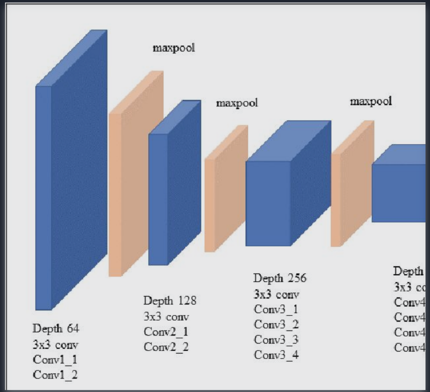
Multimedia Information Retrieval and Computer Vision Project

Gabriele Martino

Contents

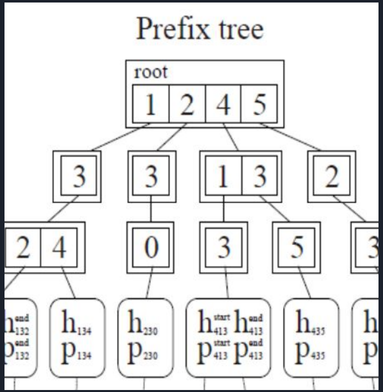
CNN

VGG16, VGG19



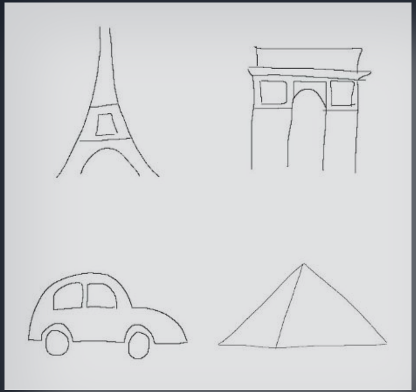
Index

PPIndex



Dataset

Sketches



Distractor Dataset

MrFlickr25k

Pipeline

1. Features Extraction from pretrained CNN
2. Performance Analysis
3. Features Extraction from Fine Tuned CNN
4. Performance Analysis

Pipeline

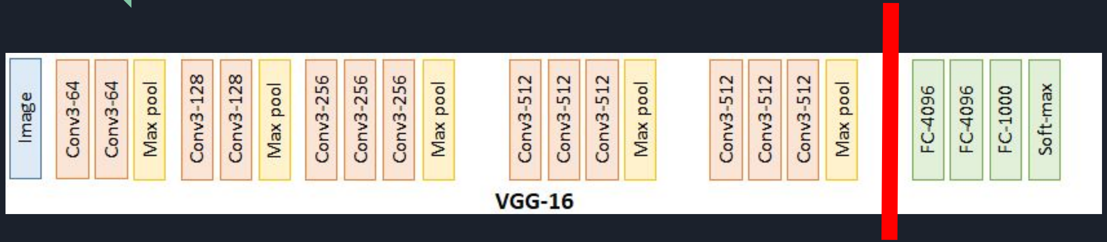
Features Extraction from pretrained CNN → Performance Analysis → Features Extraction from Fine Tuned CNN → Performance Analysis

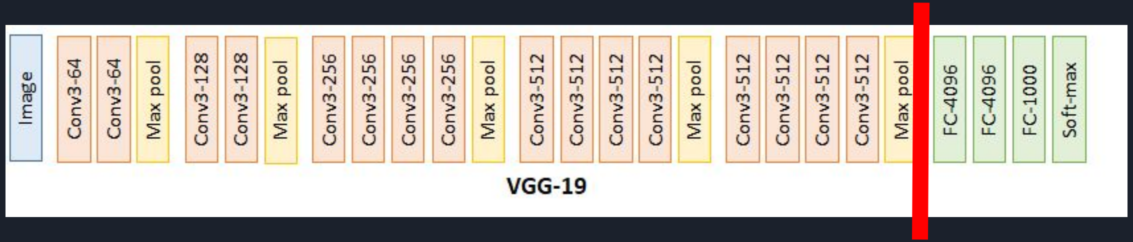
Features Extraction from pretrained CNN (i)

VGG-16

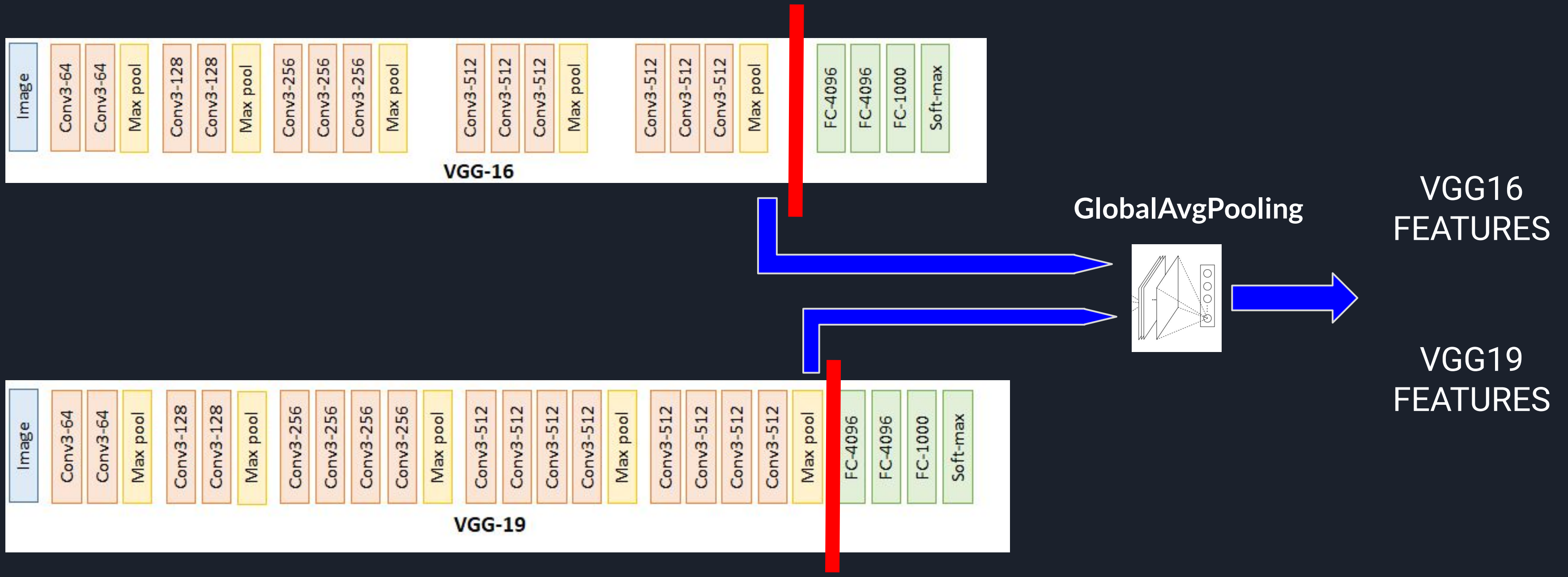
VGG-19

Features Extraction from pretrained CNN (ii)

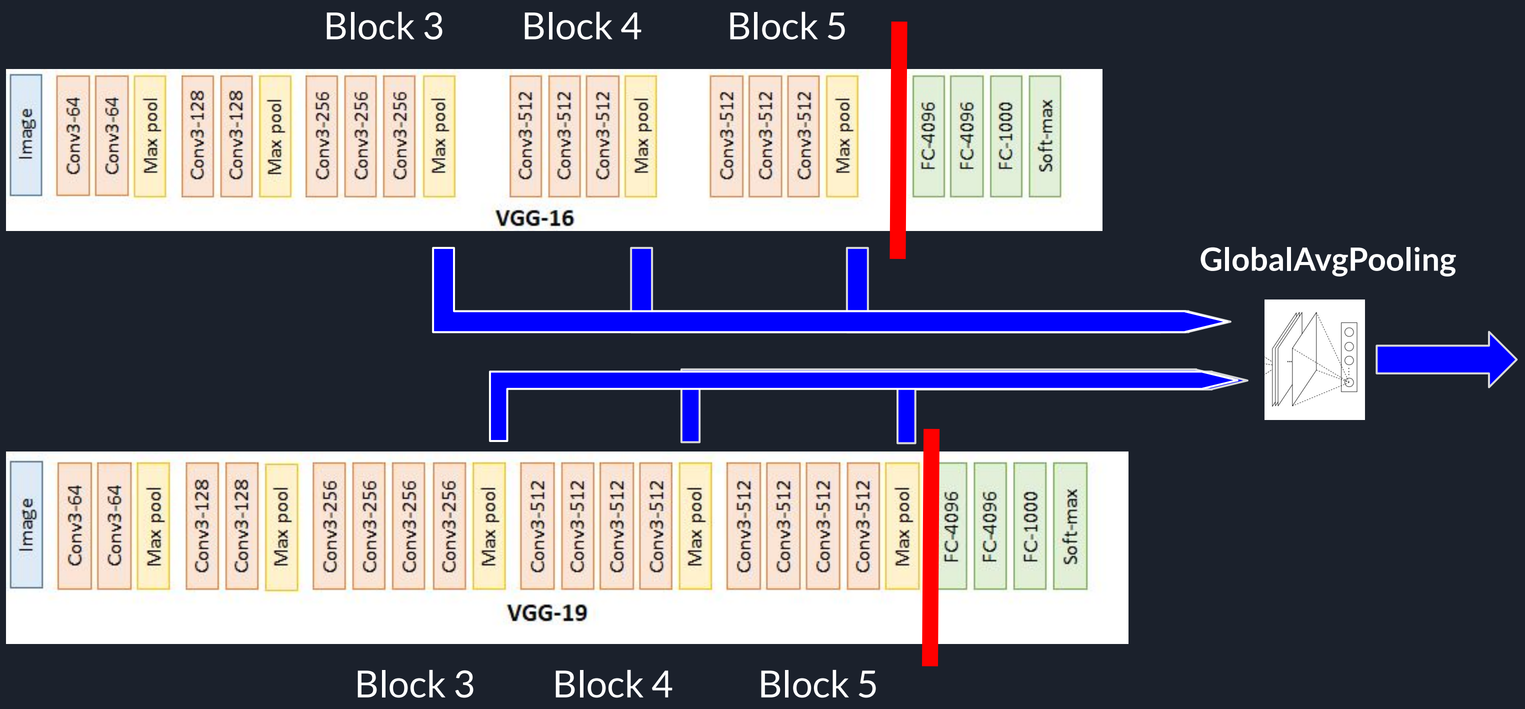




Features Extraction from pretrained CNN (iii)



Features Extraction from pretrained CNN (iv)



VGG16 FEATURES

- Block 5
- Block 4
- Block 3

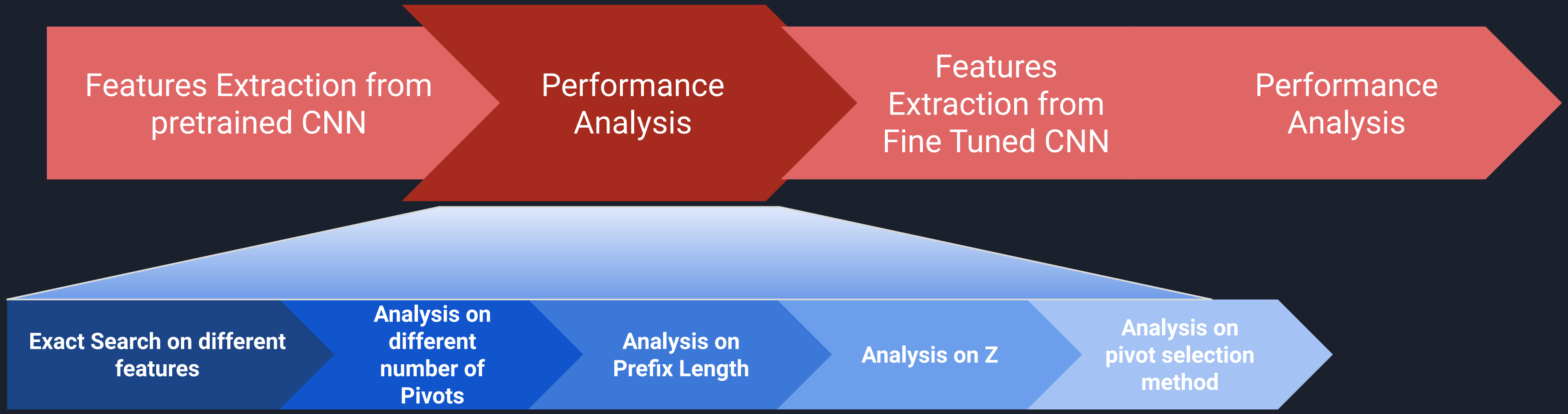
VGG19 FEATURES

- Block 5
- Block 4
- Block 3

Pipeline


Features Extraction from pretrained CNN
Performance Analysis
Features Extraction from Fine Tuned CNN
Performance Analisis

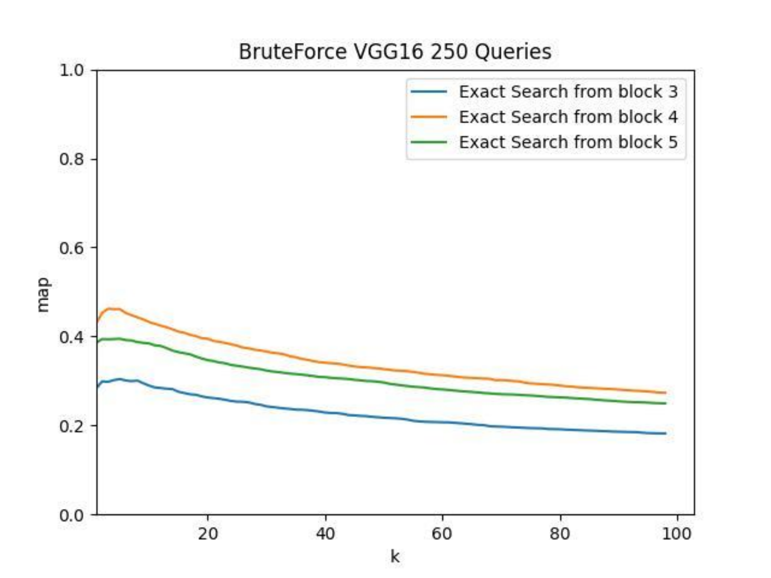
Pipeline

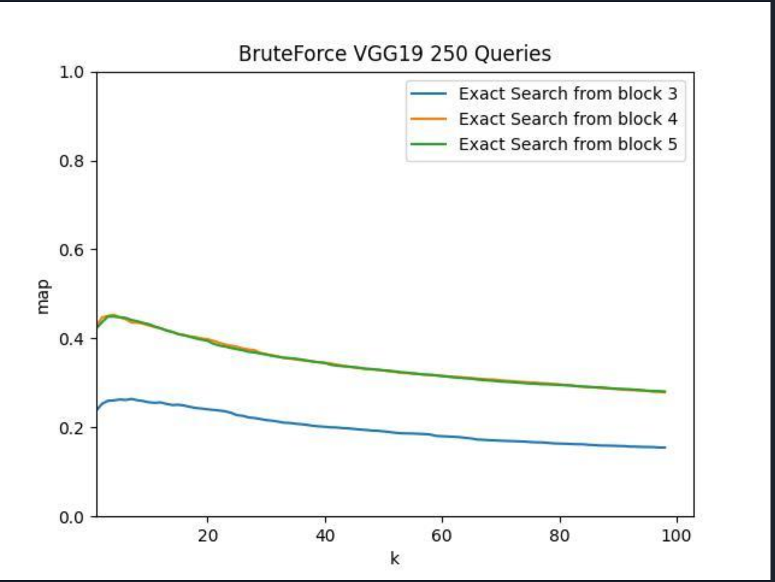


Features Extraction from pretrained CNN
Performance Analysis
Features Extraction from Fine Tuned CNN
Performance Analysis
Exact Search on different features
Analysis on different number of Pivots
Analysis on Prefix Length
Analysis on Z
Analysis on pivot selection method

Performance Analysis - Exact Search on different features

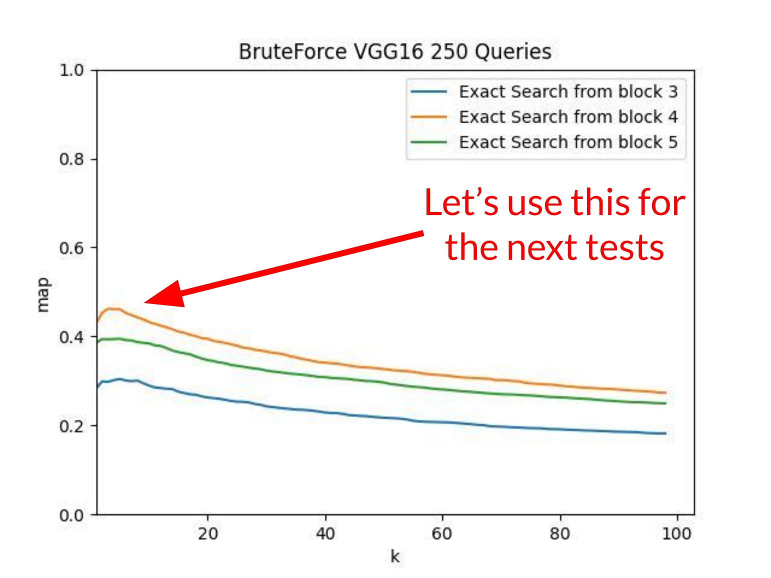
KNN search on query extracted from the dataset with a bucket of dataset + distractor. Queries are the same for all the analysis; 250, one for each class.

