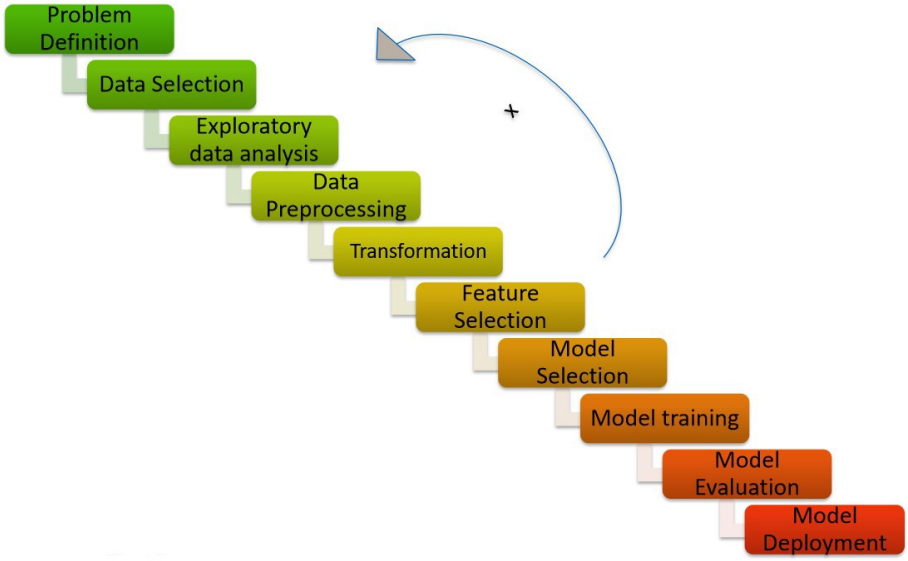
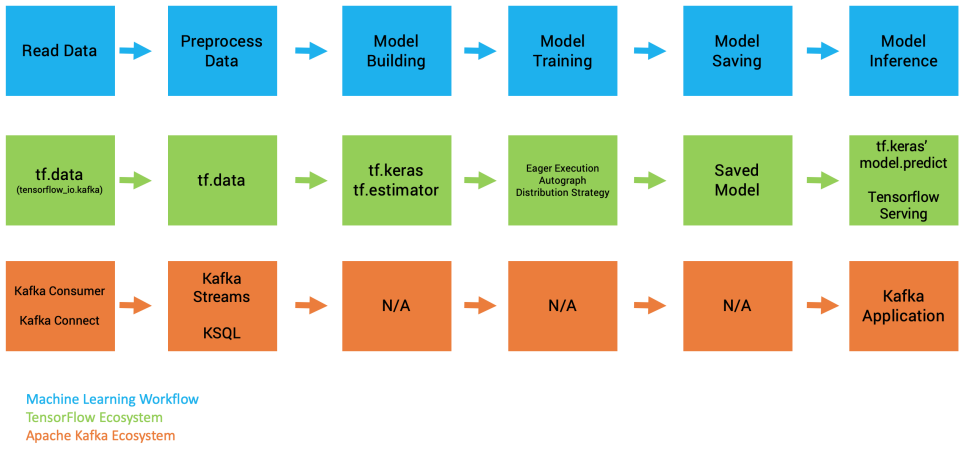
# Project - Computer Vision

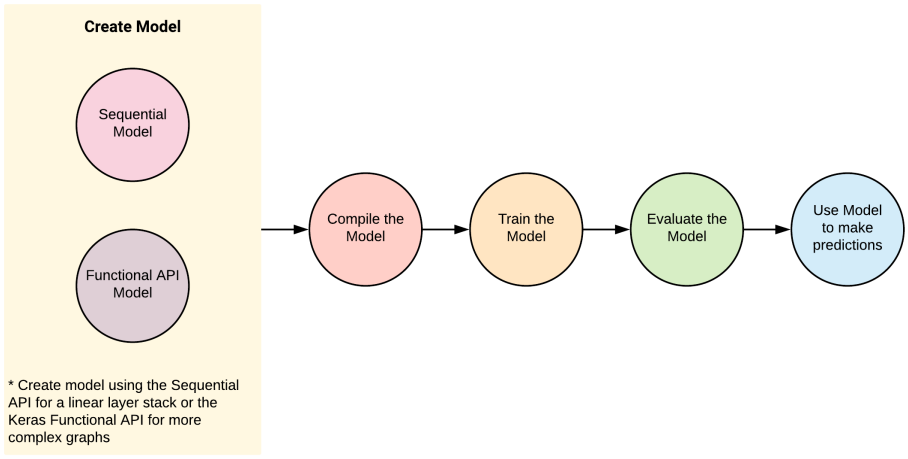
In [1]:
```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

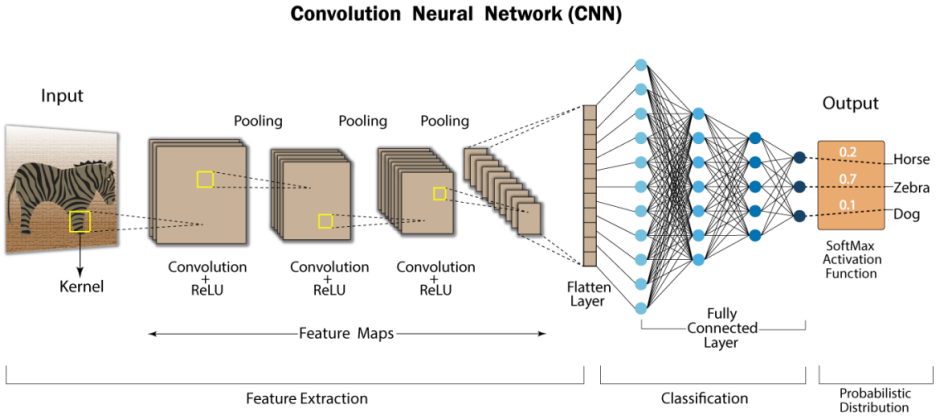
## Project Workflow

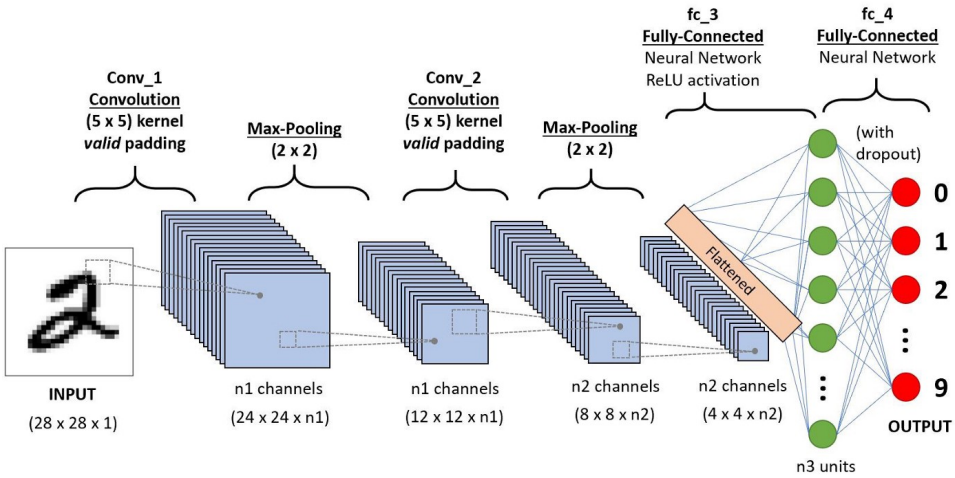





# CNN




In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn

import cv2
import os
import PIL
```

```
import tensorflow as tf
from tensorflow import keras
```

