
  ␣
↪Conv2D(filters=32,kernel_size=3,activation='relu',padding="same",kernel_regularizer=regular
↪L2(1e-4)),
   MaxPooling2D(),
   Dropout(0.2),

  ␣
↪Conv2D(filters=64,kernel_size=3,activation='relu',padding="same",kernel_regularizer=regular
↪L2(1e-4)),
  ␣
↪Conv2D(filters=64,kernel_size=3,activation='relu',padding="same",kernel_regularizer=regular
↪L2(1e-4)),
   MaxPooling2D(),
   Dropout(0.3),

  ␣
↪Conv2D(filters=128,kernel_size=3,activation='relu',padding="same",kernel_regularizer=regula
↪L2(1e-4)),
  ␣
↪Conv2D(filters=128,kernel_size=3,activation='relu',padding="same",kernel_regularizer=regula
↪L2(1e-4)),
   MaxPooling2D(),
   Dropout(0.2),

  ␣
↪Conv2D(filters=256,kernel_size=3,activation='relu',padding="same",kernel_regularizer=regula
↪L2(1e-4)),
  ␣
↪Conv2D(filters=256,kernel_size=3,activation='relu',padding="same",kernel_regularizer=regula
↪L2(1e-4)),
   MaxPooling2D(),
   Dropout(0.2),

  ␣
↪Conv2D(filters=512,kernel_size=3,activation='relu',padding="same",kernel_regularizer=regula
↪L2(1e-4)),
  ␣
↪Conv2D(filters=512,kernel_size=3,activation='relu',padding="same",kernel_regularizer=regula
↪L2(1e-4)),
   MaxPooling2D(),
   Dropout(0.2),

   GlobalAveragePooling2D(),

   Dense(units=4,activation='softmax')
```

```
])
```

[29]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 16)      448

 conv2d_1 (Conv2D)           (None, 126, 126, 16)      2320

 dropout (Dropout)           (None, 126, 126, 16)      0

 conv2d_2 (Conv2D)           (None, 126, 126, 32)      4640

 conv2d_3 (Conv2D)           (None, 126, 126, 32)      9248

 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
 )

 dropout_1 (Dropout)         (None, 63, 63, 32)        0

 conv2d_4 (Conv2D)           (None, 63, 63, 64)        18496

 conv2d_5 (Conv2D)           (None, 63, 63, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 31, 31, 64)       0
 2D)

 dropout_2 (Dropout)         (None, 31, 31, 64)        0

 conv2d_6 (Conv2D)           (None, 31, 31, 128)       73856

 conv2d_7 (Conv2D)           (None, 31, 31, 128)       147584

 max_pooling2d_2 (MaxPooling  (None, 15, 15, 128)      0
 2D)

 dropout_3 (Dropout)         (None, 15, 15, 128)       0

 conv2d_8 (Conv2D)           (None, 15, 15, 256)       295168

 conv2d_9 (Conv2D)           (None, 15, 15, 256)       590080

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 256)        0
 2D)
```

```
 dropout_4 (Dropout)         (None, 7, 7, 256)         0

 conv2d_10 (Conv2D)          (None, 7, 7, 512)         1180160

 conv2d_11 (Conv2D)          (None, 7, 7, 512)         2359808

 max_pooling2d_4 (MaxPooling  (None, 3, 3, 512)        0
 2D)

 dropout_5 (Dropout)         (None, 3, 3, 512)         0

 global_average_pooling2d (G  (None, 512)              0
 lobalAveragePooling2D)

 dense (Dense)               (None, 4)                 2052

=================================================================
Total params: 4,720,788
Trainable params: 4,720,788
Non-trainable params: 0
_________________________________________________________________
```

[30]:
```
from tensorflow.keras.callbacks import EarlyStopping,ModelCheckpoint
```

[31]:
```
model.compile(optimizer='adam',
                loss='sparse_categorical_crossentropy',
                metrics=['accuracy'])
filepath = "C://Users//gaura//Documents//GitHub//Scaler-Projects//Computer␣
 ↪Vision//Project//Models//"
callbacks = [
        tf.keras.callbacks.ReduceLROnPlateau(
            monitor="val_loss", factor=0.3, patience=3, min_lr=0.00001
        ),
        tf.keras.callbacks.ModelCheckpoint(filepath, monitor='val_accuracy',␣
 ↪mode='max', save_best_only=True),
        tf.keras.callbacks.EarlyStopping(
            monitor="val_loss", patience=3, min_delta=0.001, mode='min'
        )
    ]
```

[32]:
```
tf.keras.backend.clear_session()
```

[33]:
```
hist = model.
 ↪fit(train_ds,validation_data=val_data,verbose=1,epochs=60,batch_size=32,callbacks=callbacks
```

```
Epoch 1/60
```

```
34/34 [==============================] - ETA: 0s - loss: 1.5214 - accuracy:
0.2602

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 48s 749ms/step - loss: 1.5214 -
accuracy: 0.2602 - val_loss: 1.4708 - val_accuracy: 0.2677 - lr: 0.0010
Epoch 2/60
34/34 [==============================] - 13s 322ms/step - loss: 1.4558 -
accuracy: 0.2717 - val_loss: 1.4456 - val_accuracy: 0.2646 - lr: 0.0010
Epoch 3/60
34/34 [==============================] - ETA: 0s - loss: 1.4217 - accuracy:
0.3149

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 16s 419ms/step - loss: 1.4217 -
accuracy: 0.3149 - val_loss: 1.3761 - val_accuracy: 0.4187 - lr: 0.0010
Epoch 4/60
34/34 [==============================] - ETA: 0s - loss: 1.1786 - accuracy:
0.4782

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 16s 427ms/step - loss: 1.1786 -
accuracy: 0.4782 - val_loss: 0.9694 - val_accuracy: 0.5344 - lr: 0.0010
Epoch 5/60
```

```
34/34 [==============================] - ETA: 0s - loss: 0.9562 - accuracy:
0.5375

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 16s 434ms/step - loss: 0.9562 -
accuracy: 0.5375 - val_loss: 1.0074 - val_accuracy: 0.5698 - lr: 0.0010
Epoch 6/60
34/34 [==============================] - ETA: 0s - loss: 0.8803 - accuracy:
0.6106

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 16s 423ms/step - loss: 0.8803 -
accuracy: 0.6106 - val_loss: 0.8137 - val_accuracy: 0.6396 - lr: 0.0010
Epoch 7/60
34/34 [==============================] - 12s 323ms/step - loss: 0.8731 -
accuracy: 0.6078 - val_loss: 0.8669 - val_accuracy: 0.5948 - lr: 0.0010
Epoch 8/60
34/34 [==============================] - ETA: 0s - loss: 0.7927 - accuracy:
0.6317

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 15s 424ms/step - loss: 0.7927 -
accuracy: 0.6317 - val_loss: 0.7682 - val_accuracy: 0.6438 - lr: 0.0010
Epoch 9/60
```