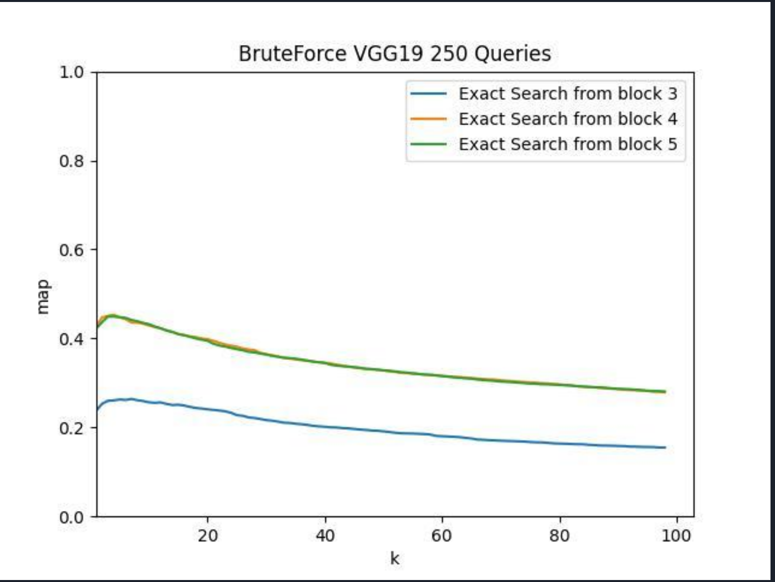




Performance Analysis - Exact Search on different features

KNN search on query extracted from the dataset with a bucket of dataset + distractor. Queries are the same for all the analysis; 250, one for each class.





Performance Analysis with PPindex (ii)

PPIndex parameters:

- **Number of pivots**
- **Pivots selection method**
- **L**, prefix length
- **Z**, number of objects retrieved before k cutting

Performance Analysis with PPindex (iii)

Experiments:

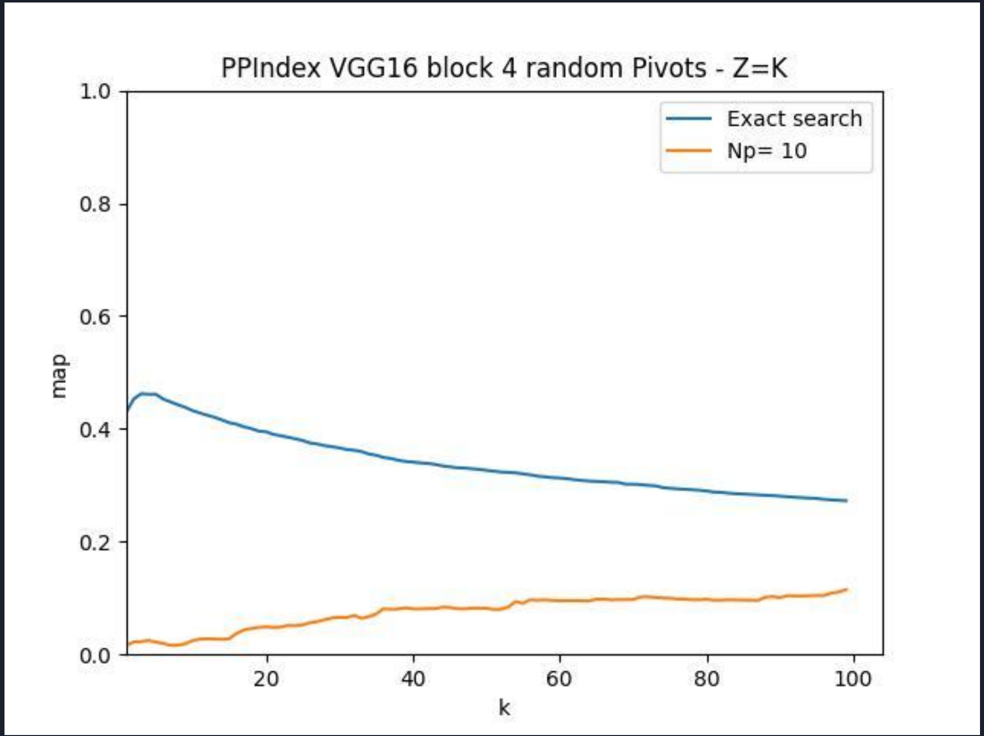
- Number of pivots: **Np∈{10, 50, 100, 200}**
- Pivot selection method: **random, 3Np method, Kmedoids**
- **L**
- **Z = K + b, b∈{20, 50, 100, 200}**
- **Same 250 queries, one for each class of the dataset, to make a real comparison**
- **K ∈[1,100]**

Analysis of number of pivots (i)

Analysis of number of pivots

Number of pivots:
Np∈{10, 50, 100, 200}

Fixed Parameter:
Pivot Selection: Random
L = 10
Z = K

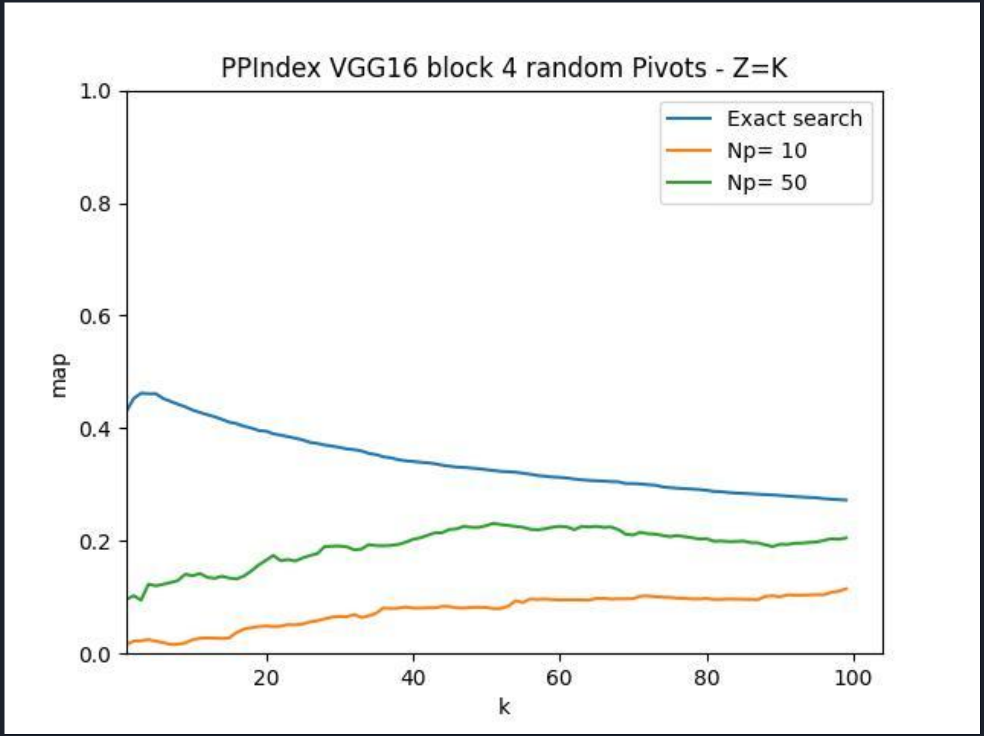


Analysis of number of pivots (ii)

Analysis of number of pivots

Number of pivots:
Np∈{10, 50, 100, 200}

Fixed Parameter:
Pivot Selection: Random
L = 10
Z = K

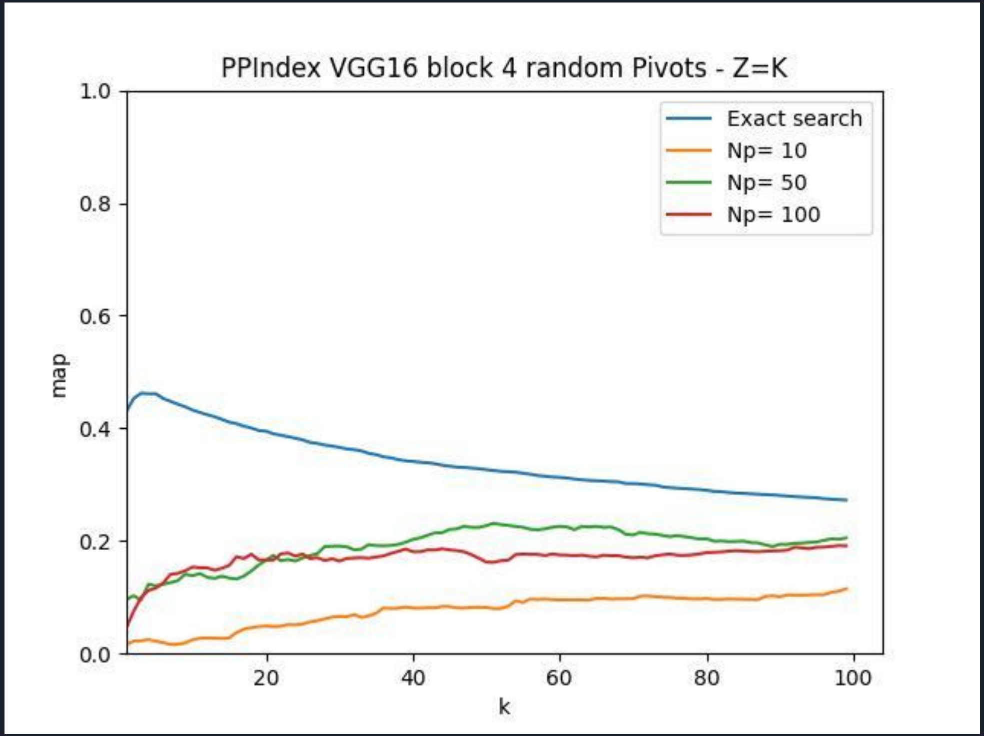


Analysis of number of pivots (iii)

Analysis of number of pivots

Number of pivots:
Np∈{10, 50, 100, 200}

Fixed Parameter:
Pivot Selection: Random
L = 10
Z = k

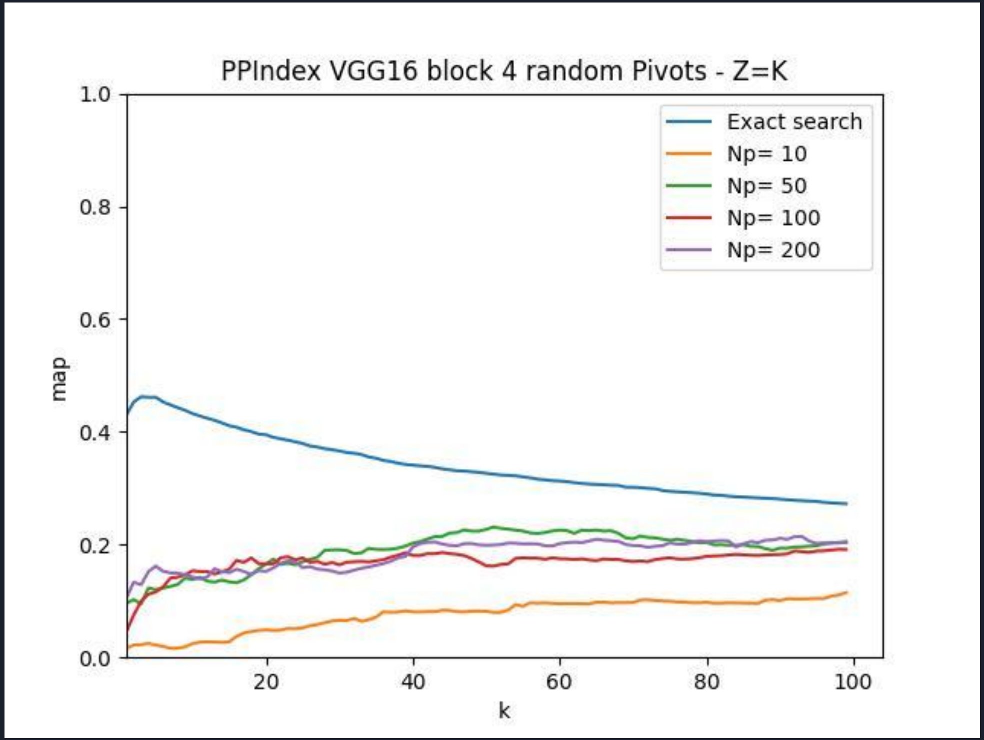


Analysis of number of pivots (iv)

Analysis of number of pivots

Number of pivots:
Np∈{10, 50, 100, 200}

Fixed Parameter:
Pivot Selection: Random
L = 10
Z = K

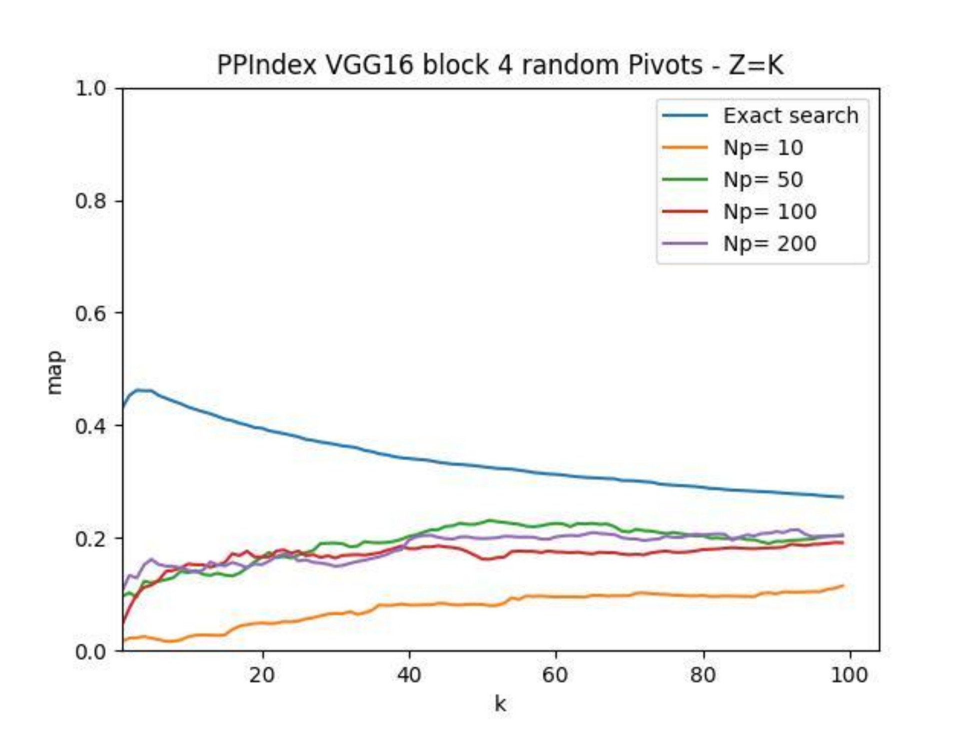


Analysis of Z (i)

Let's vary Z properly using

Z = k + b

b∈{20, 50, 100, 200}

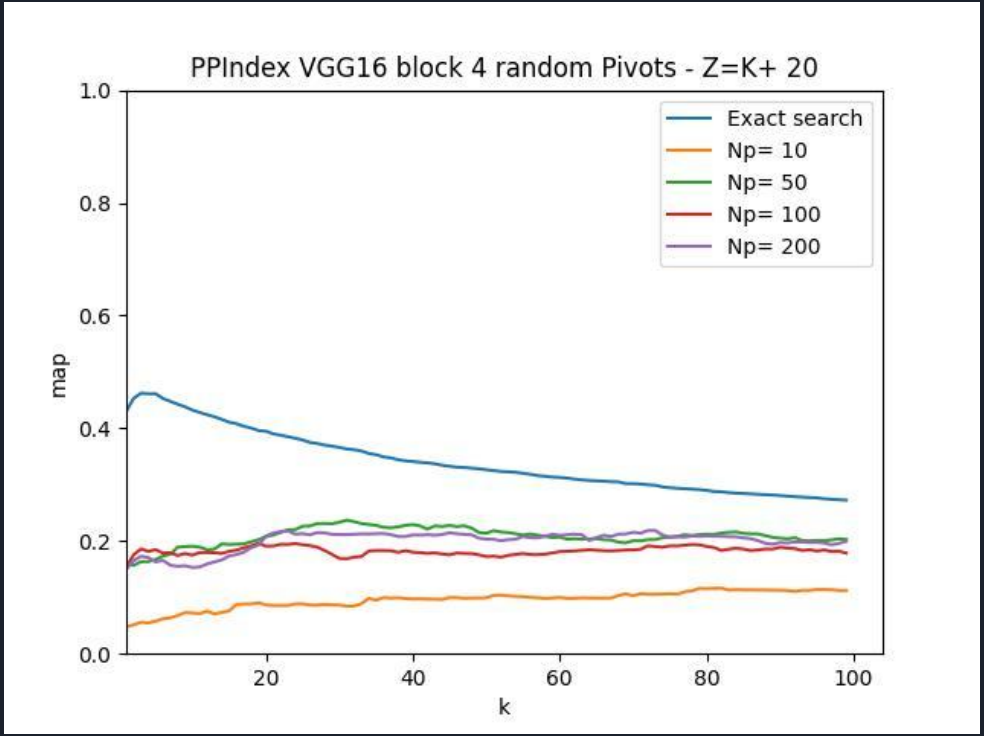


Analysis of Z(ii)

Let’s vary Z properly using

$Z = k + b$

$b \in \{20, 50, 100, 200\}$

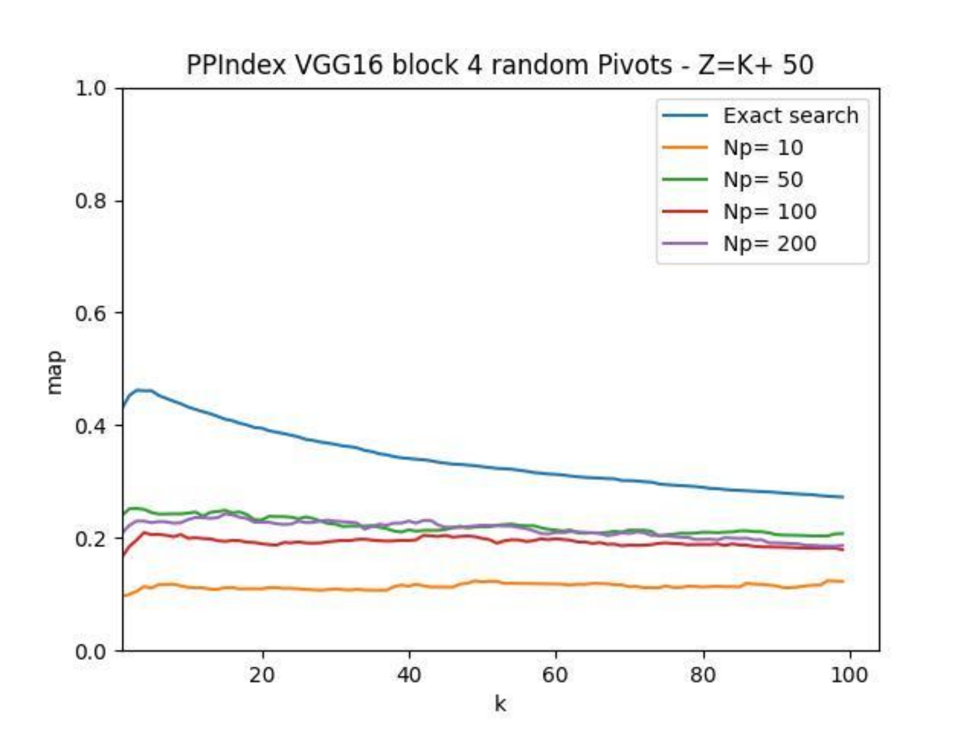


Analysis of Z(iii)

