KNOW YOUR HABITAT G-24

PROJECT: MANDI LENS

TABLE OF CONTENTS

01 PROBLEM STATEMENT

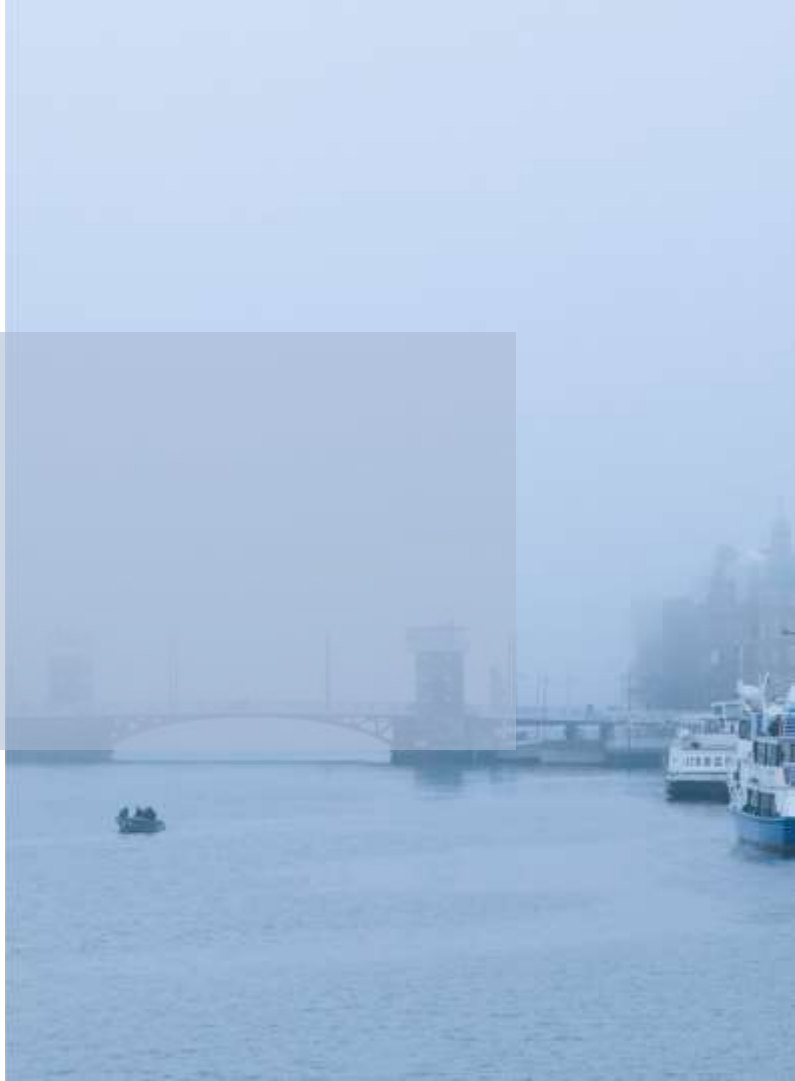
02 VGG 19 NEURAL NETWORK

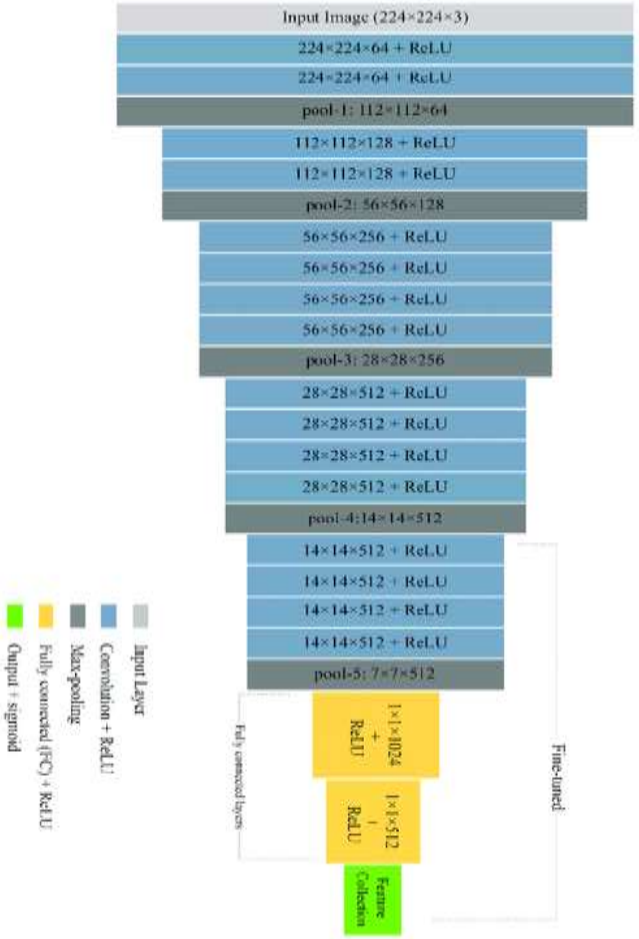
03 CODE EXPLANATION

04 CONVOLUTIONAL NEURAL NETWORKS (CNN)

PROBLEM STATEMENT

To develop a (web/android) platform which performs reverse image search, and tells the objects present in the image.