# 01. Get Data

In [ ]:
```
dataset_url = "https://storage.googleapis.com/download.tensorflow.org/example_image

data_dir = tf.keras.utils.get_file('flower_photos', origin=dataset_url, cache_dir=
# cache_dir indicates where to download data. -> '.' which means current directory
# `untar` true will unzip it

data_dir
```

Out[ ]:
```
'.\\datasets\\flower_photos'
```

In [ ]:
```
import pathlib
data_dir = pathlib.Path(data_dir)
data_dir
```

Out[ ]:
```
WindowsPath('datasets/flower_photos')
```

In [ ]:
```
data_dir.glob('*.*')
```

Out[ ]:
```
<generator object Path.glob at 0x000002910CEE1A80>
```

In [ ]:
```
list(data_dir.glob('*/*'))[:5]
```

Out[ ]:
```
[WindowsPath('datasets/flower_photos/daisy/100080576_f52e8ee070_n.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10140303196_b88d3d6cec.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10172379554_b296050f82_n.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10172567486_2748826a8b.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10172636503_21bededa75_n.jpg')]
```

# 02. Exploratory Data

In [ ]:
```
len(list(data_dir.glob('*/*')))
```

Out[ ]:
```
3670
```

In [ ]:
```
image_count = len(list(data_dir.glob('*/*.jpg')))
image_count
```

Out[ ]:
```
3670
```

In [ ]:
```
roses = list(data_dir.glob('roses/*'))
roses[:5]
```

Out[ ]:
```
[WindowsPath('datasets/flower_photos/roses/10090824183_d02c613f10_m.jpg'),
 WindowsPath('datasets/flower_photos/roses/102501987_3cdb8e5394_n.jpg'),
 WindowsPath('datasets/flower_photos/roses/10503217854_e66a804309.jpg'),
 WindowsPath('datasets/flower_photos/roses/10894627425_ec76bbc757_n.jpg'),
 WindowsPath('datasets/flower_photos/roses/110472418_87b6a3aa98_m.jpg')]
```

In [ ]:
```
PIL.Image.open(roses[0])
```

Out[ ]:

In [ ]:
```
tulips = list(data_dir.glob('tulips/*'))
tulips[:5]
```

Out[ ]:
```
[WindowsPath('datasets/flower_photos/tulips/100930342_92e8746431_n.jpg'),
 WindowsPath('datasets/flower_photos/tulips/10094729603_eeca3f2cb6.jpg'),
 WindowsPath('datasets/flower_photos/tulips/10094731133_94a942463c.jpg'),
 WindowsPath('datasets/flower_photos/tulips/10128546863_8de70c610d.jpg'),
 WindowsPath('datasets/flower_photos/tulips/10163955604_ae0b830975_n.jpg')]
```

In [ ]:
```
PIL.Image.open(tulips[0])
```

Out[ ]:

In [ ]:
```
flowers_images_dict = {
    'roses': list(data_dir.glob('roses/*')),
    'daisy': list(data_dir.glob('daisy/*')),
    'dandelion': list(data_dir.glob('dandelion/*')),
    'sunflowers': list(data_dir.glob('sunflowers/*')),
    'tulips': list(data_dir.glob('tulips/*')),
}
```

In [ ]:
```
flowers_labels_dict = {
    'roses': 0,
    'daisy': 1,
    'dandelion': 2,
    'sunflowers': 3,
    'tulips': 4,
}
```

In [ ]:
```
flowers_images_dict['roses'][:5]
```

Out[ ]:
```
[WindowsPath('datasets/flower_photos/roses/10090824183_d02c613f10_m.jpg'),
 WindowsPath('datasets/flower_photos/roses/102501987_3cdb8e5394_n.jpg'),
 WindowsPath('datasets/flower_photos/roses/10503217854_e66a804309.jpg'),
 WindowsPath('datasets/flower_photos/roses/10894627425_ec76bbc757_n.jpg'),
 WindowsPath('datasets/flower_photos/roses/110472418_87b6a3aa98_m.jpg')]
```

In [ ]:
```
str(flowers_images_dict['roses'][0])
```

Out[ ]:
```
'datasets\\flower_photos\\roses\\10090824183_d02c613f10_m.jpg'
```

In [ ]:
```
img = cv2.imread(str(flowers_images_dict['roses'][0]))
img.shape
```

Out[ ]:
```
(240, 179, 3)
```

In [ ]:
```
cv2.resize(img, (180, 180)).shape
```

Out[ ]:
```
(180, 180, 3)
```

In [ ]:
```
for flower_name, images in flowers_images_dict.items():
    print(flower_name, len(images))
```

```
roses 641
daisy 633
dandelion 898
sunflowers 699
tulips 799
```

In [ ]:
```
X = []
y = []
for flower_name, images in flowers_images_dict.items():
    for image in images:
        img = cv2.imread(str(image))
        img_resized = cv2.resize(img, (180, 180))
        X.append(img_resized)
        y.append(flowers_labels_dict[flower_name])
```