Let's vary Z properly using

$Z = k + b$

$b \in \{20, 50, 100, 200\}$

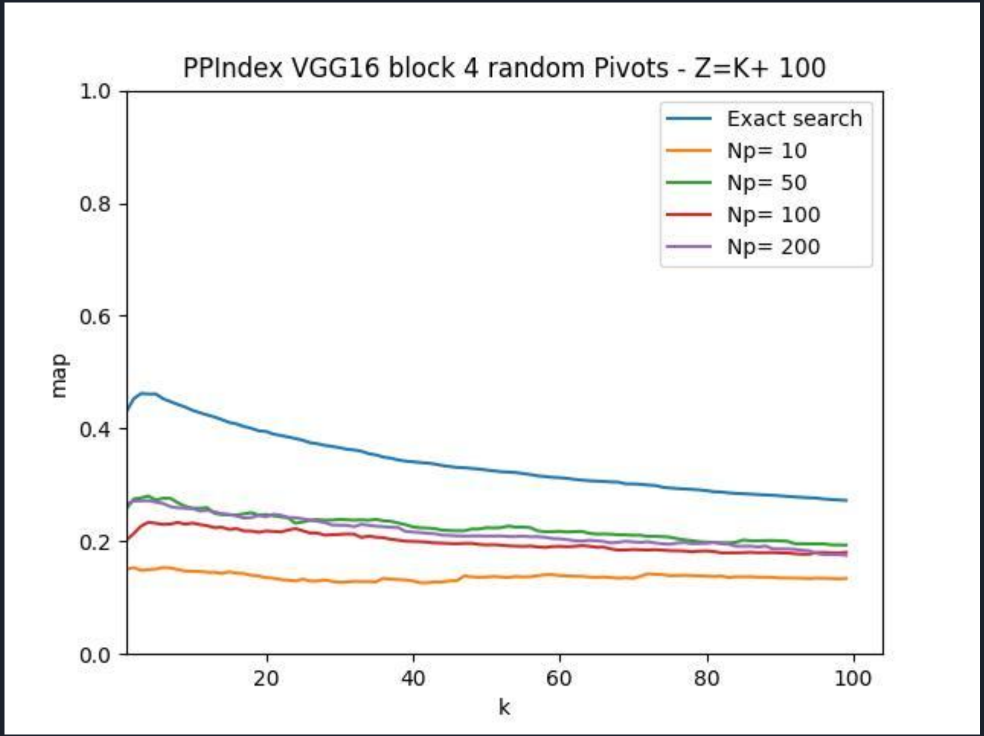


Analysis of Z(iv)

Let's vary Z properly using

$Z = k + b$

$b \in \{20, 50, 100, 200\}$

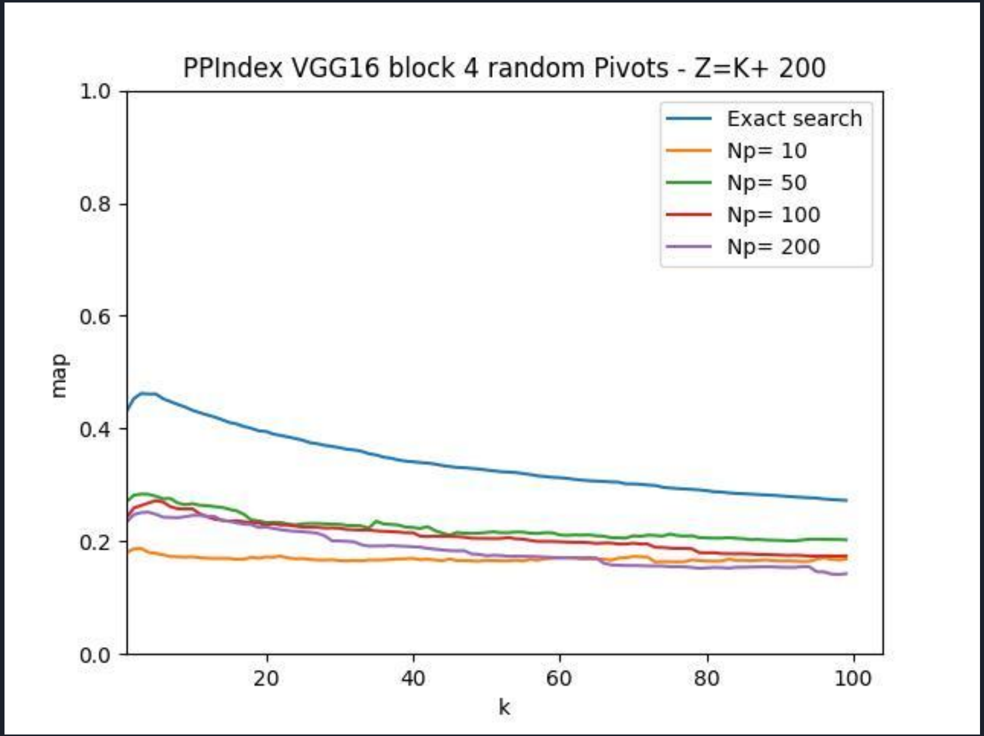


Analysis of Z(v)

Let's vary Z properly using

$$Z = k + b$$

$$b \in \{20, 50, 100, 200\}$$

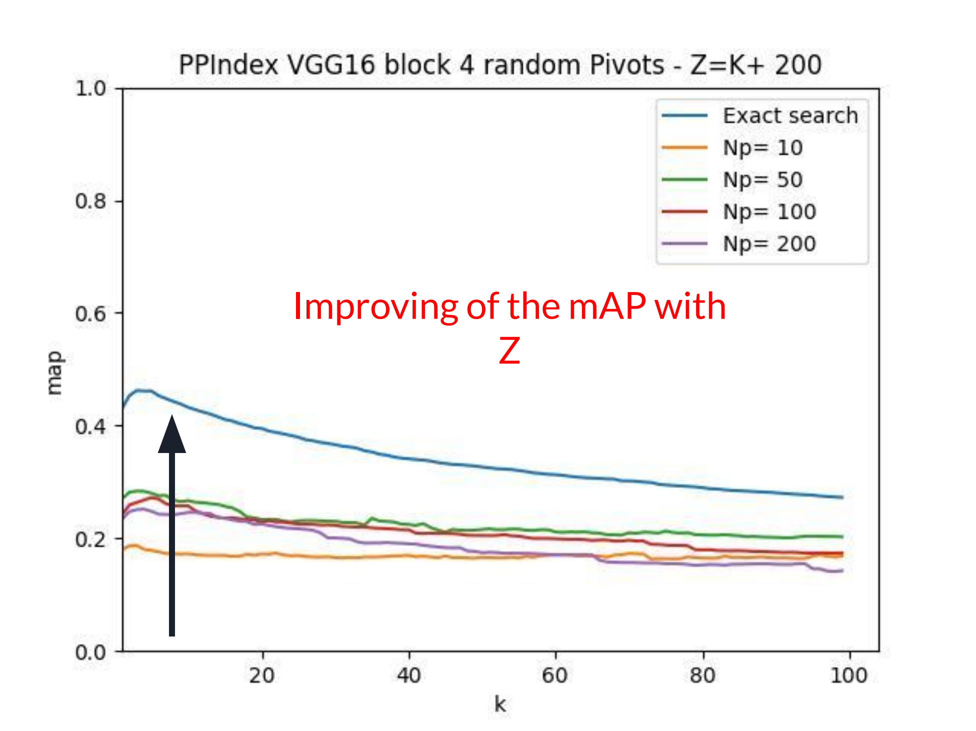


Analysis of Z(vi)

Let's vary Z properly using

Z = k + b

b∈{20, 50, 100, 200}



Analysis of Z - Note

Considering a dataset size of 45’000 (quite small) a low number of pivots is enough. For a small dataset the number of pivots affect a lot the results also because of the kind of search algorithm.

Given a k-NN query for an object $q \in O$, the basic search function of the PP-Index consists of computing the permutation prefix Πl p and ***searching for the longest prefix match in the prefix tree whose subtree points to at least z candidate objects*** *.[1]*

This means that for a large number of pivots the objects are already spread a lot at the first level of the tree of the index. This could also lead to an impossibility to retrieve a certain number of objects.

Analysis of Z - Note (ii)

Considering a dataset size of 45’000 (quite small) a low number of pivots is enough. For a small dataset the number of pivots affect a lot the results also because of the kind of search algorithm.

Given a k-NN query for an object q ∈ O, the basic search function of the PP-Index consists of computing the permutation prefix Πl p and ***searching for the longest prefix match in the prefix tree whose subtree points to at least z candidate objects*** *.[1]*

This means that for a large number of pivots the objects are already spread a lot at the first level of the tree of the index. This could also lead to an impossibility to retrieve a certain number of objects.

Example:
N = 45’000, Np = 250 N/Np = 180 (assuming a uniform distribution)

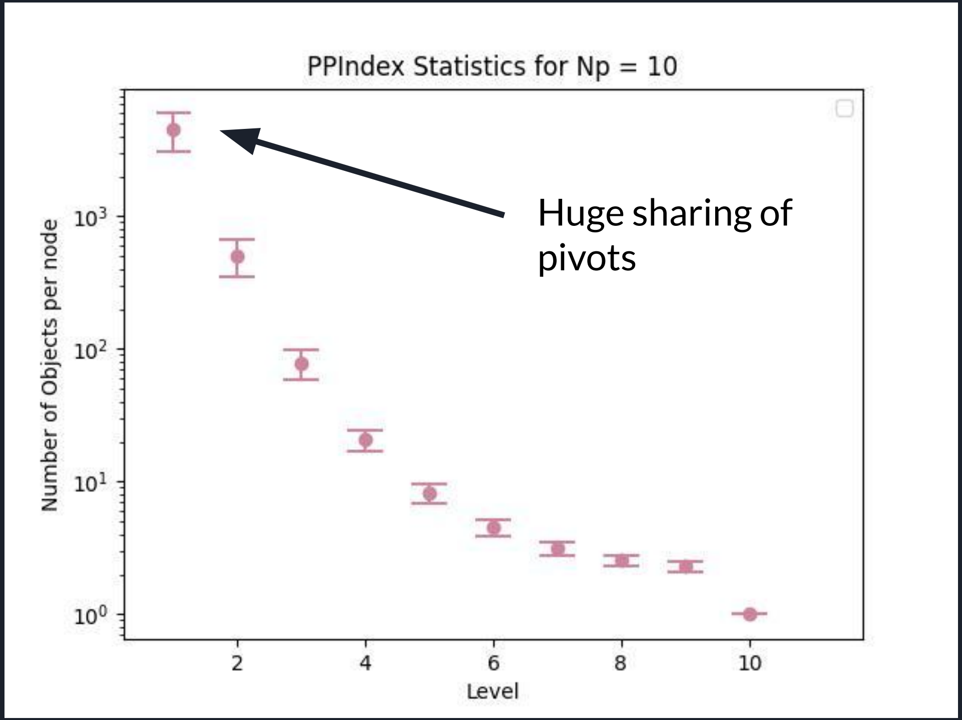
This means that at first level of the tree, each subtree contains around 180 object.
Hence impossible to achieve Z = K + 200 for any K, leading to incorrect results. More on this later.

Analysis of Prefix length and objects distribution (i)

xAxis = Tree level
yAxis = objects per node of that level

Random pivots

The first value is exactly N/Np



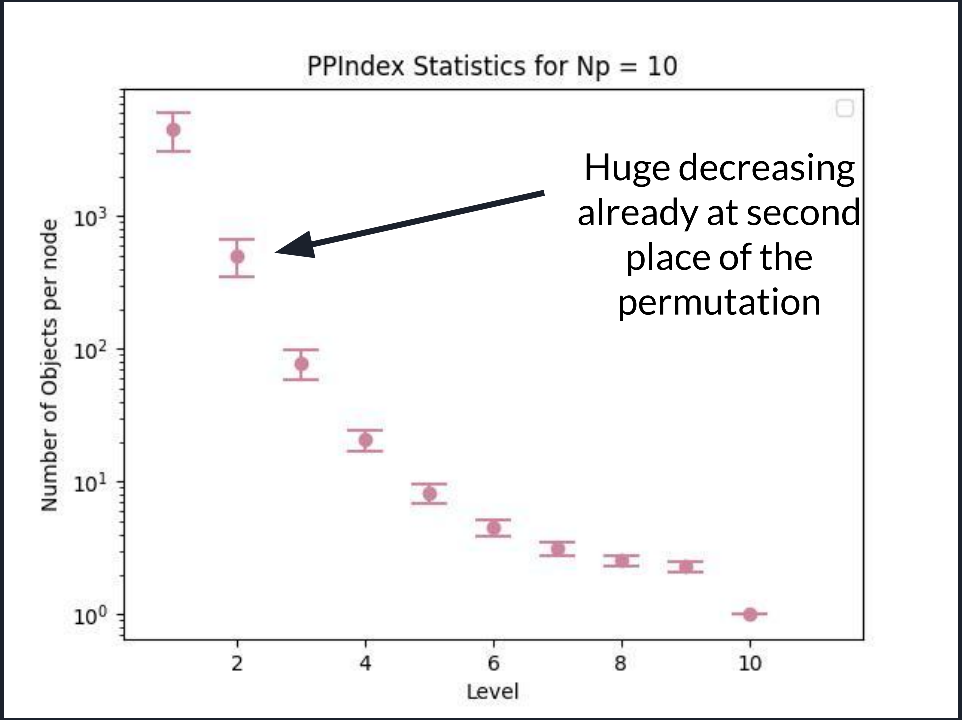
Confidence Interval at 95%

Analysis of Prefix length and objects distribution (ii)

xAxis = Tree level
yAxis = objects per node of that level

Random pivots

The first value is exactly N/Np



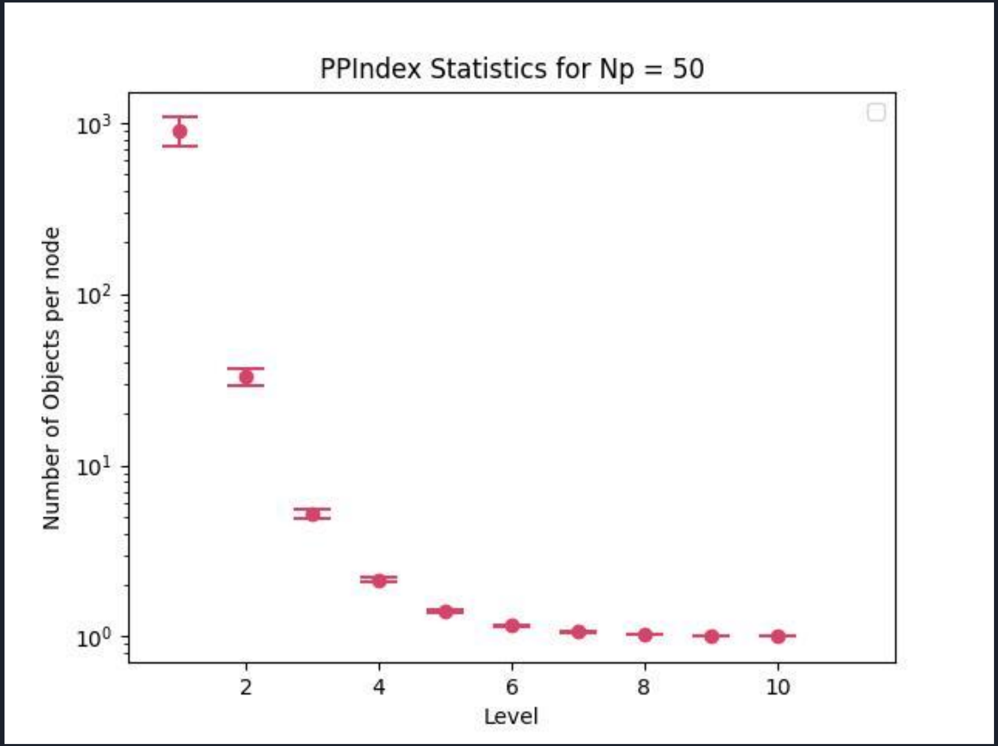
Confidence Interval at 95%

Analysis of Prefix lenght and objects distribution (iii)

