Business Problem

In today's digital age, automating the recognition of animals in images has become increasingly essential across various domains. The project aims to address this need by developing an accurate image classification system capable of distinguishing between dogs and cats. Leveraging Convolutional Neural Networks (CNNs), the system will be trained on a diverse dataset of images to automatically extract relevant features and make precise predictions. Despite challenges such as variations in animal poses and backgrounds, the project seeks to streamline the process of animal identification, contributing to advancements in computer vision technology with potential applications in wildlife conservation, veterinary science, and beyond.

In [2]:
```
# importing necessary libraries
import tensorflow as tf
import keras
```

```
WARNING:tensorflow:From C:\Users\PC\anaconda3\lib\site-packages\keras\src\losses.p
y:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [3]:
```
from keras.preprocessing.image import ImageDataGenerator
```

In [4]:
```
# Set up ImageDataGenerator for data augmentation and preprocessing
train_datagen = ImageDataGenerator(rescale=1/255, # Rescale pixel values to the ran
                                   shear_range=0.2,# Apply random shearing transform
                                   zoom_range=0.2)# Apply random zooming transformat
```

Importing Training data

In [5]:
```
training_set = train_datagen.flow_from_directory('dataset/training_set',
                                                 target_size=(64,64),
                                                 class_mode='binary')
```

```
Found 8048 images belonging to 2 classes.
```

In [6]:
```
training_set.class_indices
```

Out[6]:
```
{'cats': 0, 'dogs': 1}
```

Dog and Cat are indicated as 1 and 0 respectively.

Importing Testing data

In [7]:
```
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [8]:
```
testing_set = test_datagen.flow_from_directory('dataset/test_set',
                                               target_size=(64,64),
                                               class_mode='binary')
```

```
Found 2000 images belonging to 2 classes.
```

In [9]:
```
testing_set.class_indices
```

Out[9]:
```
{'cats': 0, 'dogs': 1}
```

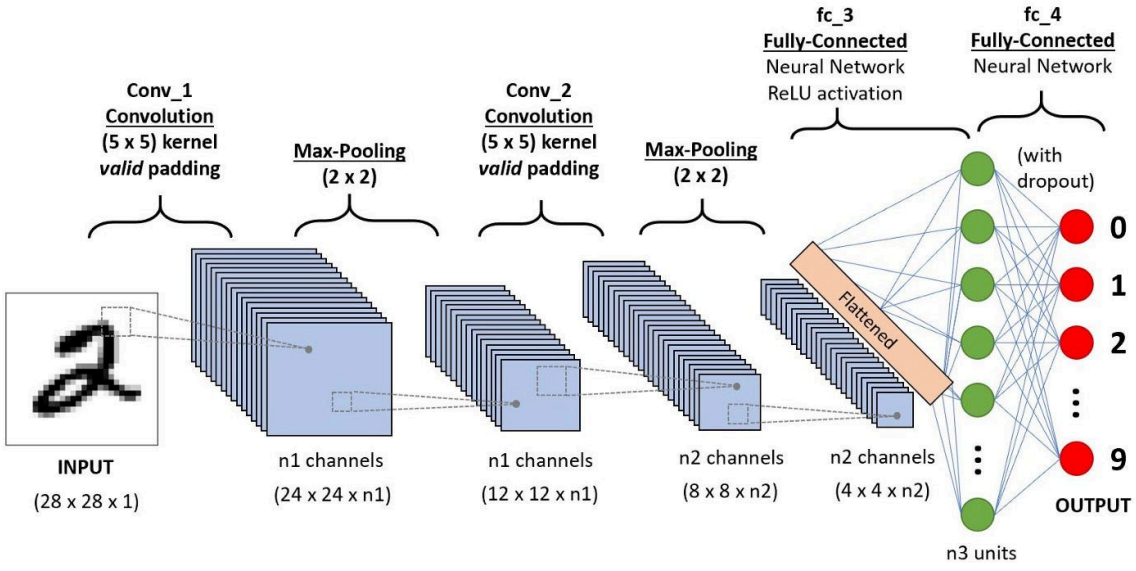
Modelling- Convolution Neural Network

initialising the CNN

Convolutional Neural Networks (CNNs) are a class of deep learning models designed specifically for processing structured grid data, such as images. They consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

In brief, CNNs work by learning hierarchical representations of input images through a process of convolution and pooling operations. The convolutional layers apply filters to extract features from the input image, capturing patterns like edges and textures. The pooling layers then downsample the feature maps, reducing their spatial dimensions while preserving important information. Finally, the fully connected layers combine the extracted features to make predictions about the input image's class or category.