In [ ]:
```
len(X)
```

Out[ ]:
```
3670
```

In [ ]:
```
len(y)
```

Out[ ]:
```
3670
```

In [ ]:
```
X = np.array(X)
y = np.array(y)
```

In [ ]:
```
X.shape
```

Out[ ]:
```
(3670, 180, 180, 3)
```

In [ ]:
```
y.shape
```

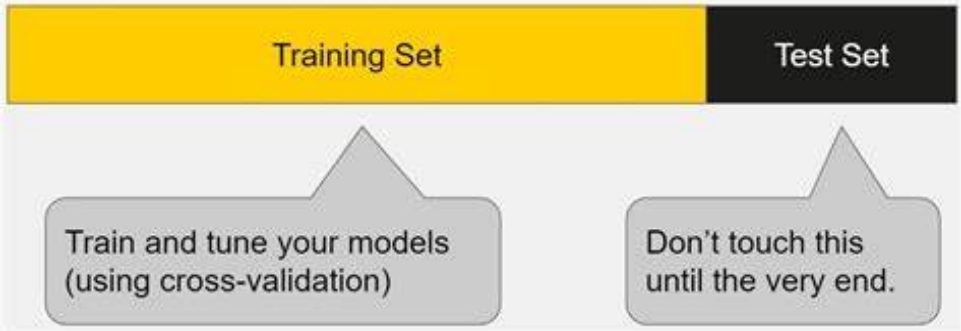
Out[ ]:
```
(3670,)
```

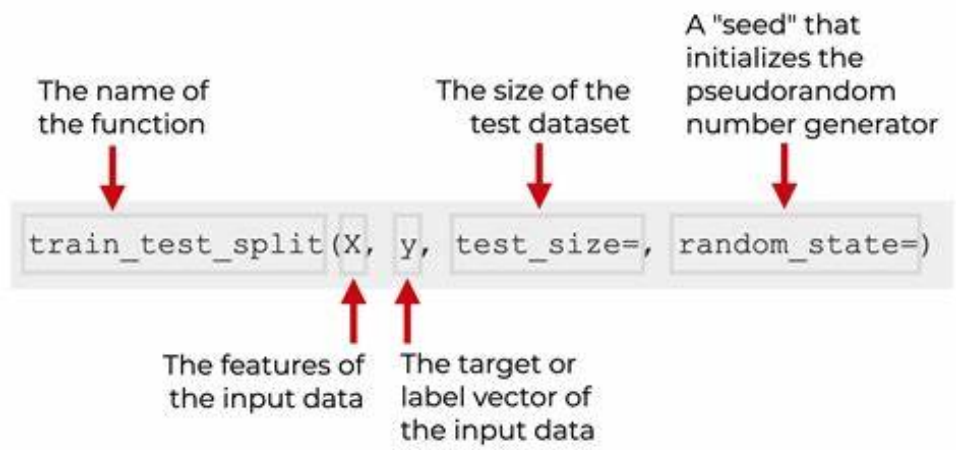
# 03. Preprocess Data

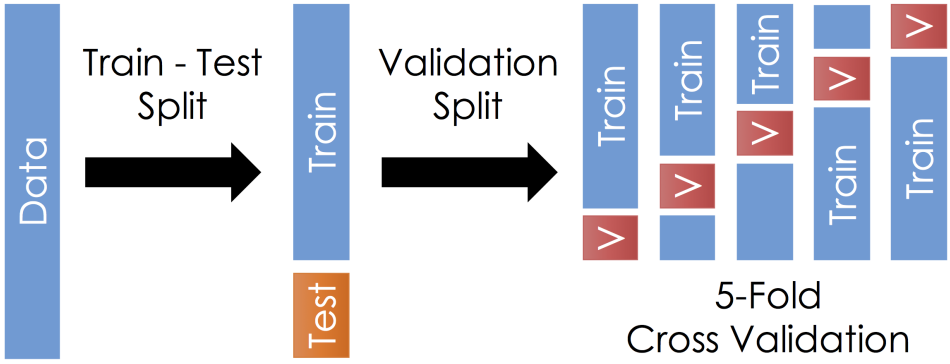
## Train-Test Split

Each split of the dataset serves a specific purpose:

| Split | Purpose | Amount of total data | How often is it used? | | ----- | ----- | ----- | ----- | | **Training set** | The model learns from this data (like the course materials you study during the semester). | ~60-80% | Always | | **Validation set** | The model gets tuned on this data (like the practice exam you take before the final exam). | ~10-20% | Often but not always | | **Testing set** | The model gets evaluated on this data to test what it has learned (like the final exam you take at the end of the semester). | ~10-20% | Always |



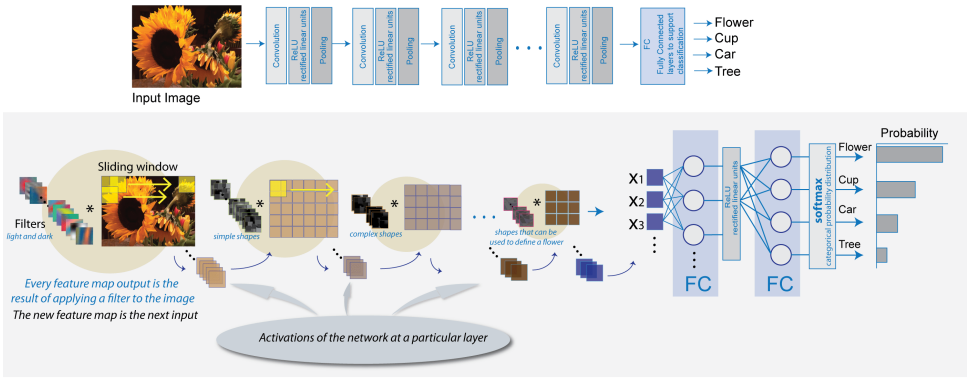



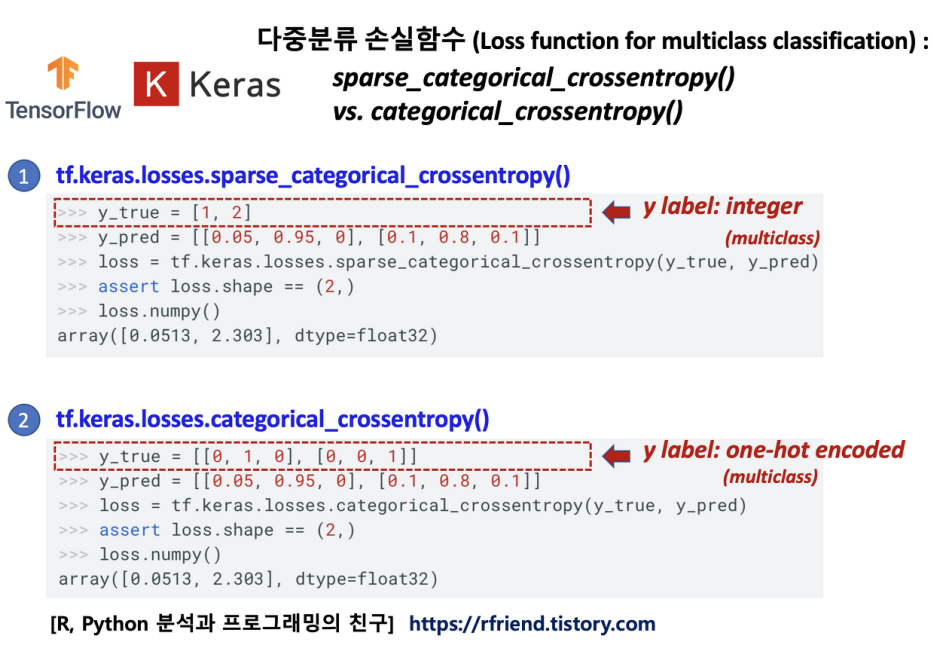




In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [ ]:
```
X_train_scaled = X_train / 255
X_test_scaled = X_test / 255
```

