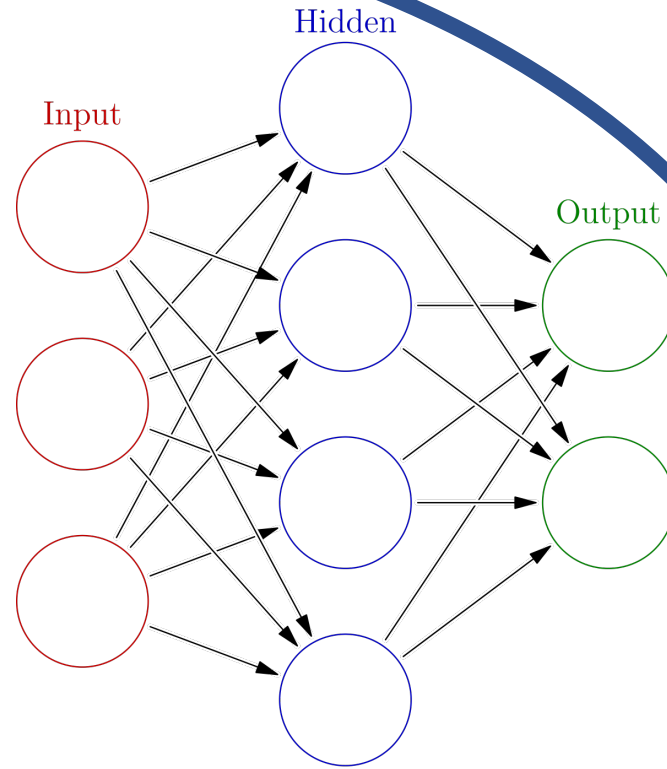
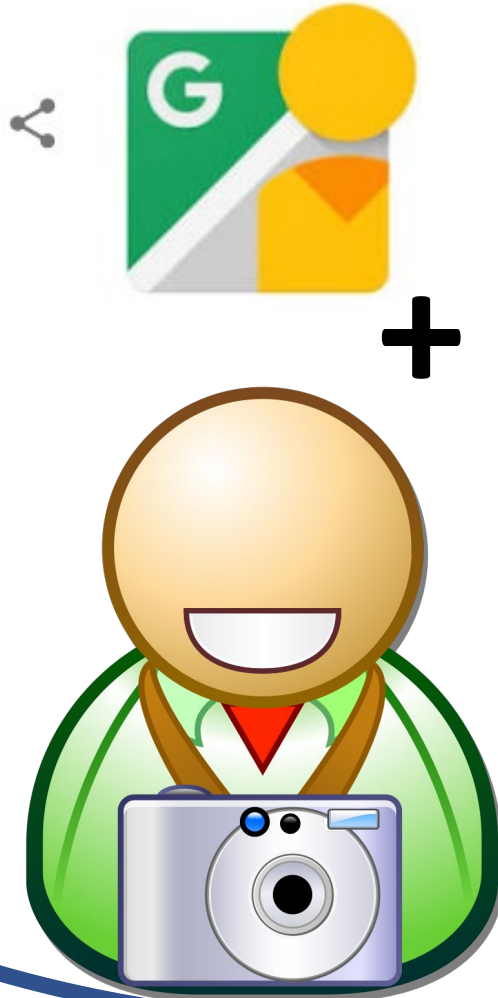




An idea

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

A screenshot of the 'paperswithcode.com' website, specifically the 'Visual Place Recognition' task page. The browser's address bar shows the URL 'paperswithcode.com/task/visual-place-recognition'. The page features a search bar, navigation links like 'Follow', 'Discuss', 'Trends', 'About', and 'Log In/Register', and a 'Browse State-of-the-Art' button. The main content area is titled 'Visual Place Recognition' and indicates '9 papers with code · Computer Vision'. Below this, there is a 'Leaderboards' section with a note that 'No evaluation results yet' and a link to 'submit evaluation metrics'. The 'Greatest papers with code' section is active, displaying a list of papers. The first paper, 'NetVLAD: CNN architecture for weakly supervised place recognition' by Relja/netvlad, is highlighted with a blue border. It has 272 stars and includes buttons for 'Paper' and 'Code'. The paper's abstract is visible, describing the task of large-scale visual place recognition. At the bottom of the highlighted paper entry, there are tags for 'IMAGE RETRIEVAL' and 'VISUAL PLACE RECOGNITION'.