xAxis = Tree level
yAxis = objects per node of that level

Random pivots

The first value is exactly N/Np

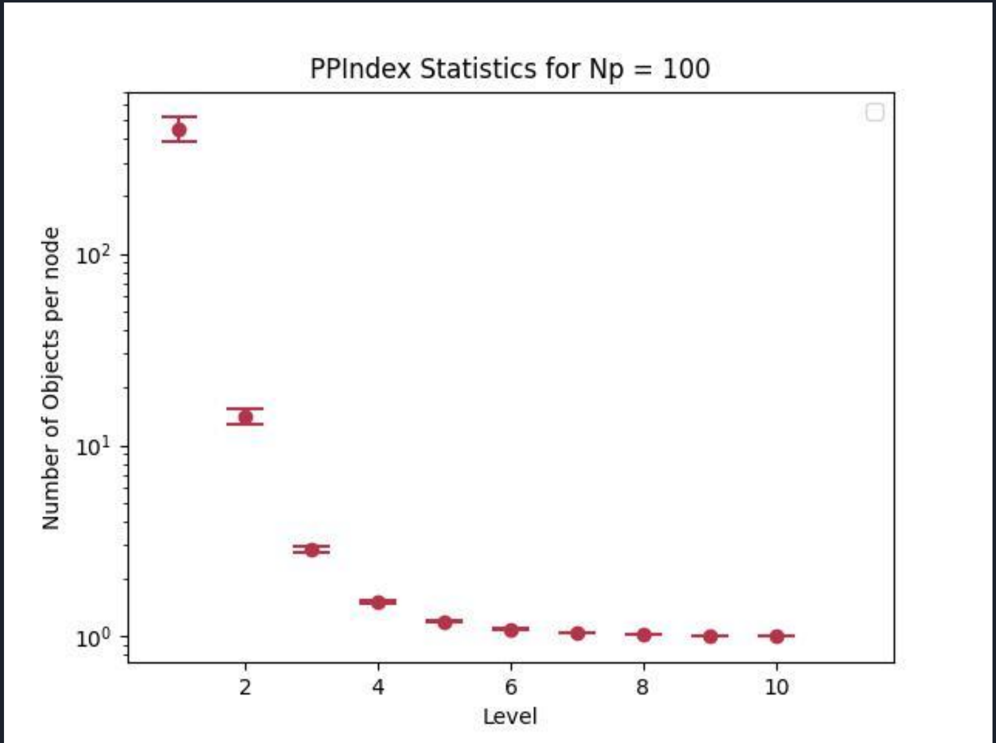


Analysis of Prefix lenght and objects distribution(iv)

xAxis = Tree level
yAxis = objects per node of that level

Random pivots

The first value is exactly N/Np

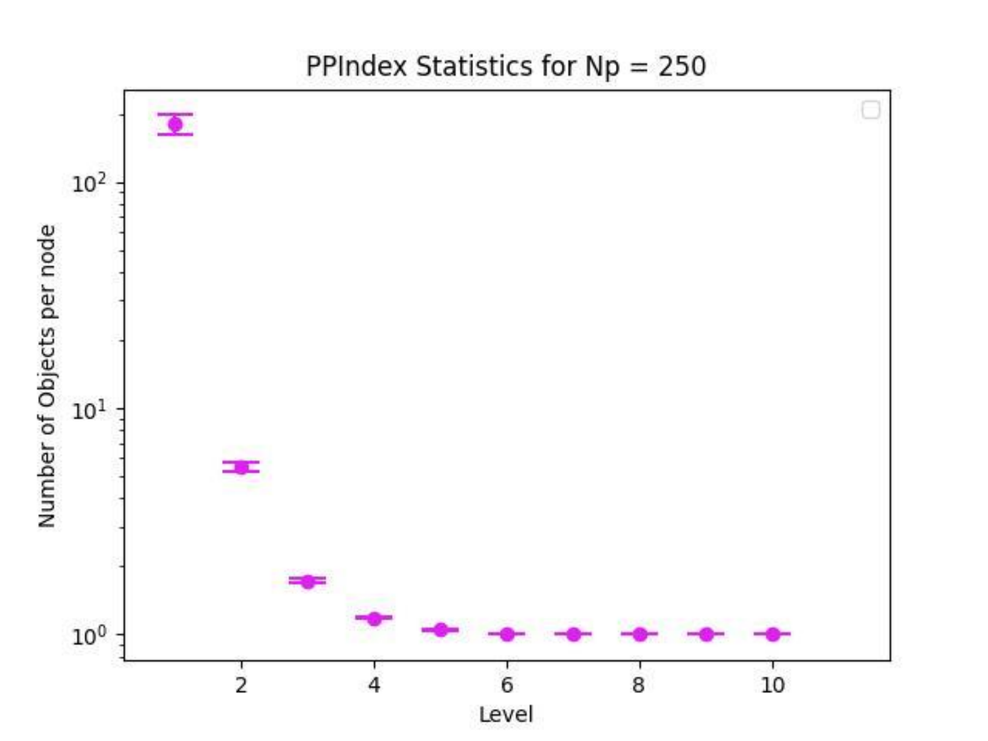


Analysis of Prefix length and objects distribution(v)

xAxis = Tree level
yAxis = objects per node of that level

Random pivots

The first value is exactly N/Np

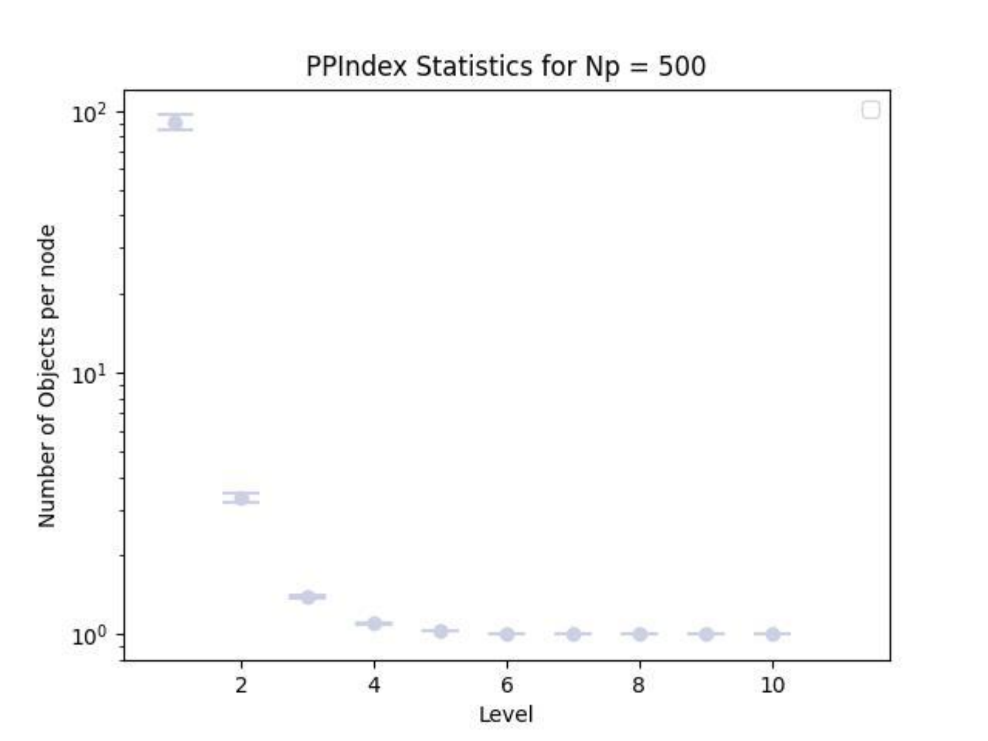


Analysis of Prefix length and objects distribution(vi)

xAxis = Tree level
yAxis = objects per node of that level

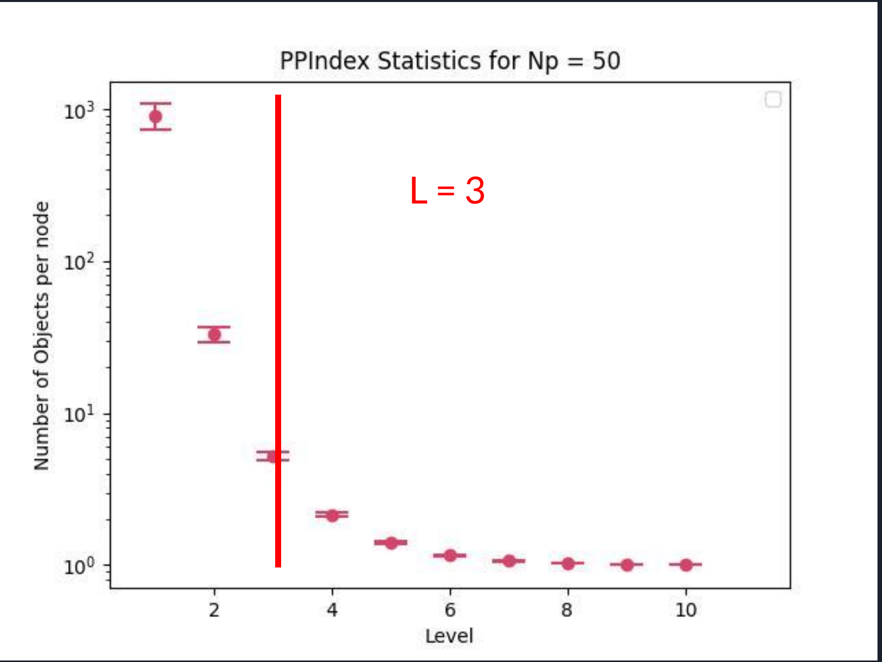
Random pivots

The first value is exactly N/Np

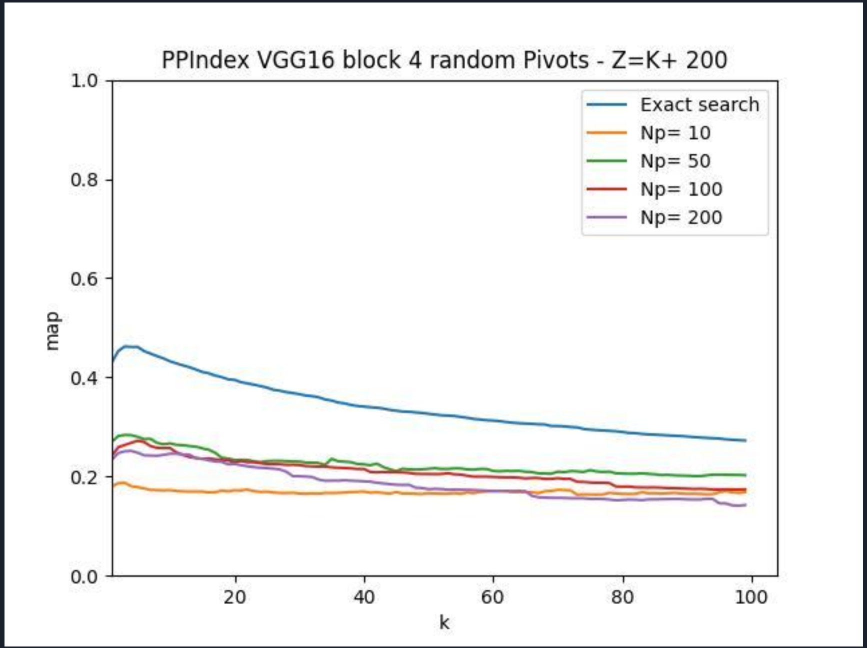


Analysis of Prefix length and objects distribution(vi)

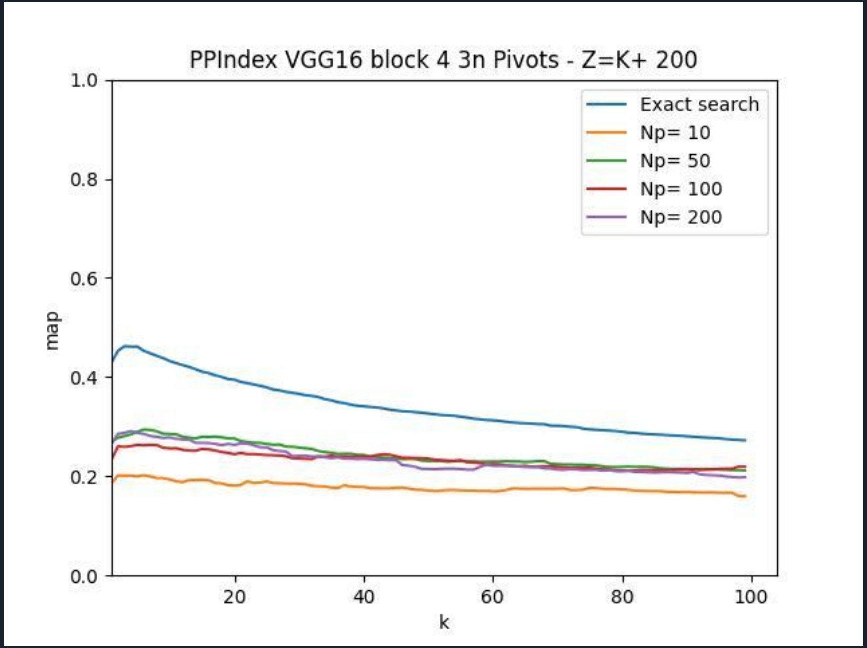
It’s possible to extract some euristics on where to cut the prefix and so the depth of the tree, considering the exponential decreasing of the distribution of the object for each level of the tree.



Pivot Selection Method Analysis (i)

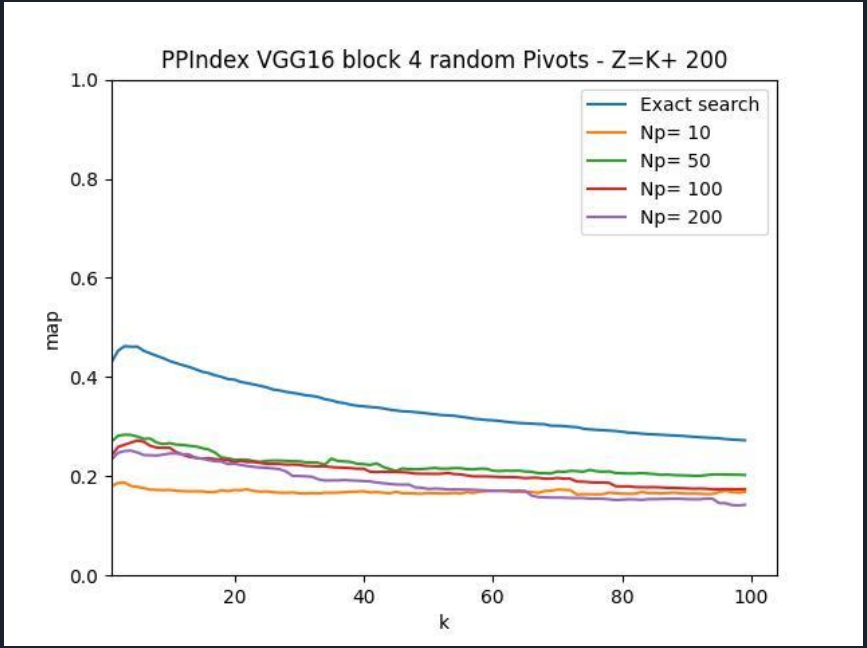


Random

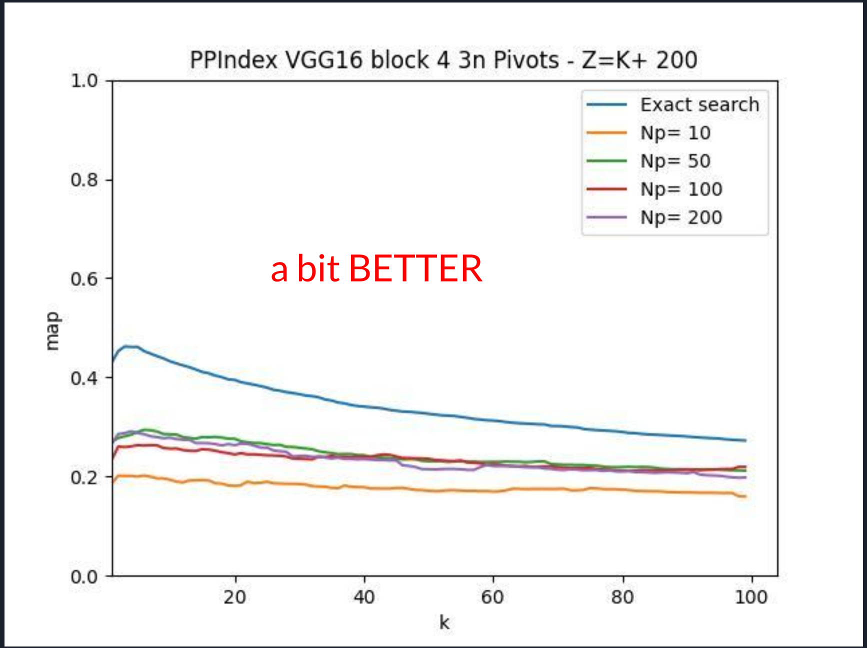


3n

Pivot Selection Method Analysis (ii)

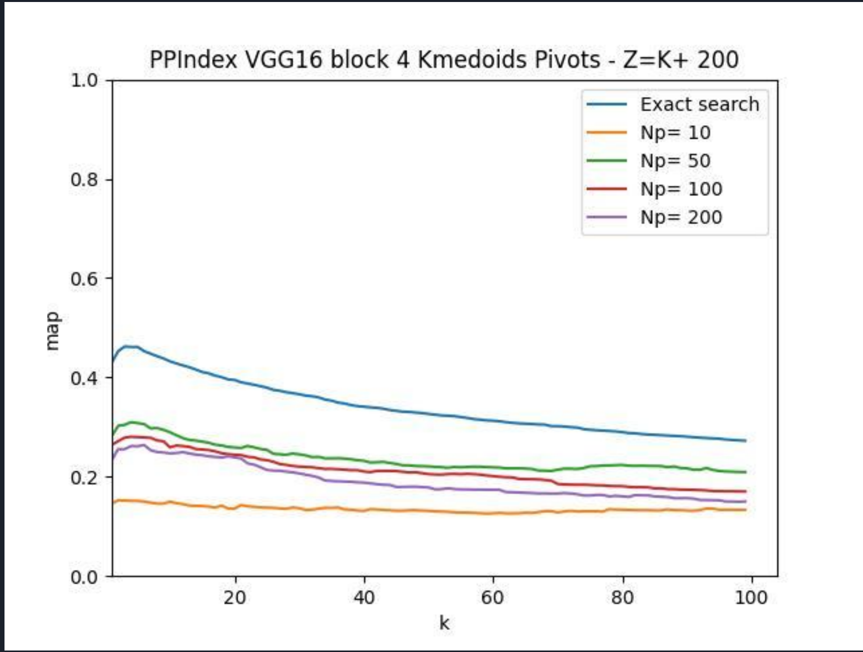


Random

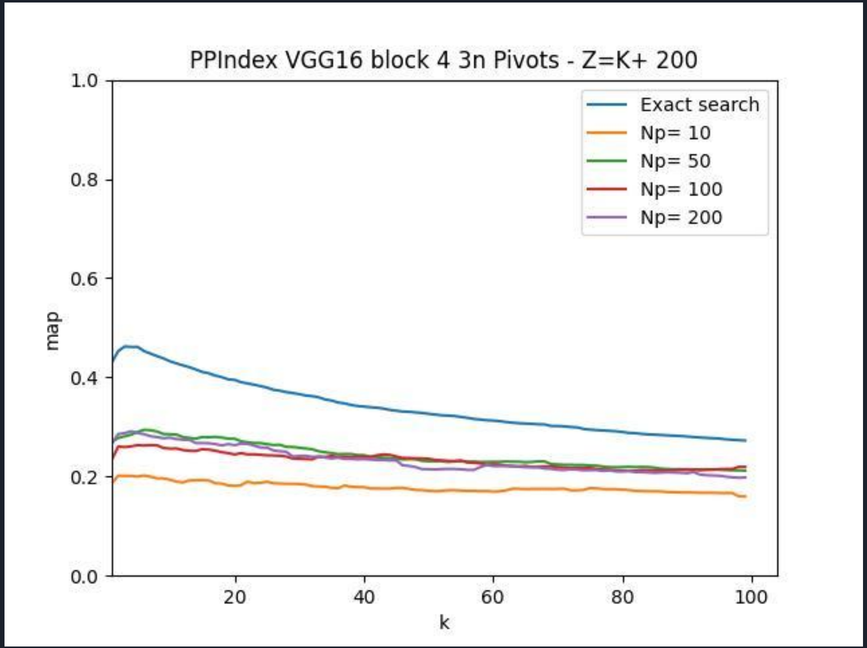


3n

Pivot Selection Method Analysis (iii)