# 04. Model Build and Train




In [ ]:
```
num_classes = 5

model = keras.Sequential([
    #CNN
    keras.layers.Conv2D(16, 3, padding='same', activation='relu'),
    keras.layers.MaxPooling2D(),
    keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
    keras.layers.MaxPooling2D(),
    keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
    keras.layers.MaxPooling2D(),
    #DENSE
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(num_classes, activation="softmax"),
])

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

model.fit(X_train_scaled, y_train, epochs=5)
```

```
Epoch 1/5
92/92 [==============================] - 47s 498ms/step - loss: 1.3959 - accuracy:
0.3931
Epoch 2/5
92/92 [==============================] - 66s 721ms/step - loss: 1.0278 - accuracy:
0.5920
Epoch 3/5
92/92 [==============================] - 66s 720ms/step - loss: 0.8666 - accuracy:
0.6696
Epoch 4/5
92/92 [==============================] - 78s 851ms/step - loss: 0.6795 - accuracy:
0.7435
Epoch 5/5
92/92 [==============================] - 78s 850ms/step - loss: 0.4852 - accuracy:
0.8089
```

Out[ ]: <keras.callbacks.History at 0x2913416e560>

In [ ]:
```
num_classes = 5

model = keras.Sequential([
    #CNN
    keras.layers.Conv2D(16, 3, padding='same', activation='relu'),
    keras.layers.MaxPooling2D(),
    keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
    keras.layers.MaxPooling2D(),
    keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
    keras.layers.MaxPooling2D(),
    #DENSE
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(num_classes),
])

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

model.fit(X_train_scaled, y_train, epochs=5)
```

```
Epoch 1/5
92/92 [==============================] - 92s 956ms/step - loss: 1.2760 - accuracy:
0.4605
Epoch 2/5
92/92 [==============================] - 82s 890ms/step - loss: 0.9863 - accuracy:
0.6189
Epoch 3/5
92/92 [==============================] - 81s 878ms/step - loss: 0.8161 - accuracy:
0.6778
Epoch 4/5
92/92 [==============================] - 76s 826ms/step - loss: 0.6247 - accuracy:
0.7698
Epoch 5/5
92/92 [==============================] - 72s 779ms/step - loss: 0.4210 - accuracy:
0.8457
```

Out[ ]: <keras.callbacks.History at 0x291345bb640>

## 05. Model Evaluate and Predict

In [ ]:
```
model.evaluate(X_test_scaled, y_test)
```

```
23/23 [==============================] - 6s 216ms/step - loss: 1.0620 - accuracy:
0.6294
```

Out[ ]:
```
[1.0619885921478271, 0.6294277906417847]
```

In [ ]:
```
y_preds = model.predict(X_test_scaled) # num_classes=5
y_preds[:5]
```

```
23/23 [==============================] - 6s 238ms/step
```

Out[ ]:
```
array([[ 2.315916  ,  4.7269154 ,  2.9969676 , -4.931148  , -0.2683714 ],
       [ 2.597604  ,  1.3329389 , -5.3565965 , -0.9266411 ,  1.7120771 ],
       [-0.4209842 ,  1.253743  ,  4.770012  , -3.5036786 , -0.8909356 ],
       [ 1.1227976 ,  0.16080914, -5.6105275 , -3.5072706 ,  4.3161116 ],
       [-1.6905533 , -3.7538605 ,  2.4151435 ,  1.2804148 ,  0.5939594 ]],
      dtype=float32)
```

In [ ]:
```
score = tf.nn.softmax(y_preds[0])
```

In [ ]:
```
score
```

Out[ ]:
```
<tf.Tensor: shape=(5,), dtype=float32, numpy=
array([7.0436381e-02, 7.8501999e-01, 1.3917907e-01, 5.0169161e-05,
       5.3144111e-03], dtype=float32)>
```

In [ ]:
```
np.argmax(score)
```

Out[ ]:
```
1
```

In [ ]:
```
y_test[0]
```

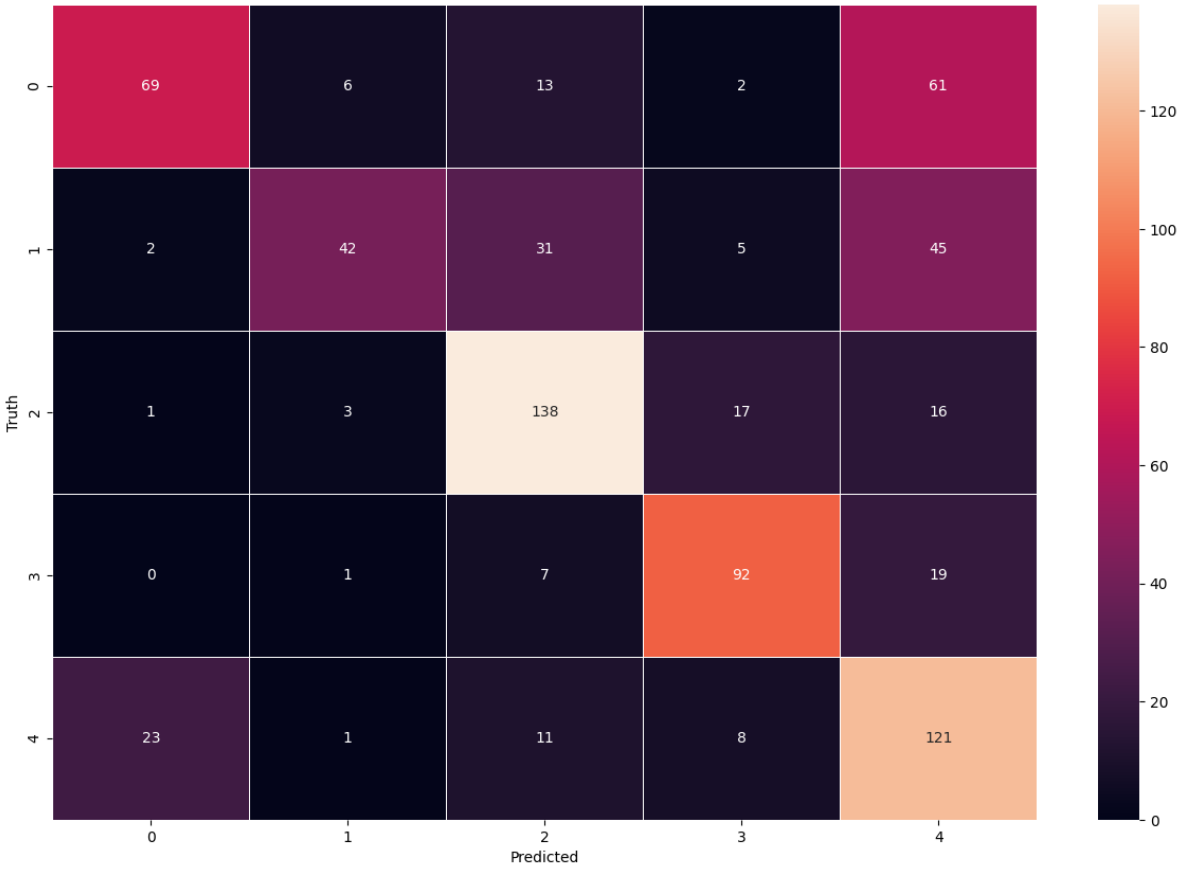
Out[ ]:
```
1
```

In [ ]:
```
y_pred_labels = [np.argmax(y_pred) for y_pred in y_preds]
y_pred_labels[:5]
```

