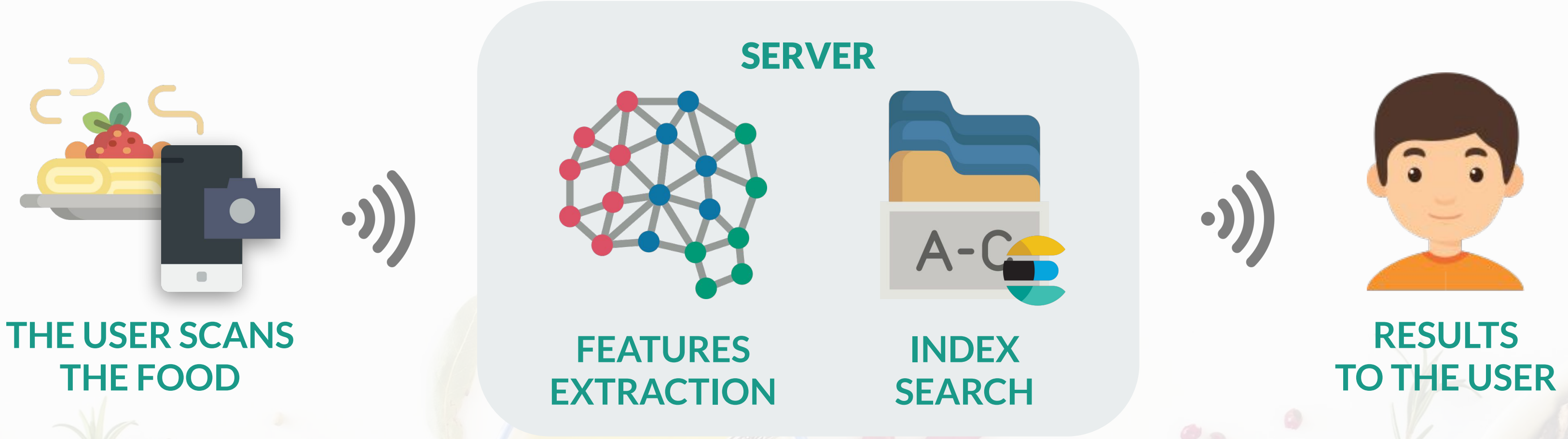
MIM Project

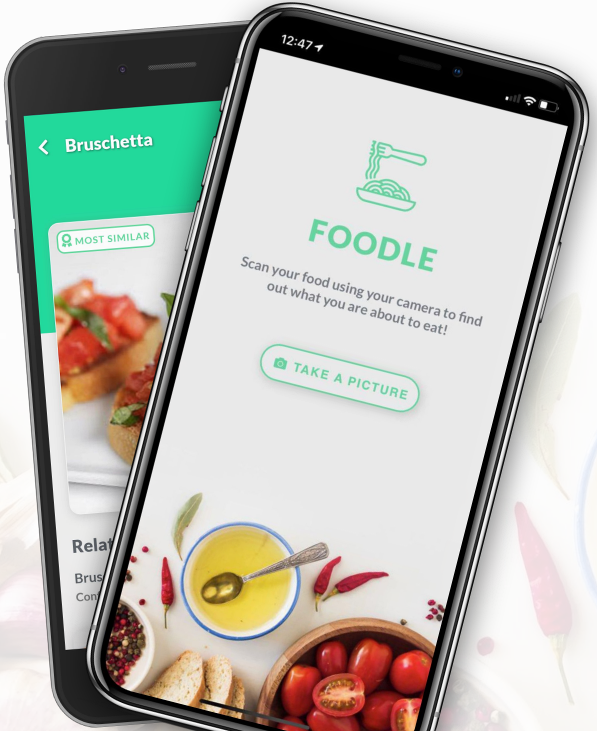
Indexing Deep Features for Food

Giada Anastasi , Chiara Bonsignori, Luca Brombin, Stefano Guazzelli, Filippo Scotto

Overall System



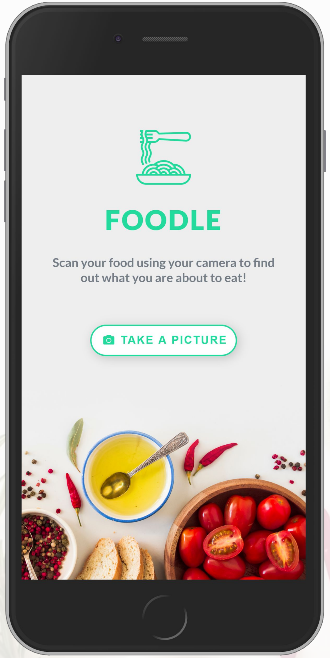
WebApp (1/2)

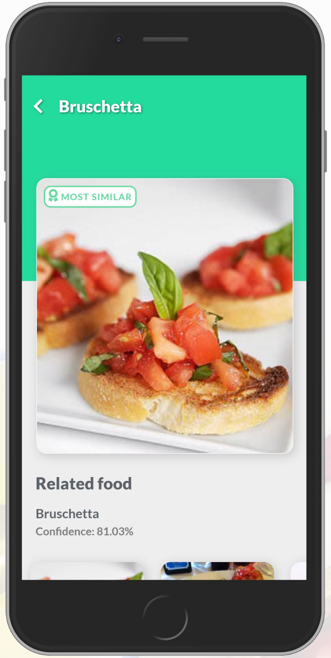


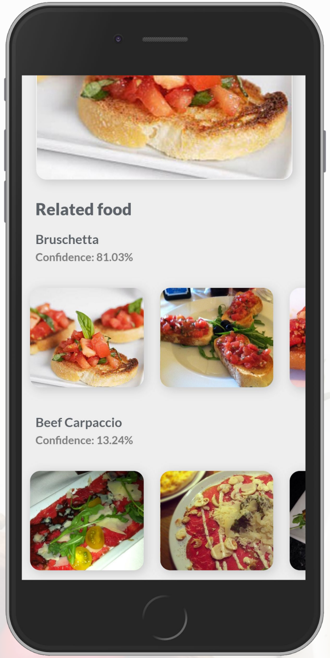
The application is an **HTML5/JS WebApp** hosted on the system's server.

1. Once the user has opened the app, they can scan the food using the **mobile camera**;
2. The image is sent to the server (**base64 encoding**);
3. The server will provide its best results (**candidate class, most similar** and **related foods**).