By automatically learning hierarchical representations, CNNs excel at tasks like image classification, object detection, and segmentation, making them indispensable in computer vision applications.



In [12]:
```
from keras.models import Sequential
classifier = Sequential()
```

```
WARNING:tensorflow:From C:\Users\PC\anaconda3\lib\site-packages\keras\src\backend.
py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_d
efault_graph instead.
```

step 1- Convolution

In [13]:
```
from keras.layers import Conv2D
classifier.add(Conv2D(input_shape=[64,64,3], filters=32,kernel_size=3,activation='r
```

Step 2 - Max Pooling

In [14]:
```
from keras.layers import MaxPooling2D
classifier.add(MaxPooling2D(pool_size=2,strides=2))
```

```
WARNING:tensorflow:From C:\Users\PC\anaconda3\lib\site-packages\keras\src\layers\p
ooling\max_pooling2d.py:161: The name tf.nn.max_pool is deprecated. Please use tf.
nn.max_pool2d instead.
```

Step 3 - Flattening

In [15]:
```
from keras.layers import Flatten
classifier.add(Flatten())
```

Step 4- Full Connection

In [16]:
```
from keras.layers import Dense

# Hidden Layers wth 128 neurons
classifier.add(Dense(units=128,activation='relu'))

#output layer
classifier.add(Dense(units=1,activation='sigmoid'))
```

In [17]:
```
classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy']
```

```
WARNING:tensorflow:From C:\Users\PC\anaconda3\lib\site-packages\keras\src\optimize
rs\__init__.py:309: The name tf.train.Optimizer is deprecated. Please use tf.compa
t.v1.train.Optimizer instead.
```