Out[ ]:
```
[1, 0, 2, 4, 2]
```

In [ ]:
```
from sklearn.metrics import confusion_matrix

cm = tf.math.confusion_matrix(
    labels=y_test,
    predictions=y_pred_labels
)

plt.figure(figsize=(15,10))
sn.heatmap(
    cm,
    annot=True,
    fmt="d",
    linewidth=0.5,
)
plt.xlabel("Predicted")
plt.ylabel("Truth")
```

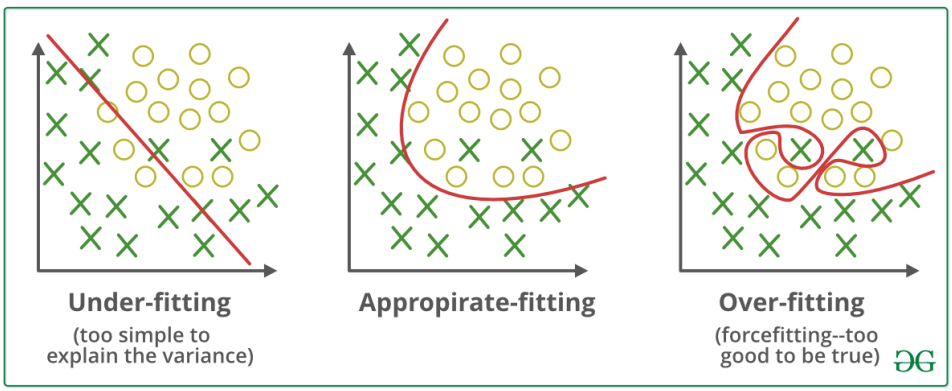
Out[ ]:
```
Text(158.22222222222223, 0.5, 'Truth')
```

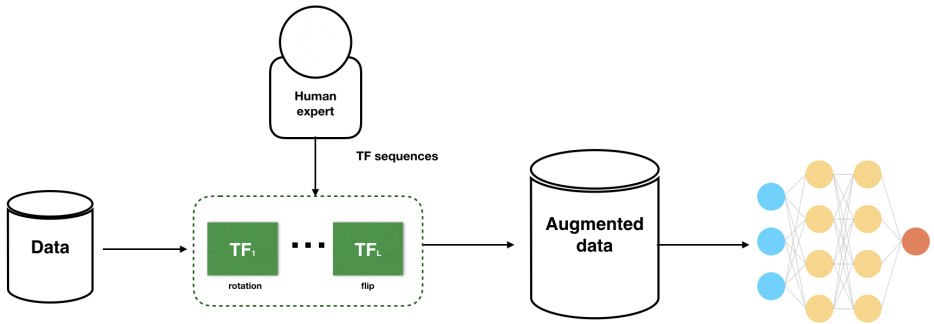


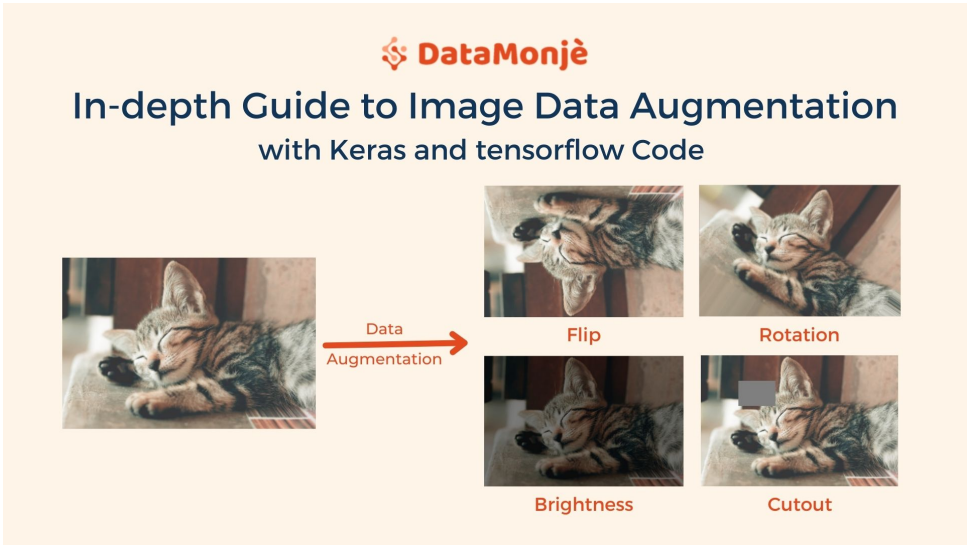

In [ ]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_labels))
```

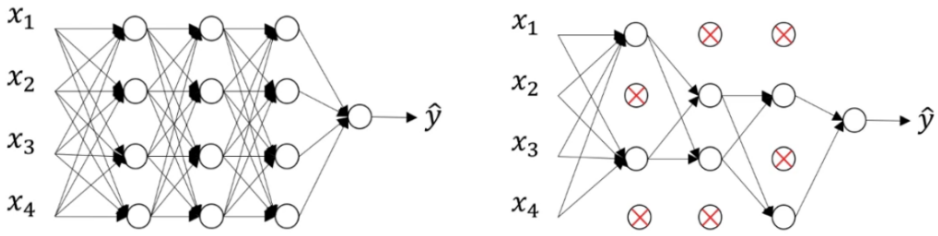
```
              precision    recall  f1-score   support

