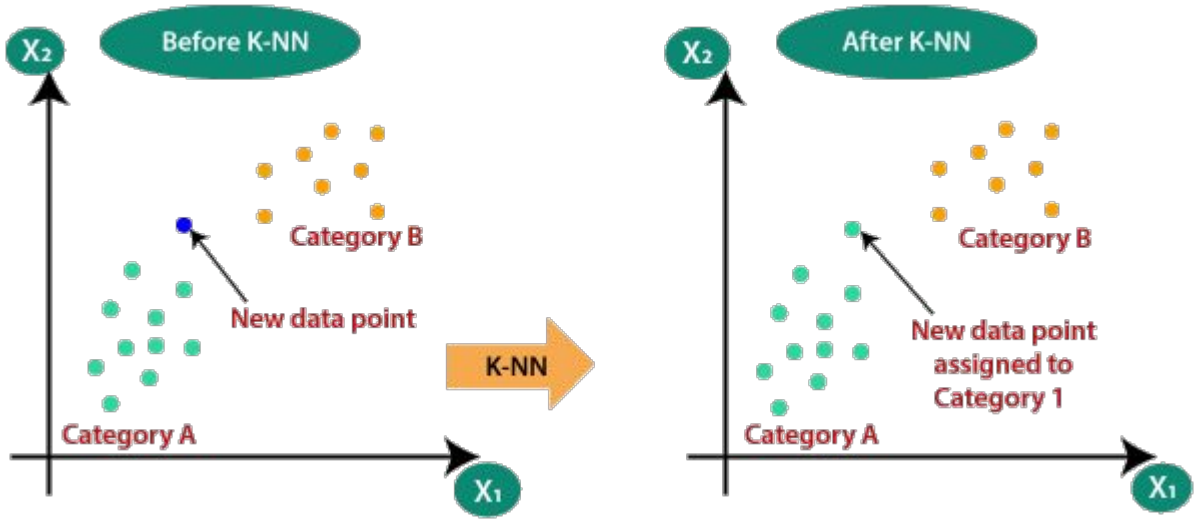
Knn as an Algorithm

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

KNN Algorithm



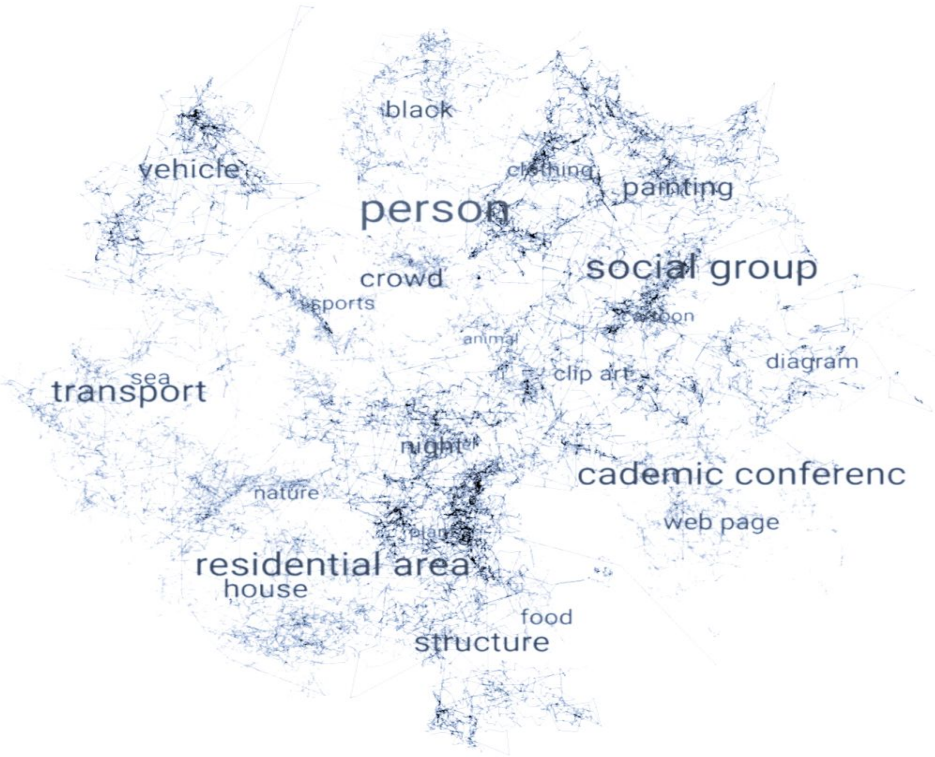
How Does it Work ?

