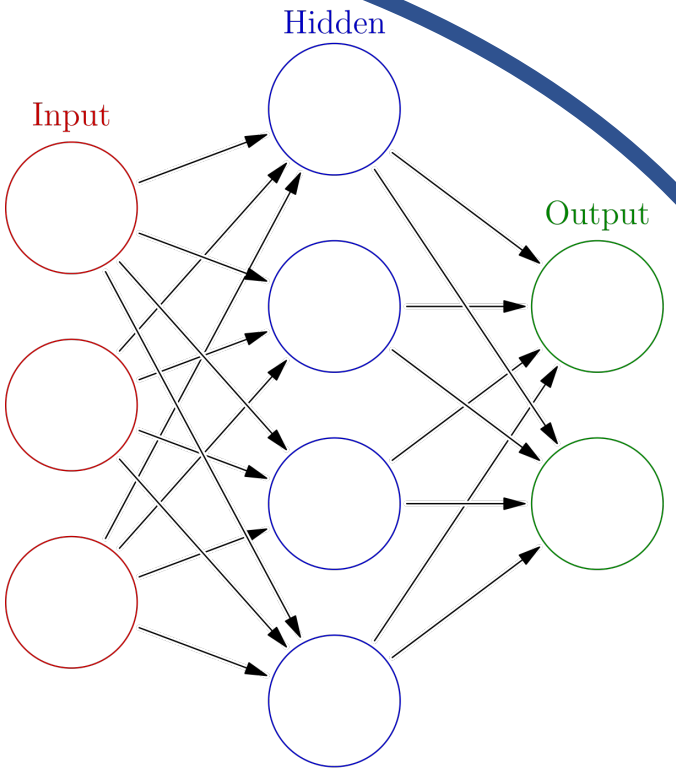
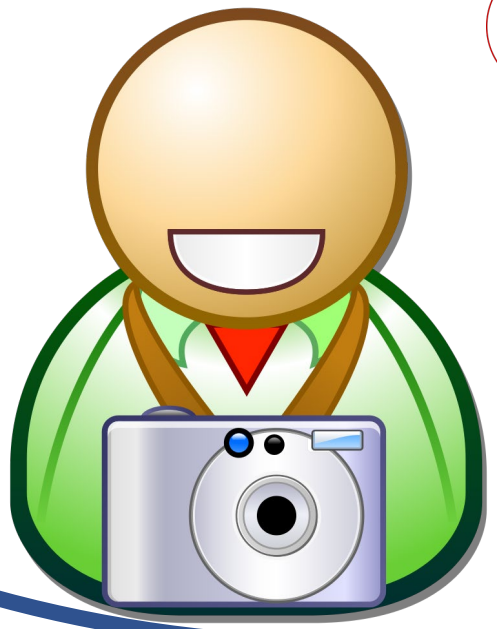