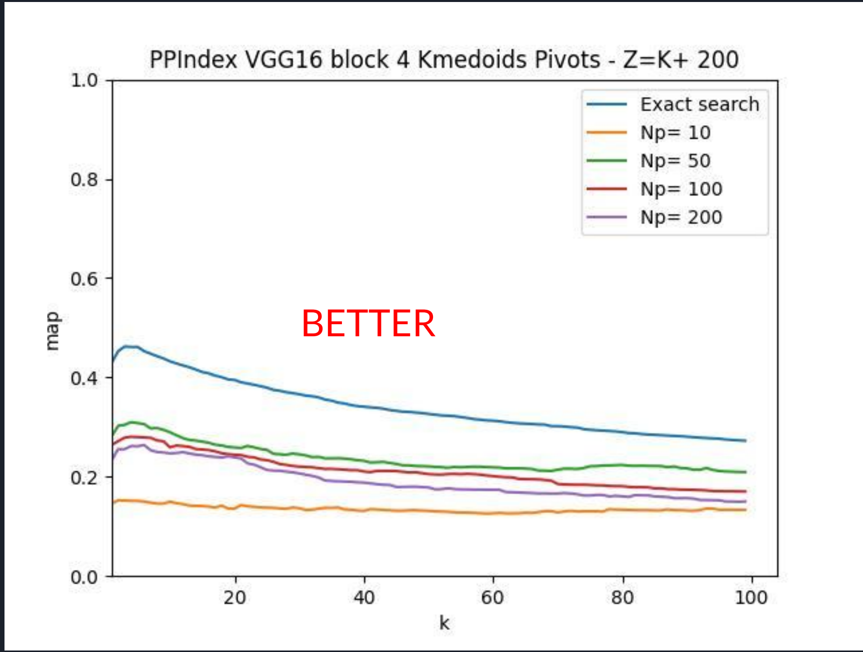


Kmedoids

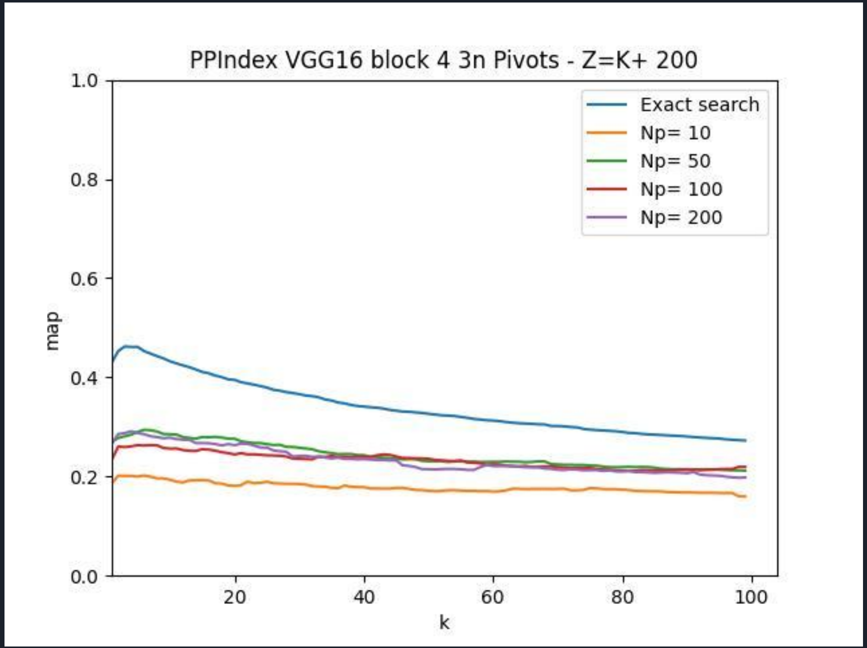


3n

Pivot Selection Method Analysis (iv)



Kmedoids



3n

Improving performance - Including Perturbation of the query in the search algorithm (i)

Given a k-NN query for an object q ∈ O, the basic search function of the PP-Index consists of computing the permutation prefix Πl p and ***searching for the longest prefix match in the prefix tree whose subtree points to at least z candidate objects*** *.[1]*

[1] PP-Index: Using Permutation Prefixes for Efficient and Scalable Similarity Search - Andrea Esuli - Istituto di Scienza e Tecnologie dell'Informazione - CNR

Improving performance - Including Perturbation of the query in the search algorithm (ii)

*Given a k-NN query for an object $q \in O$, the basic search function of the PP-Index consists of computing the permutation prefix Πl p and **searching for the longest prefix match in the prefix tree whose subtree points to at least z candidate objects** .[1]*

*Multiple query: at search time, p additional permutation prefixes from the query permutation prefix Πl q are generated, **by swapping the position of some of its elements**. The geometric rationale is that a permutation prefix Πl x differing from another permutation prefix Πl y for **the swap of two adjacent/near elements identifies an area $V\Pi l$ x of the similarity space adjacent/near to $V\Pi l$ y** . Performing a search with **additional "swapped" permutation prefixes extends the search process to areas of the search space that are likely to contain relevant objects.***

[1] PP-Index: Using Permutation Prefixes for Efficient and Scalable Similarity Search - Andrea Esuli - Istituto di Scienza e Tecnologie dell'Informazione - CNR

Improving performance - Including Perturbation of the query in the search algorithm (ii)

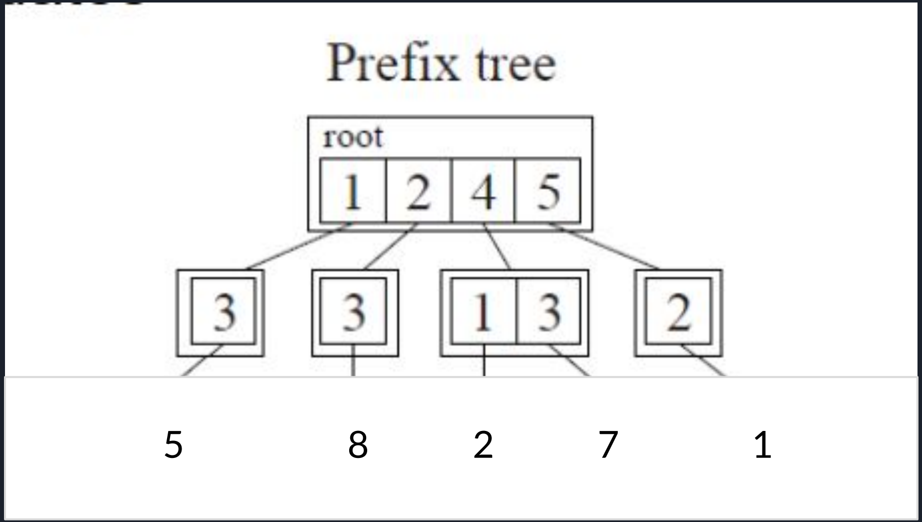
Given a k-NN query for an object q ∈ O, the basic search function of the PP-Index consists of computing the permutation prefix Πl p and ***searching for the longest prefix match in the prefix tree whose subtree points to at least z candidate objects*** *.[1]*

Multiple query: at search time, p additional permutation prefixes from the query permutation prefix Πl q are generated, ***by swapping the position of some of its elements****. The geometric rationale is that a permutation prefix Πl x differing from another permutation prefix Πl y for* ***the swap of two adjacent/near elements identifies an area VΠl x of the similarity space adjacent/near to VΠl y*** *. Performing a search with* ***additional "swapped" permutation prefixes extends the search process to areas of the search space that are likely to contain relevant objects.***

[1] PP-Index: Using Permutation Prefixes for Efficient and Scalable Similarity Search - Andrea Esuli - Istituto di Scienza e Tecnologie dell'Informazione - CNR

Improving performance - Including Perturbation of the query in the search algorithm (ii)

Example:



Number of objects under the node

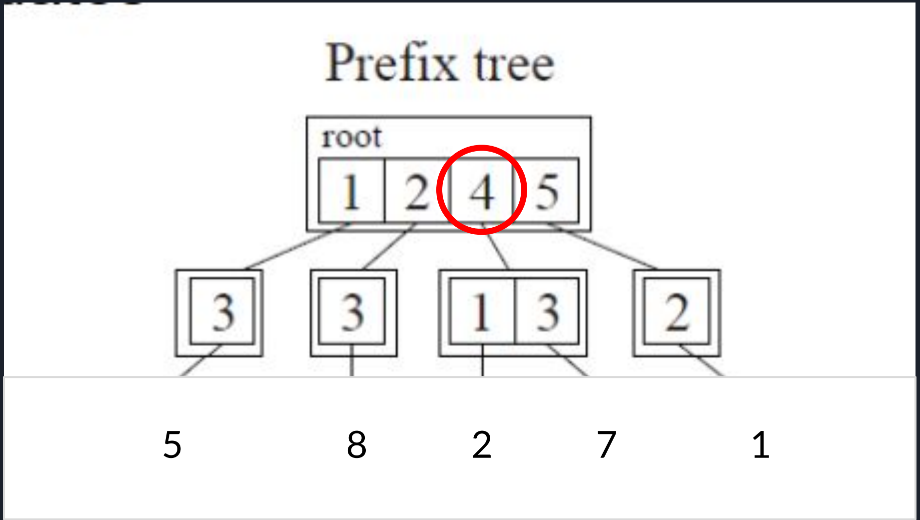
q = [4, 1, 3, 5]

Z = 10

Object captured = 0

Improving performance - Including Perturbation of the query in the search algorithm (ii)

Example:



Number of objects under the node

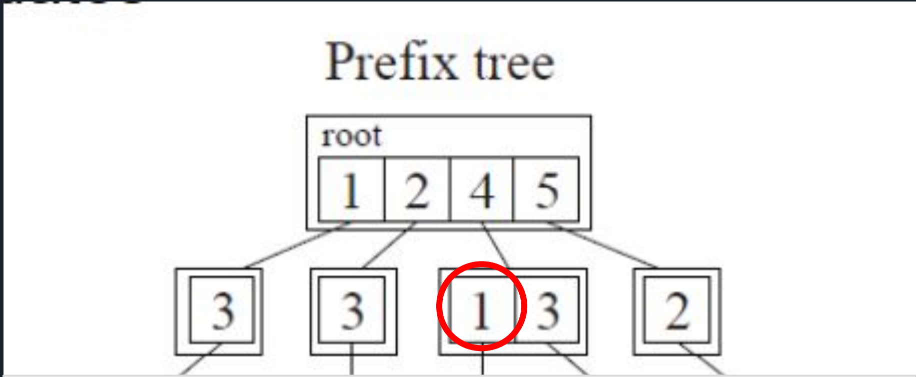
$q = [4, 1, 3, 5]$

$Z = 10$

Object captured = 0

Improving performance - Including Perturbation of the query in the search algorithm (ii)

Example:



Number of objects under the node	5	8	2	7	1

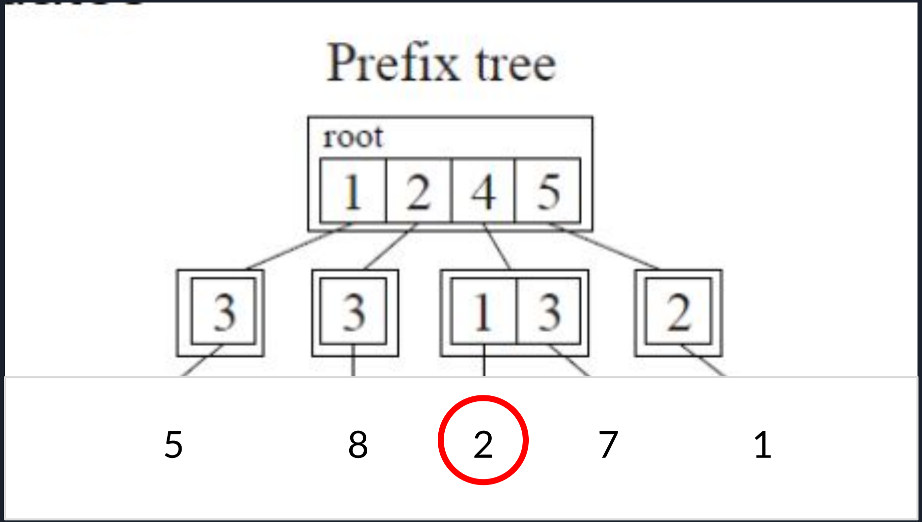
q = [4, 1, 3, 5]

Z = 10

Object captured = 0

Improving performance - Including Perturbation of the query in the search algorithm (ii)

Example:



Number of objects under the node

Take object inside

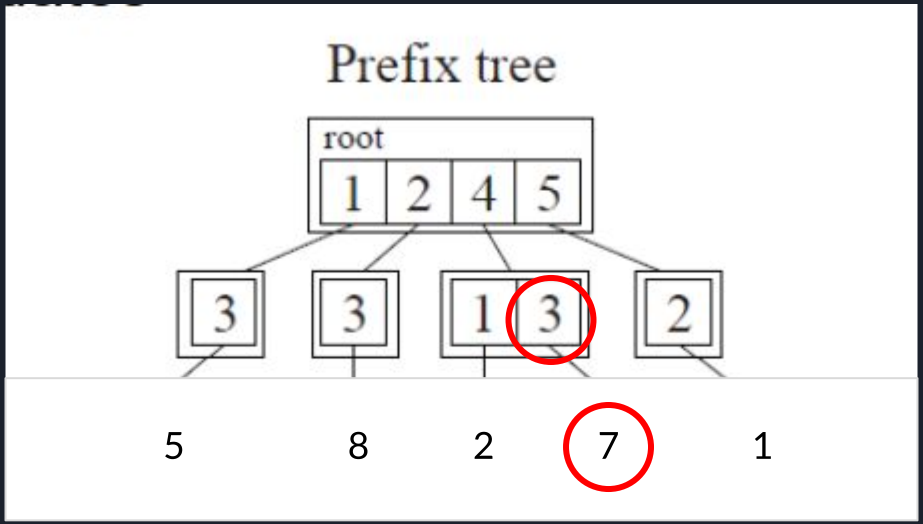
q = [4, 1, 3, 5]

Z = 10

Object captured = 2

Improving performance - Including Perturbation of the query in the search algorithm (ii)

Example:



Number of objects under the node

q = [4, 1, 3, 5]

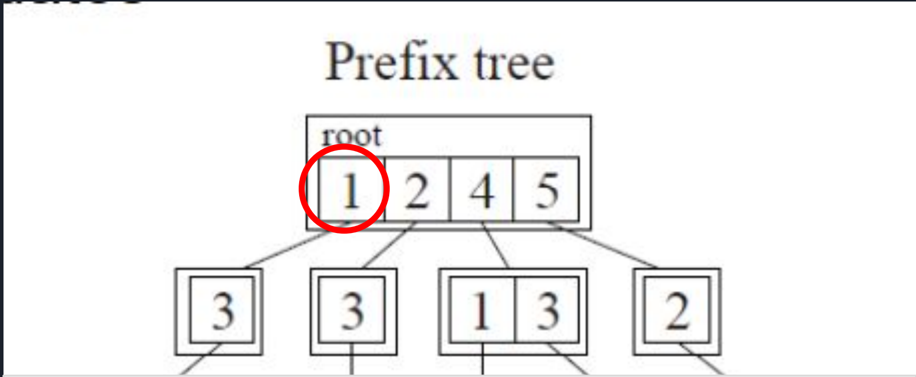
Z = 10

Object captured = 2 + 7

Go to the sibling and take all the objects

Improving performance - Including Perturbation of the query in the search algorithm (ii)

Example:



5	8	2	7	1

Number of objects under the node

q = [4, 1, 3, 5]

Z = 10

Object captured = 2 + 7 +1

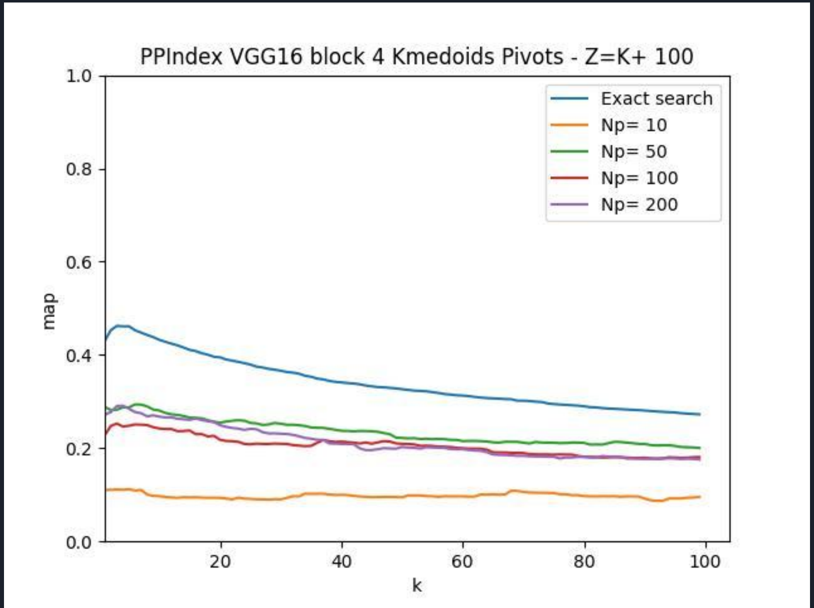
Still not enough go back to the root node and take element from the second best subtree and repeat the search

