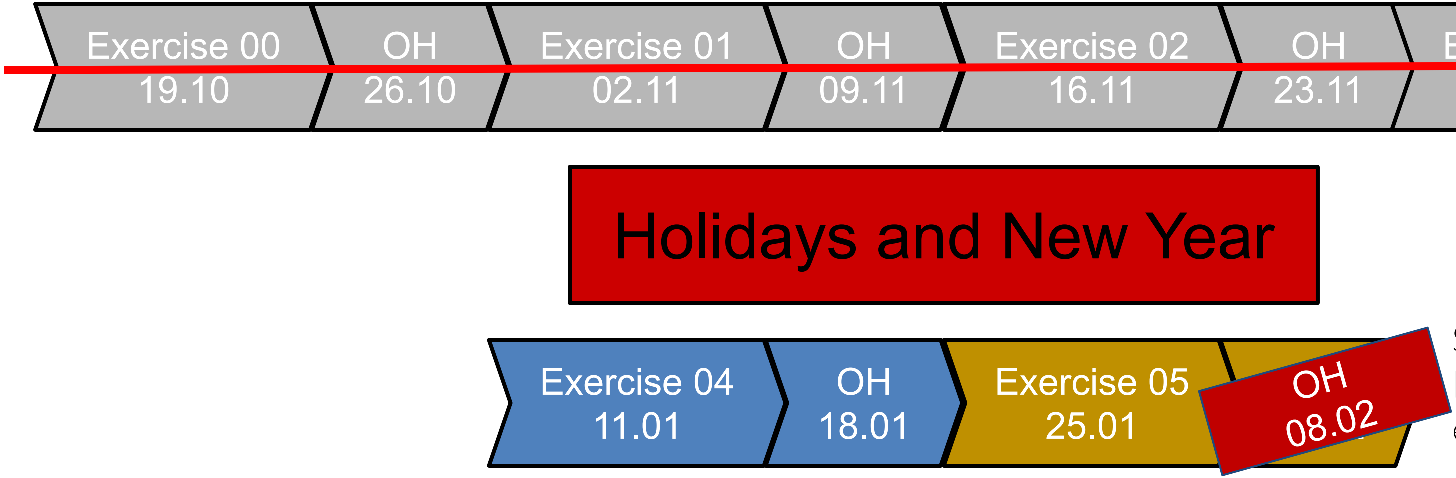
Exercise 4

About the Exercise Session

- 2 weeks for each exercise + Office hours (OH) for questions in between

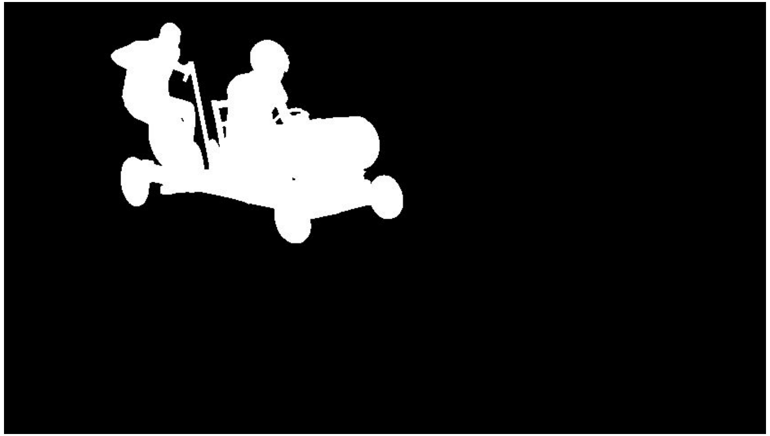


Submission for Exercise 05 also extended by 1 week

Deadline always 23:59 CET on due date

Recap: Segmentation

Going from bounding boxes to per-pixel predictions



Recap: Segmentation

Going from bounding boxes to per-pixel predictions

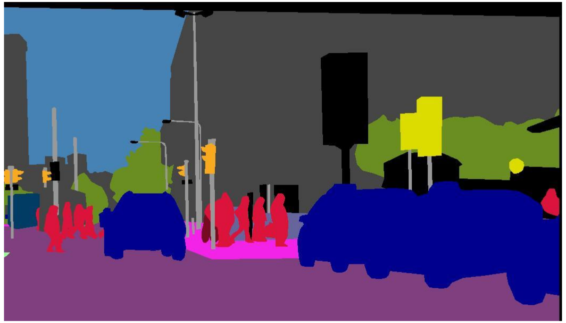
Recap: Segmentation

What is shown here?