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

 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

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:	<pre>pip3 install torch==1.3.1 torchvision==0.4.2 -f https://download.pytorch.org/whl/torch_stable.html</pre>				



developer.nvidia.com/cuda-downloads

bit.ly/2Rup7gO

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System

Windows

Linux

Mac OSX

Architecture

x86_64

Version

10

8.1

7

Server 2019

Server 2016

Server 2012 R2

Installer Type


exe (network)

exe (local)

Download Installer for Windows 10 x86_64

The base installer is available for download below.

> Base Installer

Download (19.7 MB) 

Installation Instructions:

1. Double click cuda_10.2.89_win10_network.exe
2. Follow on-screen prompts

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

```
main.py x hard_triplet_loss.py x _init_.py x
1109 def _get_anchor_negative_triplet_mask(labels):
1110     # Return a 2D mask where mask[a, n] is True iff a and n have distinct
1111     # Check if labels[i] != labels[k]
1112     labels_equal = torch.unsqueeze(labels, 0) == torch.unsqueeze(labels, 1)
1113     mask = labels_equal ^ 1
1114     return mask
1115
1116 def _get_triplet_mask(labels):
1117     """Return a 3D mask where mask[a, p, n] is True iff the triplet (a, p, n)
1118     A triplet (i, j, k) is valid if:
1119     - i, j, k are distinct
1120     - labels[i] == labels[j] and labels[i] != labels[k]
1121     """
1122     device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
1123     # Check that i, j and k are distinct
1124     indices_not_same = torch.eye(labels.shape[0]).to(device).byte() ^ 1
1125     i_not_equal_j = torch.unsqueeze(indices_not_same, 2)
1126     i_not_equal_k = torch.unsqueeze(indices_not_same, 1)
1127     j_not_equal_k = torch.unsqueeze(indices_not_same, 0)
1128     distinct_indices = i_not_equal_j * i_not_equal_k * j_not_equal_k
1129
1130     # Check if labels[i] == labels[j] and labels[i] != labels[k]
1131     label_equal = torch.eq(torch.unsqueeze(labels, 0), torch.unsqueeze(labels, 1)).to(device)
1132     i_equal_j = torch.unsqueeze(label_equal, 2)
1133     i_equal_k = torch.unsqueeze(label_equal, 1)
1134     valid_labels = i_equal_j * (i_equal_k ^ 1)
1135
1136     mask = distinct_indices * valid_labels # Combine the two masks
1137     return mask
1138
1139
```

NetVLAD (Vector of Locally Aggregated Descriptors)