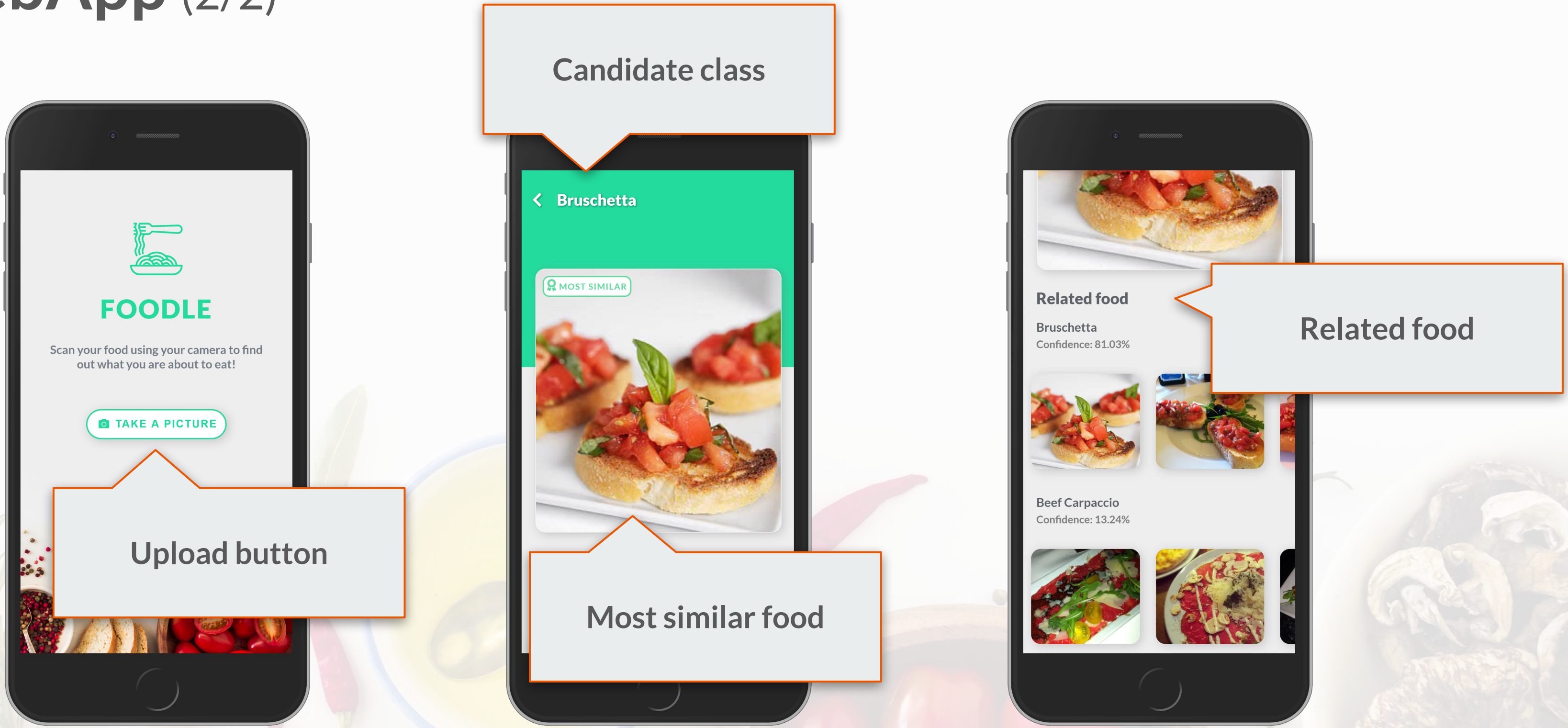
WebApp (2/2)


FOODLE
Scan your food using your camera to find out what you are about to eat!
TAKE A PICTURE

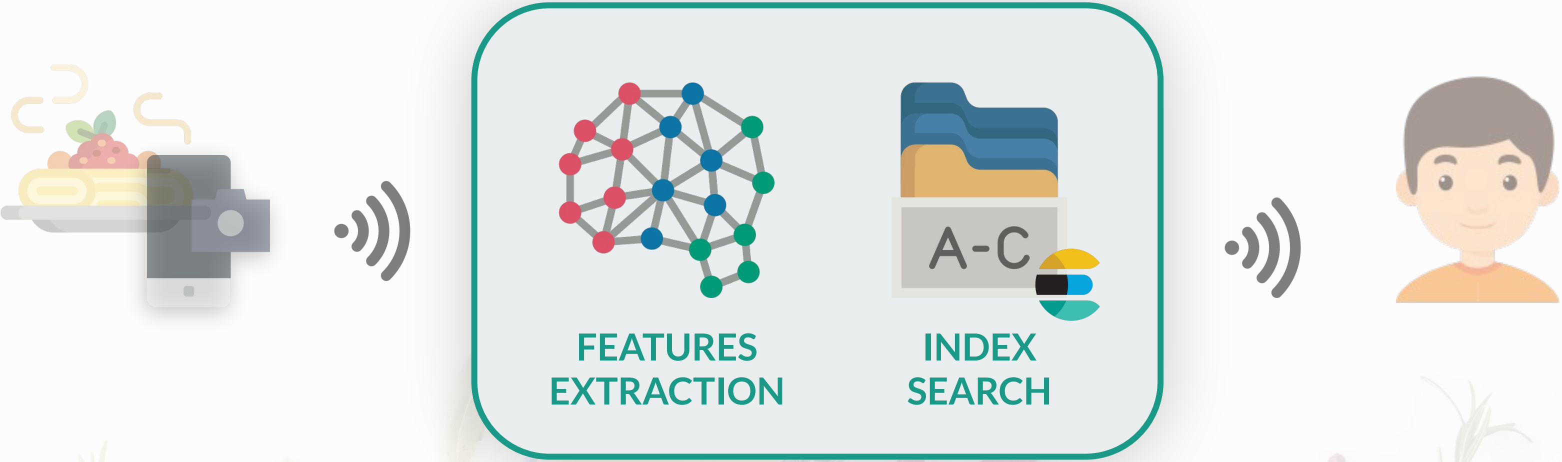

Bruschetta
MOST SIMILAR
Related food
Bruschetta
Confidence: 81.03%


Related food
Bruschetta
Confidence: 81.03%
Beef Carpaccio
Confidence: 13.24%

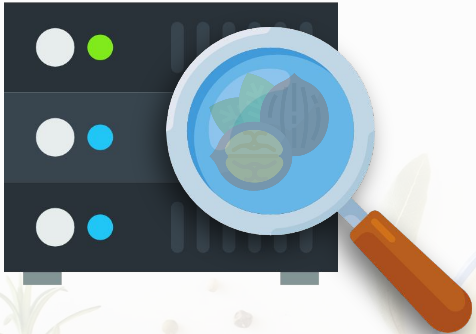
WebApp (2/2)


FOODLE
Scan your food using your camera to find out what you are about to eat!
TAKE A PICTURE
Upload button
Candidate class
Bruschetta
MOST SIMILAR
Most similar food
Related food
Bruschetta
Confidence: 81.03%
Beef Carpaccio
Confidence: 13.24%
Related food