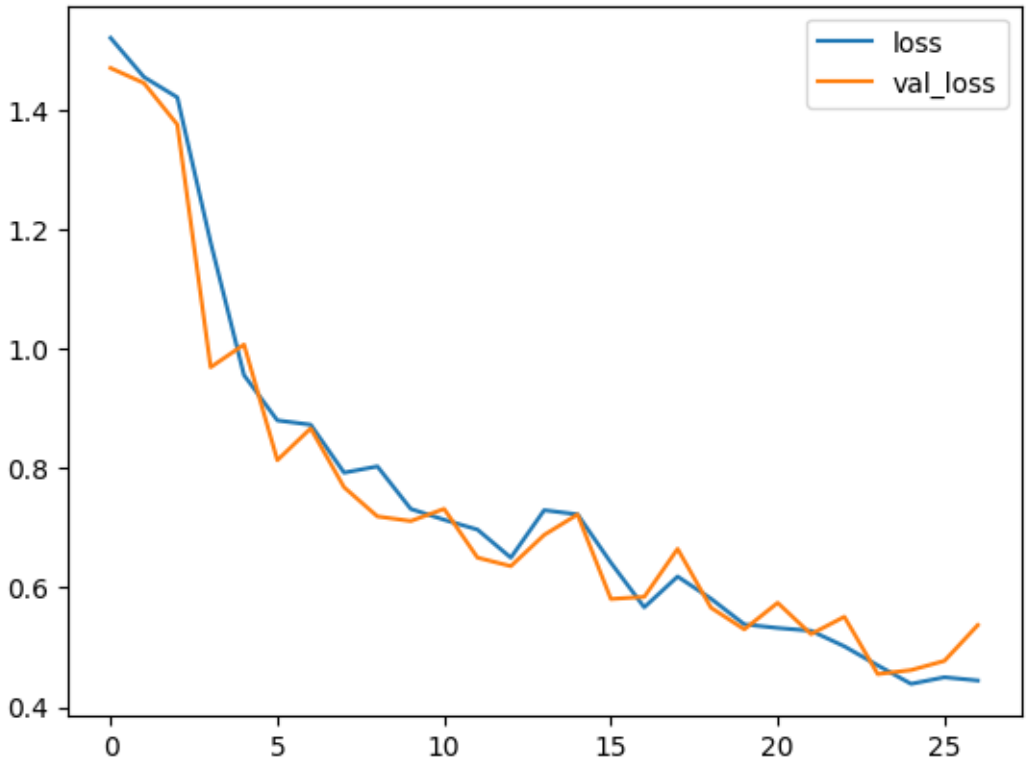
```
34/34 [==============================] - ETA: 0s - loss: 0.8032 - accuracy:
0.6487

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 15s 413ms/step - loss: 0.8032 -
accuracy: 0.6487 - val_loss: 0.7193 - val_accuracy: 0.6865 - lr: 0.0010
Epoch 10/60
34/34 [==============================] - 12s 318ms/step - loss: 0.7319 -
accuracy: 0.6874 - val_loss: 0.7117 - val_accuracy: 0.6750 - lr: 0.0010
Epoch 11/60
34/34 [==============================] - 12s 319ms/step - loss: 0.7138 -
accuracy: 0.6947 - val_loss: 0.7320 - val_accuracy: 0.6635 - lr: 0.0010
Epoch 12/60
34/34 [==============================] - ETA: 0s - loss: 0.6973 - accuracy:
0.7140

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 15s 422ms/step - loss: 0.6973 -
accuracy: 0.7140 - val_loss: 0.6503 - val_accuracy: 0.7240 - lr: 0.0010
Epoch 13/60
34/34 [==============================] - 12s 321ms/step - loss: 0.6503 -
accuracy: 0.7361 - val_loss: 0.6361 - val_accuracy: 0.7146 - lr: 0.0010
Epoch 14/60
34/34 [==============================] - ETA: 0s - loss: 0.7298 - accuracy:
0.6791

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets
```

```
INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 16s 445ms/step - loss: 0.7298 -
accuracy: 0.6791 - val_loss: 0.6884 - val_accuracy: 0.7302 - lr: 0.0010
Epoch 15/60
34/34 [==============================] - 12s 325ms/step - loss: 0.7228 -
accuracy: 0.6952 - val_loss: 0.7227 - val_accuracy: 0.6573 - lr: 0.0010
Epoch 16/60
34/34 [==============================] - ETA: 0s - loss: 0.6418 - accuracy:
0.7434

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 16s 424ms/step - loss: 0.6418 -
accuracy: 0.7434 - val_loss: 0.5810 - val_accuracy: 0.7688 - lr: 0.0010
Epoch 17/60
34/34 [==============================] - 12s 322ms/step - loss: 0.5671 -
accuracy: 0.7770 - val_loss: 0.5846 - val_accuracy: 0.7688 - lr: 0.0010
Epoch 18/60
34/34 [==============================] - 12s 321ms/step - loss: 0.6186 -
accuracy: 0.7655 - val_loss: 0.6650 - val_accuracy: 0.7115 - lr: 0.0010
Epoch 19/60
34/34 [==============================] - 12s 319ms/step - loss: 0.5816 -
accuracy: 0.7623 - val_loss: 0.5665 - val_accuracy: 0.7656 - lr: 0.0010
Epoch 20/60
34/34 [==============================] - ETA: 0s - loss: 0.5385 - accuracy:
0.7926

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 16s 428ms/step - loss: 0.5385 -
accuracy: 0.7926 - val_loss: 0.5301 - val_accuracy: 0.7865 - lr: 0.0010
Epoch 21/60
```

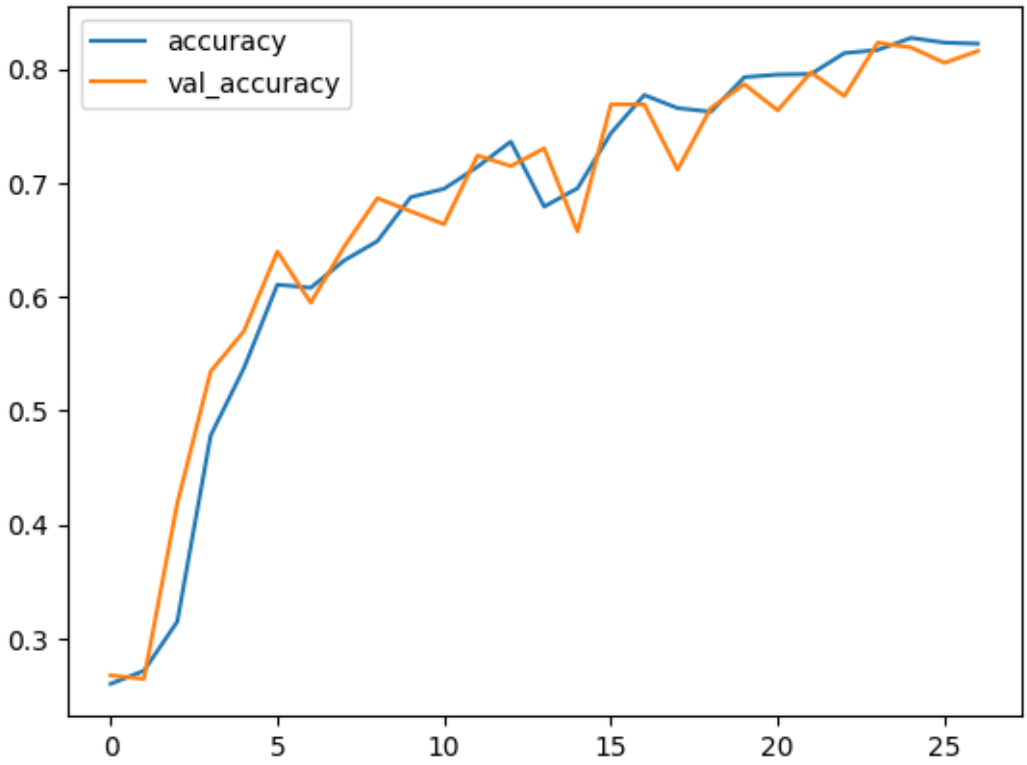
```
34/34 [==============================] - 13s 325ms/step - loss: 0.5324 -
accuracy: 0.7949 - val_loss: 0.5746 - val_accuracy: 0.7635 - lr: 0.0010
Epoch 22/60
34/34 [==============================] - ETA: 0s - loss: 0.5277 - accuracy:
0.7954

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 16s 436ms/step - loss: 0.5277 -
accuracy: 0.7954 - val_loss: 0.5222 - val_accuracy: 0.7969 - lr: 0.0010
Epoch 23/60
34/34 [==============================] - 12s 324ms/step - loss: 0.5017 -
accuracy: 0.8138 - val_loss: 0.5512 - val_accuracy: 0.7760 - lr: 0.0010
Epoch 24/60
34/34 [==============================] - ETA: 0s - loss: 0.4697 - accuracy:
0.8166

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 12). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

INFO:tensorflow:Assets written to: C://Users//gaura//Documents//GitHub//Scaler-
Projects//Computer Vision//Project//Models\/assets

34/34 [==============================] - 15s 425ms/step - loss: 0.4697 -
accuracy: 0.8166 - val_loss: 0.4552 - val_accuracy: 0.8229 - lr: 0.0010
Epoch 25/60
34/34 [==============================] - 12s 324ms/step - loss: 0.4388 -
accuracy: 0.8271 - val_loss: 0.4619 - val_accuracy: 0.8188 - lr: 0.0010
Epoch 26/60
34/34 [==============================] - 12s 323ms/step - loss: 0.4498 -
accuracy: 0.8230 - val_loss: 0.4773 - val_accuracy: 0.8052 - lr: 0.0010
Epoch 27/60
34/34 [==============================] - 12s 323ms/step - loss: 0.4444 -
accuracy: 0.8221 - val_loss: 0.5372 - val_accuracy: 0.8156 - lr: 0.0010
```