Hello there

An idea

Google Street View

+





= LOCATION

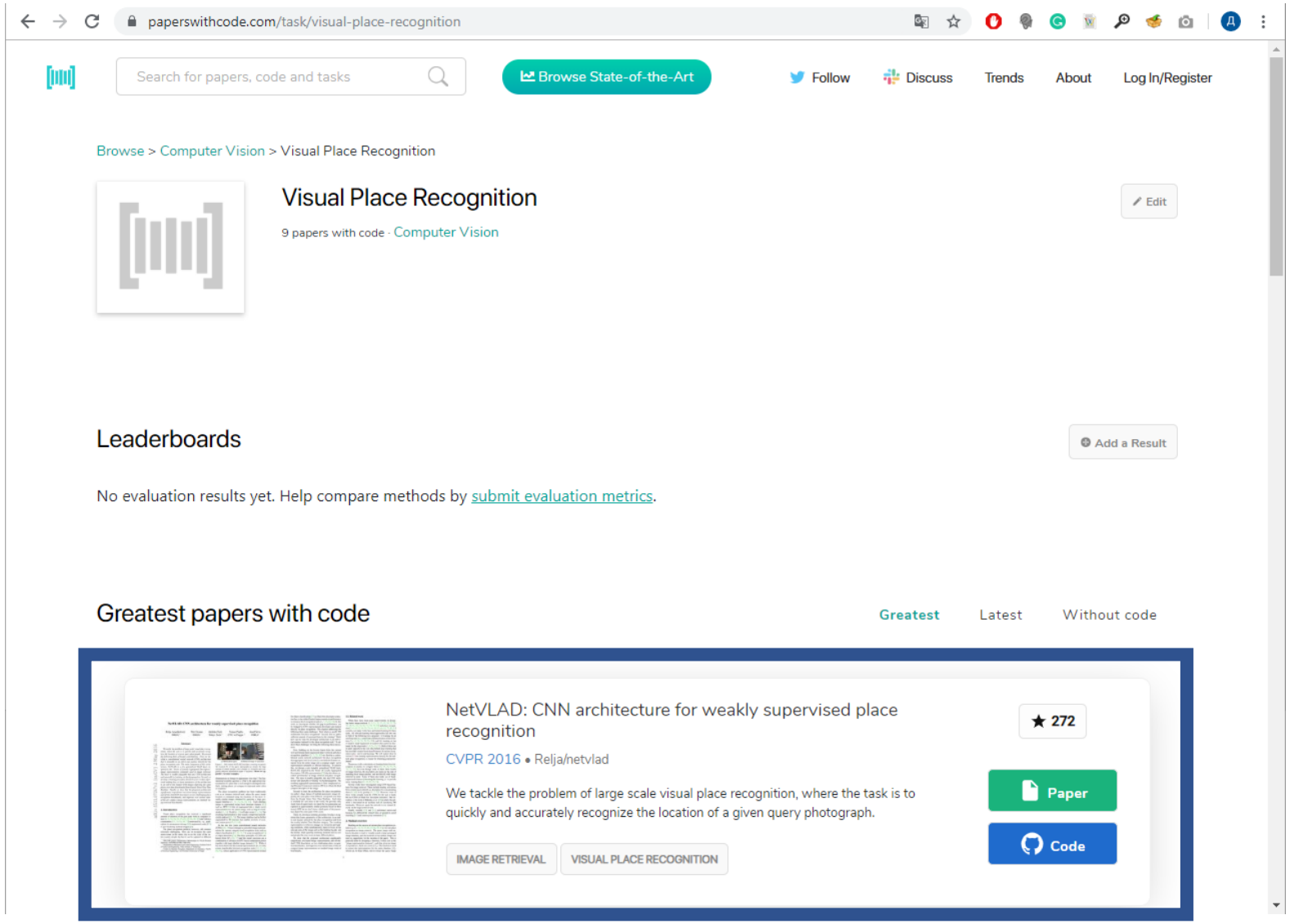
Possible ways of use

- Detecting interesting places for traveler, e. g. train passengers can use their phones to seek attractions while they are traveling
- Offline navigation
- Etc.

Steps of solving

paperswithcode.com/task/visual-place-recognition

bit.ly/2YmNoXs



Code

Edit

Relja/netvlad	★ 273	
lyakaap/NetVLAD-pytorch	★ 148	PyTorch
Nanne/pytorch-NetVlad	★ 49	PyTorch
keetsky/Net_ghostVLAD-pytorch	★ 4	PyTorch
chengricky/PanoramicScenePlaceRecognition	★ 3	PyTorch

 **lyakaap** Update README.md — Latest commit `7c1f62f` on 3 Nov 2018

LICENSE	Initial commit	2 years ago
README.md	Update README.md	last year
hard_triplet_loss.py	Rename hard_triplet_loss to hard_triplet_loss.py	2 years ago
netvlad.py	Add netvlad	2 years ago

README.md

NetVLAD-pytorch

Pytorch implementation of NetVLAD & Online Hardest Triplet Loss. In NetVLAD, broadcasting is used to calculate residuals of clusters and it makes whole calculation time much faster.

NetVLAD: https://arxiv.org/abs/1511.07247

In Defense of the Triplet Loss for Person Re-Identification: https://arxiv.org/abs/1703.07737
https://omoindrot.github.io/triplet-loss

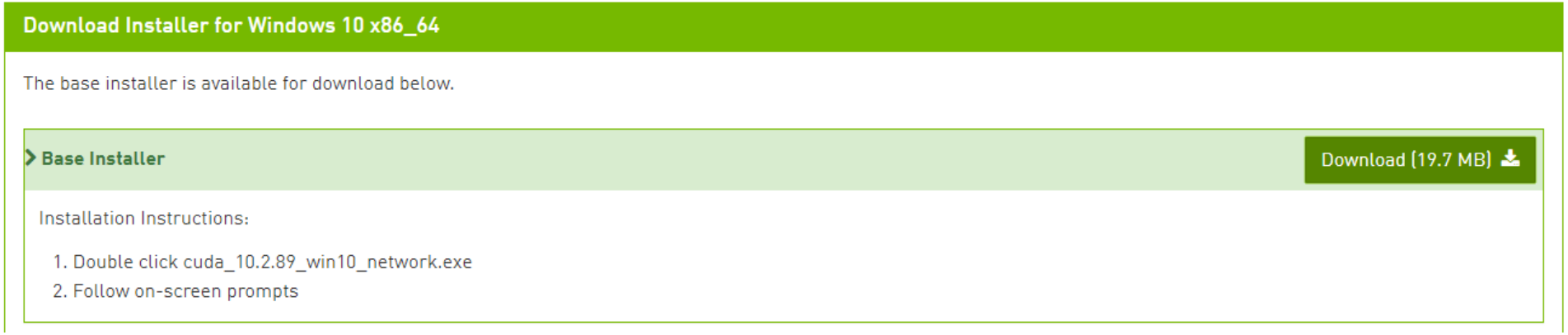
Usage

```
import torch
import torch.nn as nn
from torch.autograd import Variable
```