Server

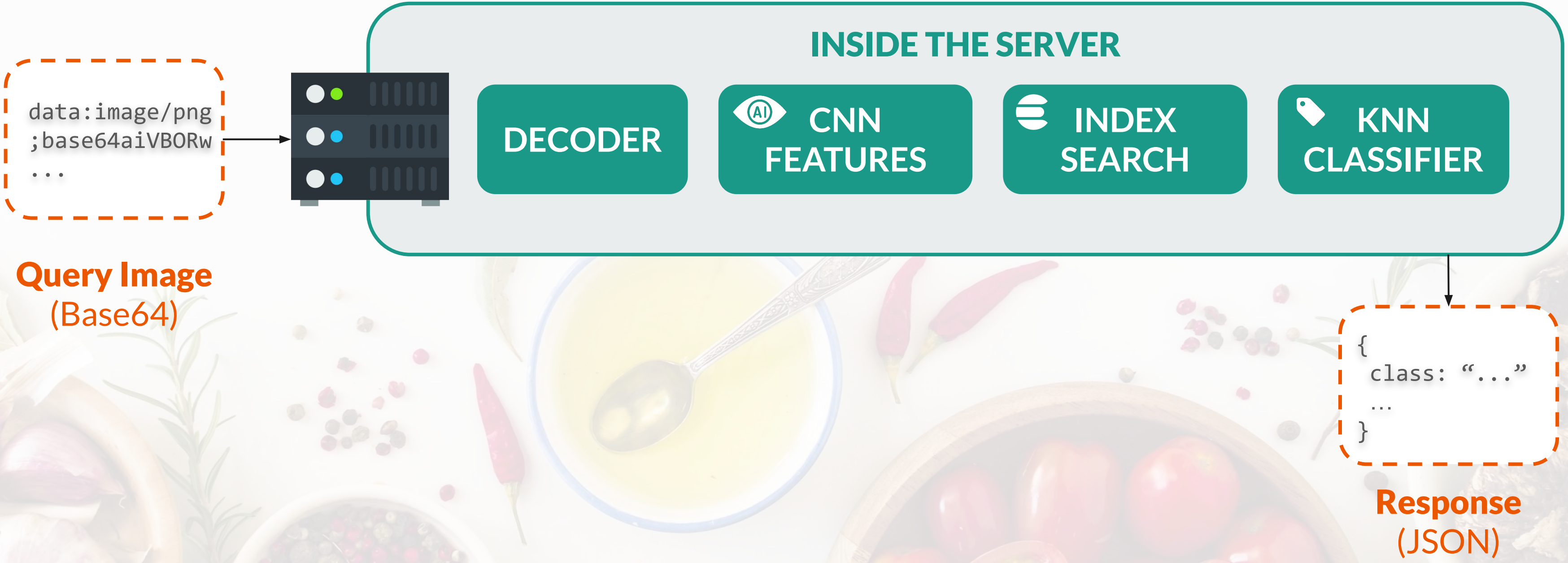


Server: In a nutshell

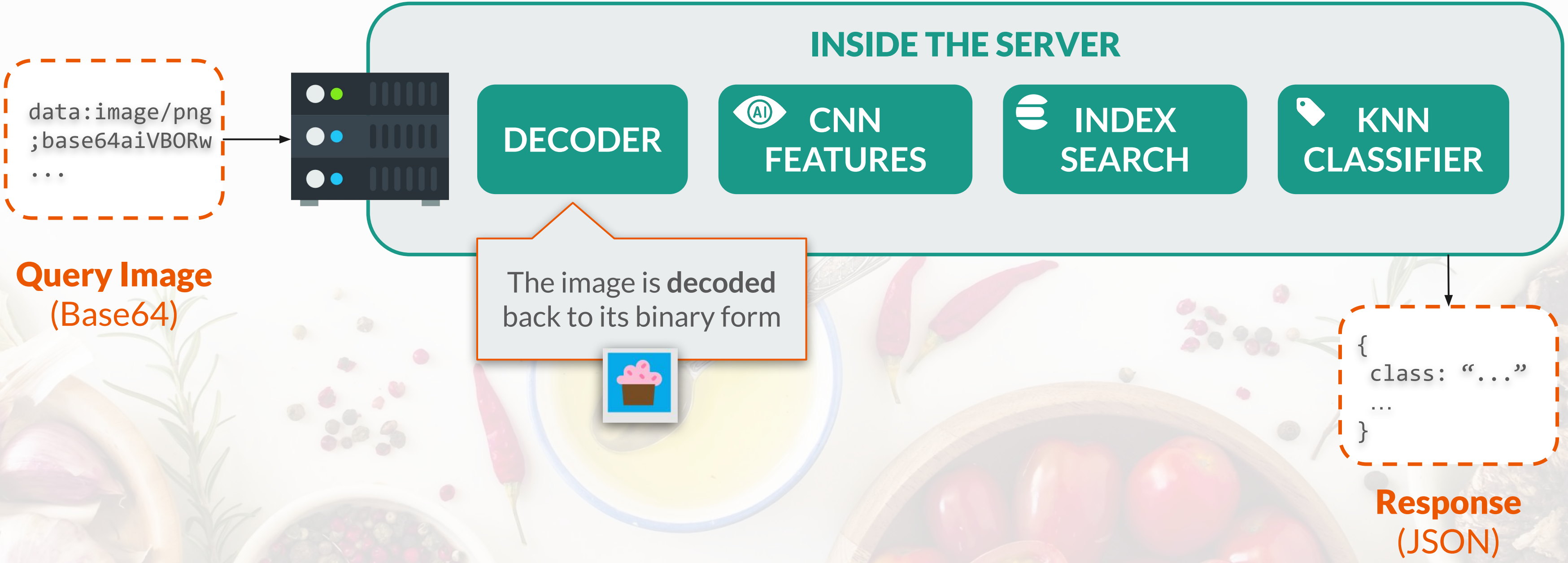


- Query image **feature extraction** using **Convolutional Neural Network**
- Search for the top K similar images using an **ElasticSearch Index**
- Perform a **K-NN Classification** to find the **candidate class.**
- Send back to the user the results of the query (**JSON Message**)