           0       0.73      0.46      0.56       151
           1       0.79      0.34      0.47       125
           2       0.69      0.79      0.74       175
           3       0.74      0.77      0.76       119
           4       0.46      0.74      0.57       164

    accuracy                           0.63       734
   macro avg       0.68      0.62      0.62       734
weighted avg       0.67      0.63      0.62       734
```

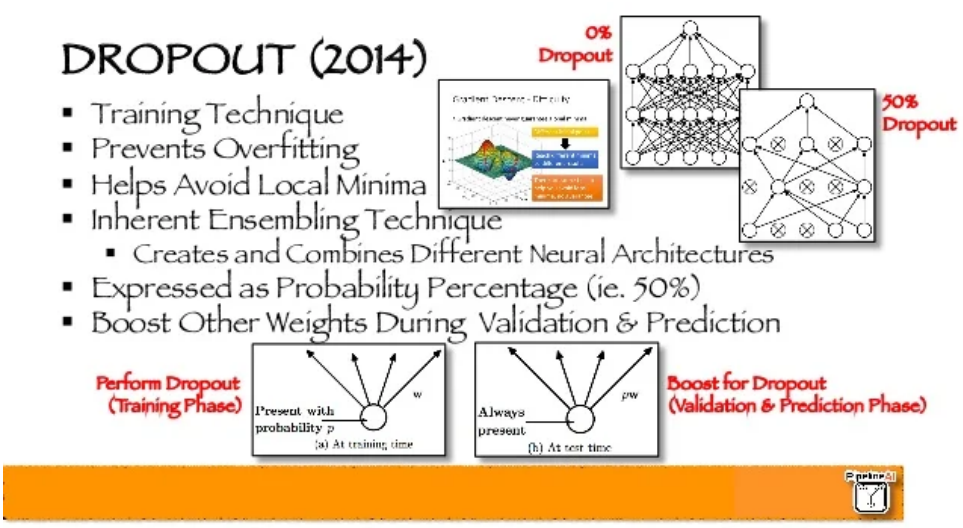
# 06. Using Data Augmentation and dropout to Address Overfitting






In [ ]:
```
data_augmentation = keras.Sequential([
    keras.layers.experimental.preprocessing.RandomZoom(0.5)
])
```

In [ ]:
```
plt.axis('off')
plt.imshow(X[0])
```

Out[ ]: <matplotlib.image.AxesImage at 0x291349aa3e0>

In [ ]:
```
type(data_augmentation(X[0]))
```

Out[ ]: tensorflow.python.framework.ops.EagerTensor

In [ ]:
```
type(X[0])
```

Out[ ]: numpy.ndarray

In [ ]:
```
type(data_augmentation(X[0]).numpy())
```

Out[ ]: numpy.ndarray

In [ ]:
```
plt.axis('off')
plt.imshow(data_augmentation(X[0]).numpy().astype("uint8"))
```

Out[ ]: <matplotlib.image.AxesImage at 0x29134a24d30>

In [ ]:
```
data_augmentation = keras.Sequential([
    keras.layers.experimental.preprocessing.RandomContrast(0.5),
])

plt.axis('off')
plt.imshow(data_augmentation(X[0]).numpy().astype("uint8"))
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x29134806500>
```

In [ ]:
```
data_augmentation = keras.Sequential([
    keras.layers.experimental.preprocessing.RandomRotation(0.5)
])
```

```
plt.axis('off')
plt.imshow(data_augmentation(X[0]).numpy().astype("uint8"))
```

Out[ ]: <matplotlib.image.AxesImage at 0x291349f6e30>

In [ ]:
```
data_augmentation = keras.Sequential([
    keras.layers.experimental.preprocessing.RandomZoom(0.6),
    keras.layers.experimental.preprocessing.RandomContrast(0.5),
    keras.layers.experimental.preprocessing.RandomRotation(0.2),
    keras.layers.experimental.preprocessing.RandomFlip(
        "horizontal",
        input_shape=(10, 10, 3)),
])

plt.axis('off')
plt.imshow(data_augmentation(X[0]).numpy().astype("uint8"))
```

Out[ ]: <matplotlib.image.AxesImage at 0x29134ab3f10>

In [ ]:
```
num_classes = 5

model = keras.Sequential([

    data_augmentation,

    keras.layers.Conv2D(16, 3, padding='same', activation='relu'),
    keras.layers.MaxPooling2D(),
    keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
    keras.layers.MaxPooling2D(),
    keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
    keras.layers.MaxPooling2D(),

    keras.layers.Dropout(0.2),

    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(num_classes)
])

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

model.fit(X_train_scaled, y_train, epochs=10)
```

```
Epoch 1/10
WARNING:tensorflow:Model was constructed with shape (180, 180, 3) for input KerasT
ensor(type_spec=TensorSpec(shape=(180, 180, 3), dtype=tf.uint8, name='random_zoom_
1_input'), name='random_zoom_1_input', description="created by layer 'random_zoom_
1_input'"), but it was called on an input with incompatible shape (None, 180, 180,
3).
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformFullInt
V2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomGetKeyCounter
cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Model was constructed with shape (180, 180, 3) for input KerasT
ensor(type_spec=TensorSpec(shape=(180, 180, 3), dtype=tf.uint8, name='random_zoom_
1_input'), name='random_zoom_1_input', description="created by layer 'random_zoom_
1_input'"), but it was called on an input with incompatible shape (None, 180, 180,
3).
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformFullInt
V2 cause there is no registered converter for this op.
```

```
WARNING:tensorflow:Using a while_loop for converting StatelessRandomGetKeyCounter
cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Model was constructed with shape (180, 180, 3) for input KerasT
ensor(type_spec=TensorSpec(shape=(180, 180, 3), dtype=tf.uint8, name='random_zoom_
1_input'), name='random_zoom_1_input', description="created by layer 'random_zoom_
1_input'"), but it was called on an input with incompatible shape (None, 180, 180,
3).
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformFullInt
V2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomGetKeyCounter
cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
92/92 [==============================] - 112s 1s/step - loss: 1.5910 - accuracy:
0.2766
Epoch 2/10
92/92 [==============================] - 95s 1s/step - loss: 1.5641 - accuracy: 0.
2987
Epoch 3/10
92/92 [==============================] - 90s 975ms/step - loss: 1.5527 - accuracy:
0.3018
Epoch 4/10
92/92 [==============================] - 90s 976ms/step - loss: 1.5365 - accuracy:
```

```
0.3035
Epoch 5/10
92/92 [==============================] - 97s 1s/step - loss: 1.5365 - accuracy: 0.
3181
Epoch 6/10
92/92 [==============================] - 98s 1s/step - loss: 1.5331 - accuracy: 0.
3120
Epoch 7/10
92/92 [==============================] - 100s 1s/step - loss: 1.5267 - accuracy:
0.3294
Epoch 8/10
92/92 [==============================] - 109s 1s/step - loss: 1.5208 - accuracy:
0.3208
Epoch 9/10
92/92 [==============================] - 93s 1s/step - loss: 1.5178 - accuracy: 0.
3256
Epoch 10/10
92/92 [==============================] - 87s 944ms/step - loss: 1.5124 - accuracy:
0.3389
```

Out[ ]: <keras.callbacks.History at 0x29134ba3d00>

In [ ]:
```
model.evaluate(X_test_scaled, y_test)
```