Improving performance - Including Perturbation of the query in the search algorithm (ii)

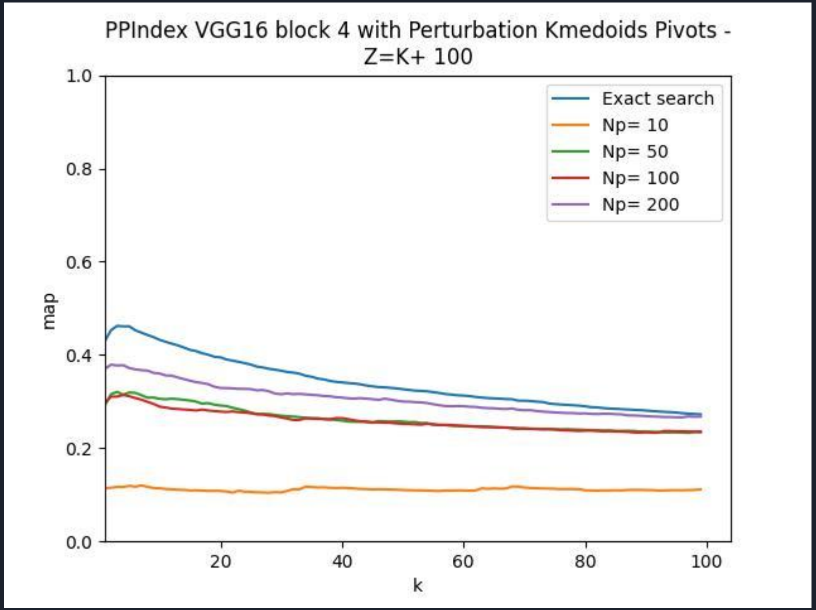
Small modifications to Algorithm:

- Search the deeper matching subtree that contains at least Z candidates
- If the deeper matching subtree contains less than Z candidates, include also the candidate of the siblings of the last matching node (maybe better the siblings in matching order respect to the query)
- If the first matching node of the Prefix tree (root) does not contains enough objects (< Z) search following the second element of the permutation of the query
- Don’t cut the permutation of the query to L for the search, to allow to explore all the subtree in order of distance if necessary

Improving performance - Including Perturbation of the query in the search algorithm (ii)



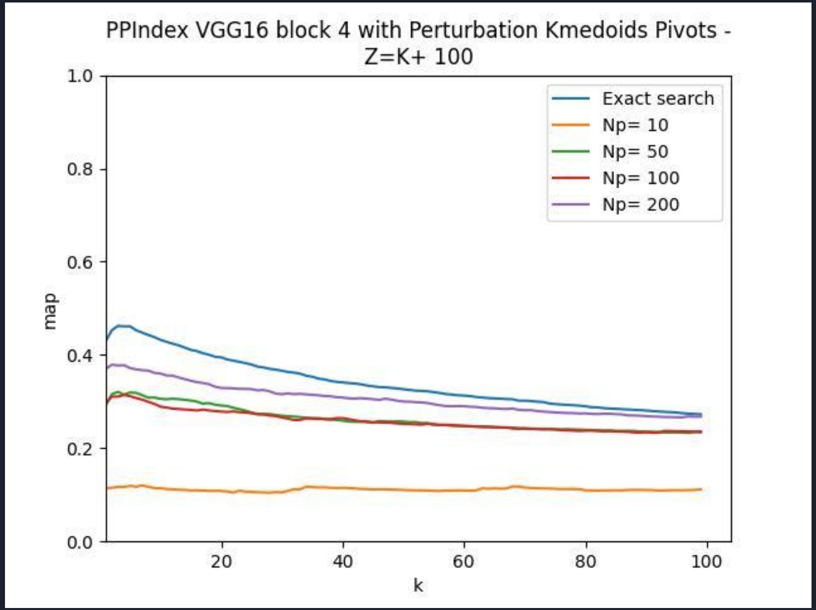
No Perturbation



With Perturbation

Improving performance - Including Perturbation of the query in the search algorithm (ii)

This approach also overcome the problem of the increasing granularity for large number of pivots for a small dataset.
Here a greater number of pivots improve the performance



With Perturbation

Query Time Analysis (i)

It is possible to estimate the query time considering the amount of computation the algorithms does.

$$M*distanceComp+M*log\left(M\right)+log\left(L\right)+Retrieval+Z*distanceComp+Z*log\left(Z\right)$$

Distances to pivots

M = Number of pivots
L = Prefix Lenght
Z = number of Candidates

Query Time Analysis (i)

It is possible to estimate the query time considering the amount of computation the algorithms does.

$$M * distanceComp + M * log(M) + log(L) + Retrieval + Z * distanceComp + Z * log(Z)$$

Ordering of Pivots for Permutations

M = Number of pivots
L = Prefix Lenght
Z = number of Candidates

Query Time Analysis (i)

It is possible to estimate the query time considering the amount of computation the algorithms does.

$$M * distanceComp + M * log(M) + log(L) + Retrieval + Z * distanceComp + Z * log(Z)$$

Searching in the Tree (Negligeble)

M = Number of pivots
L = Prefix Lenght
Z = number of Candidates

Query Time Analysis (i)

It is possible to estimate the query time considering the amount of computation the algorithms does.

$$M*distanceComp+M*log\,(M)+log\,(L)+Retrieval+Z*distanceComp+Z*log\,(Z)$$

Retrieval from disk

M = Number of pivots
L = Prefix Lenght
Z = number of Candidates

Query Time Analysis (i)

It is possible to estimate the query time considering the amount of computation the algorithms does.

$$M * distanceComp + M * log(M) + log(L) + Retrieval + Z * distanceComp + Z * log(Z)$$

Real distance to Z candidates

M = Number of pivots
L = Prefix Lenght
Z = number of Candidates

Query Time Analysis (i)

It is possible to estimate the query time considering the amount of computation the algorithms does.

$$M * distanceComp + M * log(M) + log(L) + Retrieval + Z * distanceComp + Z * log(Z)$$

Rerank the results

M = Number of pivots
L = Prefix Lenght
Z = number of Candidates

Query Time Analysis (i)

To avoid using seconds as time evaluation because strictly depends on the machine, we used the adimensional time ratio, evaluating the Query Time using the PPindex and the Query Time using the NaiveSearch (Total Scan of the dataset).

Query Time Analysis (ii)

To avoid using seconds as time evaluation because strictly depends on the machine, we used the adimensional time ratio, evaluating the Query Time using the PPindex and the Query Time using the NaiveSearch (Total Scan of the dataset).

We used the *Fieller's theorem, that finds the CI of a ratio of two random variables*

Fieller showed that if *a* and *b* are (possibly correlated) means of two samples with expectations μ_a and μ_b, and variances $\nu_{11}\sigma^2$ and $\nu_{22}\sigma^2$ and covariance $\nu_{12}\sigma^2$, and if $\nu_{11}, \nu_{12}, \nu_{22}$ are all known, then a (1 − *α*) confidence interval (m_L, m_U) for μ_a/μ_b is given by

$$(m_L, m_U) = \frac{1}{(1-g)}\left[\frac{a}{b} - \frac{g\nu_{12}}{\nu_{22}} \mp \frac{t_{r,\alpha}s}{b}\sqrt{\nu_{11} - 2\frac{a}{b}\nu_{12} + \frac{a^2}{b^2}\nu_{22} - g\left(\nu_{11} - \frac{\nu_{12}^2}{\nu_{22}}\right)}\right]$$

where

$$g = \frac{t_{r,\alpha}^2 s^2 \nu_{22}}{b^2}.$$

Here s^2 is an unbiased estimator of σ^2 based on r degrees of freedom, and $t_{r,\alpha}$ is the α-level deviate from the Student's t-distribution based on *r* degrees of freedom.

Fieller EC: The biological standardization of Insulin. Suppl to J R Statist Soc 1940, 7:1-64.

Query Time Analysis (ii)

To avoid using seconds as time evaluation because strictly depends on the machine, we used the adimensional time ratio, evaluating the Query Time using the PPindex and the Query Time using the NaiveSearch (Total Scan of the dataset).

We used the *Fieller's theorem, that finds the CI of a ratio of*

$$IE = \frac{Cost(Q)}{Cost^A(Q)}$$

Fieller showed that if *a* and *b* are (possibly correlated) means of two samples with expectations μ_a and μ_b, and va then a (1 − *α*) confidence interval (m_L, m_U) for μ_a/μ_b is given by

$$(m_L, m_U) = \frac{1}{(1-g)}\left[\frac{a}{b} - \frac{g\nu_{12}}{\nu_{22}} \mp \frac{t_{r,\alpha}s}{b}\sqrt{\nu_{11} - 2\frac{a}{b}\nu_{12} + \frac{a^2}{b^2}\nu_{22} - g\left(\nu_{11} - \frac{\nu_{12}^2}{\nu_{22}}\right)}\right]$$

where

$$g = \frac{t_{r,\alpha}^2 s^2 \nu_{22}}{b^2}.$$

Here s^2 is an unbiased estimator of σ^2 based on r degrees of freedom, and $t_{r,\alpha}$ is the α-level deviate from the Student's t-distribution based on *r* degrees of freedom.

Fieller EC: The biological standardization of Insulin. Suppl to J R Statist Soc 1940, 7:1-64.

Query Time Analysis (iii)

In our case the two random variable are the made with a query test of the same 250 Queries, from which we can extract the mean and the variance.

Experiments Parameters:

- **Pivots Selection method: Kmedoids**
- **Number of Pivots: 50**
- **K = 20**
- **l=3**
- **All the dataset is in memory (since the machine has and SSD the linear scanning of the reordered blocks could not be a lot visible)**

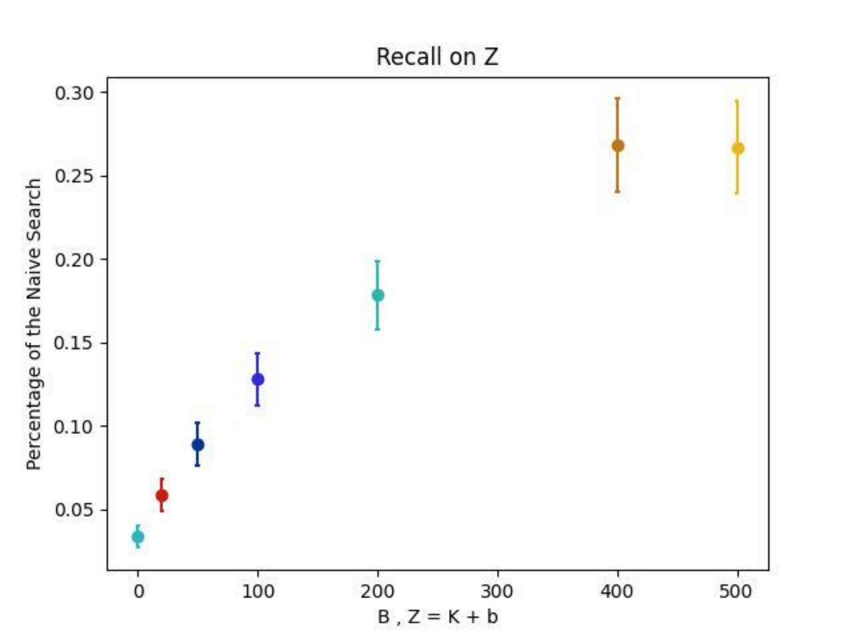
Evaluation metrics:

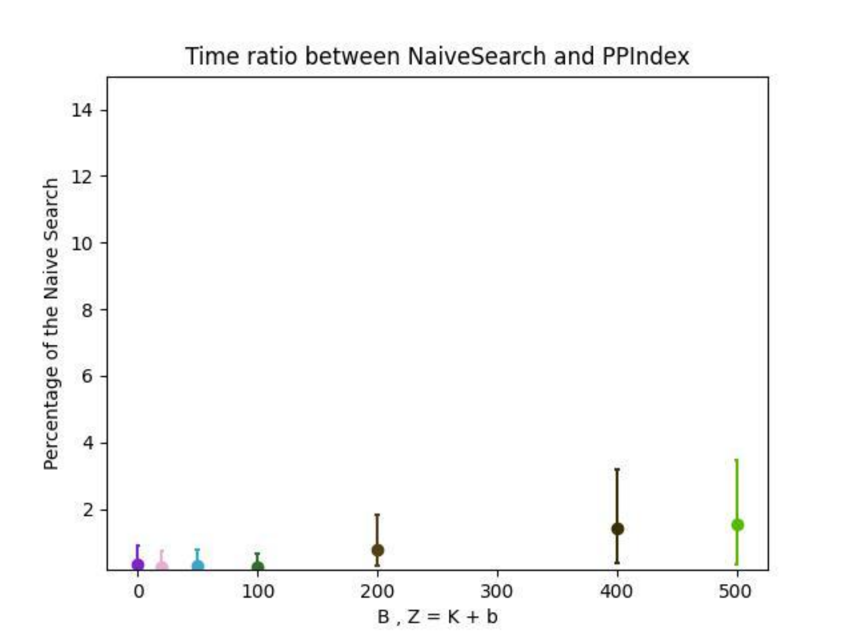
- **Ratio of the Query Time means**
- **Recall between the retrieved set (regardless the ordering)**

$$Recall(k) = \frac{|D_q^k \cap P_q^k|}{k}$$

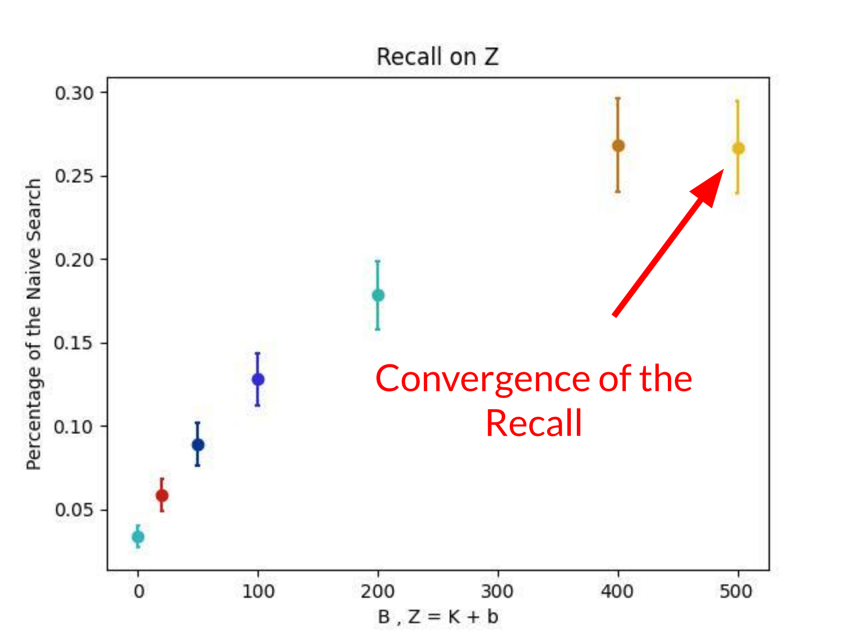
Dq : Exact Search Retrieved Object; Pq: Approximate Search Retrieved Object

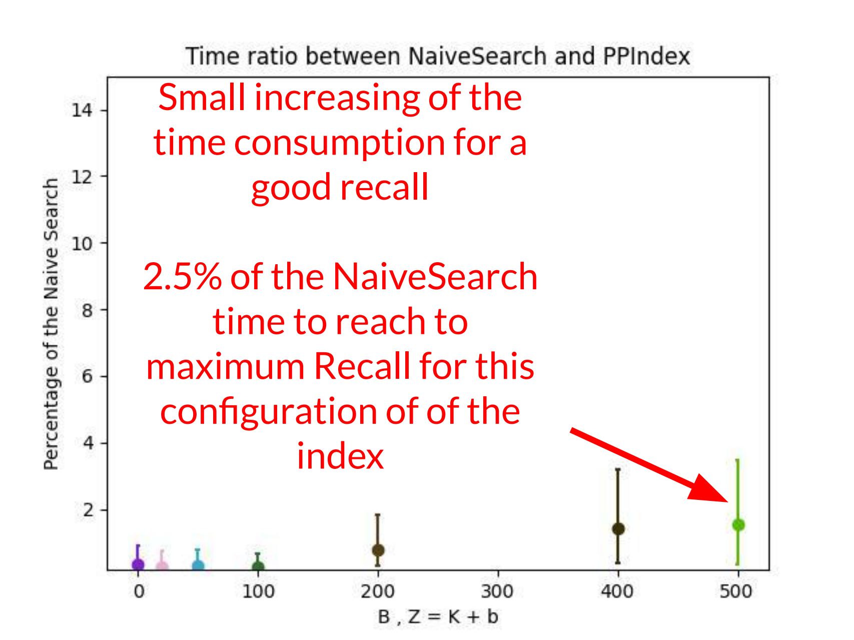
Query Time Analysis (iv)