(a) Mobile phone query



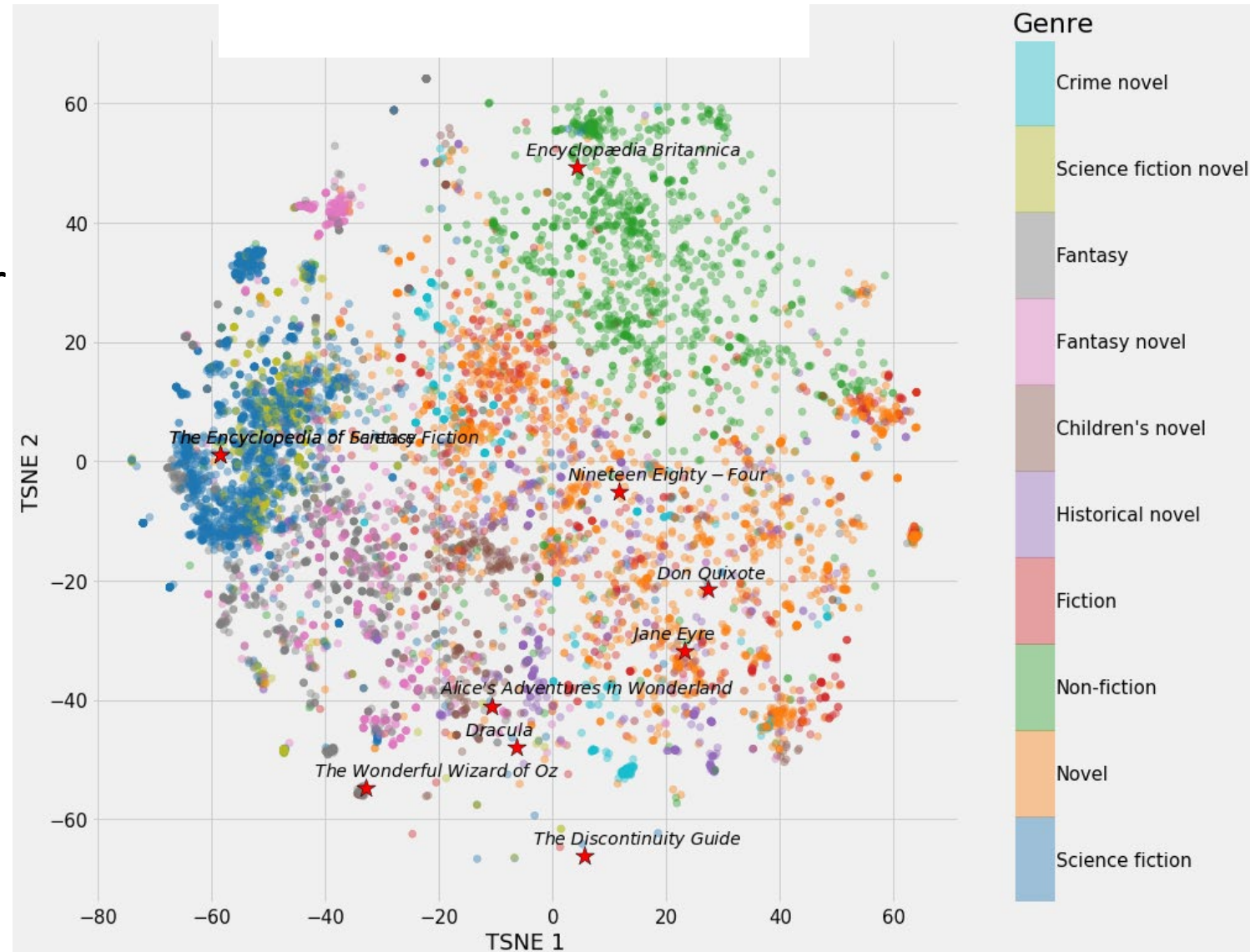
(b) Retrieved image of same place

Embedding (Вкладення)

