assignment9-2

April 15, 2021

[14]:
```
!unzip -qq /home/hiraditya/Desktop/HomeWork/SJSU/cs156/jupiter/JupyterBooks/
↪homework9_input_data.zip
```

```
replace __MACOSX/._flowers? [y]es, [n]o, [A]ll, [N]one, [r]ename: ^C
```

[15]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import os
import matplotlib.pyplot as plt
import pydot
from skimage import io
import numpy as np
import pandas as pd
import seaborn as sb
from sklearn.metrics import accuracy_score
```

[17]:
```
num_skipped = 0
for folder_name in ("daisy", "dandelion", "rose", "sunflower", "tulip"):
    folder_path = os.path.join("/home/hiraditya/Desktop/HomeWork/SJSU/cs156/
↪jupiter/JupyterBooks/flowers/training/", folder_name)
    for fname in os.listdir(folder_path):
        fpath = os.path.join(folder_path, fname)
        try:
            fobj = open(fpath, "rb")
            is_jfif = tf.compat.as_bytes("JFIF") in fobj.peek(10)
        finally:
            fobj.close()

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

print("Deleted %d images" % num_skipped)
```

```
Deleted 5 images
```

[18]:

```
num_skipped = 0
for folder_name in ("daisy", "dandelion", "rose", "sunflower", "tulip"):
    folder_path = os.path.join("/home/hiraditya/Desktop/HomeWork/SJSU/cs156/
 ↪jupiter/JupyterBooks/flowers/test/", folder_name)
    for fname in os.listdir(folder_path):
        fpath = os.path.join(folder_path, fname)
        try:
            fobj = open(fpath, "rb")
            is_jfif = tf.compat.as_bytes("JFIF") in fobj.peek(10)
        finally:
            fobj.close()

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

print("Deleted %d images" % num_skipped)
```

```
Deleted 0 images
```

[19]:

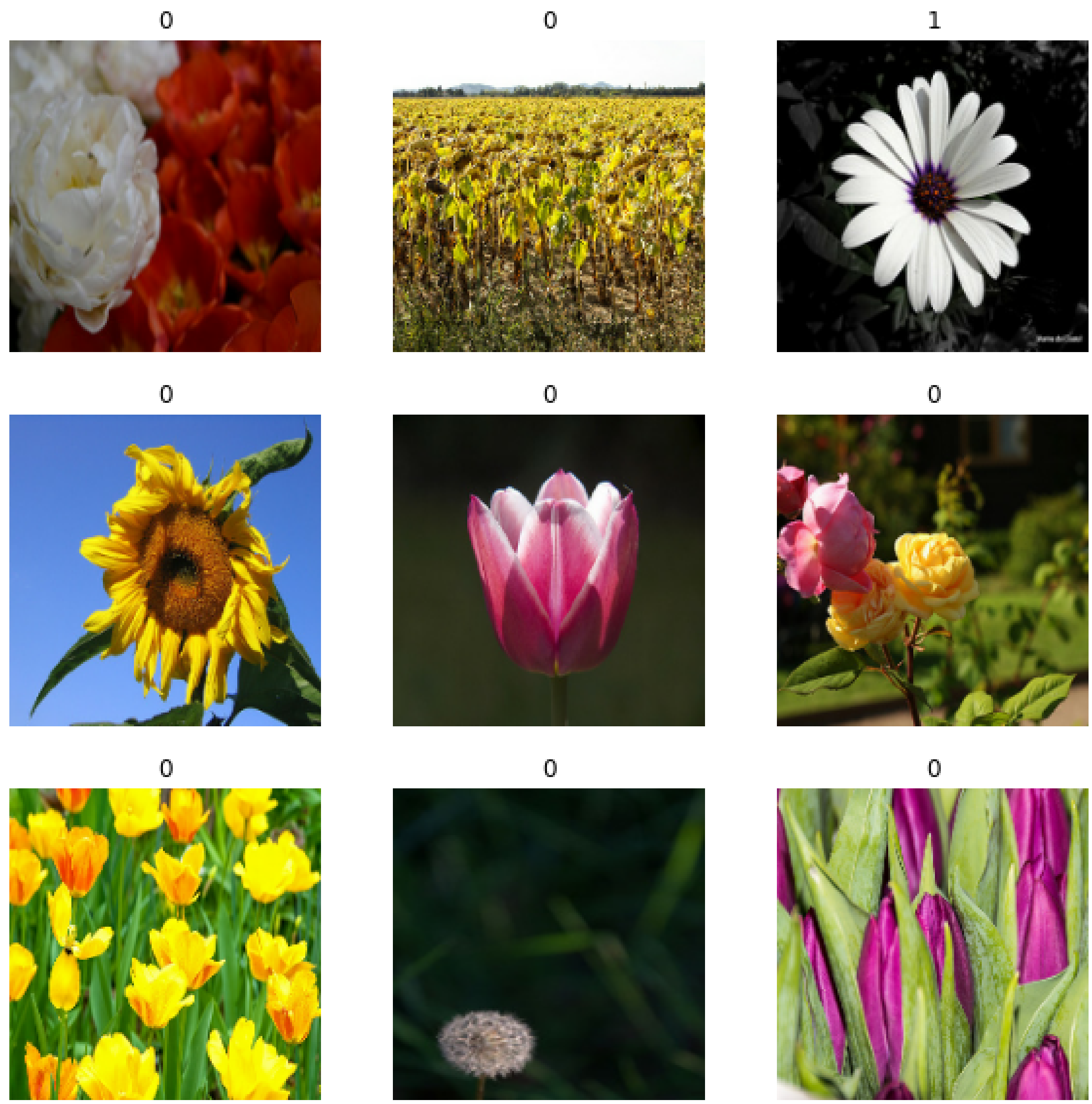
```
image_size = (180, 180)
batch_size = 32

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "/home/hiraditya/Desktop/HomeWork/SJSU/cs156/jupiter/JupyterBooks/flowers/
 ↪training/",
    validation_split=0.2,
    subset="training",
    seed=42,
    image_size=image_size,
    batch_size=batch_size,
    label_mode="categorical",
    labels='inferred'
)
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "/home/hiraditya/Desktop/HomeWork/SJSU/cs156/jupiter/JupyterBooks/flowers/
 ↪training/",
    validation_split=0.2,
    subset="validation",
    seed=42,
    image_size=image_size,
    batch_size=batch_size,
    label_mode="categorical",
    labels='inferred'
)
```

```
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "/home/hiraditya/Desktop/HomeWork/SJSU/cs156/jupiter/JupyterBooks/flowers/
 ↪test/",
    seed=42,
    image_size=image_size,
    batch_size=1,
    label_mode="categorical",
    labels='inferred',
    shuffle=False
)
```

```
Found 3456 files belonging to 5 classes.
Using 2765 files for training.
Found 3456 files belonging to 5 classes.
Using 691 files for validation.
Found 861 files belonging to 5 classes.
```

1 Sample images

[20]:
```
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(int(labels[i][0]))
        plt.axis("off")
```



2 Augmentation

[21]:
```
data_augmentation = keras.Sequential(
    [
        layers.experimental.preprocessing.RandomFlip("horizontal"),
        layers.experimental.preprocessing.RandomRotation(0.1),
    ]
)
```

[22]:

```
plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_images[0].numpy().astype("uint8"))
        plt.axis("off")
```

[23]:

```
train_ds = train_ds.prefetch(buffer_size=32)
val_ds = val_ds.prefetch(buffer_size=32)

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

```python
    # Image augmentation block
    x = data_augmentation(inputs)

    # Entry block
    x = layers.experimental.preprocessing.Rescaling(1.0 / 255)(x)
    x = layers.Conv2D(32, 3, strides=2, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    x = layers.Conv2D(64, 3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    previous_block_activation = x  # Set aside residual

    for size in [128, 256, 512, 728]:
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.MaxPooling2D(3, strides=2, padding="same")(x)

        # Project residual
        residual = layers.Conv2D(size, 1, strides=2, padding="same")(
            previous_block_activation
        )
        x = layers.add([x, residual])  # Add back residual
        previous_block_activation = x  # Set aside next residual

    x = layers.SeparableConv2D(1024, 3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    x = layers.GlobalAveragePooling2D()(x)

    activation = "softmax"
    units = num_classes

    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(units, activation=activation)(x)
    return keras.Model(inputs, outputs)
```

```python
model = make_model(input_shape=image_size + (3,), num_classes=5)
#keras.utils.plot_model(model, show_shapes=True)
model.summary()
```

```
Model: "model"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_1 (InputLayer)            [(None, 180, 180, 3) 0
________________________________________________________________________________
__________________
sequential (Sequential)         (None, 180, 180, 3)  0           input_1[0][0]
________________________________________________________________________________
__________________
rescaling (Rescaling)           (None, 180, 180, 3)  0
sequential[0][0]
________________________________________________________________________________
__________________
conv2d (Conv2D)                 (None, 90, 90, 32)   896         rescaling[0][0]
________________________________________________________________________________
__________________
batch_normalization (BatchNorma (None, 90, 90, 32)   128         conv2d[0][0]
________________________________________________________________________________
__________________
activation (Activation)         (None, 90, 90, 32)   0
batch_normalization[0][0]
________________________________________________________________________________
__________________
conv2d_1 (Conv2D)               (None, 90, 90, 64)   18496
activation[0][0]
________________________________________________________________________________
__________________
batch_normalization_1 (BatchNor (None, 90, 90, 64)   256         conv2d_1[0][0]
________________________________________________________________________________
__________________
activation_1 (Activation)       (None, 90, 90, 64)   0
batch_normalization_1[0][0]
________________________________________________________________________________
__________________
activation_2 (Activation)       (None, 90, 90, 64)   0
activation_1[0][0]
________________________________________________________________________________
__________________
separable_conv2d (SeparableConv (None, 90, 90, 128)  8896
activation_2[0][0]
```

```
__________________________________________________________________________________
__________________
batch_normalization_2 (BatchNor (None, 90, 90, 128)  512
separable_conv2d[0][0]

__________________________________________________________________________________
__________________
activation_3 (Activation)       (None, 90, 90, 128)  0
batch_normalization_2[0][0]

__________________________________________________________________________________
__________________
separable_conv2d_1 (SeparableCo (None, 90, 90, 128)  17664
activation_3[0][0]

__________________________________________________________________________________
__________________
batch_normalization_3 (BatchNor (None, 90, 90, 128)  512
separable_conv2d_1[0][0]

__________________________________________________________________________________
__________________
max_pooling2d (MaxPooling2D)    (None, 45, 45, 128)  0
batch_normalization_3[0][0]

__________________________________________________________________________________
__________________
conv2d_2 (Conv2D)               (None, 45, 45, 128)  8320
activation_1[0][0]

__________________________________________________________________________________
__________________
add (Add)                       (None, 45, 45, 128)  0
max_pooling2d[0][0]
                                                                 conv2d_2[0][0]

__________________________________________________________________________________
__________________
activation_4 (Activation)       (None, 45, 45, 128)  0           add[0][0]

__________________________________________________________________________________
__________________
separable_conv2d_2 (SeparableCo (None, 45, 45, 256)  34176
activation_4[0][0]

__________________________________________________________________________________
__________________
batch_normalization_4 (BatchNor (None, 45, 45, 256)  1024
separable_conv2d_2[0][0]

__________________________________________________________________________________
__________________
activation_5 (Activation)       (None, 45, 45, 256)  0
batch_normalization_4[0][0]

__________________________________________________________________________________
__________________
separable_conv2d_3 (SeparableCo (None, 45, 45, 256)  68096
activation_5[0][0]
```

```
__________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 45, 45, 256)  1024
separable_conv2d_3[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 23, 23, 256)  0
batch_normalization_5[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 23, 23, 256)  33024       add[0][0]
__________________________________________________________________________________________________
add_1 (Add)                     (None, 23, 23, 256)  0
max_pooling2d_1[0][0]
                                                                 conv2d_3[0][0]
__________________________________________________________________________________________________
activation_6 (Activation)       (None, 23, 23, 256)  0           add_1[0][0]
__________________________________________________________________________________________________
separable_conv2d_4 (SeparableCo (None, 23, 23, 512)  133888
activation_6[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 23, 23, 512)  2048
separable_conv2d_4[0][0]
__________________________________________________________________________________________________
activation_7 (Activation)       (None, 23, 23, 512)  0
batch_normalization_6[0][0]
__________________________________________________________________________________________________
separable_conv2d_5 (SeparableCo (None, 23, 23, 512)  267264
activation_7[0][0]
__________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, 23, 23, 512)  2048
separable_conv2d_5[0][0]
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 12, 12, 512)  0
batch_normalization_7[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 12, 12, 512)  131584      add_1[0][0]
__________________________________________________________________________________________________
```

```
add_2 (Add)                     (None, 12, 12, 512)  0
max_pooling2d_2[0][0]
                                                                 conv2d_4[0][0]
________________________________________________________________________________
__________________
activation_8 (Activation)       (None, 12, 12, 512)  0           add_2[0][0]
________________________________________________________________________________
__________________
separable_conv2d_6 (SeparableCo (None, 12, 12, 728)  378072
activation_8[0][0]
________________________________________________________________________________
__________________
batch_normalization_8 (BatchNor (None, 12, 12, 728)  2912
separable_conv2d_6[0][0]
________________________________________________________________________________
__________________
activation_9 (Activation)       (None, 12, 12, 728)  0
batch_normalization_8[0][0]
________________________________________________________________________________
__________________
separable_conv2d_7 (SeparableCo (None, 12, 12, 728)  537264
activation_9[0][0]
________________________________________________________________________________
__________________
batch_normalization_9 (BatchNor (None, 12, 12, 728)  2912
separable_conv2d_7[0][0]
________________________________________________________________________________
__________________
max_pooling2d_3 (MaxPooling2D)  (None, 6, 6, 728)    0
batch_normalization_9[0][0]
________________________________________________________________________________
__________________
conv2d_5 (Conv2D)               (None, 6, 6, 728)    373464      add_2[0][0]
________________________________________________________________________________
__________________
add_3 (Add)                     (None, 6, 6, 728)    0
max_pooling2d_3[0][0]
                                                                 conv2d_5[0][0]
________________________________________________________________________________
__________________
separable_conv2d_8 (SeparableCo (None, 6, 6, 1024)   753048      add_3[0][0]
________________________________________________________________________________
__________________
batch_normalization_10 (BatchNo (None, 6, 6, 1024)   4096
separable_conv2d_8[0][0]
________________________________________________________________________________
__________________
activation_10 (Activation)      (None, 6, 6, 1024)   0
```

```
batch_normalization_10[0][0]

--------------------------------------------------------------------------------
------------------
global_average_pooling2d (Globa (None, 1024)         0
activation_10[0][0]

--------------------------------------------------------------------------------
------------------
dropout (Dropout)               (None, 1024)         0
global_average_pooling2d[0][0]

--------------------------------------------------------------------------------
------------------
dense (Dense)                   (None, 5)            5125        dropout[0][0]
================================================================================
==================
Total params: 2,786,749
Trainable params: 2,778,013
Non-trainable params: 8,736

--------------------------------------------------------------------------------
------------------
```

3 Training

[24]:
```
epochs = 50

callbacks = [
    keras.callbacks.ModelCheckpoint("save_at_{epoch}.h5"),
]
model.compile(
    optimizer=keras.optimizers.Adam(1e-3),
    loss="categorical_crossentropy",
    metrics=["accuracy"],
)
model.fit(
    train_ds, epochs=epochs, callbacks=callbacks, validation_data=val_ds,
)
```

```
Epoch 1/50
87/87 [==============================] - 224s 3s/step - loss: 1.3465 - accuracy:
0.4832 - val_loss: 1.7483 - val_accuracy: 0.2590
Epoch 2/50
87/87 [==============================] - 218s 3s/step - loss: 1.0236 - accuracy:
0.6029 - val_loss: 2.1732 - val_accuracy: 0.2590
Epoch 3/50
87/87 [==============================] - 219s 3s/step - loss: 0.9266 - accuracy:
0.6574 - val_loss: 3.0028 - val_accuracy: 0.2590
Epoch 4/50
```

```
87/87 [==============================] - 232s 3s/step - loss: 0.8780 - accuracy:
0.6635 - val_loss: 3.1405 - val_accuracy: 0.2590
Epoch 5/50
87/87 [==============================] - 218s 3s/step - loss: 0.8106 - accuracy:
0.6918 - val_loss: 3.3701 - val_accuracy: 0.2590
Epoch 6/50
87/87 [==============================] - 218s 3s/step - loss: 0.7251 - accuracy:
0.7340 - val_loss: 1.7965 - val_accuracy: 0.4182
Epoch 7/50
87/87 [==============================] - 218s 3s/step - loss: 0.6986 - accuracy:
0.7460 - val_loss: 1.3393 - val_accuracy: 0.5427
Epoch 8/50
87/87 [==============================] - 220s 3s/step - loss: 0.6548 - accuracy:
0.7494 - val_loss: 0.7131 - val_accuracy: 0.7308
Epoch 9/50
87/87 [==============================] - 217s 2s/step - loss: 0.6536 - accuracy:
0.7660 - val_loss: 1.2157 - val_accuracy: 0.6078
Epoch 10/50
87/87 [==============================] - 223s 3s/step - loss: 0.6052 - accuracy:
0.7608 - val_loss: 1.0400 - val_accuracy: 0.6816
Epoch 11/50
87/87 [==============================] - 218s 3s/step - loss: 0.5760 - accuracy:
0.7839 - val_loss: 0.7353 - val_accuracy: 0.7352
Epoch 12/50
87/87 [==============================] - 218s 3s/step - loss: 0.5531 - accuracy:
0.7971 - val_loss: 0.8489 - val_accuracy: 0.7265
Epoch 13/50
87/87 [==============================] - 218s 3s/step - loss: 0.5359 - accuracy:
0.8015 - val_loss: 0.7992 - val_accuracy: 0.7511
Epoch 14/50
87/87 [==============================] - 218s 2s/step - loss: 0.4897 - accuracy:
0.8097 - val_loss: 0.5440 - val_accuracy: 0.8177
Epoch 15/50
87/87 [==============================] - 218s 2s/step - loss: 0.5172 - accuracy:
0.8010 - val_loss: 0.8306 - val_accuracy: 0.7337
Epoch 16/50
87/87 [==============================] - 218s 3s/step - loss: 0.4708 - accuracy:
0.8260 - val_loss: 0.8424 - val_accuracy: 0.7192
Epoch 17/50
87/87 [==============================] - 218s 3s/step - loss: 0.4718 - accuracy:
0.8249 - val_loss: 1.3616 - val_accuracy: 0.6107
Epoch 18/50
87/87 [==============================] - 218s 3s/step - loss: 0.4373 - accuracy:
0.8307 - val_loss: 0.6515 - val_accuracy: 0.8003
Epoch 19/50
87/87 [==============================] - 218s 3s/step - loss: 0.4367 - accuracy:
0.8436 - val_loss: 0.9319 - val_accuracy: 0.7308
Epoch 20/50
```

```
87/87 [==============================] - 220s 3s/step - loss: 0.4311 - accuracy:
0.8426 - val_loss: 0.9469 - val_accuracy: 0.7236
Epoch 21/50
87/87 [==============================] - 218s 2s/step - loss: 0.3968 - accuracy:
0.8579 - val_loss: 0.6814 - val_accuracy: 0.8017
Epoch 22/50
87/87 [==============================] - 217s 2s/step - loss: 0.4027 - accuracy:
0.8565 - val_loss: 0.8472 - val_accuracy: 0.7641
Epoch 23/50
87/87 [==============================] - 217s 2s/step - loss: 0.3894 - accuracy:
0.8553 - val_loss: 0.6170 - val_accuracy: 0.7931
Epoch 24/50
87/87 [==============================] - 218s 2s/step - loss: 0.3799 - accuracy:
0.8643 - val_loss: 0.6488 - val_accuracy: 0.7945
Epoch 25/50