[34]:
```
plt.plot(hist.history['loss'],label='loss')
plt.plot(hist.history['val_loss'],label='val_loss')
```

```
plt.legend()
plt.show()
```



[35]:
```
plt.plot(hist.history['accuracy'],label='accuracy')
plt.plot(hist.history['val_accuracy'],label='val_accuracy')
plt.legend()
plt.show()
```



[36]:
```
load_model = tf.keras.models.load_model('C://Users//gaura//Documents//GitHub//
↪Scaler-Projects//Computer Vision//Project//Models')
```

[37]:
```
# Evaluate the model
loss, acc = load_model.evaluate(test_data, verbose=2)
print("Restored model, accuracy: {:5.2f}%".format(100 * acc))
```

```
6/6 - 4s - loss: 0.5625 - accuracy: 0.7521 - 4s/epoch - 600ms/step
Restored model, accuracy: 75.21%
```

[38]:
```
y_pred = load_model.predict(test_data)
```

```
6/6 [==============================] - 1s 55ms/step
```

[39]:
```
pred = np.argmax(y_pred,axis=1)
```

[40]:
```
ytest = []
for img,label in test_data:
    labels = label.numpy()
    ytest.extend(labels)
```

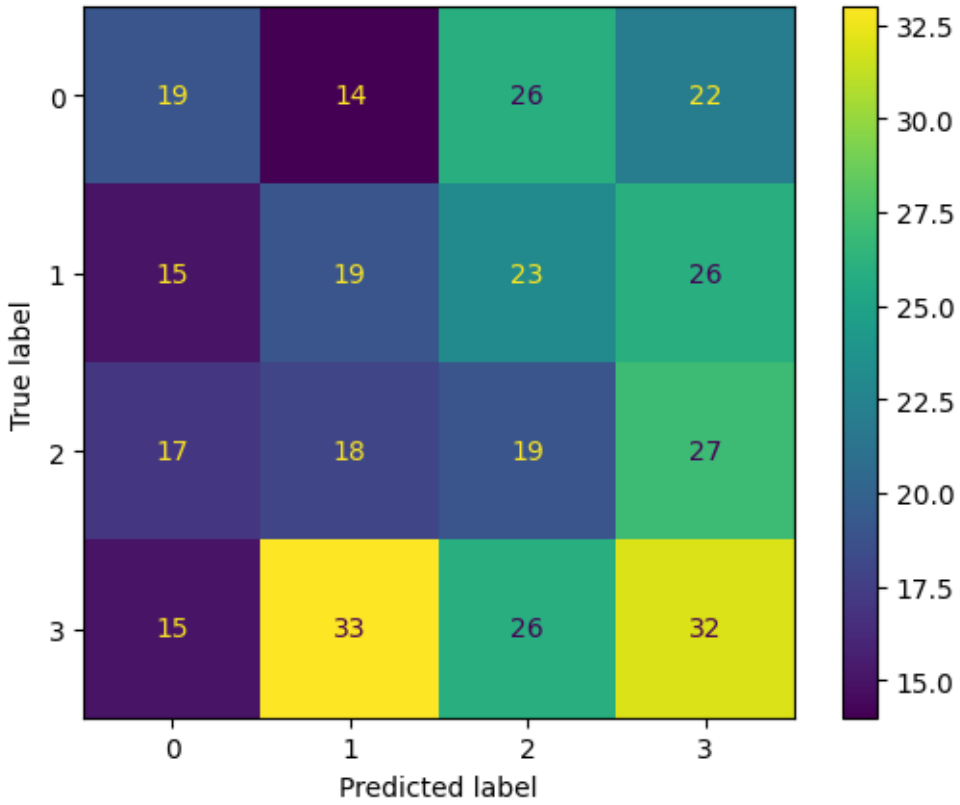
[41]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
↪classification_report
```

[42]:
```
print(classification_report(ytest,pred))
```

```
              precision    recall  f1-score   support

           0       0.29      0.23      0.26        81
           1       0.23      0.23      0.23        83
           2       0.20      0.23      0.22        81
           3       0.30      0.30      0.30       106

    accuracy                           0.25       351
   macro avg       0.25      0.25      0.25       351
weighted avg       0.26      0.25      0.25       351
```

[43]:
```
ConfusionMatrixDisplay(confusion_matrix(ytest,pred)).plot()
```

[43]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x225772bfeb0>
```



[44]:
```
images,label = next(iter(test_data))
images = images.numpy()
label = label.numpy()
```

[45]:
```
reference_mapping_label = {
    2:'potato',
    3:'tomato',
    0:'indian market',
    1:'onion'
}
```

[46]:
```
CLASS_NAMES = ['indian market','onion','potato','tomato']
```

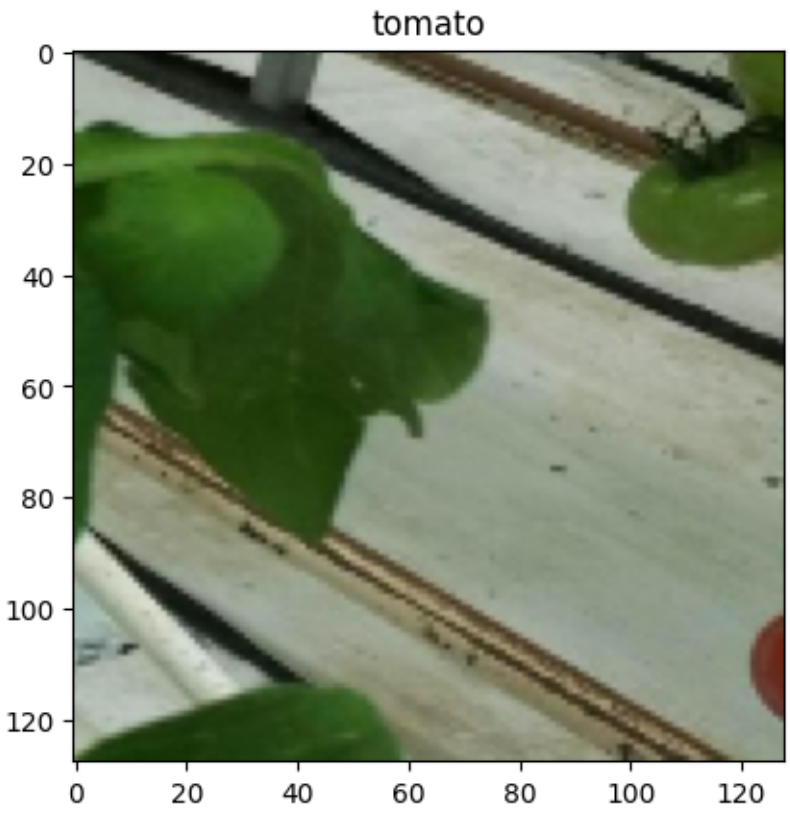
[47]:
```
for image, label in test_data.take(2):
      plt.imshow(image[2])
      plt.title(CLASS_NAMES[label[2].numpy()])
      plt.axis('off')
prediction = load_model.predict(test_data.take(2))[2]
prediction = np.argmax(prediction)
#print(prediction)
scores = [1 - prediction, prediction]
for score, name in zip(scores, CLASS_NAMES):
    print("This image is %.2f percent %s" % ((100 * score), name))
%time
```

```
2/2 [==============================] - 0s 149ms/step
This image is -200.00 percent indian market
This image is 300.00 percent onion
CPU times: total: 0 ns
Wall time: 0 ns
```