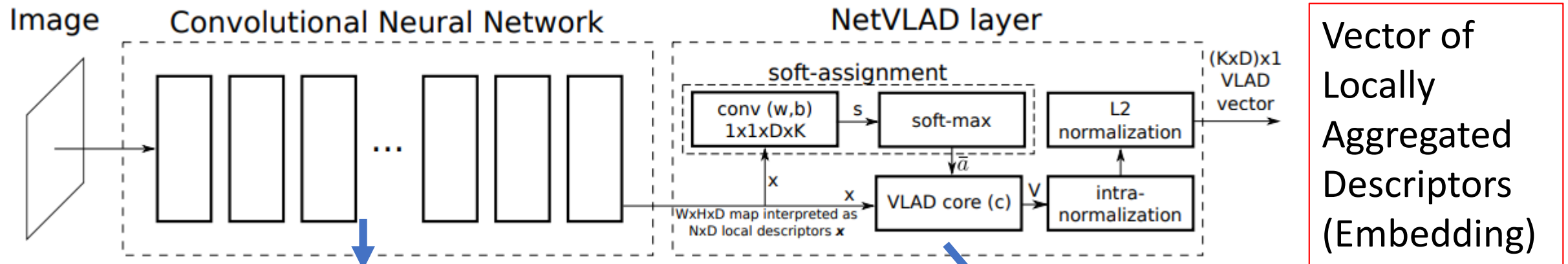
An embedding is a mapping of a discrete — categorical — variable to a vector of continuous numbers. In the context of neural networks, embeddings are *low-dimensional, learned* continuous vector representations of discrete variables. Neural network embeddings are useful because they can *reduce the dimensionality*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



```
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

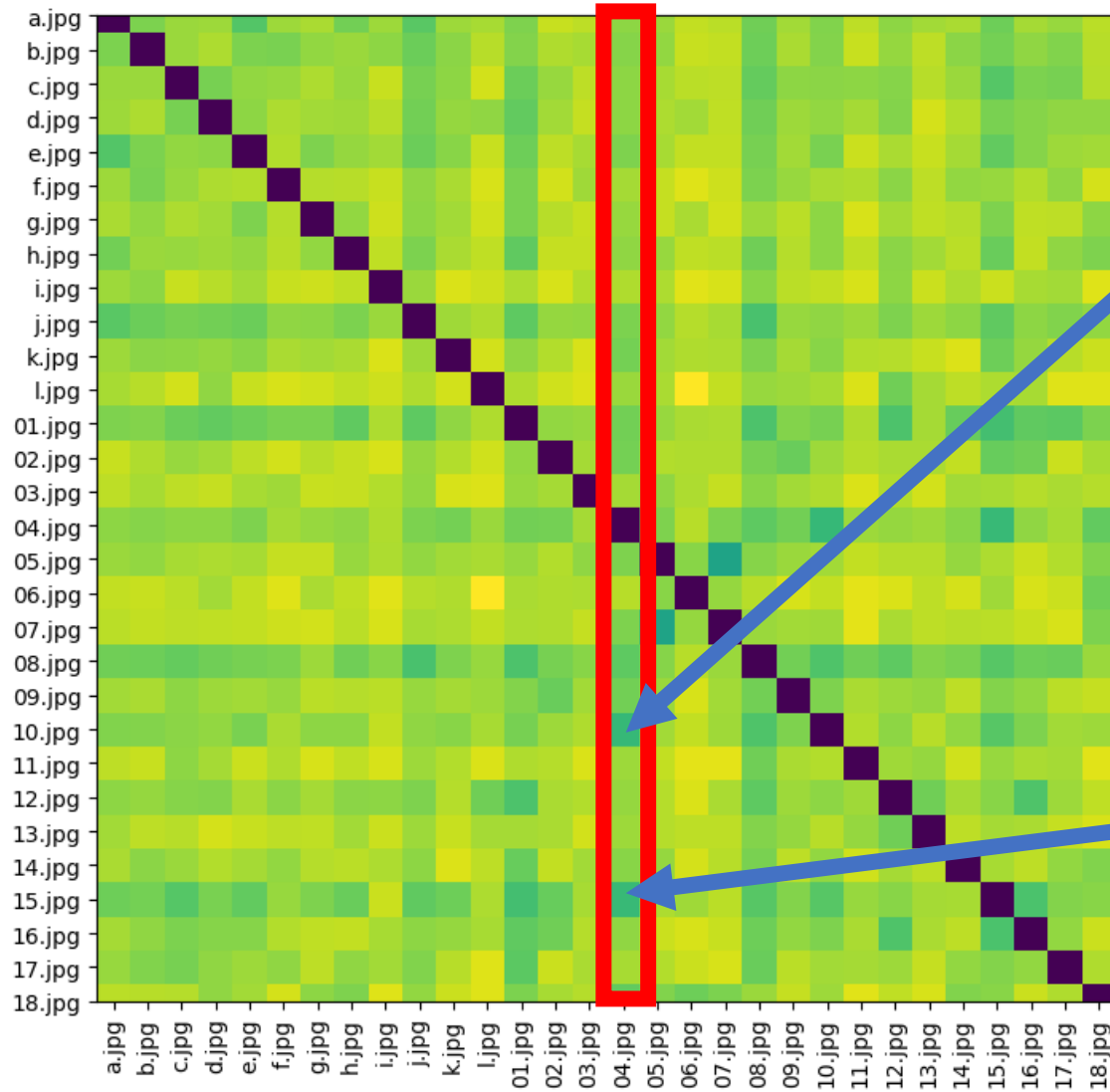
Distinguish by a photo whether you at CSC faculty building or KPI first building



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 than other

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