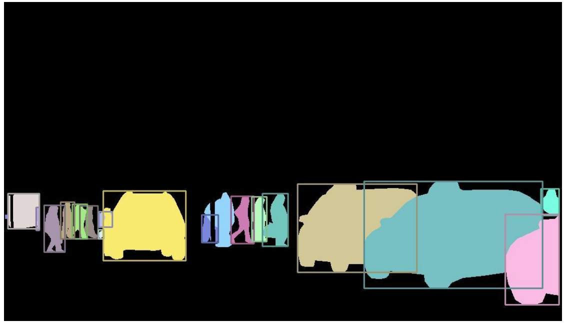
Instance, Semantic, or Panoptic Segmentation?

Recap: Segmentation

semantic segmentation



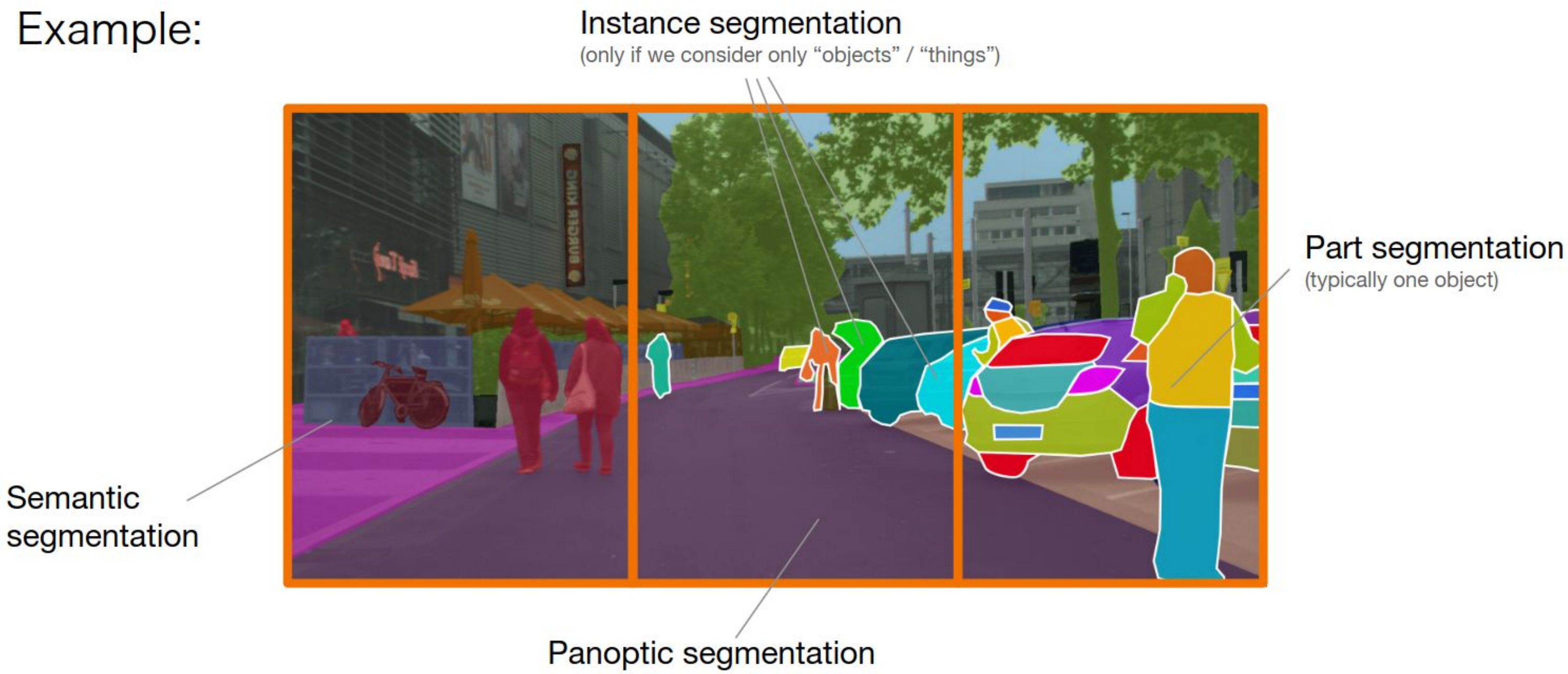
instance segmentation



panoptic segmentation

Recap Segmentation

Example:



Exercise 4

- **Object Segmentation (Semantic segmentation with binary class)**
- Supervision available
- Pretrained powerful image embeddings (learned with self-supervision)
- Upsampling: Learn to take advantage of pixel-adaptive convolutional neural nets

Exercise 4: Object Segmentation

Image

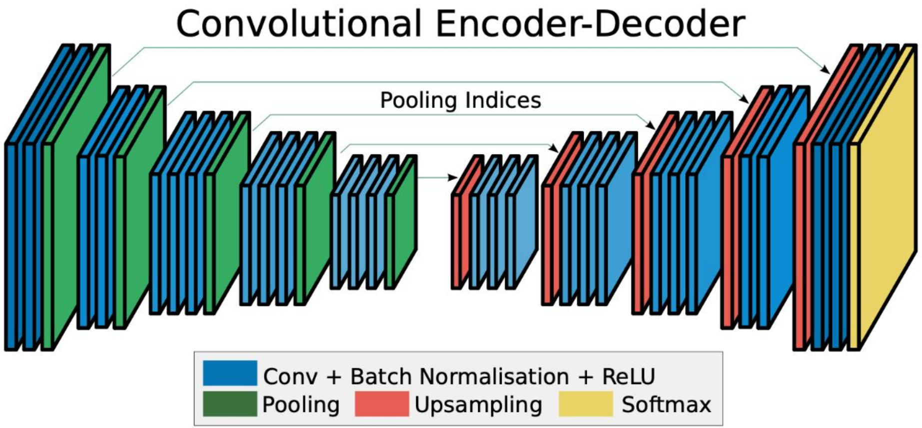
With Annotations

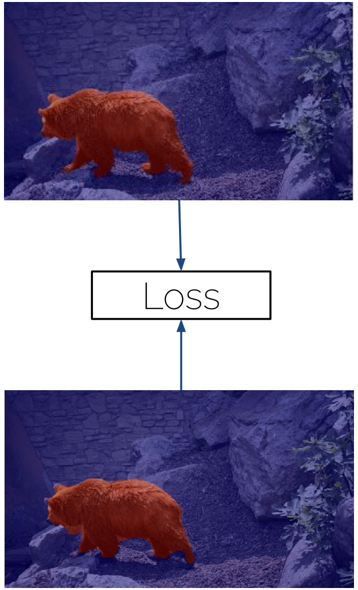
Exercise 4

- Object Segmentation (Semantic segmentation with binary class)
- **Supervision available**
- Pretrained powerful image embeddings (learned with self-supervision)
- Upsampling: Learning about pixel-adaptive convolutional neural nets

Classical Supervised Method

W,H,3





W,H,1

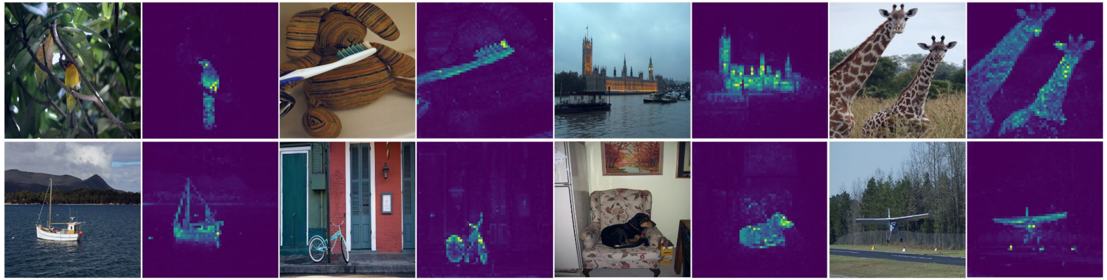
Badrinarayanan et al. “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation”