```
WARNING:tensorflow:Model was constructed with shape (180, 180, 3) for input KerasT
ensor(type_spec=TensorSpec(shape=(180, 180, 3), dtype=tf.uint8, name='random_zoom_
1_input'), name='random_zoom_1_input', description="created by layer 'random_zoom_
1_input'"), but it was called on an input with incompatible shape (None, 180, 180,
3).
23/23 [==============================] - 6s 240ms/step - loss: 1.5706 - accuracy:
0.3297
```

Out[ ]: [1.570554494857788, 0.3297002613544464]

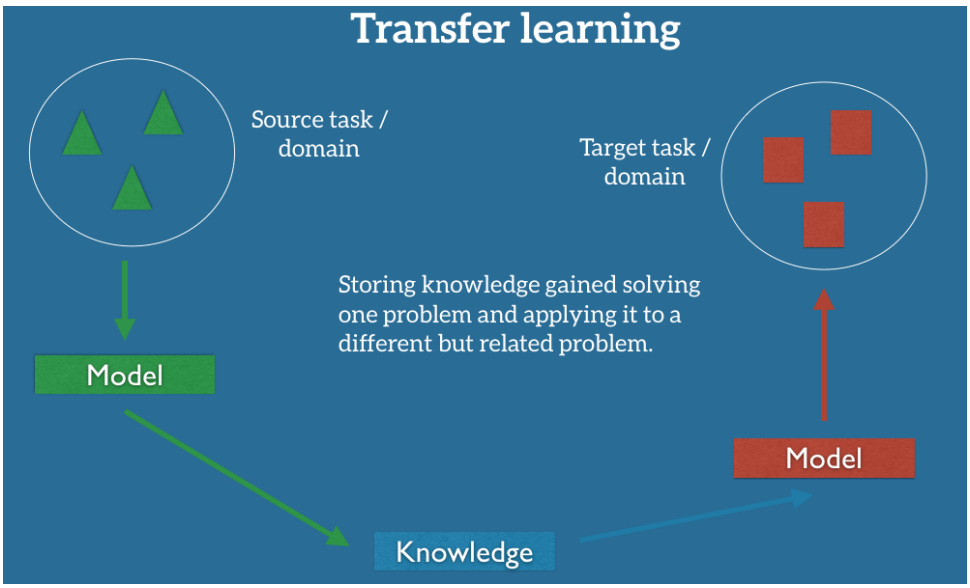
In [ ]:
```
model.summary()
```

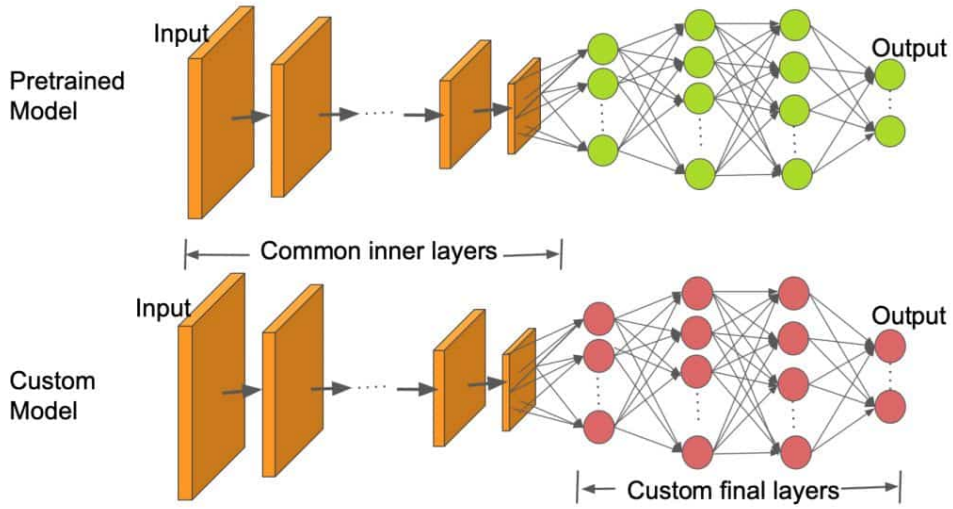
```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_5 (Sequential)   (180, 180, 3)             0

 conv2d_6 (Conv2D)           (None, 180, 180, 16)      448

 max_pooling2d_6 (MaxPooling  (None, 90, 90, 16)       0
 2D)

 conv2d_7 (Conv2D)           (None, 90, 90, 32)        4640

 max_pooling2d_7 (MaxPooling  (None, 45, 45, 32)       0
 2D)

 conv2d_8 (Conv2D)           (None, 45, 45, 64)        18496

 max_pooling2d_8 (MaxPooling  (None, 22, 22, 64)       0
 2D)

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

 flatten_2 (Flatten)         (None, 30976)             0

 dense_4 (Dense)             (None, 128)               3965056

 dense_5 (Dense)             (None, 5)                 645

=================================================================
Total params: 3,989,285
Trainable params: 3,989,285
Non-trainable params: 0
_________________________________________________________________
```

# 07. Using Transfer Learning to Train

## TensorFlow Hub

## tf2-preview/mobilenet_v2/classification






In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn

import tensorflow as tf
from tensorflow import keras
```

In [ ]:
```
dataset_url = "https://storage.googleapis.com/download.tensorflow.org/example_image

data_dir = tf.keras.utils.get_file('flower_photos', origin=dataset_url, cache_dir=
# cache_dir indicates where to download data. -> '.' which means current directory
# `untar` true will unzip it

data_dir
```

Out[ ]:
```
'.\\datasets\\flower_photos'
```

In [ ]:
```
import pathlib
data_dir = pathlib.Path(data_dir)
data_dir
```

Out[ ]:
```
WindowsPath('datasets/flower_photos')
```

In [ ]:
```
data_dir.glob('*.*')
```

Out[ ]:
```
<generator object Path.glob at 0x00000291394A7CA0>
```

In [ ]:
```
list(data_dir.glob('*/*'))[:5]
```

Out[ ]:
```
[WindowsPath('datasets/flower_photos/daisy/100080576_f52e8ee070_n.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10140303196_b88d3d6cec.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10172379554_b296050f82_n.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10172567486_2748826a8b.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10172636503_21bededa75_n.jpg')]
```

In [ ]:
```
!pip install tensorflow-hub
```

```
Requirement already satisfied: tensorflow-hub in c:\users\user\appdata\local\packa
ges\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\p
ython310\site-packages (0.12.0)
Requirement already satisfied: protobuf>=3.8.0 in c:\users\user\appdata\local\pack
ages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages
\python310\site-packages (from tensorflow-hub) (3.19.6)
Requirement already satisfied: numpy>=1.12.0 in c:\users\user\appdata\local\packag
es\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\py
thon310\site-packages (from tensorflow-hub) (1.23.3)
```