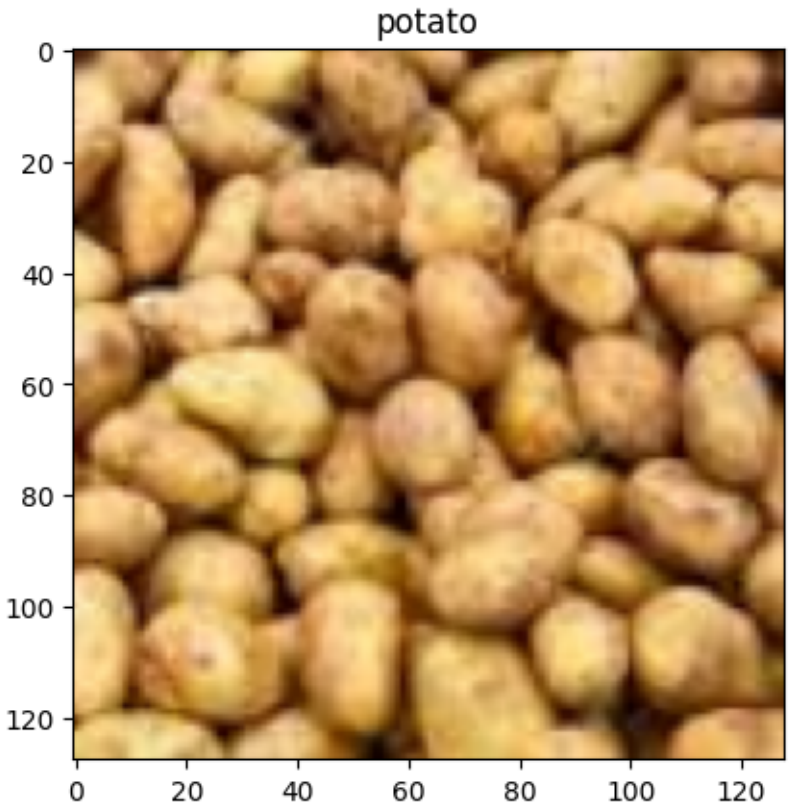
onion

[48]:
```
for img in images[:5]:
    #print(img.shape)
    temp = (np.expand_dims(img,axis=0))
    pred = load_model.predict(temp)
    labels = reference_mapping_label[np.argmax(pred)]
    plt.imshow(img)
    plt.title(labels)
    plt.show()
```

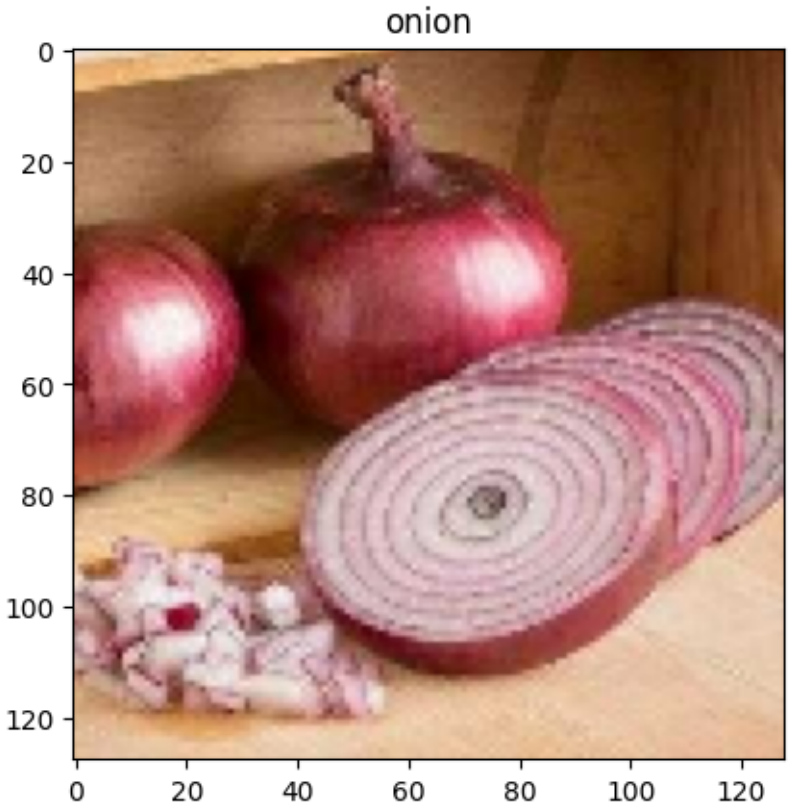
```
1/1 [==============================] - 1s 1s/step
```



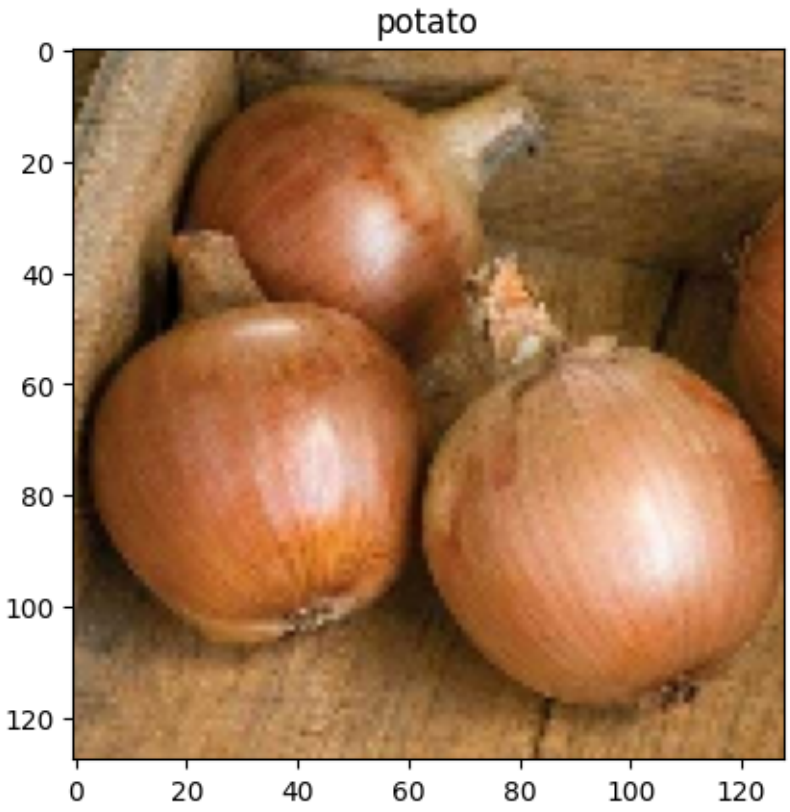

```
1/1 [==============================] - 0s 34ms/step
```