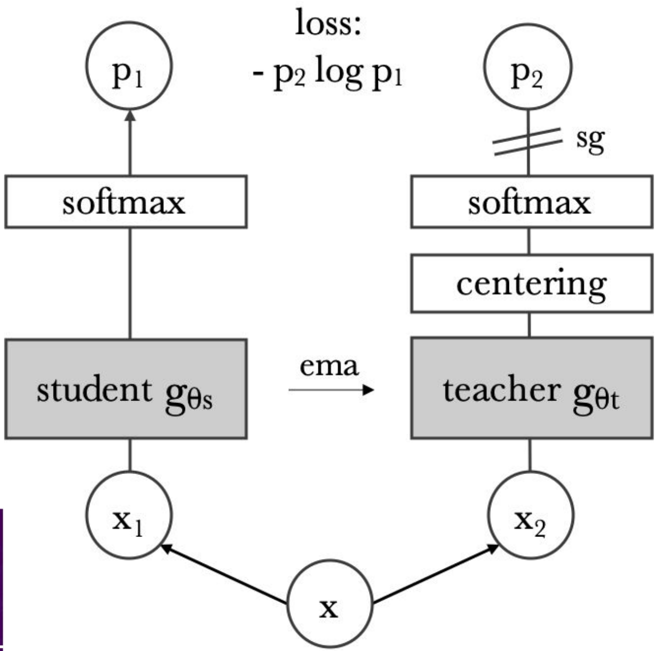
Exercise 4

- Object Segmentation (Semantic segmentation with binary class)
- Supervision available
- **Pretrained powerful image embeddings (learned with self-supervision)**
- Upsampling: Learn to take advantage of pixel-adaptive convolutional neural nets

DINO - Self-Distillation with no Labels

- Vision Transformer trained in a self-supervised fashion
- Student-teacher approach to train the network
- Results in powerful image representations





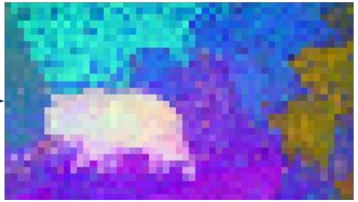
Caron et al. “Emerging Properties in Self-Supervised Vision Transformers”

DINO: Self Supervised Learning

W,H,3

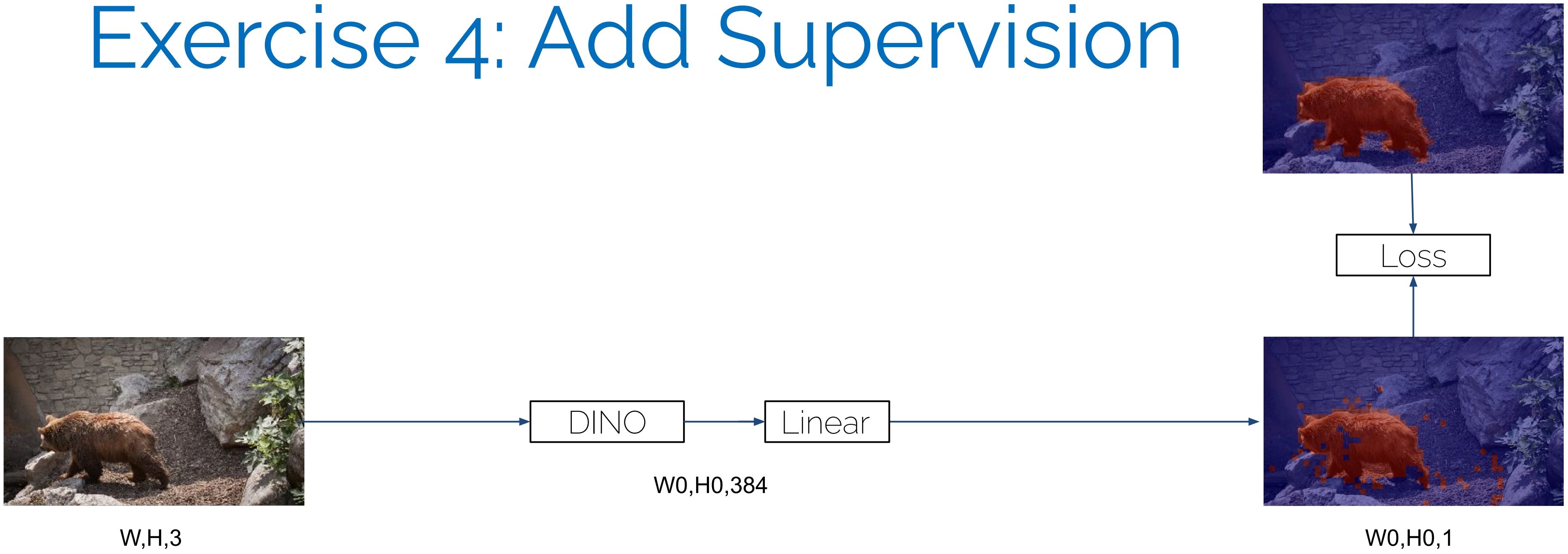
DINO → Linear (PCA)