In [ ]:
```
import tensorflow_hub as hub
```

In [ ]:
```
IMAGE_SHAPE = (224, 224)

classifier_model = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/classificatic

classifier = keras.Sequential([
    hub.KerasLayer(
        classifier_model,
        input_shape=IMAGE_SHAPE+(3,))
])
```

```
WARNING:tensorflow:Please fix your imports. Module tensorflow.python.training.trac
king.data_structures has been moved to tensorflow.python.trackable.data_structure
s. The old module will be deleted in version 2.11.
WARNING:tensorflow:From C:\Users\User\AppData\Local\Packages\PythonSoftwareFoundat
ion.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\te
nsorflow\python\autograph\pyct\static_analysis\liveness.py:83: Analyzer.lamba_chec
k (from tensorflow.python.autograph.pyct.static_analysis.liveness) is deprecated a
nd will be removed after 2023-09-23.
Instructions for updating:
Lambda fuctions will be no more assumed to be used in the statement where they are
used, or at least in the same block. https://github.com/tensorflow/tensorflow/issu
es/56089
```

```
WARNING:tensorflow:From C:\Users\User\AppData\Local\Packages\PythonSoftwareFoundat
ion.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\te
nsorflow\python\autograph\pyct\static_analysis\liveness.py:83: Analyzer.lamba_chec
k (from tensorflow.python.autograph.pyct.static_analysis.liveness) is deprecated a
nd will be removed after 2023-09-23.
Instructions for updating:
Lambda fuctions will be no more assumed to be used in the statement where they are
used, or at least in the same block. https://github.com/tensorflow/tensorflow/issu
es/56089
```

In [ ]:
```
import cv2
import os
import PIL
```

In [ ]:
```
flowers_images_dict = {
    'roses': list(data_dir.glob('roses/*')),
    'daisy': list(data_dir.glob('daisy/*')),
    'dandelion': list(data_dir.glob('dandelion/*')),
    'sunflowers': list(data_dir.glob('sunflowers/*')),
    'tulips': list(data_dir.glob('tulips/*')),
}
```

In [ ]:
```
flowers_labels_dict = {
    'roses': 0,
    'daisy': 1,
    'dandelion': 2,
```

```
    'sunflowers': 3,
    'tulips': 4,
}
```

In [ ]:
```
img = cv2.imread(str(flowers_images_dict['roses'][0]))
img.shape
```

Out[ ]:
```
(240, 179, 3)
```

In [ ]:
```
img = cv2.resize(img, IMAGE_SHAPE)
img.shape
```

Out[ ]:
```
(224, 224, 3)
```

In [ ]:
```
for flower_name, images in flowers_images_dict.items():
    print(flower_name, len(images))
```

```
roses 641
daisy 633
dandelion 898
sunflowers 699
tulips 799
```

In [ ]:
```
X = []
y = []
for flower_name, images in flowers_images_dict.items():
    for image in images:
        img = cv2.imread(str(image))
        img_resized = cv2.resize(img, IMAGE_SHAPE)
        X.append(img_resized)
        y.append(flowers_labels_dict[flower_name])
```

In [ ]:
```
X = np.array(X)
y = np.array(y)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [ ]:
```
X_train_scaled = X_train / 255
X_test_scaled = X_test / 255
```

In [ ]:
```
plt.axis("off")
plt.imshow(X[0])
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x2913497e470>
```

In [ ]:
```
plt.axis("off")
plt.imshow(X[1])
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x291345e8e80>
```

In [ ]:
```
plt.axis("off")
plt.imshow(X[2])
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x29139e90d60>
```

In [ ]:
```
preds = classifier.predict(np.array([X[0], X[1], X[2]]))
preds_labels = np.argmax(preds, axis=1)
preds_labels
```

```
1/1 [==============================] - 1s 889ms/step
```

Out[ ]:
```
array([795, 880, 795], dtype=int64)
```

In [ ]:
```
x0_resized = cv2.resize(X[0], IMAGE_SHAPE)
x1_resized = cv2.resize(X[1], IMAGE_SHAPE)
x2_resized = cv2.resize(X[2], IMAGE_SHAPE)
```

In [ ]:
```
preds = classifier.predict(np.array([x0_resized, x1_resized, x2_resized]))
preds_labels = np.argmax(preds, axis=1)
preds_labels
```

```
1/1 [==============================] - 0s 82ms/step
```

Out[ ]:
```
array([795, 880, 795], dtype=int64)
```

## Take pre-trained model and retrain it using flowers images

In [ ]:
```
feature_extractor_model = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/featur
```

In [ ]:
```
pretrained_model_without_top_layer = hub.KerasLayer(
    feature_extractor_model,
    input_shape=(224, 224, 3),
    trainable=False
)
```

In [ ]:
```
num_of_flowers = 5

model = keras.Sequential([
  pretrained_model_without_top_layer,
```

```
  keras.layers.Dense(num_of_flowers)
])
```

In [ ]:
```
preds = model.predict(np.array([x0_resized, x1_resized, x2_resized]))
preds_labels = np.argmax(preds, axis=1)
preds_labels
```

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

Out[ ]:
```
array([4, 4, 4], dtype=int64)
```

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

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_1 (KerasLayer)  (None, 1280)              2257984

 dense_6 (Dense)             (None, 5)                 6405

=================================================================
Total params: 2,264,389
Trainable params: 6,405
Non-trainable params: 2,257,984
_________________________________________________________________
```

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

model.fit(X_train_scaled, y_train, epochs=10)
```

```
Epoch 1/10
92/92 [==============================] - 110s 1s/step - loss: 0.8404 - accuracy:
0.6829
Epoch 2/10
92/92 [==============================] - 119s 1s/step - loss: 0.4113 - accuracy:
0.8600
Epoch 3/10
92/92 [==============================] - 123s 1s/step - loss: 0.3167 - accuracy:
0.9009
Epoch 4/10
92/92 [==============================] - 109s 1s/step - loss: 0.2692 - accuracy:
0.9149
Epoch 5/10
92/92 [==============================] - 109s 1s/step - loss: 0.2284 - accuracy:
0.9319
Epoch 6/10
92/92 [==============================] - 106s 1s/step - loss: 0.2024 - accuracy:
0.9496
Epoch 7/10
92/92 [==============================] - 108s 1s/step - loss: 0.1804 - accuracy:
0.9503
Epoch 8/10
92/92 [==============================] - 116s 1s/step - loss: 0.1578 - accuracy:
0.9595
Epoch 9/10
92/92 [==============================] - 113s 1s/step - loss: 0.1434 - accuracy:
0.9666
Epoch 10/10
92/92 [==============================] - 123s 1s/step - loss: 0.1287 - accuracy:
0.9717
```

Out[ ]:
```
<keras.callbacks.History at 0x29148f1df60>
```

In [ ]:
```
model.evaluate(X_test_scaled,y_test)
```

```
23/23 [==============================] - 30s 1s/step - loss: 0.3947 - accuracy: 0.
8651
```

Out[ ]:
```
[0.3946790099143982, 0.8651226162910461]
```

-- Project End --