pytorch.org/get-started/locally/ [bit.ly/2Rup7gO](bit.ly/2Rup7gO)

PyTorch Build	**Stable (1.3)**	Preview (Nightly)		
Your OS	Linux	Mac	**Windows**	
Package	Conda	**Pip**	LibTorch	Source
Language	Python 2.7	Python 3.5	Python 3.6	**Python 3.7**	C++
CUDA	9.2	**10.1**	None	
Run this Command:	`pip3 install torch===1.3.1 torchvision===0.4.2 -f https://download.pytorch.org/whl/torch_stable.html`			

developer.nvidia.com/cuda-downloads bit.ly/2Rup7gO



Fix pytorch bugs

```
D:\Programs\Python37\python.exe C:/Users/Dmitry/Desktop/NetVLAD-pytorch-master/main.py
Traceback (most recent call last):
  File "C:/Users/Dmitry/Desktop/NetVLAD-pytorch-master/main.py", line 37, in <module>
    triplet_loss = criterion(output, labels)
  File "D:\Programs\Python37\lib\site-packages\torch\nn\modules\module.py", line 541, in __call__
    result = self.forward(*input, **kwargs)
  File "C:\Users\Dmitry\Desktop\NetVLAD-pytorch-master\hard_triplet_loss.py", line 62, in forward
    mask = _get_triplet_mask(labels).float()
  File "C:\Users\Dmitry\Desktop\NetVLAD-pytorch-master\hard_triplet_loss.py", line 141, in _get_triplet_mask
    mask = distinct_indices * valid_labels   # Combine the two masks
RuntimeError: expected device cuda:0 but got device cpu

Process finished with exit code 1
```

Fix pytorch bugs

NetVLAD (Vector of Locally Aggregated Descriptors)