WHY CHOOSE VGG 19 AS OUR PRE-BUILT MODEL ?

VGG-19 is a convolutional neural network that is 19 layers deep. You can load a pre trained version of the network trained on more than a million images. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

VGG-19 NEURAL NETWORK

ZERO - CENTRE - NORMALIZATION

The Centralization and Normalization towards Origo. It normalizes and reduces dimensions - to keep scale centralized - in terms of when we will perform Convolutions later on.

The reason this is important - is to bring everything “down in line” in a normalized, streamlined and orderly fashion - so we have some sense of normality condition.

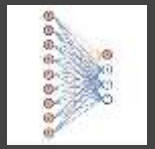
As in - we want the general structure of what we are parsing - to be normalized and centralized - so that we have a pre-defined boundary that we are being relative towards.

CONVOLUTION

Convolutions is the functional operation of performing concatenation of Functional Curves - so that they “add up to encapsulate how they affect each other - in terms of multiplicative relationships” - roughly speaking.

What you’re doing - is that you are basically “multiplying a function relationship with another” - so you get the average of how they affect each other - on average.

VGG-19 NEURAL NETWORK



A FULLY CONNECTED LAYER

A Fully Connected Layer is a layer that is utilized for Classification.

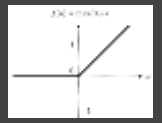
Since this is used so sparingly - it's at the end - when the rough feature extraction and averaging of Functional relationships have had it's sequence. Now - it summarizes to run tally on the total output.

MAX POOLING

Max Pooling is when you pool together the largest sample you can find - in an average area of taking strides. Strides - is the average functional kernel mapping - that you have in a square diagram plot - that averages out the value samplings of a certain space .So - you may have a 8x8 total square - with 4 4x4 squares - So - you reduce that to being the max of every 4x4 square - put them together - and get a 4x4 total Square to replace your earlier 8x8.

This effectively takes "the biggest impact features" and there of, "the largest information to retrieve" (read: coarsest features) - And then ignores the smaller features. So you are left with a smaller sample space (lower dimensionality) - But - you keep the coarsest features - you keep the largest representative of information.

VGG-19 NEURAL NETWORK



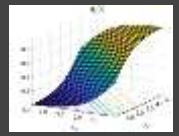
ReLU

ReLU is a rectifying linear unit. A unit that is made to rectify and compensate for signal parsing errors - such as when we are forced to encapsulate epsilon (infinitely small) - and we have to work around that.

So ReLU “Steers back the signal” to where it’s supposed to be headed - so that the signal does not explode or vanish.

There of - ReLU’s act kind of like “Safety stops” on the sides of the roads - so you don’t steer off the road - when you are driving..

SOFTMAX



Softmax is a applicative algorithm that normalizes a distribution dynamic - from K amount of composite functional concatenations. This means, that instead of having a spread where the values can be anything :- 0, -1, 1, 2, etc.

They are all normalized to be in line to a distribution (as the Softmax act as a regularizer over a distribution).

And then there of generalized to be across the j-th Linear function as it has become baked into the Distribution we have regressed forward with Softmax.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