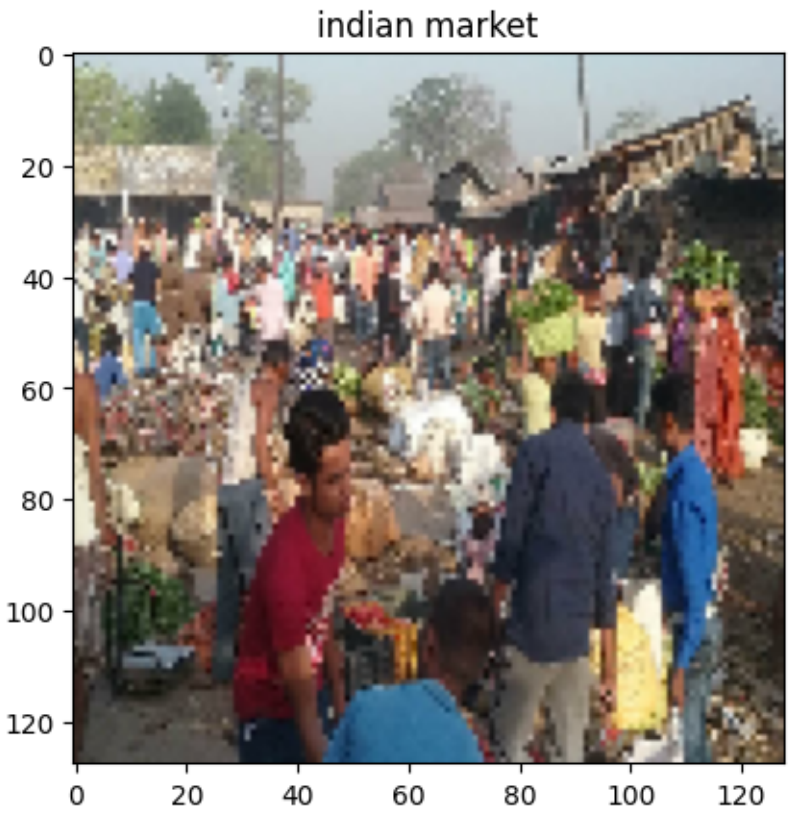

```
1/1 [==============================] - 0s 31ms/step
```




```
1/1 [==============================] - 0s 40ms/step
```




```
1/1 [==============================] - 0s 40ms/step
```



9 Applying Transfer Learning Method

[49]:
```
base_model = tf.keras.applications.
  ↪ResNet50(input_shape=(128,128,3),include_top=False,weights='imagenet')

for layers in base_model.layers[-1:-1000:-1]:
    #print(layers,layers.trainable)
    layers.trainable = False

inputs = tf.keras.Input(shape=(128,128,3))
x = base_model(inputs,training=False)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
outputs = tf.keras.layers.Dense(4,activation='softmax')(x)

n_model = tf.keras.Model(inputs,outputs)
```

[50]:
```
# for layers in base_model.layers:
#     if(layers.trainable == False):
#         print(layers,layers.trainable)
```

[51]:
```
# for layers in base_model.layers:
#     print(layers,layers.trainable)
```

[52]:
```
n_model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 128, 128, 3)]     0

 resnet50 (Functional)       (None, 4, 4, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 4)                 8196

=================================================================
Total params: 23,595,908
Trainable params: 8,196
Non-trainable params: 23,587,712
_________________________________________________________________
```

[53]:
```
n_model.compile(optimizer=tf.keras.optimizers.Adam(),
                loss='sparse_categorical_crossentropy',
                metrics=['accuracy'])

filepath = "C://Users//gaura//Documents//GitHub//Scaler-Projects//Computer␣
 ↪Vision//Project//TL_MODEL"
callbacks = [
        tf.keras.callbacks.ReduceLROnPlateau(
            monitor="val_loss", factor=0.3, patience=5, min_lr=0.00001
        ),
        tf.keras.callbacks.ModelCheckpoint(filepath, save_weights_only=True,␣
 ↪monitor='val_accuracy', mode='max', save_best_only=True),
        tf.keras.callbacks.EarlyStopping(
            monitor="val_loss", patience=10, min_delta=0.001, mode='min'
        )
    ]
```