Workflow

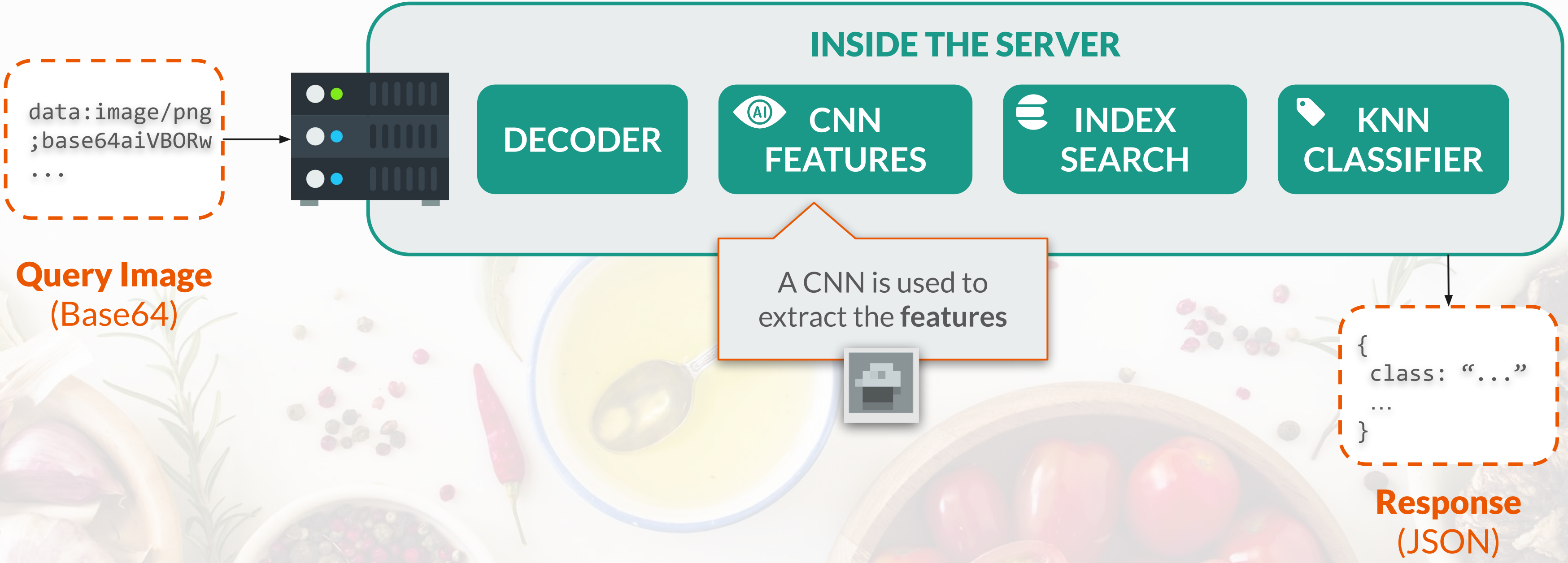

data:image/png
;base64aiVBORw
...
Query Image
(Base64)
INSIDE THE SERVER
DECODER
CNN
FEATURES
INDEX
SEARCH
KNN
CLASSIFIER
{
class: "..."
...
}
Response
(JSON)

Workflow

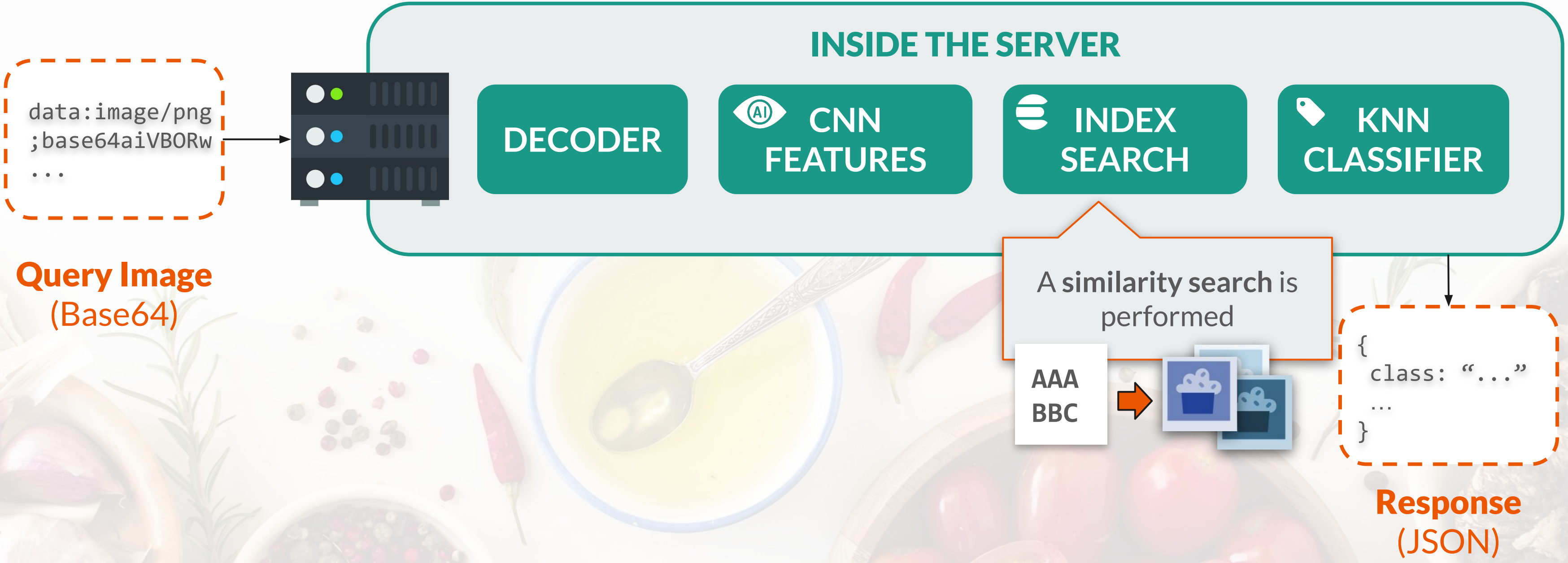


data:image/png
;base64aiVBORw
...
Query Image
(Base64)
INSIDE THE SERVER
DECODER
CNN FEATURES
INDEX SEARCH
KNN CLASSIFIER
The image is **decoded** back to its binary form
{
class: "..."
...
}
Response
(JSON)

