*# Image classification using CNN*

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

## Data Augumentation

```
train_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.2,
horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
```

### # Data Augumentation on training data

```
x_train = train_datagen.flow_from_directory('S:/Studies/Nalaiya
thiran/assignment 3/Flowers-Dataset/flowers', target_size=(64,64),
class_mode='categorical', batch_size=100)
```

```
Found 4317 images belonging to 5 classes.
```

## Data Augumentation on testing data

```
x_test = test_datagen.flow_from_directory('S:/Studies/Nalaiya
thiran/assignment 3/Flowers-Dataset/flowers', target_size=(64,64),
class_mode='categorical', batch_size=100)
```

```
Found 4317 images belonging to 5 classes.
```

## CNN Model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D, MaxPooling2D,
Flatten, Dense
```

```
model = Sequential()
model.add(Convolution2D(32,(3,3), activation='relu',
input_shape=(64,64,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(300, activation='relu'))
model.add(Dense(150, activation='relu'))
model.add(Dense(5, activation='softmax'))
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
model.fit(x_train, steps_per_epoch=len(x_train), epochs=10,
validation_data=x_test, validation_steps=len(x_test))
```

```
Epoch 1/10
44/44 [==============================] - 75s 2s/step - loss: 1.4514 -
```

```
accuracy: 0.4225 - val_loss: 1.1998 - val_accuracy: 0.5080
Epoch 2/10
44/44 [==============================] - 137s 3s/step - loss: 1.0843 -
accuracy: 0.5645 - val_loss: 1.0983 - val_accuracy: 0.5768
Epoch 3/10
44/44 [==============================] - 136s 3s/step - loss: 0.9835 -
accuracy: 0.6148 - val_loss: 1.0368 - val_accuracy: 0.6152
Epoch 4/10
44/44 [==============================] - 91s 2s/step - loss: 0.9115 -
accuracy: 0.6491 - val_loss: 0.8315 - val_accuracy: 0.6912
Epoch 5/10
44/44 [==============================] - 118s 3s/step - loss: 0.8700 -
accuracy: 0.6671 - val_loss: 0.8722 - val_accuracy: 0.6806
Epoch 6/10
44/44 [==============================] - 81s 2s/step - loss: 0.8228 -
accuracy: 0.6882 - val_loss: 0.7335 - val_accuracy: 0.7301
Epoch 7/10
44/44 [==============================] - 82s 2s/step - loss: 0.7649 -
accuracy: 0.7160 - val_loss: 0.7981 - val_accuracy: 0.6963
Epoch 8/10
44/44 [==============================] - 69s 2s/step - loss: 0.7552 -
accuracy: 0.7114 - val_loss: 0.7307 - val_accuracy: 0.7237
Epoch 9/10
44/44 [==============================] - 84s 2s/step - loss: 0.6976 -
accuracy: 0.7313 - val_loss: 0.6288 - val_accuracy: 0.7647
Epoch 10/10
44/44 [==============================] - 83s 2s/step - loss: 0.6604 -
accuracy: 0.7487 - val_loss: 0.5567 - val_accuracy: 0.7948

<keras.callbacks.History at 0x1f9c8250490>

model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 flatten (Flatten)           (None, 30752)             0

 dense (Dense)               (None, 300)               9225900

 dense_1 (Dense)             (None, 150)               45150

 dense_2 (Dense)             (None, 5)                 755

=================================================================
```

```
Total params: 9,272,701
Trainable params: 9,272,701
Non-trainable params: 0
_________________________________________________________________
```

## Saving the data model

```
model.save('S:/Studies/Nalaiya thiran/assignment 3/classifier.h5')
from tensorflow import keras

import numpy as np
from tensorflow.keras.preprocessing import image

img = image.load_img('S:/Studies/Nalaiya thiran/assignment 3/Flowers-Dataset/flowers/sunflower/1008566138_6927679c8a.jpg',target_size=(64,64))
img
```

```
x = image.img_to_array(img)
x
x = np.expand_dims(x,axis=0)
x
model.predict(x)

1/1 [==============================] - 0s 278ms/step

array([[0., 0., 0., 1., 0.]], dtype=float32)

x_train.class_indices

{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

## Testing the model

```
op = ['daisy','dandelion','rose','sunflower','tulip']
pred = np.argmax(model.predict(x))
op[pred]

1/1 [==============================] - 0s 47ms/step

'sunflower'

from numpy.lib.type_check import imag
img = image.load_img('S:/Studies/Nalaiya thiran/assignment 3/Flowers-Dataset/flowers/tulip/10128546863_8de70c610d.jpg',target_size=(64,64))
img
```

```
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
pred = np.argmax(model.predict(x))
op[pred]
```

```
1/1 [==============================] - 0s 63ms/step
```

```
'tulip'
```