[54]:
```
tf.keras.backend.clear_session()
```

[55]:
```
t_hist = n_model.
 ↪fit(train_ds,validation_data=val_data,verbose=1,epochs=100,batch_size=32,callbacks=callback
```

```
Epoch 1/100
```

```
34/34 [==============================] - 14s 247ms/step - loss: 1.3703 -
accuracy: 0.2855 - val_loss: 1.3536 - val_accuracy: 0.2865 - lr: 0.0010
Epoch 2/100
34/34 [==============================] - 8s 210ms/step - loss: 1.3423 -
accuracy: 0.3131 - val_loss: 1.3322 - val_accuracy: 0.3729 - lr: 0.0010
Epoch 3/100
34/34 [==============================] - 7s 186ms/step - loss: 1.3207 -
accuracy: 0.3756 - val_loss: 1.3173 - val_accuracy: 0.3646 - lr: 0.0010
Epoch 4/100
34/34 [==============================] - 8s 196ms/step - loss: 1.3010 -
accuracy: 0.3825 - val_loss: 1.3040 - val_accuracy: 0.3625 - lr: 0.0010
Epoch 5/100
34/34 [==============================] - 9s 218ms/step - loss: 1.2878 -
accuracy: 0.4014 - val_loss: 1.2867 - val_accuracy: 0.3760 - lr: 0.0010
Epoch 6/100
34/34 [==============================] - 9s 224ms/step - loss: 1.2731 -
accuracy: 0.4230 - val_loss: 1.2723 - val_accuracy: 0.3833 - lr: 0.0010
Epoch 7/100
34/34 [==============================] - 8s 214ms/step - loss: 1.2589 -
accuracy: 0.4299 - val_loss: 1.2561 - val_accuracy: 0.4573 - lr: 0.0010
Epoch 8/100
34/34 [==============================] - 7s 185ms/step - loss: 1.2417 -
accuracy: 0.4667 - val_loss: 1.2540 - val_accuracy: 0.4354 - lr: 0.0010
Epoch 9/100
34/34 [==============================] - 8s 196ms/step - loss: 1.2317 -
accuracy: 0.4749 - val_loss: 1.2362 - val_accuracy: 0.4490 - lr: 0.0010
Epoch 10/100
34/34 [==============================] - 8s 197ms/step - loss: 1.2273 -
accuracy: 0.4464 - val_loss: 1.2383 - val_accuracy: 0.4292 - lr: 0.0010
Epoch 11/100
34/34 [==============================] - 8s 195ms/step - loss: 1.2122 -
accuracy: 0.4892 - val_loss: 1.2097 - val_accuracy: 0.4521 - lr: 0.0010
Epoch 12/100
34/34 [==============================] - 9s 227ms/step - loss: 1.2027 -
accuracy: 0.5076 - val_loss: 1.1983 - val_accuracy: 0.4833 - lr: 0.0010
Epoch 13/100
34/34 [==============================] - 8s 190ms/step - loss: 1.1924 -
accuracy: 0.5053 - val_loss: 1.2043 - val_accuracy: 0.4708 - lr: 0.0010
Epoch 14/100
34/34 [==============================] - 12s 321ms/step - loss: 1.1797 -
accuracy: 0.5159 - val_loss: 1.1924 - val_accuracy: 0.4875 - lr: 0.0010
Epoch 15/100
34/34 [==============================] - 8s 180ms/step - loss: 1.1777 -
accuracy: 0.5076 - val_loss: 1.1900 - val_accuracy: 0.4812 - lr: 0.0010
Epoch 16/100
34/34 [==============================] - 9s 223ms/step - loss: 1.1714 -
accuracy: 0.4970 - val_loss: 1.1774 - val_accuracy: 0.5052 - lr: 0.0010
Epoch 17/100
```

```
34/34 [==============================] - 8s 216ms/step - loss: 1.1619 -
accuracy: 0.5168 - val_loss: 1.1684 - val_accuracy: 0.5104 - lr: 0.0010
Epoch 18/100
34/34 [==============================] - 8s 195ms/step - loss: 1.1542 -
accuracy: 0.5333 - val_loss: 1.1629 - val_accuracy: 0.5000 - lr: 0.0010
Epoch 19/100
34/34 [==============================] - 8s 196ms/step - loss: 1.1478 -
accuracy: 0.5255 - val_loss: 1.1693 - val_accuracy: 0.4792 - lr: 0.0010
Epoch 20/100
34/34 [==============================] - 8s 193ms/step - loss: 1.1366 -
accuracy: 0.5343 - val_loss: 1.1596 - val_accuracy: 0.4927 - lr: 0.0010
Epoch 21/100
34/34 [==============================] - 9s 223ms/step - loss: 1.1295 -
accuracy: 0.5434 - val_loss: 1.1515 - val_accuracy: 0.5260 - lr: 0.0010
Epoch 22/100
34/34 [==============================] - 8s 214ms/step - loss: 1.1251 -
accuracy: 0.5554 - val_loss: 1.1380 - val_accuracy: 0.5281 - lr: 0.0010
Epoch 23/100
34/34 [==============================] - 8s 189ms/step - loss: 1.1263 -
accuracy: 0.5490 - val_loss: 1.1341 - val_accuracy: 0.5156 - lr: 0.0010
Epoch 24/100
34/34 [==============================] - 8s 195ms/step - loss: 1.1171 -
accuracy: 0.5586 - val_loss: 1.1309 - val_accuracy: 0.5167 - lr: 0.0010
Epoch 25/100
34/34 [==============================] - 8s 193ms/step - loss: 1.1096 -
accuracy: 0.5582 - val_loss: 1.1187 - val_accuracy: 0.5188 - lr: 0.0010
Epoch 26/100
34/34 [==============================] - 9s 226ms/step - loss: 1.1067 -
accuracy: 0.5453 - val_loss: 1.1226 - val_accuracy: 0.5375 - lr: 0.0010
Epoch 27/100
34/34 [==============================] - 8s 189ms/step - loss: 1.0959 -
accuracy: 0.5683 - val_loss: 1.1190 - val_accuracy: 0.5292 - lr: 0.0010
Epoch 28/100
34/34 [==============================] - 8s 196ms/step - loss: 1.1077 -
accuracy: 0.5379 - val_loss: 1.1167 - val_accuracy: 0.5271 - lr: 0.0010
Epoch 29/100
34/34 [==============================] - 8s 197ms/step - loss: 1.0888 -
accuracy: 0.5678 - val_loss: 1.1130 - val_accuracy: 0.5281 - lr: 0.0010
Epoch 30/100
34/34 [==============================] - 8s 208ms/step - loss: 1.0835 -
accuracy: 0.5623 - val_loss: 1.1104 - val_accuracy: 0.5198 - lr: 0.0010
Epoch 31/100
34/34 [==============================] - 8s 199ms/step - loss: 1.0836 -
accuracy: 0.5724 - val_loss: 1.1021 - val_accuracy: 0.5344 - lr: 0.0010
Epoch 32/100
34/34 [==============================] - 8s 204ms/step - loss: 1.0717 -
accuracy: 0.5807 - val_loss: 1.1107 - val_accuracy: 0.5177 - lr: 0.0010
Epoch 33/100
```

```
34/34 [==============================] - 9s 224ms/step - loss: 1.0689 -
accuracy: 0.5747 - val_loss: 1.0952 - val_accuracy: 0.5500 - lr: 0.0010
Epoch 34/100
34/34 [==============================] - 8s 204ms/step - loss: 1.0672 -
accuracy: 0.5802 - val_loss: 1.0988 - val_accuracy: 0.5312 - lr: 0.0010
Epoch 35/100
34/34 [==============================] - 9s 232ms/step - loss: 1.0651 -
accuracy: 0.5775 - val_loss: 1.0749 - val_accuracy: 0.5531 - lr: 0.0010
Epoch 36/100
34/34 [==============================] - 9s 215ms/step - loss: 1.0631 -
accuracy: 0.5816 - val_loss: 1.0812 - val_accuracy: 0.5396 - lr: 0.0010
Epoch 37/100
34/34 [==============================] - 9s 223ms/step - loss: 1.0589 -
accuracy: 0.5743 - val_loss: 1.0912 - val_accuracy: 0.5312 - lr: 0.0010
Epoch 38/100
34/34 [==============================] - 6s 148ms/step - loss: 1.0494 -
accuracy: 0.5793 - val_loss: 1.0860 - val_accuracy: 0.5375 - lr: 0.0010
Epoch 39/100
34/34 [==============================] - 6s 144ms/step - loss: 1.0506 -
accuracy: 0.5802 - val_loss: 1.0759 - val_accuracy: 0.5500 - lr: 0.0010
Epoch 40/100
34/34 [==============================] - 7s 166ms/step - loss: 1.0424 -
accuracy: 0.5867 - val_loss: 1.0761 - val_accuracy: 0.5635 - lr: 0.0010
Epoch 41/100
34/34 [==============================] - 6s 136ms/step - loss: 1.0387 -
accuracy: 0.5940 - val_loss: 1.0767 - val_accuracy: 0.5469 - lr: 3.0000e-04
Epoch 42/100
34/34 [==============================] - 6s 132ms/step - loss: 1.0383 -
accuracy: 0.5954 - val_loss: 1.0771 - val_accuracy: 0.5479 - lr: 3.0000e-04
Epoch 43/100
34/34 [==============================] - 6s 135ms/step - loss: 1.0358 -
accuracy: 0.5991 - val_loss: 1.0655 - val_accuracy: 0.5573 - lr: 3.0000e-04
Epoch 44/100
34/34 [==============================] - 5s 133ms/step - loss: 1.0418 -
accuracy: 0.5913 - val_loss: 1.0743 - val_accuracy: 0.5490 - lr: 3.0000e-04
Epoch 45/100
34/34 [==============================] - 6s 148ms/step - loss: 1.0396 -
accuracy: 0.5903 - val_loss: 1.0703 - val_accuracy: 0.5708 - lr: 3.0000e-04
Epoch 46/100
34/34 [==============================] - 6s 138ms/step - loss: 1.0353 -
accuracy: 0.5977 - val_loss: 1.0683 - val_accuracy: 0.5510 - lr: 3.0000e-04
Epoch 47/100
34/34 [==============================] - 6s 152ms/step - loss: 1.0388 -
accuracy: 0.5908 - val_loss: 1.0564 - val_accuracy: 0.5792 - lr: 3.0000e-04
Epoch 48/100
34/34 [==============================] - 6s 136ms/step - loss: 1.0348 -
accuracy: 0.5949 - val_loss: 1.0747 - val_accuracy: 0.5437 - lr: 3.0000e-04
Epoch 49/100
```

```
34/34 [==============================] - 6s 134ms/step - loss: 1.0331 -
accuracy: 0.5926 - val_loss: 1.0637 - val_accuracy: 0.5583 - lr: 3.0000e-04
Epoch 50/100
34/34 [==============================] - 6s 135ms/step - loss: 1.0266 -
accuracy: 0.6014 - val_loss: 1.0674 - val_accuracy: 0.5698 - lr: 3.0000e-04
Epoch 51/100
34/34 [==============================] - 6s 133ms/step - loss: 1.0387 -
accuracy: 0.5899 - val_loss: 1.0623 - val_accuracy: 0.5625 - lr: 3.0000e-04
Epoch 52/100
34/34 [==============================] - 6s 137ms/step - loss: 1.0367 -
accuracy: 0.5931 - val_loss: 1.0768 - val_accuracy: 0.5510 - lr: 3.0000e-04
Epoch 53/100
34/34 [==============================] - 6s 141ms/step - loss: 1.0243 -
accuracy: 0.5982 - val_loss: 1.0484 - val_accuracy: 0.5781 - lr: 9.0000e-05
Epoch 54/100
34/34 [==============================] - 6s 142ms/step - loss: 1.0274 -
accuracy: 0.5926 - val_loss: 1.0583 - val_accuracy: 0.5635 - lr: 9.0000e-05
Epoch 55/100
34/34 [==============================] - 6s 134ms/step - loss: 1.0221 -
accuracy: 0.5986 - val_loss: 1.0615 - val_accuracy: 0.5615 - lr: 9.0000e-05
Epoch 56/100
34/34 [==============================] - 6s 132ms/step - loss: 1.0331 -
accuracy: 0.5917 - val_loss: 1.0691 - val_accuracy: 0.5594 - lr: 9.0000e-05
Epoch 57/100
34/34 [==============================] - 6s 134ms/step - loss: 1.0313 -
accuracy: 0.5926 - val_loss: 1.0703 - val_accuracy: 0.5531 - lr: 9.0000e-05
Epoch 58/100
34/34 [==============================] - 6s 136ms/step - loss: 1.0267 -
accuracy: 0.5945 - val_loss: 1.0573 - val_accuracy: 0.5698 - lr: 9.0000e-05
Epoch 59/100
34/34 [==============================] - 6s 138ms/step - loss: 1.0288 -
accuracy: 0.5926 - val_loss: 1.0700 - val_accuracy: 0.5500 - lr: 2.7000e-05
Epoch 60/100
34/34 [==============================] - 6s 132ms/step - loss: 1.0277 -
accuracy: 0.5963 - val_loss: 1.0611 - val_accuracy: 0.5635 - lr: 2.7000e-05
Epoch 61/100
34/34 [==============================] - 6s 136ms/step - loss: 1.0253 -
accuracy: 0.5926 - val_loss: 1.0610 - val_accuracy: 0.5615 - lr: 2.7000e-05
Epoch 62/100
34/34 [==============================] - 6s 135ms/step - loss: 1.0336 -
accuracy: 0.5899 - val_loss: 1.0579 - val_accuracy: 0.5625 - lr: 2.7000e-05
Epoch 63/100
34/34 [==============================] - 6s 134ms/step - loss: 1.0277 -
accuracy: 0.5995 - val_loss: 1.0613 - val_accuracy: 0.5615 - lr: 2.7000e-05
```