Depth Wise Separable Convolution

Higher Dimensional Database and Embedding Space

- High-dimensional spaces arise as a way of modelling datasets with many attributes .
- Such a dataset can be directly represented in a space spanned by its attributes, with each record represented as a point in the space with its position depending on its attribute values.
- Such spaces are not easy to work with because of their high dimensionality .
- A low-resolution image might be a 32-by-32 pixel thumbnail. Even though it's represented visually as a square, you can imagine stretching it into a line with 1024 = 32 × 32 pixels. **So what's the point?** What space does this image live in? Well, we now have 1024 real numbers, so it must exist in the 1024-dimensional space $\mathbb{R}^{1024}$.

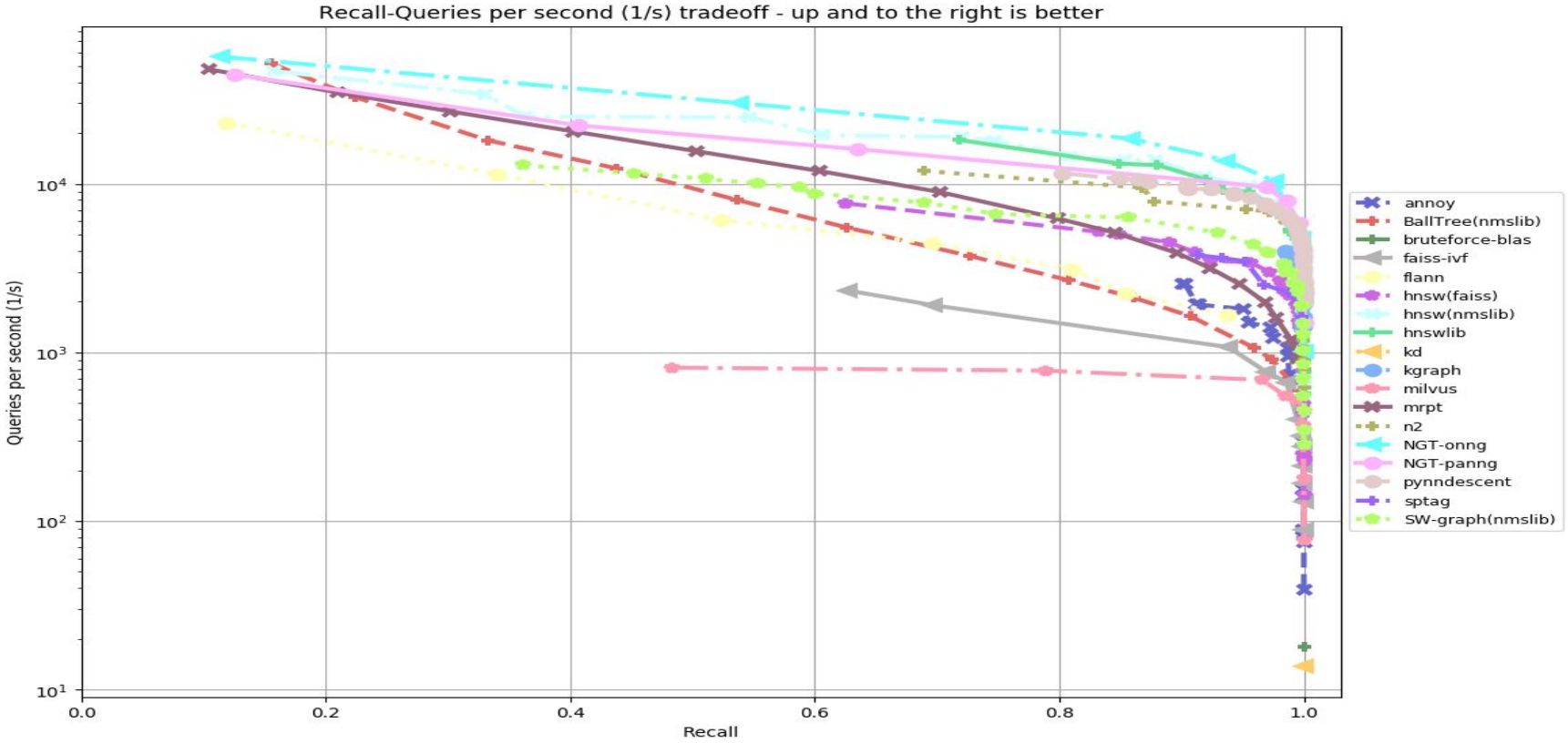
Embedding Space



Benchmarking nearest neighbors

- https://github.com/erikbern/ann-benchmarks
- Some of the major used HDDB are as follows :
  1) Annoy
  2) Faiss
  3) Elastic-Knn
  4) NMSLIB

Comparisons of Recall-Queries for famous HDDB



ALL About Data : Imaterialist Product

- https://www.kaggle.com/c/imaterialist-product-2019/overview
- Contains products by labels ,and has product such as beer,cloths,furniture etc.
- Contains id, class, url for each file in json , So we will need an python script to download the dataset as image as we are working on Image-to-Image Search.