(a) Mobile phone query

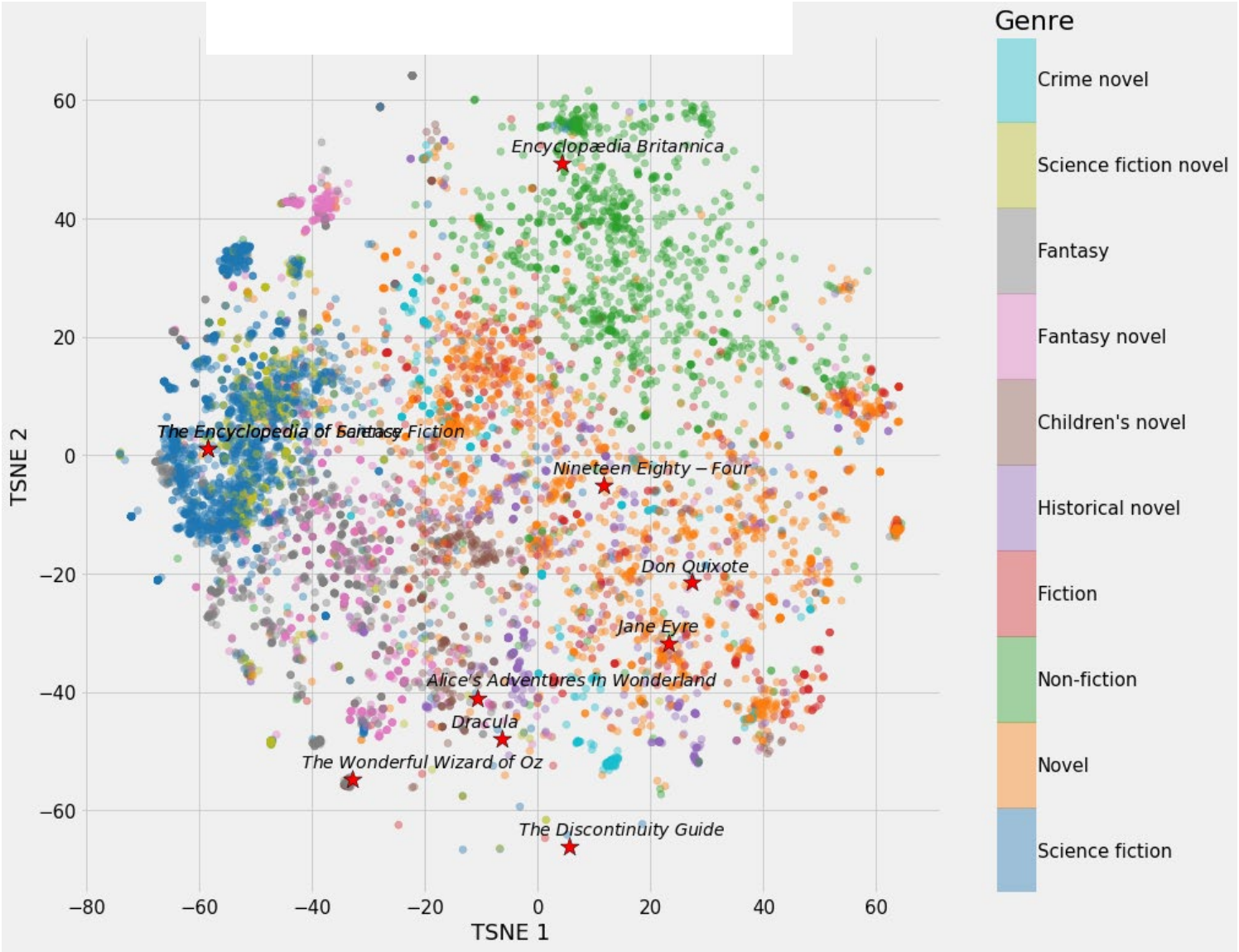
(b) Retrieved image of same place

Embedding (Вкладення)

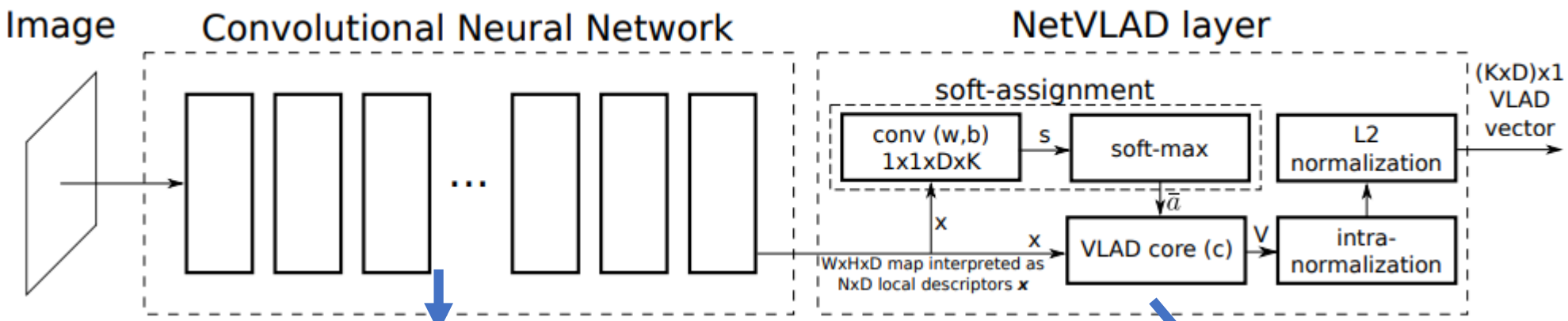
An embedding is a mapping of a discrete — categorical — variable to a vector of continuous numbers. In the context of neural networks, embeddings are *low-dimensional, learned* continuous vector representations of discrete variables. Neural network embeddings are useful because they can *reduce the dimensionality* of categorical variables and *meaningfully represent* categories in the transformed space.

Neural network embeddings have 3 primary purposes:

- Finding nearest neighbors in the embedding space. These can be used to make recommendations based on user interests or cluster categories.
- As input to a machine learning model for a supervised task.
- For visualization of concepts and relations between categories.