[64]:
```
# n_model.save('tl_model.pb')
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
```

```
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 53). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: tl_model.pb\assets

INFO:tensorflow:Assets written to: tl_model.pb\assets
```

[65]:
```
load_model = tf.keras.models.load_model("C://Users//gaura//Documents//GitHub//
  ↪Scaler-Projects//Computer Vision//Project//Notebook//tl_model.pb")
```

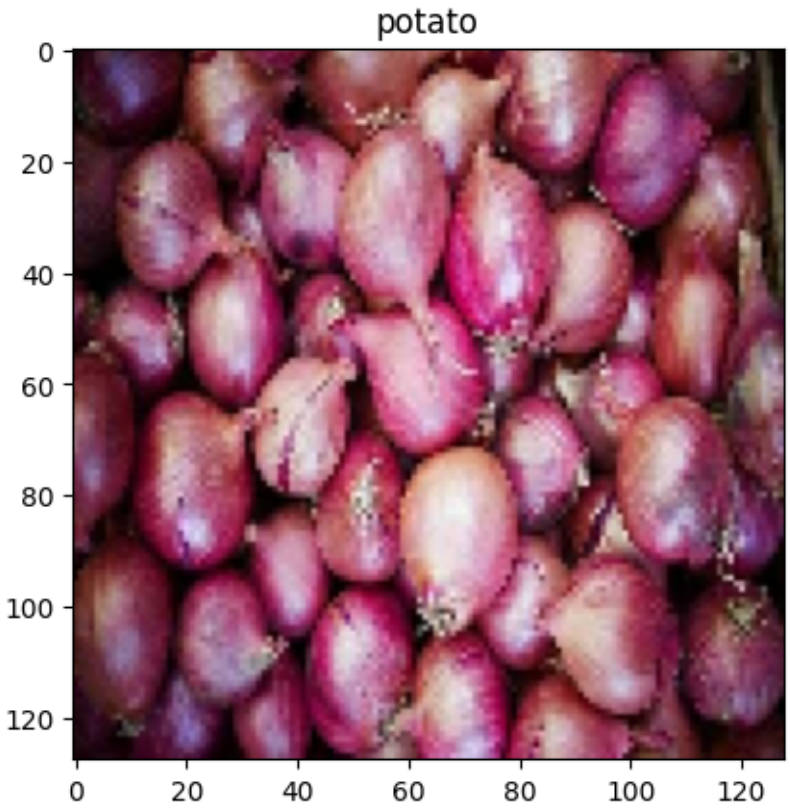
[66]:
```
# Evaluate the model
loss, acc = load_model.evaluate(test_data, verbose=2)
print("Restored model, accuracy: {:5.2f}%".format(100 * acc))
```

```
6/6 - 2s - loss: 1.1665 - accuracy: 0.4644 - 2s/epoch - 338ms/step
Restored model, accuracy: 46.44%
```

[67]:
```
images,label = next(iter(test_data))
```

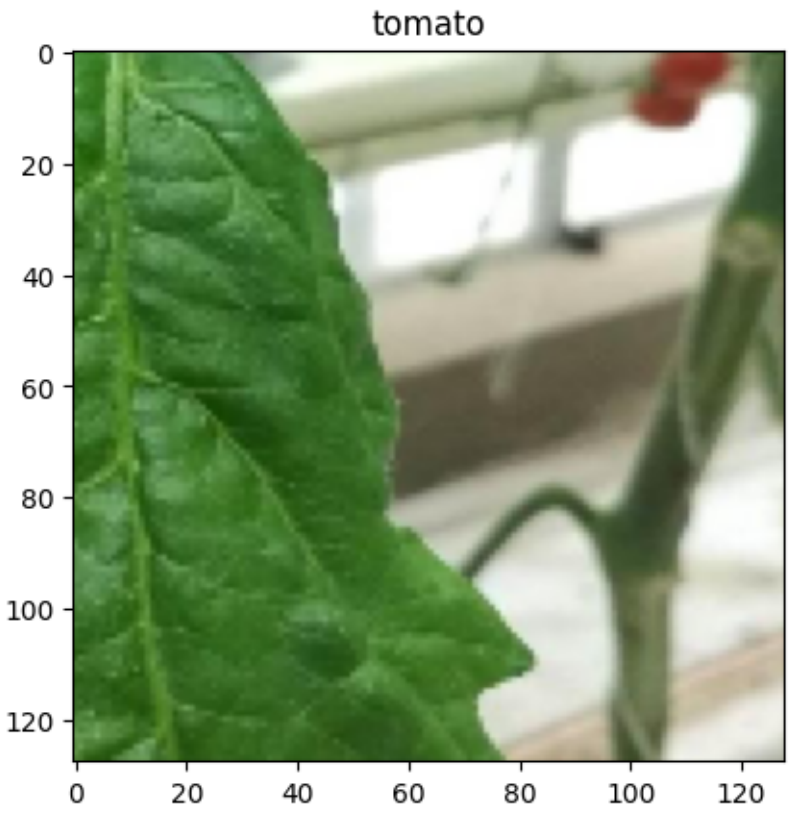
[68]:
```
for img in images.numpy()[:5]:
    #print(img.shape)
    temp = (np.expand_dims(img,axis=0))
    pred = load_model.predict(temp)
    labels = reference_mapping_label[np.argmax(pred)]
    plt.imshow(img)
    plt.title(labels)
    plt.show()
```

```
1/1 [==============================] - 1s 1s/step
```