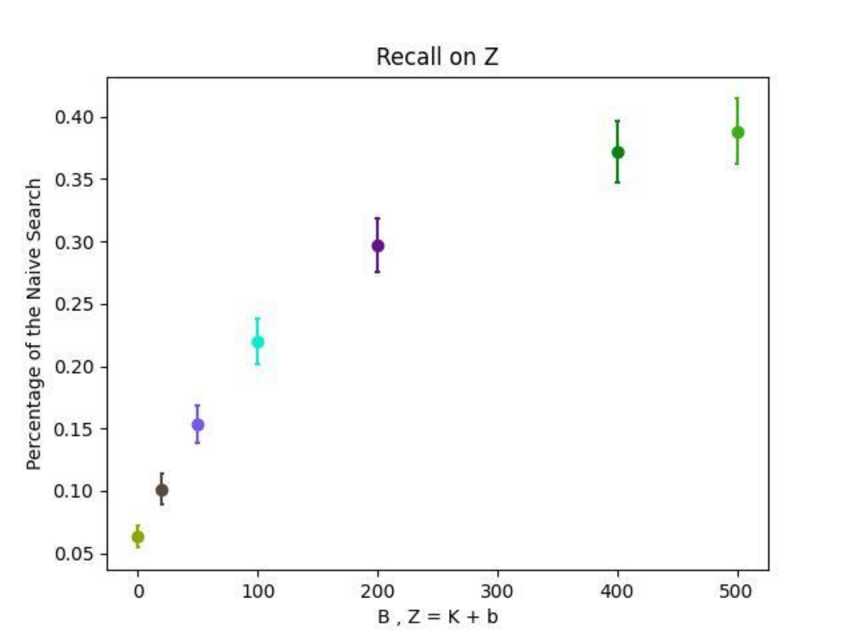


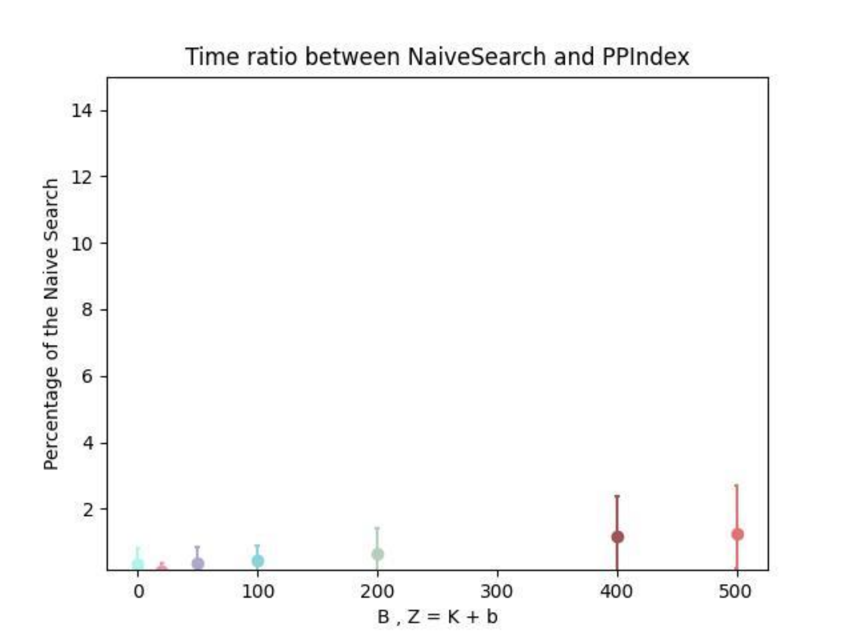
Query Time Analysis (v)





Query Time Analysis - With Perturbation (vii)





With perturbation search with almost no increasing of the time

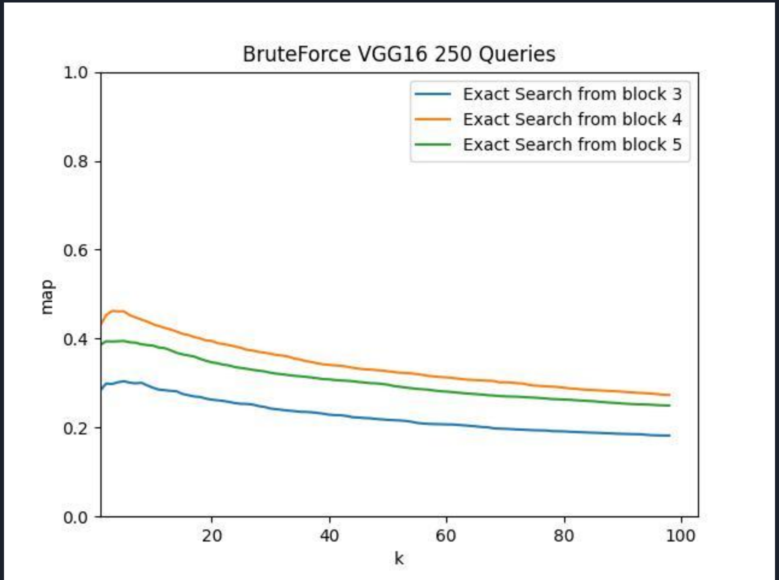
Conclusions

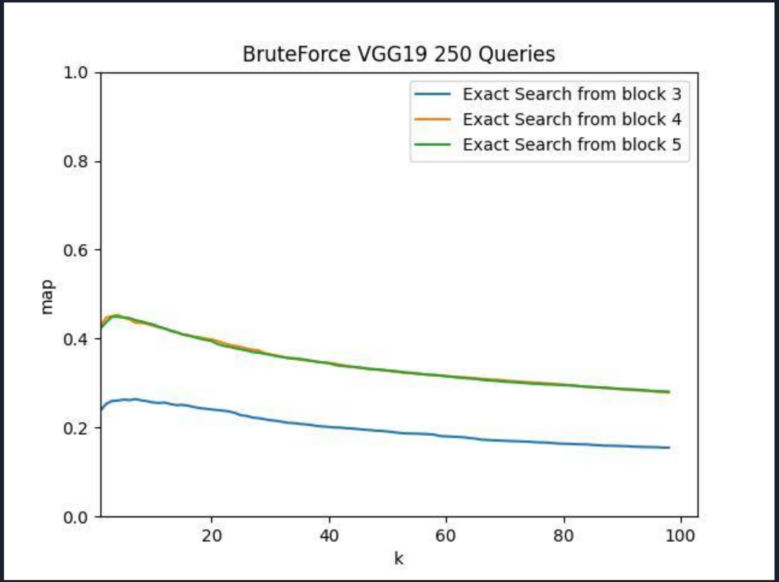
- Best pivots selection method: ***Kmedoids***
- Number of Pivots: **around 0.5% of the size of the dataset**
- **Use perturbation in the search to better explore the space**
- **Cut L to small value considering an exponential decreasing of the object for each level**

Pipeline

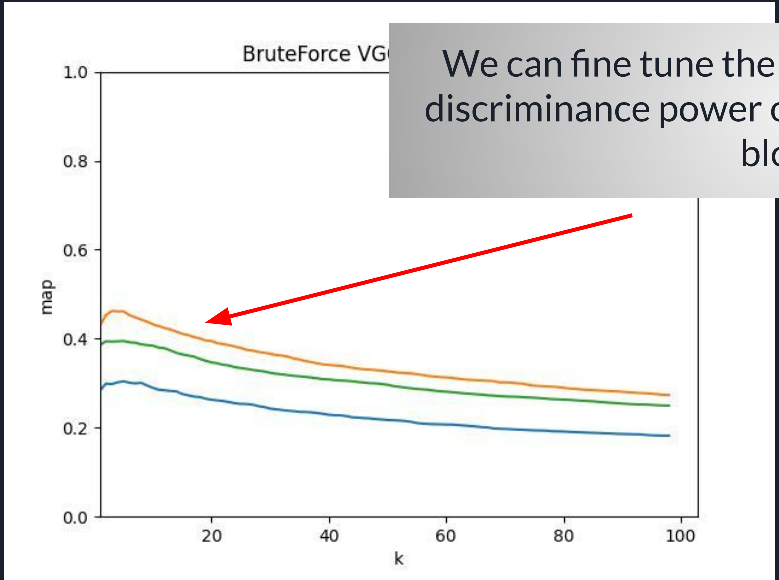
1. Features Extraction from pretrained CNN
2. Performance Analysis
3. Features Extraction from Fine Tuned CNN
4. Performance Analysis

Features Extraction from Fine Tuned CNN (i)

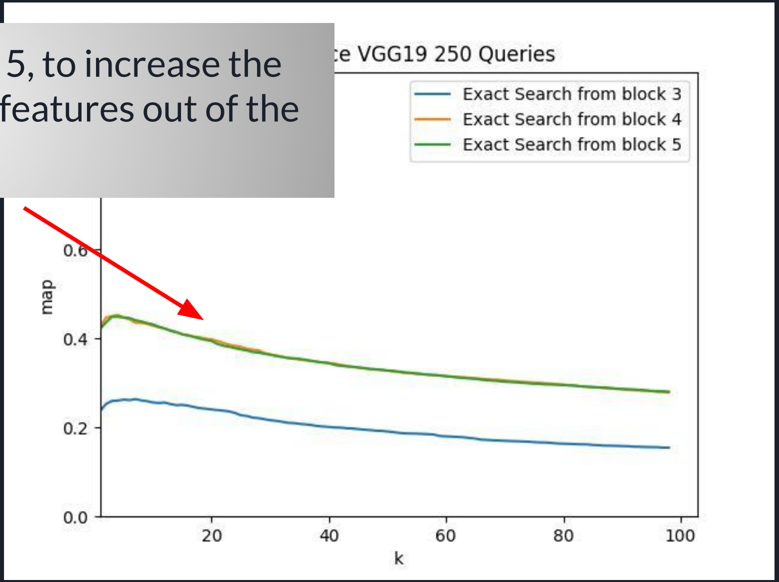




Features Extraction from Fine Tuned CNN (i)

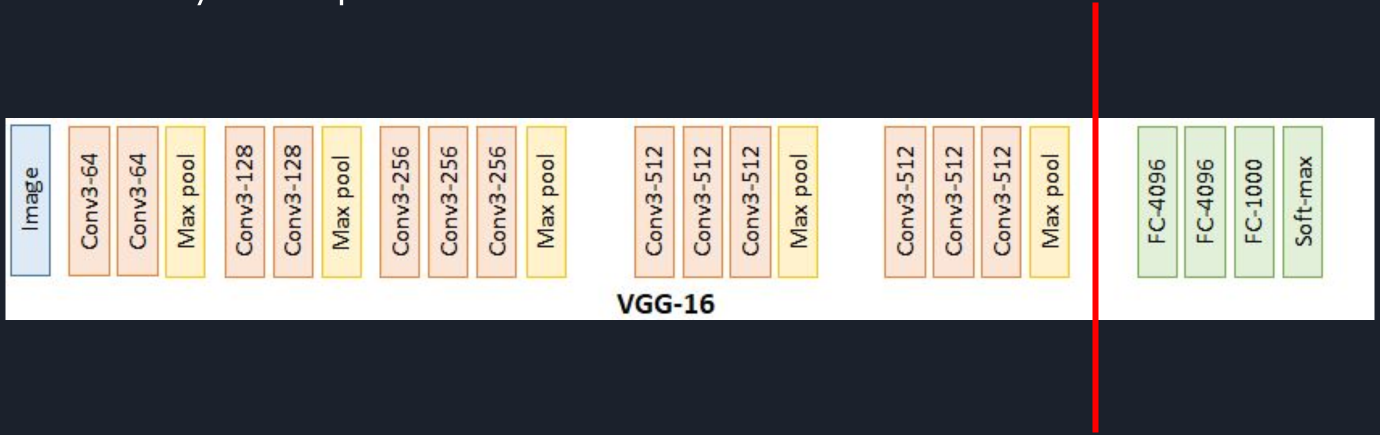


We can fine tune the block 5, to increase the discriminance power of the features out of the block 4



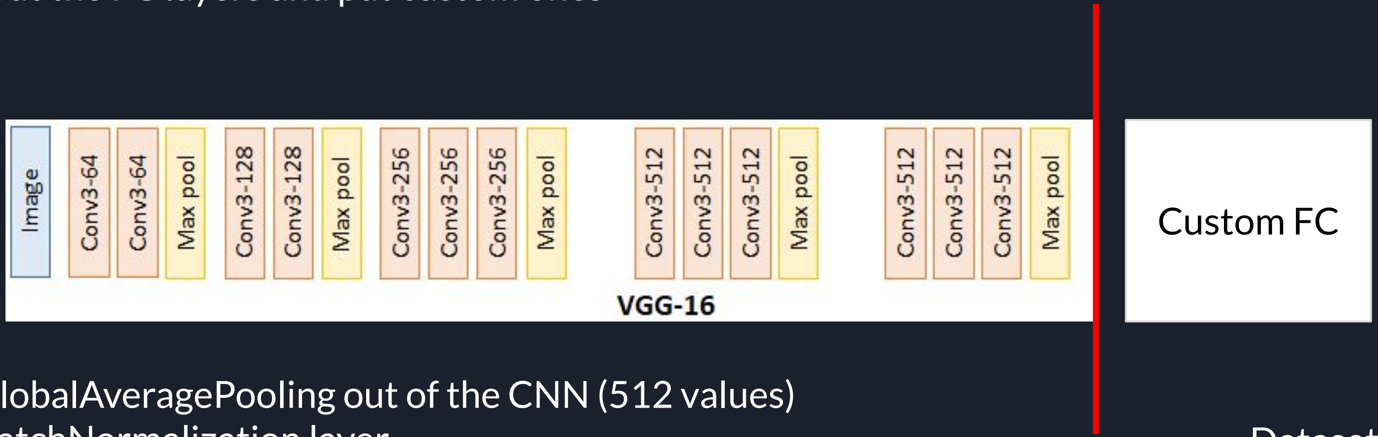
Fine Tuning Pipeline (i)

1) Cut the FC layers and put custom ones



Fine Tuning Pipeline (ii)

1) Cut the FC layers and put custom ones

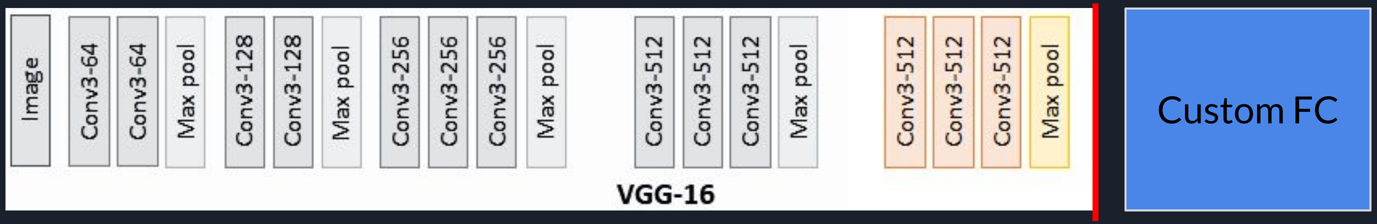


- GlobalAveragePooling out of the CNN (512 values)
- BatchNormalization layer
- 1 Hidden layer with 1024 neurons
- Dropout
- Data augmentation for the training set (training = 0.8, validation = 0.2)
- 250 classes

Dataset of 20’000 samples

Fine Tuning Pipeline (iii)

2) Freeze all the CNN and Train only the Custom FC and the final block

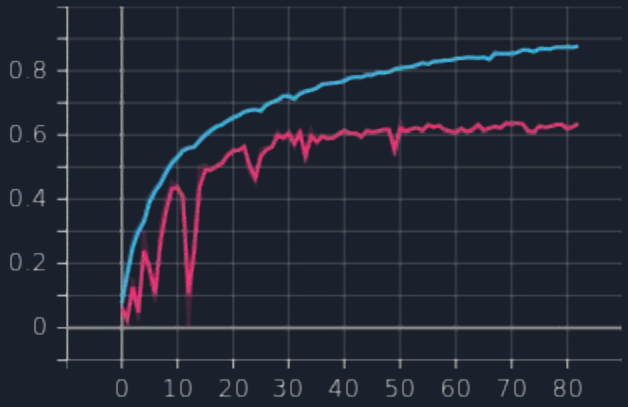


Freezed To train

Actually we should first train only the Custom FC layers, then unfreezing the last block of the VGG and using a smaller learning rate to not the destroy totally the pretrained weights, but considering that has been shown that the last layer is not so useful for the features, that our machine is powerful enough for the training from sckretch of that amount of parameters, we trained totally the last part of the network.

Fine Tuning Pipeline (iv)

2) Freeze all the CNN and Train only the Custom FC and the final block. Results

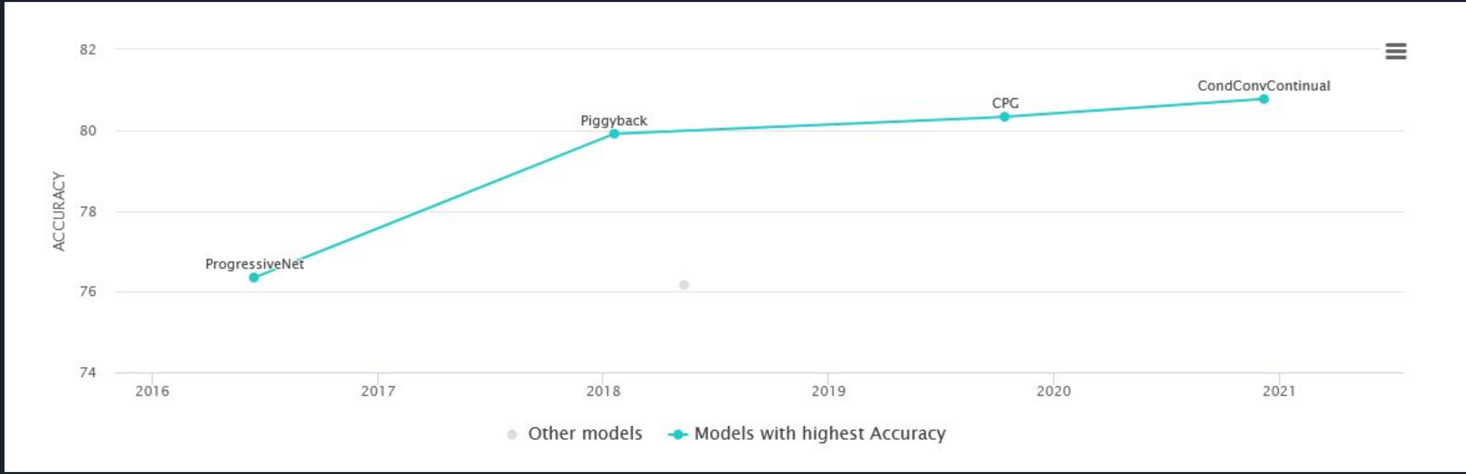


Epoch Accuracy
Test accuracy : 0.62

Epoch Loss
Test loss : 2.04

Fine Tuning Pipeline (iv)

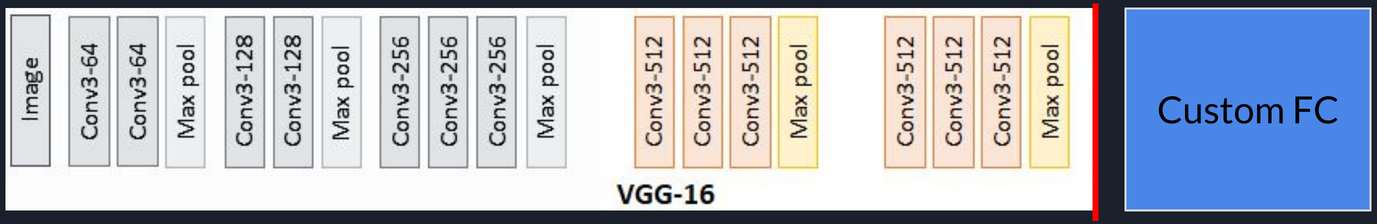
Not to bad considering the state of the art of the classification on this dataset



We can do BETTER!

Fine Tuning Pipeline (iii)

3) Unfreeze part of the CNN that we want to fine tune and retrain with a smaller learning rate



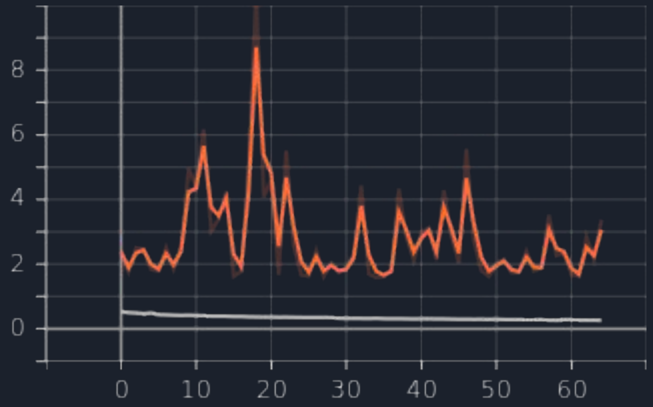
Freezed

To Fine Tune

Fine Tuning Pipeline (iii)

3) Unfreeze part of the CNN that we want to fine tune and retrain with a smaller learning rate.
Results:

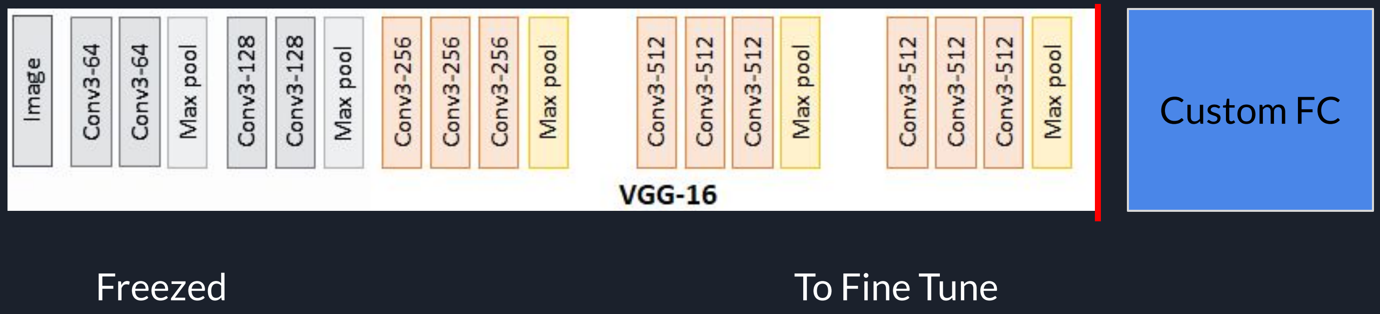
Epoch Accuracy
Test accuracy : 0.70



Epoch Loss
Test loss : 1.57

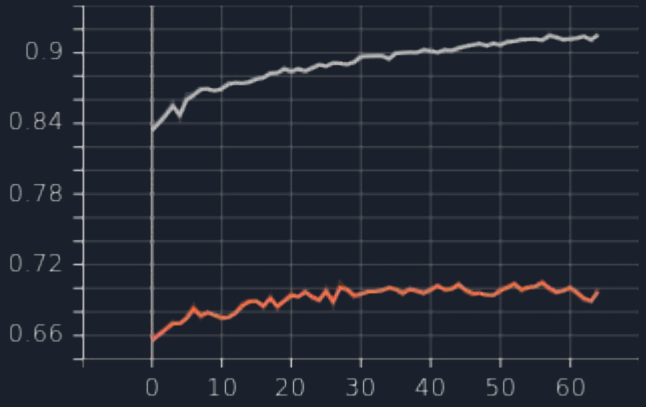
Fine Tuning Pipeline (iii)

3) Unfreeze part of the CNN that we want to fine tune and retrain with a smaller learning rate

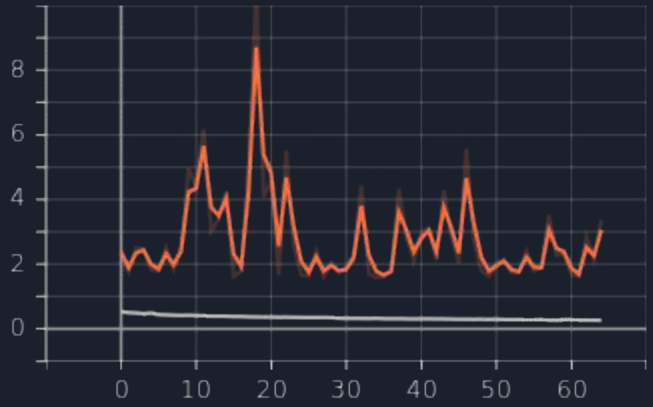


Fine Tuning Pipeline (iv)

2) Freeze all the CNN and Train only the Custom FC



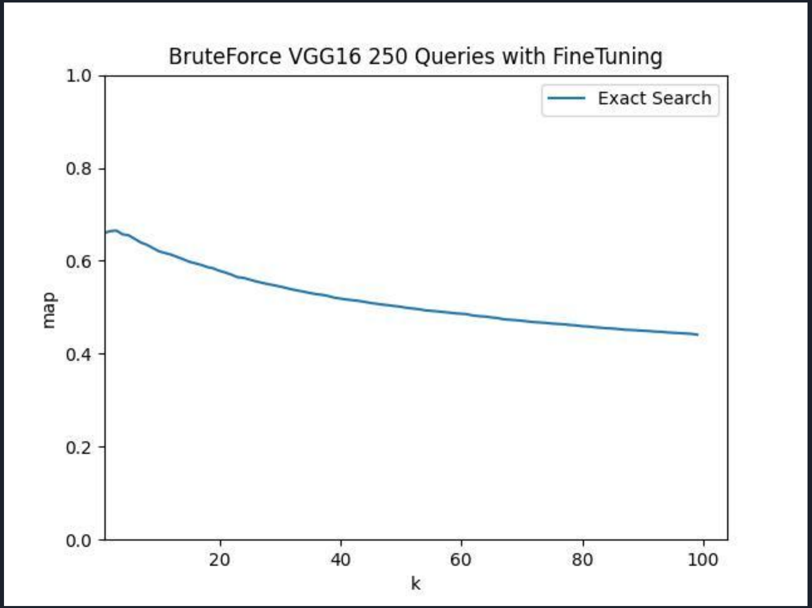
Epoch Accuracy
Test accuracy : 0.71



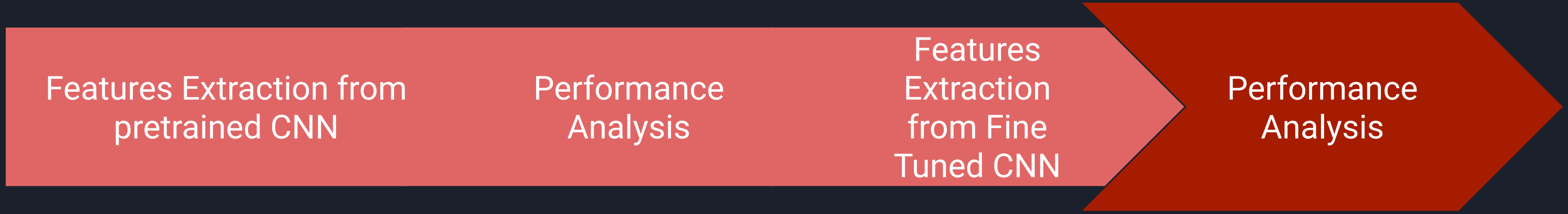
Epoch Loss
Test loss : 1.39

Fine Tuning Pipeline (iv)

3) Extract Features from the CNN and evaluate mAP



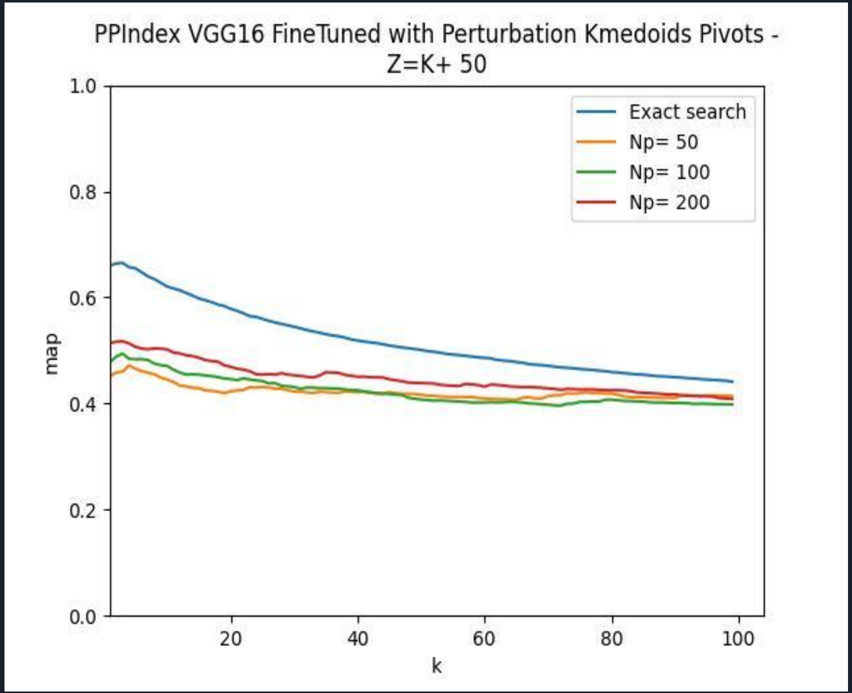
Pipeline


Features Extraction from pretrained CNN
Performance Analysis
Features Extraction from Fine Tuned CNN
Performance Analysis

Performance Analysis Fine Tuning

HyperParameter:

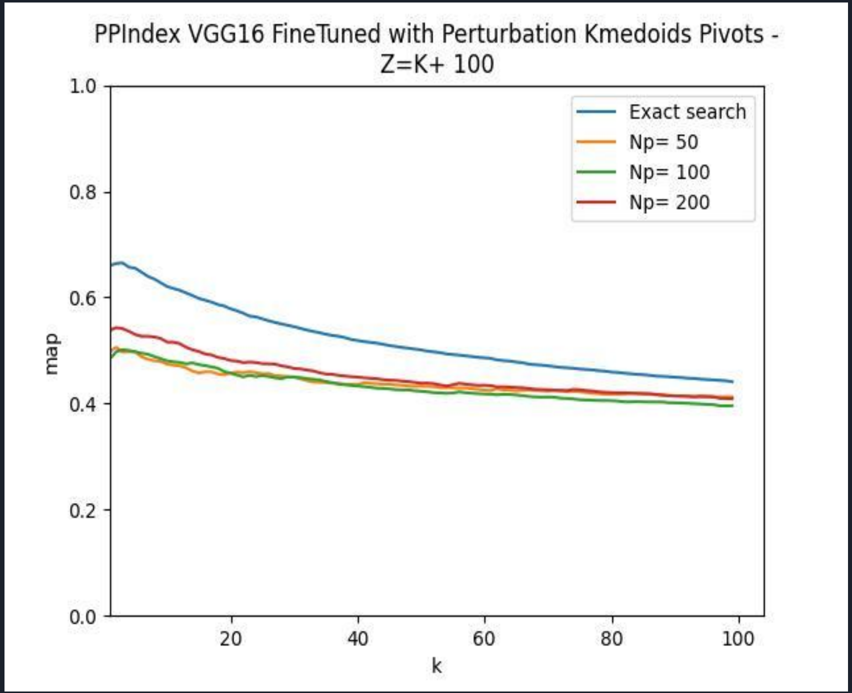
- Pivot Selection Method: Kmedoids
- Number of Pivots : {50, 100, 200}
- Z = K + {50, 100, 200}



Performance Analysis Fine Tuning

HyperParameter:

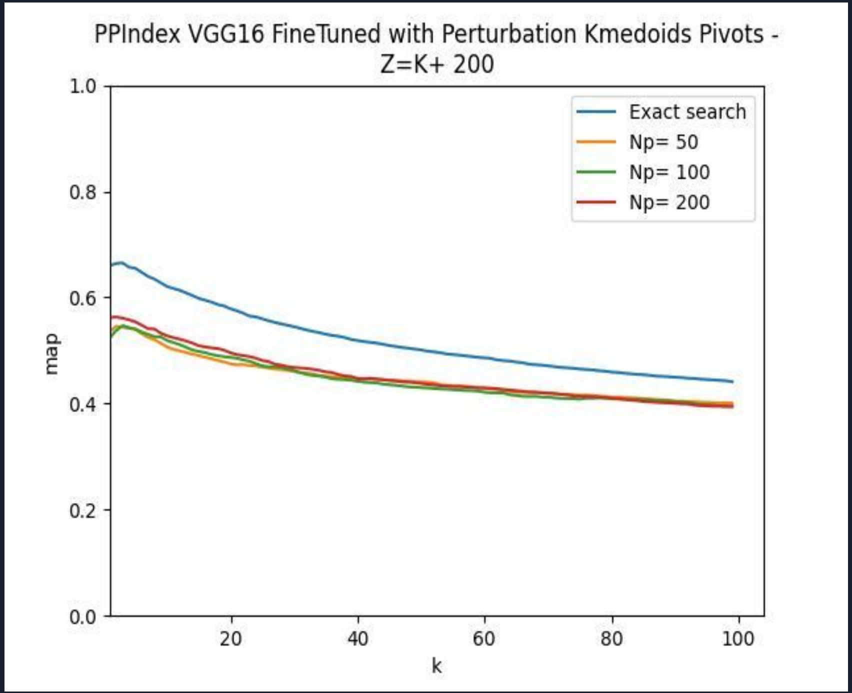
- Pivot Selection Method: Kmedoids
- Number of Pivots : {50, 100, 200}
- Z = K + {50, 100, 200}



Performance Analysis Fine Tuning

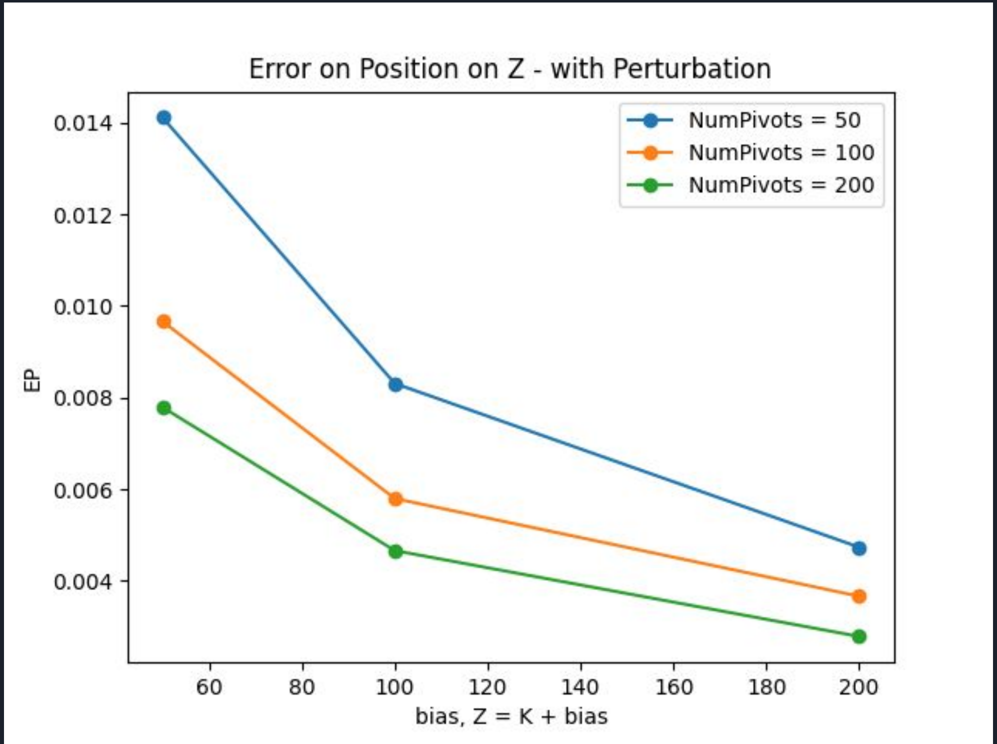
HyperParameter:

- Pivot Selection Method: Kmedoids
- Number of Pivots : {50, 100, 200}
- Z = K + {50, 100, 200}



Performance Analysis Fine Tuning - EP

Hyperparameters:
- Selection Method: Kmedoids
- L = 3
- Fine Tuned Features



The End

References

- PP-Index: Using Permutation Prefixes for Efficient and Scalable Similarity Search - Andrea Esuli - Istituto di Scienza e Tecnologie dell'Informazione - CNR
- https://keras.io/api/applications/
- Fieller EC: The biological standardization of Insulin. Suppl to J R Statist Soc 1940, 7:1-64.
- https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
- https://paperswithcode.com/dataset/sketch
- Very Deep Convolutional Networks for Large-Scale Image Recognition - Karen Simonyan, Andrew Zisserman
- The many Facets of Approximate Similarity Search - Marco Patella - Paolo Ciaccia