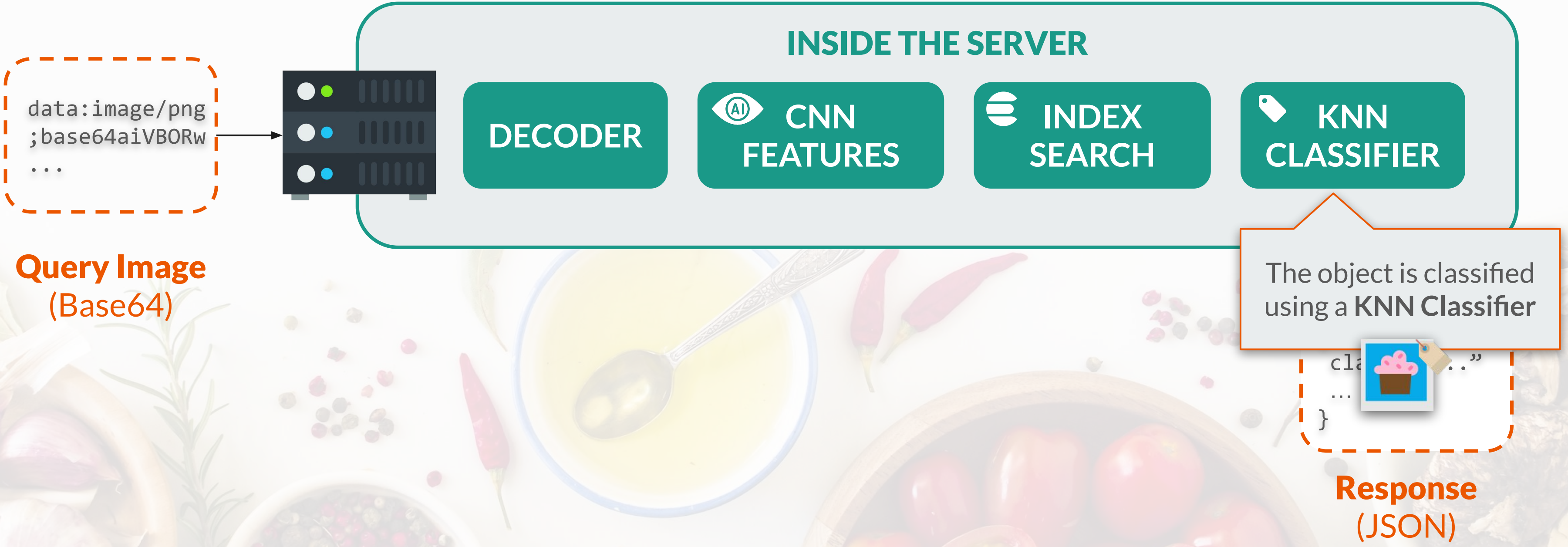
Workflow


data:image/png
;base64aiVBORw
...
Query Image
(Base64)
INSIDE THE SERVER
DECODER
CNN FEATURES
INDEX SEARCH
KNN CLASSIFIER
A CNN is used to extract the **features**
{
class: "..."
...
}
Response
(JSON)

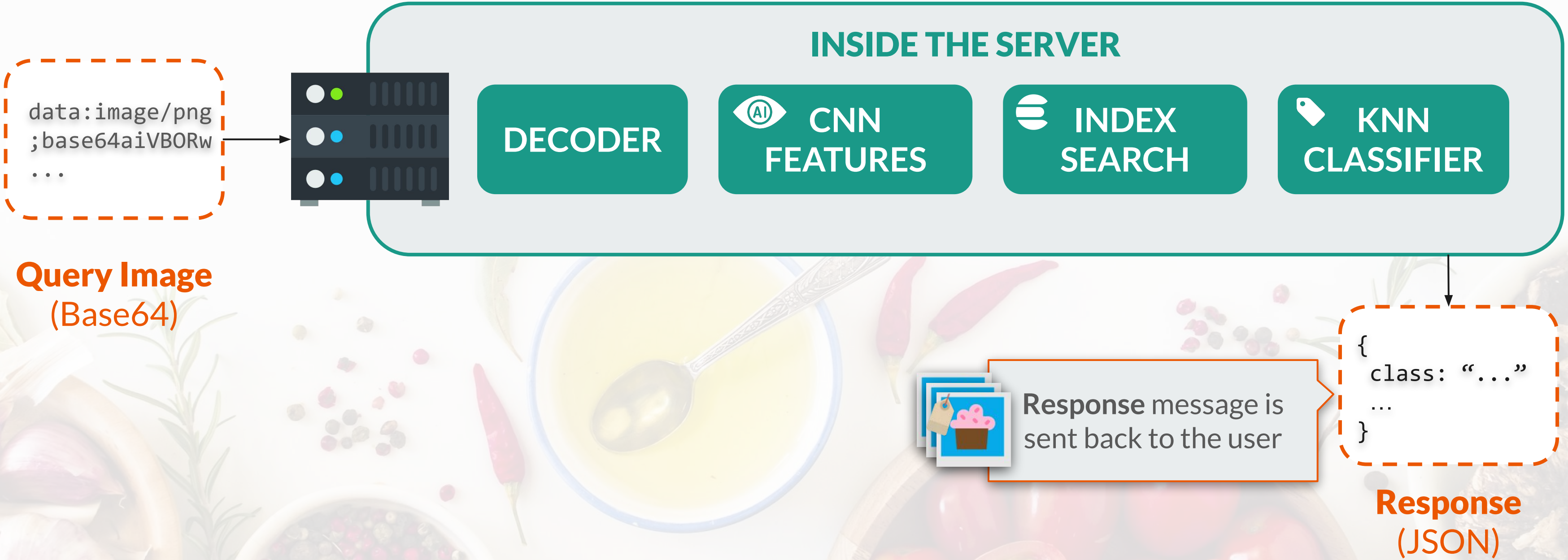
Workflow


data:image/png
;base64aiVBORw
...
Query Image
(Base64)
INSIDE THE SERVER
DECODER
CNN FEATURES
INDEX SEARCH
KNN CLASSIFIER
A similarity search is performed
AAA
BBC
{
class: "..."
...
}
Response
(JSON)

Workflow


data:image/png
;base64aiVBORw
...
Query Image
(Base64)
INSIDE THE SERVER
DECODER
CNN FEATURES
INDEX SEARCH
KNN CLASSIFIER
The object is classified using a **KNN Classifier**
}
Response
(JSON)

Workflow


data:image/png
;base64aiVBORw
...
Query Image
(Base64)
INSIDE THE SERVER
DECODER
CNN
FEATURES
INDEX
SEARCH
KNN
CLASSIFIER
{
class: “...”
...
}
Response
(JSON)
Response message is
sent back to the user

ElasticImgSearching

The java class developed during the **lab sessions** is extended and adapted for the project.
The goal is to search in the index for the most similar images, given an **ImgDescriptor**.

1. The image descriptor is converted into a **Surrogate Text Representation**;
2. A **SearchRequest** object is built and the **ElasticSearch** index is queried;
3. The result set is **reordered** according to the **real distance**.

Datasets

Two twin datasets

UMPC
Food 101

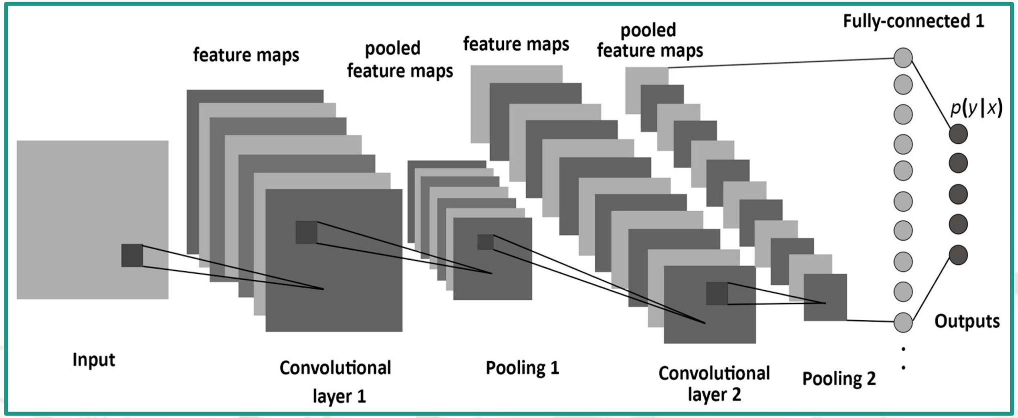
- 101 food classes
- 700-900 images per class

ETHZ
Food 101

- 101 food classes
- 1000 images per class

Features Extraction

In order to develop a **Content-Based Image Retrieval System**, we use **Transfer Learning** techniques to extract features from the images (**Convolutional Neural Network pretrained models**)



For our project we consider: **GoogleNet** and **VGG**.

The features are extracted by considering the output at **specific layers** of the networks.

Features Extraction

Extracts the **features** of a single image, using a **Caffe Framework Model**:

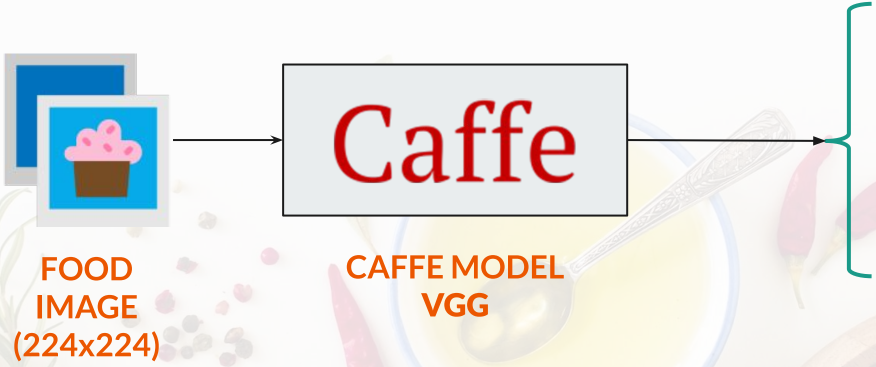


IMAGE DESCRIPTOR
DEEP LAYER: FC7

String **ImageID**
String **Class**
float **NormalizedVector[4096]**

Features Extraction

Extracts the **features** of a single image, using a **Caffe Framework Model**:

FOOD
IMAGE
(224x224)

Caffe

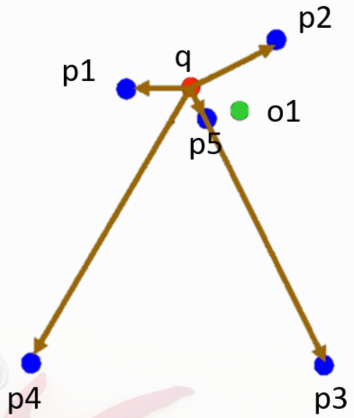
CAFFE MODEL
GOOGLENET

IMAGE DESCRIPTOR
DEEP LAYER: **POOL5/7x7_S1**

String **ImageID**
String **Class**
float **NormalizedVector[1024]**

Pivot Selection

FEATURES EXTRACTION → INDEXING →



$$O_1 \text{ -> } \Pi_1 = (p_5, p_2, p_1, p_3, p_4)$$

STRING OF PIVOTS FOR EACH IMG

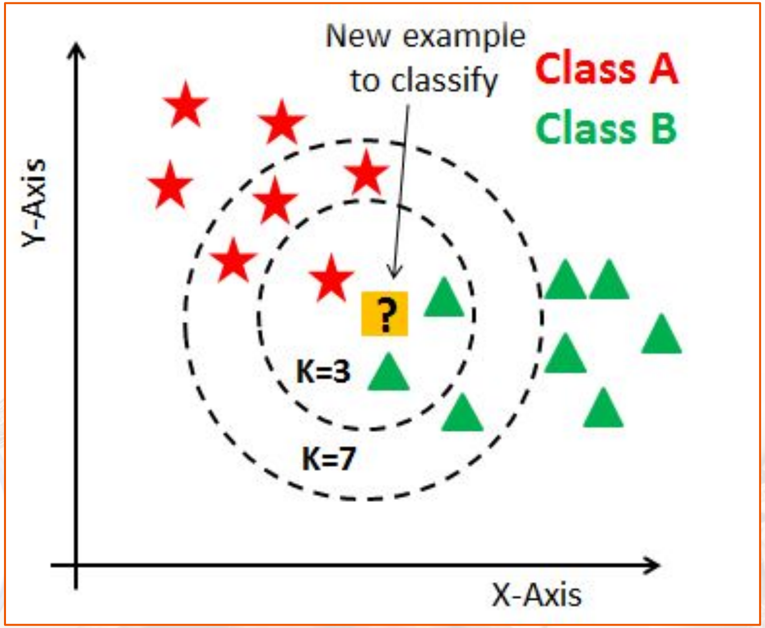
Most similar images selection

- Select most similar images to a certain image query
- Perform a sequential scan search
  - computes the distance between each pivot descriptor and the query
  - sorts the results
  - returns the k best results

KNN Classification (1/2)

Goal

- Evaluate the **class of an image** and the **most related**



KNN details

- Supervised Learning technique
- No explicit training
- Two types of voting techniques

KNN Classification (2/2)

Simple Voting

- Counting the number of results belonging to each class

Weighted Voting

- Summing all distances of elements belonging to the same class

$$\boxed{w \equiv \frac{1}{d(x_q, x_i)^2}}$$

Test

Goals

- find the **best** combination of **parameters** for each index
- find the **most performing** configuration
- **compare** results with classification performed using CNN from De Bonis' thesis
  (*"Development of a mobile application for Food Recognition using Convolutional Neural Networks"*)

Four similarity search indices to analyze

- 2 CNNs
  - **BVLC GoogleNet** and **VGG - 16 layers**
- 2 twin datasets
  - **ETHZ Food-101** and **UMPC Food-101**

Test configurations

Performance metrics:

- **accuracy**, **time to search**, **time to index**

Dataset used for testing:

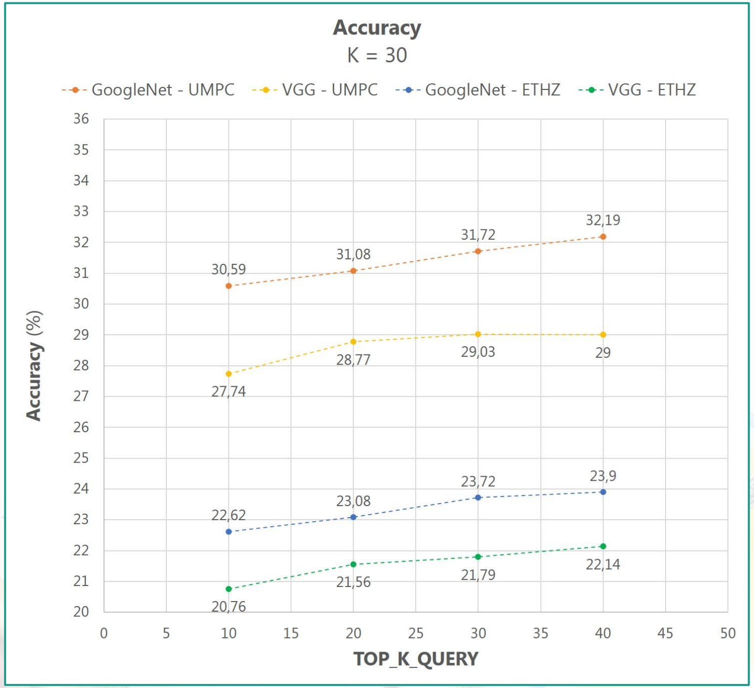
- **test** dataset of UMPC Food-101

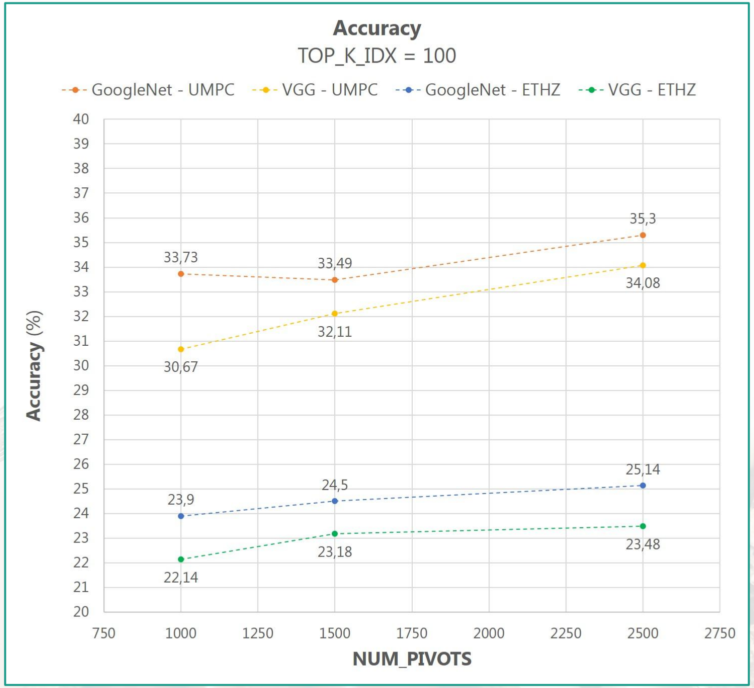
Parameters to tune

NUM_PIVOTS	1000	1500	2500
TOP_K_IDX	50	100	200

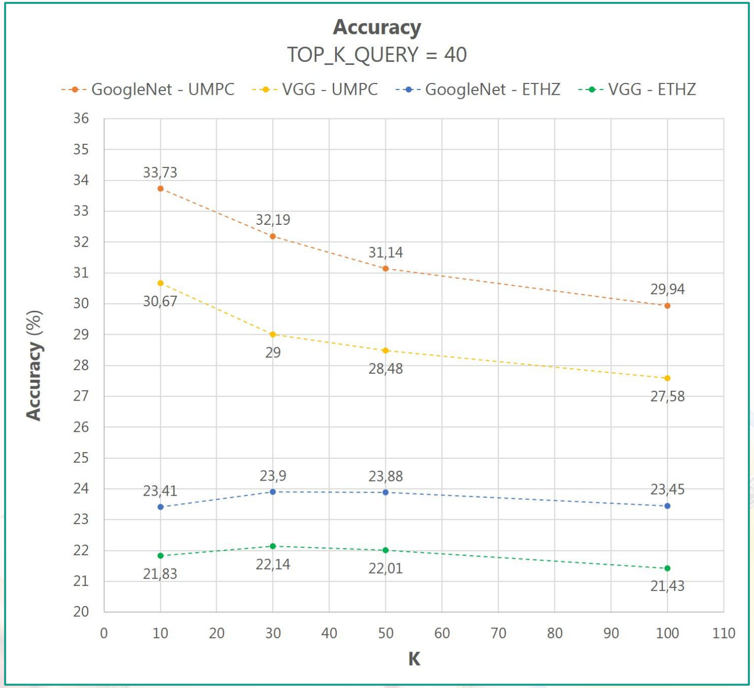
TOP_K_QUERY	10	20	30	40
K	10	30	50	100

Accuracy (1/2)


Accuracy
K = 30
GoogleNet - UMPC
VGG - UMPC
GoogleNet - ETHZ
VGG - ETHZ
Accuracy (%)
TOP_K_QUERY
30,59
31,08
31,72
32,19
27,74
28,77
29,03
29
22,62
23,08
23,72
23,9
20,76
21,56
21,79
22,14

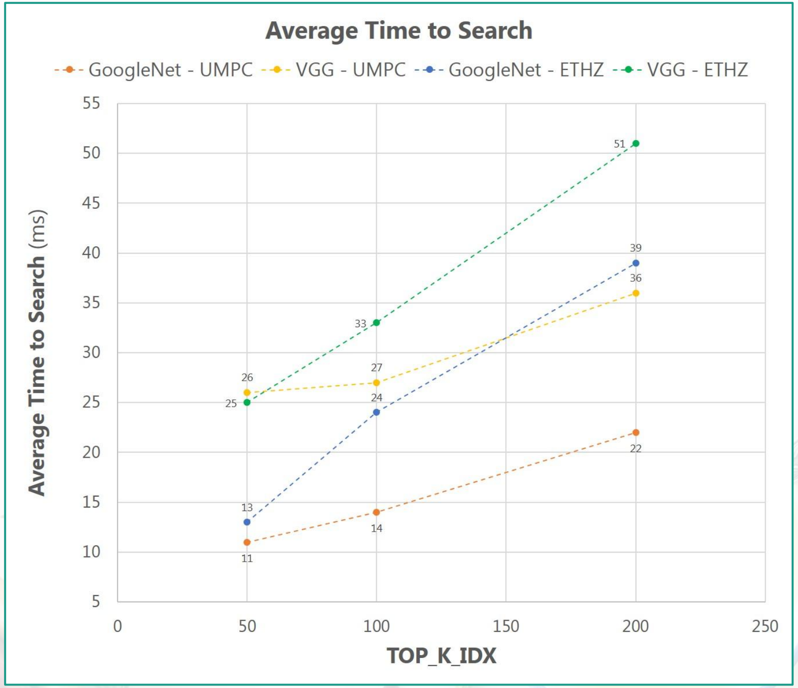

Accuracy
TOP_K_IDX = 100
GoogleNet - UMPC
VGG - UMPC
GoogleNet - ETHZ
VGG - ETHZ
Accuracy (%)
NUM_PIVOTS
33,73
33,49
35,3
30,67
32,11
34,08
23,9
24,5
25,14
22,14
23,18
23,48

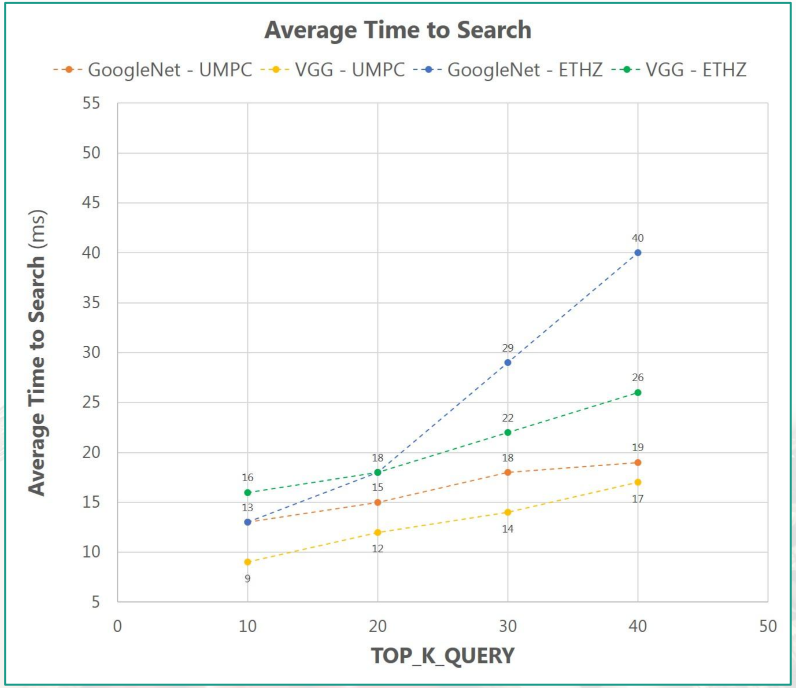
Accuracy (2/2)


Accuracy
TOP_K_QUERY = 40
GoogleNet - UMPC
VGG - UMPC
GoogleNet - ETHZ
VGG - ETHZ
Accuracy (%)
K
33,73
32,19
31,14
29,94
30,67
29
28,48
27,58
23,41
23,9
23,88
23,45
21,83
22,14
22,01
21,43

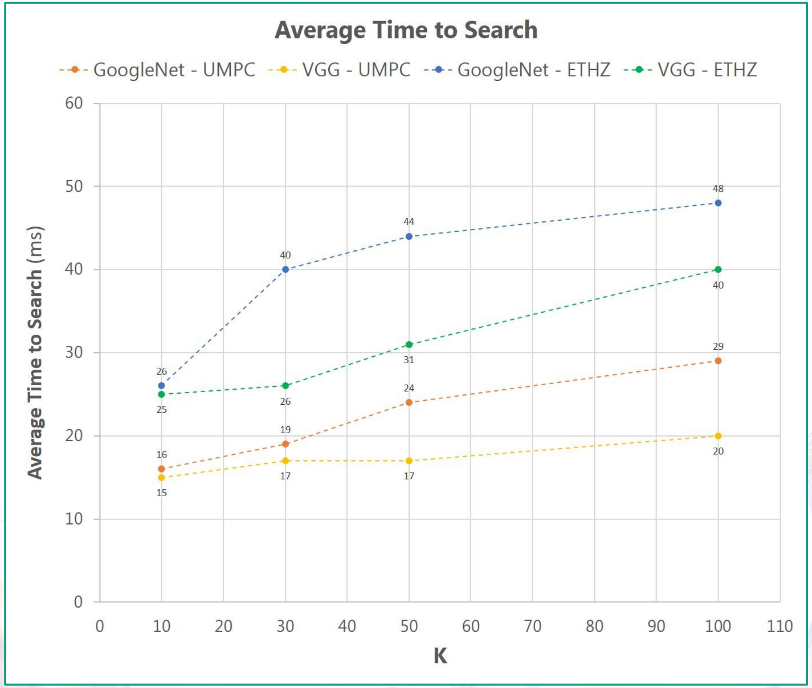

Accuracy
NUM_PIVOTS = 2500
GoogleNet - UMPC
VGG - UMPC
GoogleNet - ETHZ
VGG - ETHZ
Accuracy (%)
TOP_K_IDX
35,23
35,3
34,66
33,85
34,08
33,01
25,41
25,14
24,96
23,7
23,48
23,05

Time to Search (1/2)





Time to Search (2/2)



Time to extract features from an image:

- **GoogleNet** : ~ 100 ms
- **VGG** : ~ 300 ms

Time that the user must wait: ~ 1.5 s

Best Configurations

		GoogleNet ETHZ	GoogleNet UMPC	VGG ETHZ	VGG UMPC
NUM_PIVOTS		2500	2500	2500	2500
TOP_K_IDX		50	100	50	100
TOP_K_QUERY		40	40	40	40
K		30	10	30	10
Accuracy (%)	KNN with voting	25.4	35.3	23.7	33.9
	Weighted KNN	**25.7**	**40.9**	**24**	**40.5**
Time to search (ms)		13	11	25	26
Time to index (min)		16	12	32	24

Accuracy per class

In the case of **GoogleNet - UMPC** :

Class	Accuracy (%)
spaghetti_carbonara	78
guacamole	77.2
deviled_eggs	67.7
prime_rib	67.3
mussels	66.7
. . .	. . .

Class	Accuracy (%)
. . .	. . .
scallops	17.5
ice_cream	17.2
steak	14.8
hot_dog	13.7
beef_tartare	12.4

Comparison with De Bonis' CNN

Compared indices		Accuracy (%) using KNN with voting		Accuracy (%) using weighted KNN	
ETHZ	GoogleNet	25.4		25.7	
	De Bonis' CNN	44.2	19% gap	44.3	19% gap
UMPC	GoogleNet	35.3		40.9	
	De Bonis' CNN	50.4	15% gap	54.2	13% gap

Using **De Bonis' CNN** for the entire classification: **accuracy** about **45.5%**

Thank you!

... any questions?