VGG-19 NEURAL NETWORK

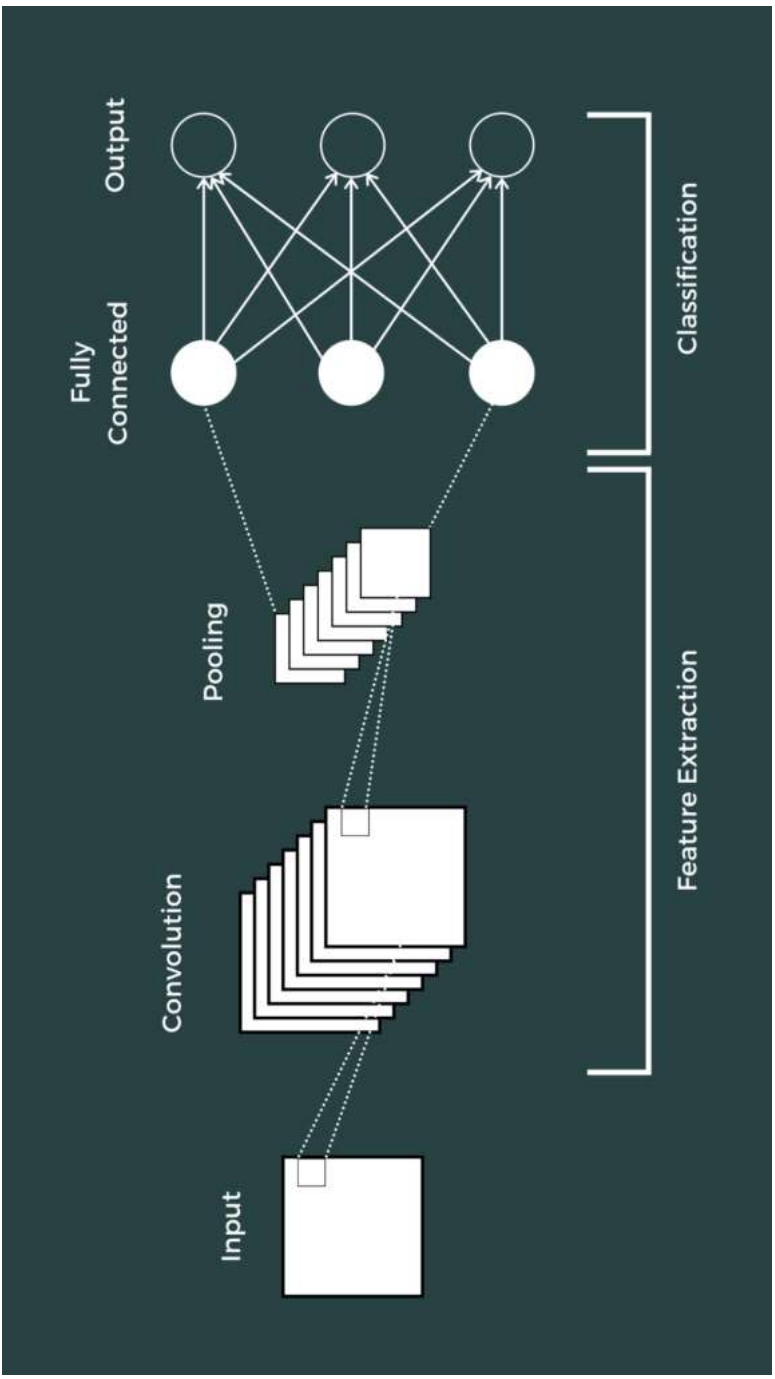
CLASSIFICATION OUTPUT

Classification output is pretty much what it sounds like.

It's the output of classification where it utilizes Cross Entropy to maximize the probability in terms of where each K-th unit came from and where it most likely belongs.

There of it stands in relation to the previous K functional relationships where it classifies and infers the probability of placement - of each.

CONVOLUTIONAL NEURAL NETWORK



CONVOLUTIONAL NEURAL NETWORK

Convolutional neural networks (CNNs) are used for classification and computer vision tasks. Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

- Convolutional layer
- Pooling layer
- Fully-connected (FC) layer

The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image.

Convolutional Layer

The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map. Let's assume that the input will be a color image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimensions—a height, width, and depth—which correspond to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution.

CONVOLUTIONAL NEURAL NETWORK

The feature detector is a two-dimensional (2-D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterwards, the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature.

Each output value in the feature map does not have to connect to each pixel value in the input image. It only needs to connect to the receptive field, where the filter is being applied. Since the output array does not need to map directly to each input value, convolutional (and pooling) layers are commonly referred to as “partially connected” layers. However, this characteristic can also be described as local connectivity.

Note that the weights in the feature detector remain fixed as it moves across the image, which is also known as parameter sharing. Some parameters, like the weight values, adjust during training through the process of backpropagation and gradient descent. However, there are three hyperparameters which affect the volume size of the output that need to be set before the training of the neural network begins. These include:

CONVOLUTIONAL NEURAL NETWORK

1. **Number of filters** affects the depth of the output. For example, three distinct filters would yield three different feature maps, creating a depth of three.

2. **Stride** is the distance, or number of pixels, that the kernel moves over the input matrix. While stride values of two or greater is rare, a larger stride yields a smaller output.