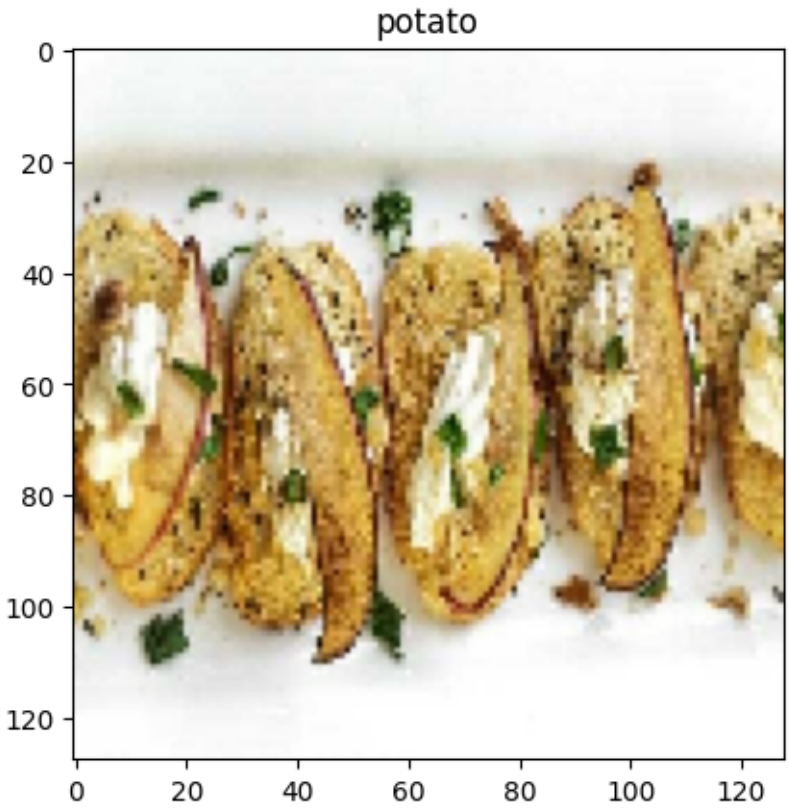



```
1/1 [==============================] - 0s 33ms/step
```


```
1/1 [==============================] - 0s 24ms/step
```

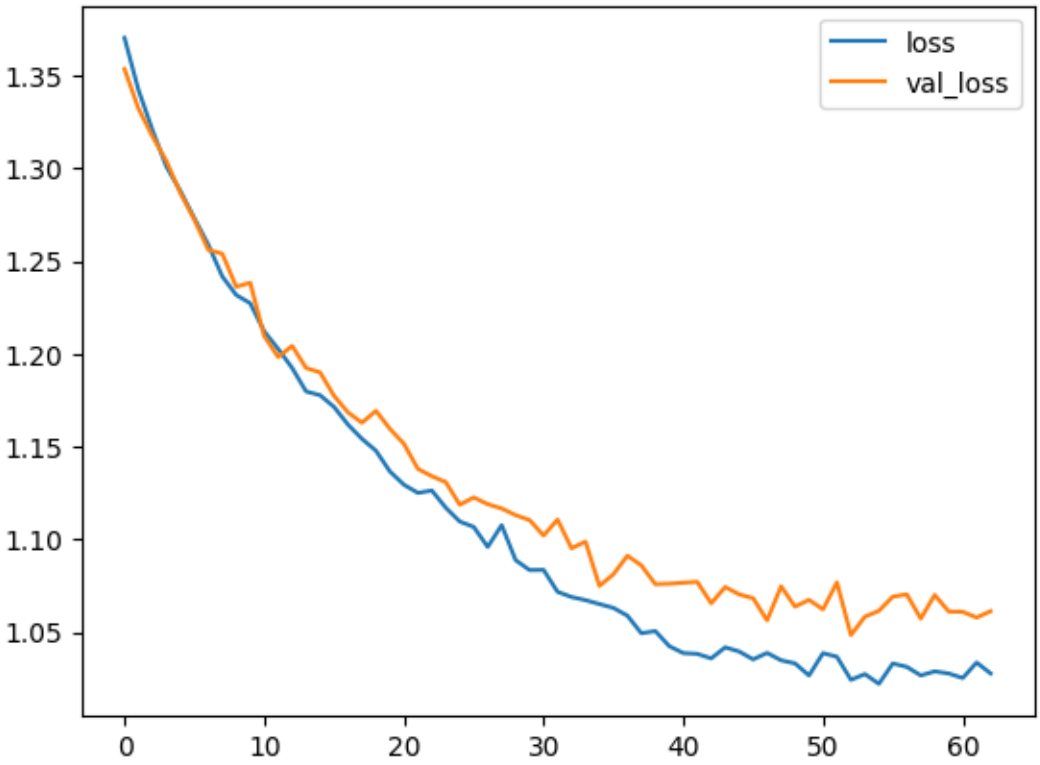



```
1/1 [==============================] - 0s 26ms/step
```

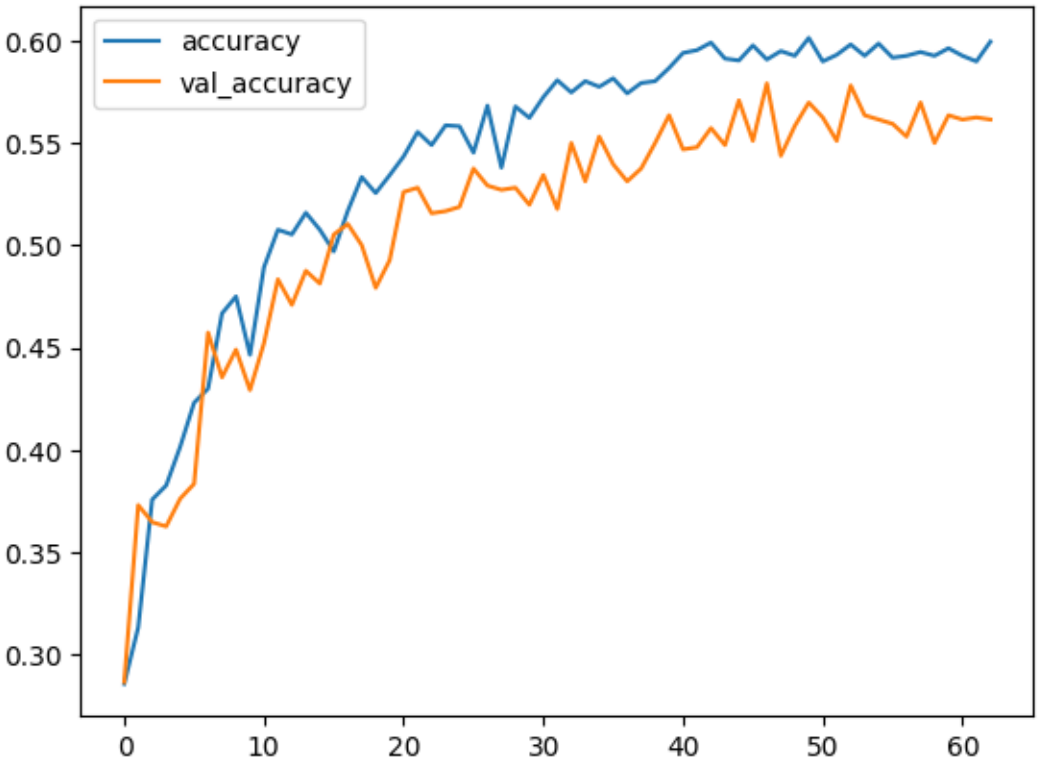



```
1/1 [==============================] - 0s 32ms/step
```

[69]:
```
plt.plot(t_hist.history['loss'],label='loss')
plt.plot(t_hist.history['val_loss'],label='val_loss')
plt.legend()
plt.show()
```



[70]:
```
plt.plot(t_hist.history['accuracy'],label='accuracy')
plt.plot(t_hist.history['val_accuracy'],label='val_accuracy')
plt.legend()
plt.show()
```



10 Summary of Model

1. Both the model have there own way of generalization.
2. CNN from scratch has achived accuracy of 75.21% but was unable to learn the params and end up with missclassification.
3. Using Transfer learning the model has achived accuracy of 46% accuracy on test but has less missclassification problems.

[ ]: