The dataset contains 14 types of flower images, including 13618 training images and 98 validation images, with a total data size of 202MB, and supports the recognition of the following flower types: carnation, iris, bluebells, golden english, roses, fallen nephews, tulips, marigolds, dandelions, chrysanthemums, black-eyed daisies, water lilies, sunflowers, and daisies.

---

1) Read the dataset using the os module.
2) Perform image preprocessing as per requirement.
3) Implement a neural network using keras. (transfer learning is not allowed)
4) Compile the model.
5) Print the summary of the model.
6) Fit and Evaluate the model.

In [1]:
```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

In [2]:
```
!kaggle datasets download -d marquis03/flower-classification
```

```
Warning: Your Kaggle API key is readable by other users on this system!
To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Downloading flower-classification.zip to /content
 96% 197M/205M [00:00<00:00, 285MB/s]
100% 205M/205M [00:00<00:00, 279MB/s]
```

In [3]:
```
import zipfile
zip_ref = zipfile.ZipFile('/content/flower-classification.zip', 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

In [4]:
```
import os
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
import keras.models
from keras import layers
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.models import Sequential
```

In [5]:
```
train_dir = '/content/train'
val_dir = '/content/val'
train_data = keras.utils.image_dataset_from_directory(
 train_dir,
    image_size=(224,224),
    batch_size=128,
    seed=3,
    subset='training',
    validation_split=0.1
)
val_data = keras.utils.image_dataset_from_directory(
    val_dir,
    seed=1,
    subset='validation',
    validation_split=0.3,
    image_size=(224,224)
)
```

```
Found 13642 files belonging to 14 classes.
Using 12278 files for training.
Found 98 files belonging to 14 classes.
Using 29 files for validation.
```

In [15]:
```
classes = train_data.class_names
num_classes = len(classes)
model = keras.Sequential([
    keras.layers.Rescaling(1./255, input_shape=(224, 224, 3)),
    keras.layers.RandomFlip('horizontal'),
    keras.layers.Conv2D(filters=16, kernel_size=(3, 3), padding='valid', a
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Conv2D(filters=32, kernel_size=(4, 4), padding='same', ac
    keras.layers.MaxPooling2D(pool_size=(4, 4)),
    keras.layers.Flatten(),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(units=32, activation='relu'),
    keras.layers.Dropout(rate=0.3),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dense(units=256, activation='relu'),
    keras.layers.Dense(units=num_classes, activation='softmax')
])
```

In [16]:
```
model.compile(optimizer='Adam', loss='sparse_categorical_crossentropy', me
```

In [17]:
```
model.fit(train_data, epochs=15, validation_data = val_data)
```

```
Epoch 1/15
96/96 [==============================] - 19s 172ms/step - loss: 1.8842 -
accuracy: 0.3338 - val_loss: 2.5538 - val_accuracy: 0.1034
Epoch 2/15
96/96 [==============================] - 18s 174ms/step - loss: 1.5196 -
accuracy: 0.4660 - val_loss: 2.5229 - val_accuracy: 0.0690
Epoch 3/15
96/96 [==============================] - 18s 176ms/step - loss: 1.3422 -
accuracy: 0.5262 - val_loss: 2.2406 - val_accuracy: 0.3448
Epoch 4/15
96/96 [==============================] - 19s 186ms/step - loss: 1.1897 -
accuracy: 0.5898 - val_loss: 2.0129 - val_accuracy: 0.3793
Epoch 5/15
96/96 [==============================] - 18s 178ms/step - loss: 1.0580 -
accuracy: 0.6337 - val_loss: 1.6357 - val_accuracy: 0.5862
Epoch 6/15
96/96 [==============================] - 18s 182ms/step - loss: 0.9607 -
accuracy: 0.6669 - val_loss: 1.2191 - val_accuracy: 0.4828
Epoch 7/15
96/96 [==============================] - 17s 170ms/step - loss: 0.8838 -
accuracy: 0.6899 - val_loss: 1.3070 - val_accuracy: 0.5172
Epoch 8/15
96/96 [==============================] - 18s 183ms/step - loss: 0.8004 -
accuracy: 0.7219 - val_loss: 0.9539 - val_accuracy: 0.6552
Epoch 9/15
96/96 [==============================] - 18s 174ms/step - loss: 0.7474 -
accuracy: 0.7395 - val_loss: 1.0196 - val_accuracy: 0.6207
Epoch 10/15
96/96 [==============================] - 18s 181ms/step - loss: 0.6970 -
accuracy: 0.7592 - val_loss: 0.9349 - val_accuracy: 0.6552
Epoch 11/15
96/96 [==============================] - 18s 177ms/step - loss: 0.6454 -
accuracy: 0.7736 - val_loss: 1.2196 - val_accuracy: 0.5517
Epoch 12/15
96/96 [==============================] - 18s 182ms/step - loss: 0.5956 -
accuracy: 0.7904 - val_loss: 0.9313 - val_accuracy: 0.5862
Epoch 13/15
96/96 [==============================] - 18s 176ms/step - loss: 0.5552 -
accuracy: 0.8049 - val_loss: 1.2195 - val_accuracy: 0.5517
Epoch 14/15
96/96 [==============================] - 18s 175ms/step - loss: 0.5298 -
accuracy: 0.8101 - val_loss: 1.1425 - val_accuracy: 0.6552
Epoch 15/15
96/96 [==============================] - 18s 175ms/step - loss: 0.5042 -
accuracy: 0.8235 - val_loss: 1.0177 - val_accuracy: 0.7241
```

Out[17]:
```
<keras.src.callbacks.History at 0x7fb87cd28970>
```

In [18]:
```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_3 (Rescaling)     (None, 224, 224, 3)       0

 random_flip_3 (RandomFlip)  (None, 224, 224, 3)       0

 conv2d_6 (Conv2D)           (None, 222, 222, 16)      448

 max_pooling2d_6 (MaxPoolin  (None, 111, 111, 16)      0
 g2D)

 conv2d_7 (Conv2D)           (None, 111, 111, 32)      8224

 max_pooling2d_7 (MaxPoolin  (None, 27, 27, 32)        0
 g2D)

 flatten_3 (Flatten)         (None, 23328)             0

 batch_normalization_3 (Bat  (None, 23328)             93312
 chNormalization)

 dense_12 (Dense)            (None, 32)                746528

 dropout_3 (Dropout)         (None, 32)                0

 dense_13 (Dense)            (None, 128)               4224

 dense_14 (Dense)            (None, 256)               33024

 dense_15 (Dense)            (None, 14)                3598

=================================================================
Total params: 889358 (3.39 MB)
Trainable params: 842702 (3.21 MB)
Non-trainable params: 46656 (182.25 KB)
_________________________________________________________________
```

In [19]:
```
evaluation_result = model.evaluate(val_data)
print("Evaluation Loss:", evaluation_result[0])
print("Evaluation Accuracy:", evaluation_result[1])
```

```
1/1 [==============================] - 0s 75ms/step - loss: 1.0177 - acc
uracy: 0.7241
Evaluation Loss: 1.0176910161972046
Evaluation Accuracy: 0.7241379022598267
```

In [ ]:
```

```