W0,H0,384



W0,H0,3

Exercise 4: Add Supervision



Exercise 4: Using DINO Feats

Loss

W,H,3

DINO

W0,H0,384

Linear

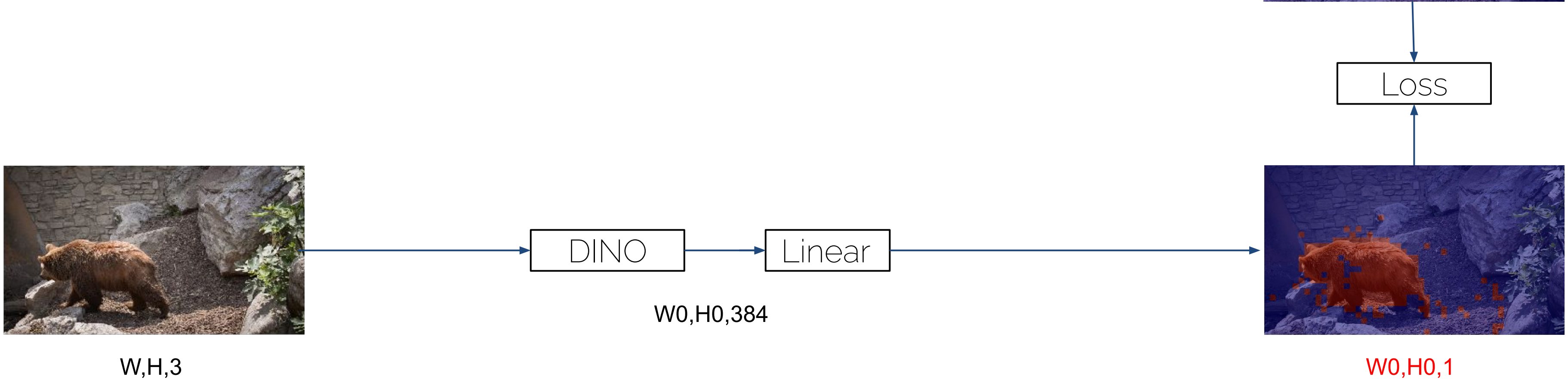
W0,H0,1

We only have to learn a small linear Layer

Exercise 4

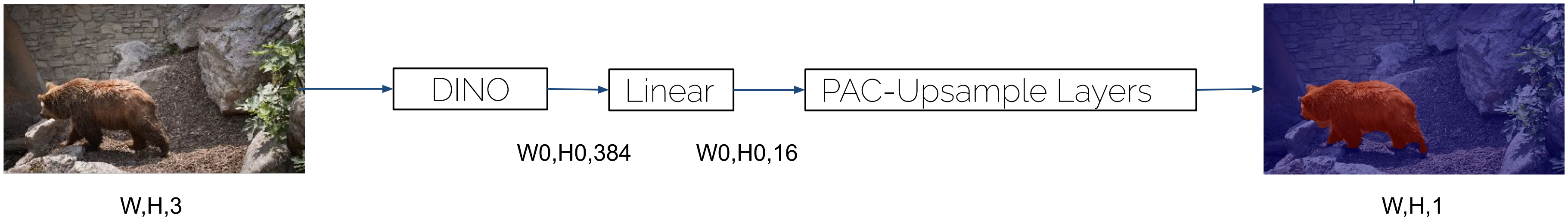
- Object Segmentation (Semantic segmentation with binary class)
- Supervision available
- Pretrained powerful image embeddings (learned with self-supervision)
- **Upsampling: Learn to take advantage of pixel-adaptive convolutional neural nets**