In [18]:
```
classifier.fit(x= training_set,validation_data = testing_set,epochs=25)
```

```
Epoch 1/25
WARNING:tensorflow:From C:\Users\PC\anaconda3\lib\site-packages\keras\src\utils\tf
_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.c
ompat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\PC\anaconda3\lib\site-packages\keras\src\engine\b
ase_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is depreca
ted. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

252/252 [==============================] - 52s 200ms/step - loss: 0.6893 - accurac
y: 0.6126 - val_loss: 0.6423 - val_accuracy: 0.6385
Epoch 2/25
252/252 [==============================] - 52s 205ms/step - loss: 0.5732 - accurac
y: 0.7066 - val_loss: 0.5637 - val_accuracy: 0.7225
Epoch 3/25
252/252 [==============================] - 56s 222ms/step - loss: 0.5375 - accurac
y: 0.7296 - val_loss: 0.5727 - val_accuracy: 0.7160
Epoch 4/25
252/252 [==============================] - 54s 215ms/step - loss: 0.5164 - accurac
y: 0.7399 - val_loss: 0.5471 - val_accuracy: 0.7360
Epoch 5/25
252/252 [==============================] - 53s 211ms/step - loss: 0.5022 - accurac
y: 0.7479 - val_loss: 0.5505 - val_accuracy: 0.7440
Epoch 6/25
252/252 [==============================] - 58s 230ms/step - loss: 0.4854 - accurac
y: 0.7679 - val_loss: 0.5771 - val_accuracy: 0.7355
Epoch 7/25
252/252 [==============================] - 62s 247ms/step - loss: 0.4612 - accurac
y: 0.7732 - val_loss: 0.5471 - val_accuracy: 0.7410
Epoch 8/25
252/252 [==============================] - 75s 297ms/step - loss: 0.4364 - accurac
y: 0.7965 - val_loss: 0.5704 - val_accuracy: 0.7485
Epoch 9/25
252/252 [==============================] - 68s 268ms/step - loss: 0.4220 - accurac
y: 0.8031 - val_loss: 0.6073 - val_accuracy: 0.7330
Epoch 10/25
252/252 [==============================] - 120s 477ms/step - loss: 0.4078 - accura
cy: 0.8121 - val_loss: 0.5393 - val_accuracy: 0.7530
Epoch 11/25
252/252 [==============================] - 193s 768ms/step - loss: 0.3708 - accura
cy: 0.8336 - val_loss: 0.6559 - val_accuracy: 0.7355
Epoch 12/25
252/252 [==============================] - 116s 461ms/step - loss: 0.3592 - accura
cy: 0.8386 - val_loss: 0.5750 - val_accuracy: 0.7515
Epoch 13/25
252/252 [==============================] - 56s 224ms/step - loss: 0.3366 - accurac
y: 0.8545 - val_loss: 0.6400 - val_accuracy: 0.7570
Epoch 14/25
252/252 [==============================] - 59s 233ms/step - loss: 0.3146 - accurac
y: 0.8625 - val_loss: 0.6723 - val_accuracy: 0.7400
Epoch 15/25
252/252 [==============================] - 53s 209ms/step - loss: 0.2884 - accurac
y: 0.8762 - val_loss: 0.7018 - val_accuracy: 0.7380
Epoch 16/25
252/252 [==============================] - 58s 230ms/step - loss: 0.2757 - accurac
y: 0.8806 - val_loss: 0.6821 - val_accuracy: 0.7550
Epoch 17/25
252/252 [==============================] - 60s 237ms/step - loss: 0.2575 - accurac
y: 0.8945 - val_loss: 0.7047 - val_accuracy: 0.7400
Epoch 18/25
252/252 [==============================] - 62s 245ms/step - loss: 0.2247 - accurac
y: 0.9100 - val_loss: 0.7964 - val_accuracy: 0.7305
Epoch 19/25
252/252 [==============================] - 79s 314ms/step - loss: 0.2204 - accurac
```

```
y: 0.9124 - val_loss: 0.7578 - val_accuracy: 0.7555
Epoch 20/25
252/252 [==============================] - 79s 314ms/step - loss: 0.2057 - accurac
y: 0.9179 - val_loss: 0.8105 - val_accuracy: 0.7465
Epoch 21/25
252/252 [==============================] - 74s 295ms/step - loss: 0.1809 - accurac
y: 0.9262 - val_loss: 0.8147 - val_accuracy: 0.7595
Epoch 22/25
252/252 [==============================] - 78s 311ms/step - loss: 0.1734 - accurac
y: 0.9330 - val_loss: 0.8620 - val_accuracy: 0.7405
Epoch 23/25
252/252 [==============================] - 75s 299ms/step - loss: 0.1631 - accurac
y: 0.9371 - val_loss: 0.8232 - val_accuracy: 0.7545
Epoch 24/25
252/252 [==============================] - 76s 303ms/step - loss: 0.1571 - accurac
y: 0.9401 - val_loss: 0.8710 - val_accuracy: 0.7555
Epoch 25/25
252/252 [==============================] - 76s 300ms/step - loss: 0.1396 - accurac
y: 0.9487 - val_loss: 0.9414 - val_accuracy: 0.7580
```

Out[18]: <keras.src.callbacks.History at 0x2036cb73640>

Evolution

In [19]:
```
import numpy as np
from PIL import Image
```

Evolution Image

In [26]:
```
test_image = Image.open("dataset/single_prediction/cat_or_dog_2.jpg")

# data preprocessing
test_image = test_image.resize((64,64))
test_image = np.array(test_image)
test_image = np.expand_dims(test_image,axis=0)
```

```python
# Prediction
result = classifier.predict(test_image)

# Evaluation

if result[0][0]==1:
    print("Dog")
else:
    print("CAt")
```

```
1/1 [==============================] - 0s 41ms/step
CAt
```