Convolutional Neural Network architecture with the NetVLAD layer



Vector of Locally Aggregated Descriptors (Embedding)

```
encoder = resnet18(pretrained=True)
base_model = nn.Sequential(
    encoder.conv1,
    encoder.bn1,
    encoder.relu,
    encoder.maxpool,
    encoder.layer1,
    encoder.layer2,
    encoder.layer3,
    encoder.layer4,
)
```

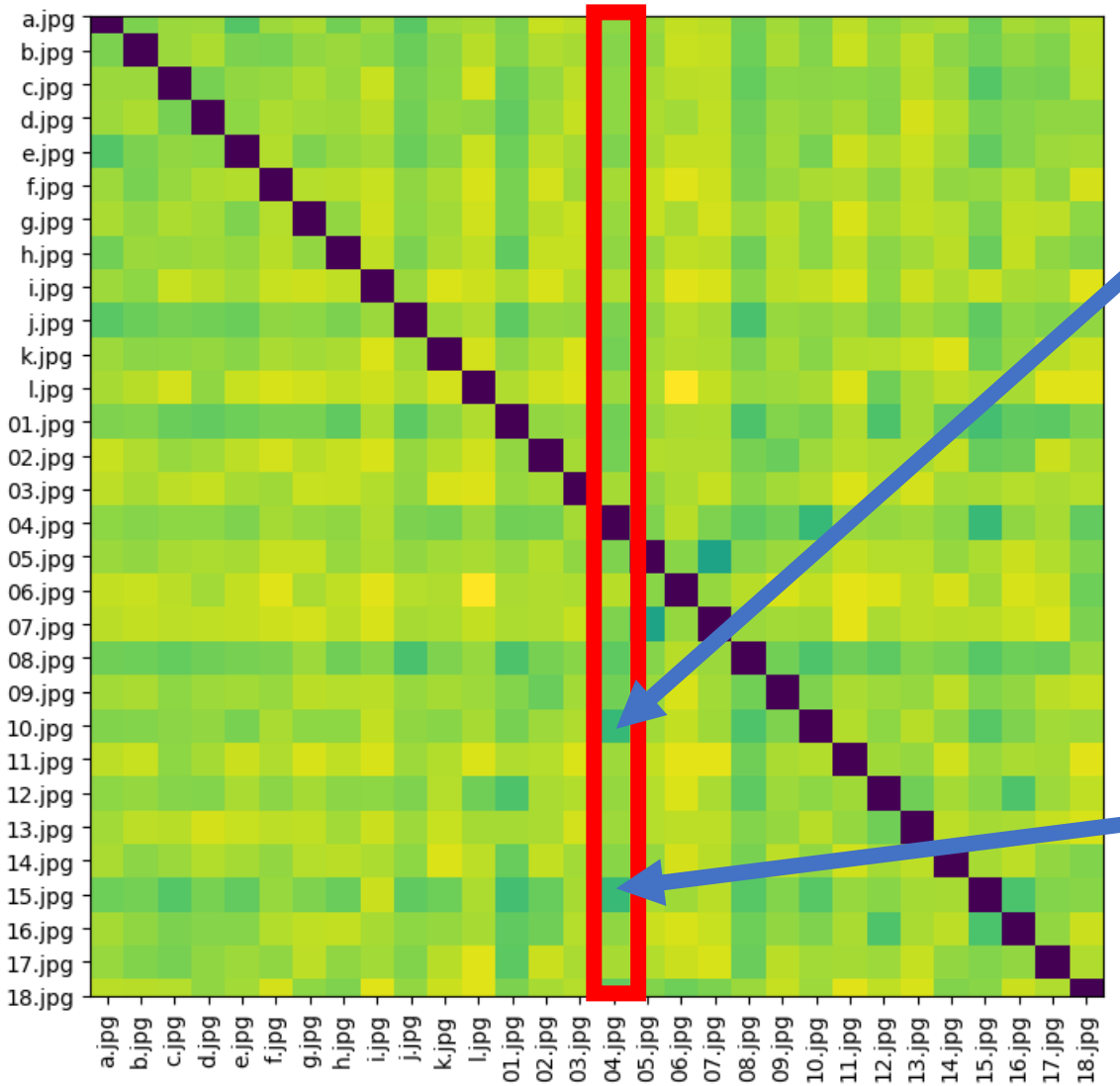
```
# Define model for embedding
net_vlad = NetVLAD(num_clusters=32, dim=dim, alpha=1.0)
# noinspection PyUnresolvedReferences
model = EmbedNet(base_model, net_vlad).cuda()
```

Simple demo

Distinguish by a photo whether you at CSC faculty building or KPI first building

Pair-wise “distance” between each two images

04.jpg - Original image



10.jpg - Compare image

15.jpg - Compare image

What do we have by the end?

- We can check if some photo corresponds some place and restore position

Why this approach is better then other

1. It is scalable
2. (very much scalable)