3. **Zero-padding** is usually used when the filters do not fit the input image. This sets all elements that fall outside of the input matrix to zero, producing a larger or equally sized output. There are three types of padding:

- **Valid padding:** This is also known as no padding. In this case, the last convolution is dropped if dimensions do not align.
- **Same padding:** This padding ensures that the output layer has the same size as the input layer
- **Full padding:** This type of padding increases the size of the output by adding zeros to the border of the input.

After each convolution operation, a CNN applies a Rectified Linear Unit (ReLU) transformation to the feature map, introducing nonlinearity to the model.

CONVOLUTIONAL NEURAL NETWORK

Pooling Layer

Pooling layers, also known as downsampling, conducts dimensionality reduction, reducing the number of parameters in the input. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array. There are two main types of pooling:

- **Max pooling**: As the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.
- **Average pooling:** As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.

While a lot of information is lost in the pooling layer, it also has a number of benefits to the CNN. They help to reduce complexity, improve efficiency, and limit risk of overfitting.

```
input_1 (InputLayer)         [(None, 299, 299, 3)]     0
block1_conv1 (Conv2D)        (None, 299, 299, 64)      1792
block1_conv2 (Conv2D)        (None, 299, 299, 64)      36928
block1_pool (MaxPooling2D)   (None, 149, 149, 64)      0
block2_conv1 (Conv2D)        (None, 149, 149, 128)     73856
block2_conv2 (Conv2D)        (None, 149, 149, 128)     147584
block2_pool (MaxPooling2D)   (None, 74, 74, 128)       0
block3_conv1 (Conv2D)        (None, 74, 74, 256)       295168
block3_conv2 (Conv2D)        (None, 74, 74, 256)       590080
block3_conv3 (Conv2D)        (None, 74, 74, 256)       590080
block3_conv4 (Conv2D)        (None, 74, 74, 256)       590080
block3_pool (MaxPooling2D)   (None, 37, 37, 256)       0
block4_conv1 (Conv2D)        (None, 37, 37, 512)       1180160
block4_conv2 (Conv2D)        (None, 37, 37, 512)       2359808
block4_conv3 (Conv2D)        (None, 37, 37, 512)       2359808
block4_conv4 (Conv2D)        (None, 37, 37, 512)       2359808
block4_pool (MaxPooling2D)   (None, 18, 18, 512)       0
block5_conv1 (Conv2D)        (None, 18, 18, 512)       2359808
block5_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
block5_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
block5_conv4 (Conv2D)        (None, 18, 18, 512)       2359808
block5_pool (MaxPooling2D)   (None, 9, 9, 512)         0
flatten (Flatten)            (None, 41472)             0
dense (Dense)                (None, 8)                 331784
=================================================================
Total params: 20,356,168
Trainable params: 331,784
Non-trainable params: 20,024,384
```

CONVOLUTIONAL NEURAL NETWORK

Fully-Connected Layer

The name of the fully-connected layer aptly describes itself. As mentioned earlier, the pixel values of the input image are not directly connected to the output layer in partially connected layers. However, in the fully-connected layer, each node in the output layer connects directly to a node in the previous layer.

This layer performs the task of classification based on the features extracted through the previous layers and their different filters. While convolutional and pooling layers tend to use ReLu functions, FC layers usually leverage a softmax activation function to classify inputs appropriately, producing a probability from 0 to 1.

CONVOLUTIONAL NEURAL NETWORK

Convolutional neural networks and computer vision

Convolutional neural networks power image recognition and computer vision tasks. Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs, and based on those inputs, it can take action. This ability to provide recommendations distinguishes it from image recognition tasks. Some common applications of this computer vision today can be seen in:

- **Marketing:** Social media platforms provide suggestions on who might be in photograph that has been posted on a profile, making it easier to tag friends in photo albums.
- **Healthcare:** Computer vision has been incorporated into radiology technology, enabling doctors to better identify cancerous tumors in healthy anatomy.
- **Retail:** Visual search has been incorporated into some e-commerce platforms, allowing brands to recommend items that would complement an existing wardrobe.
- **Automotive**: While the age of driverless cars hasn't quite emerged, the underlying technology has started to make its way into automobiles, improving driver and passenger safety through features like lane line detection.

```
from keras.utils import img_to_array
import numpy as np
#t_img = test_path + 'OAK/20221125_143623.jpg'
t_img = test_path + 'A9/20221125_141501.jpg'
```

```
from PIL import Image
from matplotlib import pyplot as plt

img = Image.open(t_img)
img = img.rotate(270)
plt.axis("off")
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fb13ff0e290>
```

INPUT FORMAT

OUTPUT FORMAT