87/87 [==============================] - 218s 3s/step - loss: 0.3280 - accuracy:
0.8767 - val_loss: 0.6389 - val_accuracy: 0.8017
Epoch 26/50
87/87 [==============================] - 218s 2s/step - loss: 0.2978 - accuracy:
0.8911 - val_loss: 0.5093 - val_accuracy: 0.8379
Epoch 27/50
87/87 [==============================] - 218s 2s/step - loss: 0.3449 - accuracy:
0.8744 - val_loss: 0.8083 - val_accuracy: 0.7482
Epoch 28/50
87/87 [==============================] - 217s 2s/step - loss: 0.3190 - accuracy:
0.8877 - val_loss: 0.8514 - val_accuracy: 0.7742
Epoch 29/50
87/87 [==============================] - 217s 2s/step - loss: 0.2916 - accuracy:
0.8945 - val_loss: 1.6230 - val_accuracy: 0.6556
Epoch 30/50
87/87 [==============================] - 217s 2s/step - loss: 0.3687 - accuracy:
0.8657 - val_loss: 0.8991 - val_accuracy: 0.7337
Epoch 31/50
87/87 [==============================] - 217s 2s/step - loss: 0.2846 - accuracy:
0.8947 - val_loss: 0.5686 - val_accuracy: 0.8278
Epoch 32/50
87/87 [==============================] - 216s 2s/step - loss: 0.2900 - accuracy:
0.8974 - val_loss: 0.9521 - val_accuracy: 0.7424
Epoch 33/50
87/87 [==============================] - 218s 2s/step - loss: 0.2620 - accuracy:
0.8994 - val_loss: 0.5064 - val_accuracy: 0.8321
Epoch 34/50
87/87 [==============================] - 217s 2s/step - loss: 0.2526 - accuracy:
0.9118 - val_loss: 0.8188 - val_accuracy: 0.7685
Epoch 35/50
87/87 [==============================] - 217s 2s/step - loss: 0.2766 - accuracy:
0.8994 - val_loss: 0.8129 - val_accuracy: 0.8046
Epoch 36/50
```

```
87/87 [==============================] - 217s 2s/step - loss: 0.3141 - accuracy:
0.8859 - val_loss: 1.4699 - val_accuracy: 0.7004
Epoch 37/50
87/87 [==============================] - 217s 2s/step - loss: 0.2486 - accuracy:
0.9156 - val_loss: 0.9625 - val_accuracy: 0.7236
Epoch 38/50
87/87 [==============================] - 217s 2s/step - loss: 0.2417 - accuracy:
0.9134 - val_loss: 1.0098 - val_accuracy: 0.7858
Epoch 39/50
87/87 [==============================] - 217s 2s/step - loss: 0.2034 - accuracy:
0.9259 - val_loss: 0.7311 - val_accuracy: 0.7959
Epoch 40/50
87/87 [==============================] - 216s 2s/step - loss: 0.2134 - accuracy:
0.9180 - val_loss: 0.8256 - val_accuracy: 0.7844
Epoch 41/50
87/87 [==============================] - 216s 2s/step - loss: 0.2087 - accuracy:
0.9213 - val_loss: 0.6728 - val_accuracy: 0.8104
Epoch 42/50
87/87 [==============================] - 217s 2s/step - loss: 0.2069 - accuracy:
0.9267 - val_loss: 0.8395 - val_accuracy: 0.8061
Epoch 43/50
87/87 [==============================] - 216s 2s/step - loss: 0.2601 - accuracy:
0.9008 - val_loss: 0.5315 - val_accuracy: 0.8524
Epoch 44/50
87/87 [==============================] - 216s 2s/step - loss: 0.2084 - accuracy:
0.9260 - val_loss: 0.5010 - val_accuracy: 0.8365
Epoch 45/50
87/87 [==============================] - 217s 2s/step - loss: 0.2366 - accuracy:
0.9242 - val_loss: 0.4863 - val_accuracy: 0.8611
Epoch 46/50
87/87 [==============================] - 216s 2s/step - loss: 0.1859 - accuracy:
0.9299 - val_loss: 0.6040 - val_accuracy: 0.8350
Epoch 47/50
87/87 [==============================] - 216s 2s/step - loss: 0.2127 - accuracy:
0.9180 - val_loss: 0.6580 - val_accuracy: 0.8336
Epoch 48/50
87/87 [==============================] - 216s 2s/step - loss: 0.1856 - accuracy:
0.9303 - val_loss: 0.6543 - val_accuracy: 0.8191
Epoch 49/50
87/87 [==============================] - 215s 2s/step - loss: 0.1757 - accuracy:
0.9379 - val_loss: 0.8392 - val_accuracy: 0.7742
Epoch 50/50
87/87 [==============================] - 217s 2s/step - loss: 0.1891 - accuracy:
0.9300 - val_loss: 0.5563 - val_accuracy: 0.8292
```

```
[24]: <tensorflow.python.keras.callbacks.History at 0x7f7dbabc12e0>
```

4 Evaluation

[25]:
```
model.evaluate(test_ds)
```

```
861/861 [==============================] - 19s 22ms/step - loss: 1.1448 -
accuracy: 0.7480
```

[25]:
```
[1.1448140144348145, 0.7479674816131592]
```

Extracting true labels and the images from test_ds

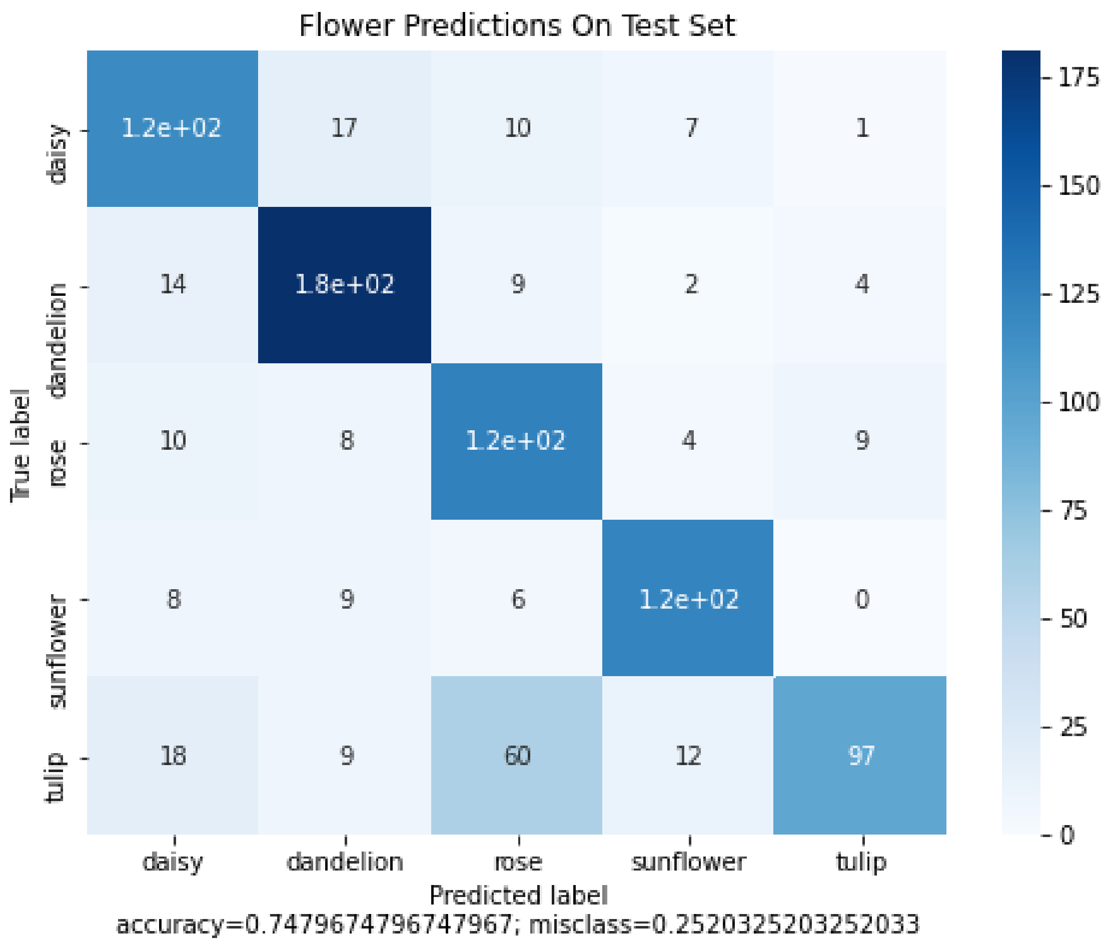
[26]:
```
# Test Labels
true = np.concatenate([y for x, y in test_ds], axis=0)
true = np.argmax(true, axis=1)
# Test Images
test_images = np.concatenate([x for x, y in test_ds], axis=0).astype('int')
```

5 Making Predictions

[27]:
```
probas = model.predict(test_ds)
predicted = np.argmax(probas, axis=-1)
```

6 Confusion Matrix

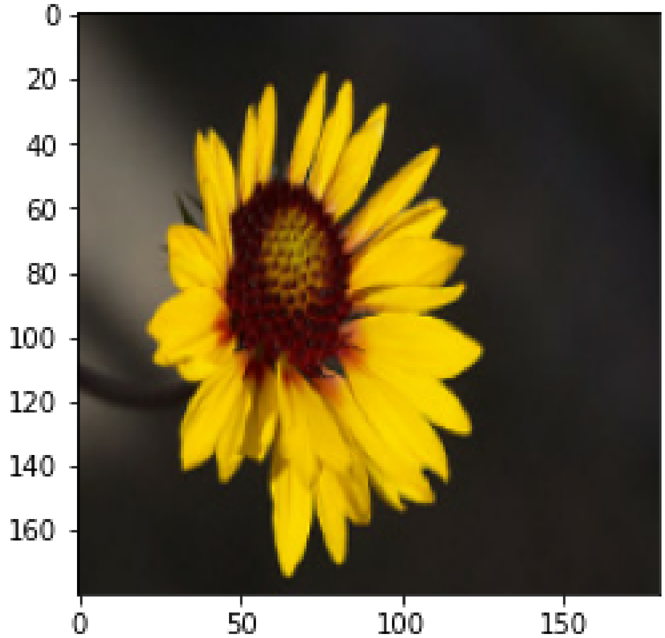
[28]:
```
cm = tf.math.confusion_matrix(labels=true, predictions=predicted).numpy()
cm_df = pd.DataFrame(cm, index = ['daisy', 'dandelion', 'rose', 'sunflower',␣
 ↪'tulip'], columns = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'])

plt.figure(figsize=(8, 6))
sb.heatmap(cm_df, annot=True, cmap=plt.cm.Blues)
plt.title('Flower Predictions On Test Set')
plt.ylabel('True label')
plt.xlabel('Predicted label\naccuracy={}; misclass={}'.
 ↪format(accuracy_score(true, predicted), 1 - accuracy_score(true, predicted)))
plt.show()
```



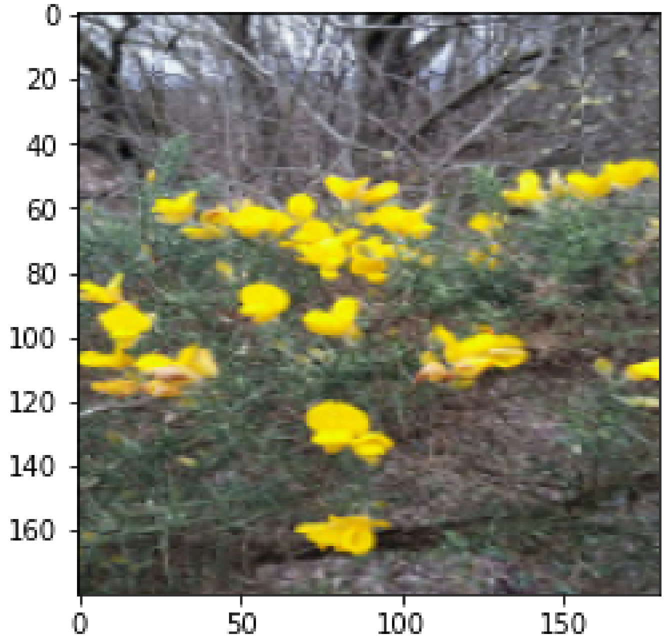
7 3 images that were misclassified

[29]:
```
counter = 0
class_names = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']
for i in range(100, len(true)):
    if true[i] != predicted[i]:
        if counter < 3:
            print(class_names[true[i]], "predicted as",␣
 ↪class_names[predicted[i]], "\n")
            plt.imshow(test_images[i])
            plt.show()
            print("\n\n\n\n")
            counter += 1
```

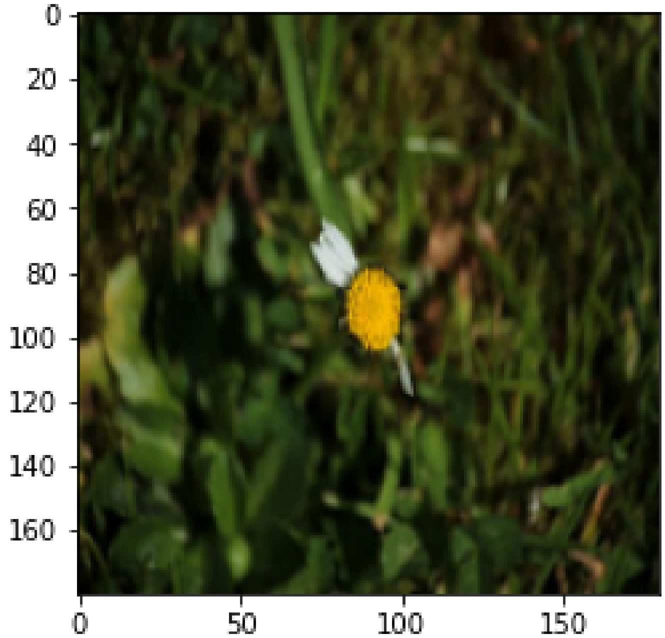
```
daisy predicted as sunflower
```



daisy predicted as dandelion



daisy predicted as dandelion


0
20
40
60
80
100
120
140
160
0
50
100
150

```
[ ]:
```