ElasticSearch Setup with Knn plugin

- https://opendistro.github.io/for-elasticsearch-docs/docs/install

  Click on the above link and follow basic instruction given there!!

ElasticSearch (KNN) from AWS Docs

- https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/knn.html

  Click on the above link and follow the instructions .

Feature Extraction using Pretrained Models

- TensorFlow has a module tensorflow.keras.applications which holds a number of pre-trained deep learning models. When loading any of such models, not only their architectures are existing but also their trained weights.
- So, you are ready to either use them for making predictions or transfer learning. What is the difference between loading the model for making a prediction or transfer learning?
- When a model is loaded for making predictions, then the entire model is loaded including the last fully connected (FC) layers. For transfer learning, these layers are not included. But why not loading these layers for transfer learning? The answer is simple.

If a model is trained by a given dataset, then the last FC layers will have a number of neurons equal to the number of classes within this dataset. For example, a dataset might have 1,000 classes and thus the last FC layers will have 1,000 neurons. For transfer learning, suppose that the new dataset has a number of classes equal to 102 as. Thus, we need the last FC layers to have 102 neurons rather than 1,000 neurons. As a result, the last FC layers in the original model is no longer needed and we have to remove them and replace them by FC layers working with just 102 classes.

According to the above discussion, there are 2 steps which are as follows:

1. Removing the last FC layers from the original model.
2. Flattening the output feature extracted!!

MobilenetV2 Architecture Explained!!

- MobileNetV1, **Depthwise Separable Convolution** is introduced which dramatically reduce the complexity cost and model size of the network .
- In MobileNetV2, a better module is introduced with **inverted residual structure**. **Non-linearities in narrow layers are removed** this time.
- With MobileNetV2 as backbone for feature extraction, state-of-the-art performances are also achieved for object detection and semantic segmentation.

MobilenetV1 vs MobilenetV2

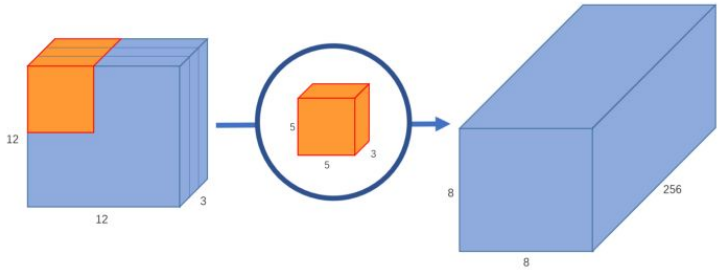
1. MobilenetV1

- In MobilenetV1, there are 2 layers.
- The **first layer** is called a **depthwise convolution**, it performs lightweight filtering by applying a single convolutional filter per input channel.
- The **second layer** is a **1×1 convolution**, called a **pointwise convolution**, which is responsible for building new features through computing linear combinations of the input channels.
- **ReLU6** is used here for comparison.

1.2. MobilenetV2

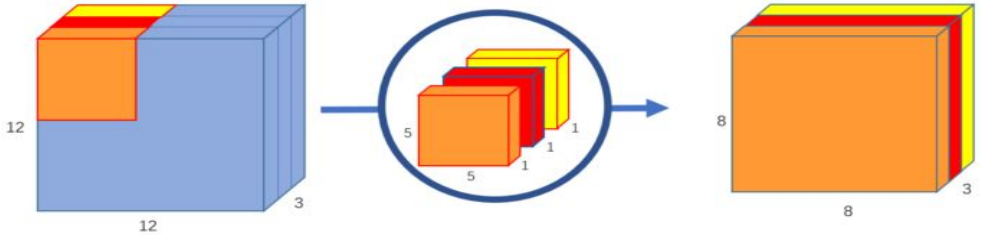
- In MobileNetV2, there are two types of blocks. One is residual block with stride of 1. Another one is block with stride of 2 for downsizing.
- There are 3 layers for both types of blocks.
- This time, the **first layer** is **1×1 convolution with ReLU6.**
- The **second layer** is the **depthwise convolution**.
- The **third layer** is another **1×1 convolution but without any non-linearity.** It is claimed that if ReLU is used again, the deep networks only have the power of a linear classifier on the non-zero volume part of the output domain.

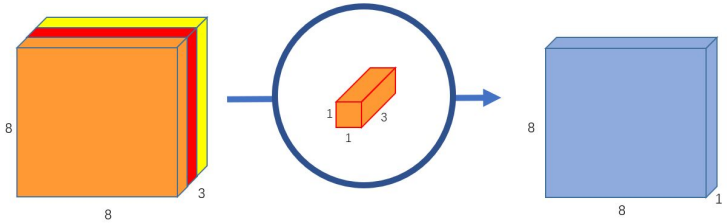
Depth Wise Separable Convolution



Convolution

Depth Wise Convolution





Point Wise Convolution