Exercise 4: PAC-Upsample



Low Resultion (16x16) after DINO net

Exercise 4: PAC-Upsample



Recap: CNNs

-5	3	2	-5	3
4	3	2	1	-3
1	0	3	3	5
-2	0	1	4	4
5	6	7	9	-1

Image **5x5**

Output **3x3**

Weight **3x3**

2	4	-2
3	1	0
-1	0	0

Recap: CNNs

-5	3	2	-5	3
4	3	2	1	-3
1	0	3	3	5
-2	0	1	4	4
5	6	7	9	-1

Image **5x5**

12		

Output **3x3**

2	4	-2
3	1	0
-1	0	0

Weight **3x3**

Recap: CNNs

-5	3	2	-5	3
4	3	2	1	-3
1	0	3	3	5
-2	0	1	4	4
5	6	7	9	-1

Image **5x5**

Weight **3x3**

2	4	-2
3	1	0
-1	0	0

12	35	

Output **3x3**

Pixel-Adaptive CNNs

-5	3	2	-5	3
4	3	2	1	-3
1	0	3	3	5
-2	0	1	4	4
5	6	7	9	-1

Image **5x5**

6		

Output **3x3**

Weight **3x3**

2	4	-2
3	1	0
-1	0	0

✖

1	2	-1
-1	0	0
0	1	3

Pixel-Adaptive Kernel **3x3**

Pixel-Adaptive CNNs

-5	3	2	-5	3
4	3	2	1	-3
1	0	3	3	5
-2	0	1	4	4
5	0	7	9	-1

Image **5x5**

6	-1	

Output **3x3**

Weight **3x3**

2	4	-2
3	1	0
-1	0	0

✖

2	1	-2
0	-1	1
1	0	0

Pixel-Adaptive Kernel **3x3**

Calculating the (gaussian) Kernel

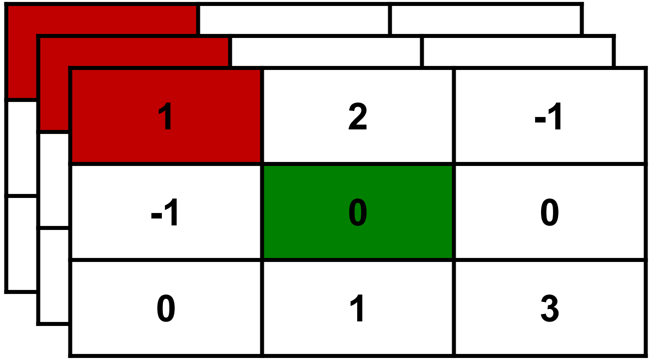
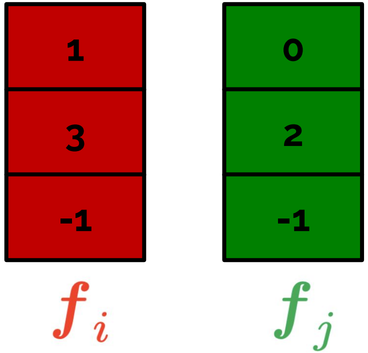


Image patch **3x3x3**

exp(-1)		

Kernel **3x3**



$$K(\boldsymbol{f}_i, \boldsymbol{f}_j) = \exp\left(-\frac{1}{2}(\boldsymbol{f}_i - \boldsymbol{f}_j)^\top(\boldsymbol{f}_i - \boldsymbol{f}_j)\right)$$

Calculating the (gaussian) Kernel

1	2	-1
-1	0	0
0	1	3

Image patch **3x3x3**

exp(-1)	exp(-3)	

Kernel **3x3**

$\boldsymbol{f}_i$	$\boldsymbol{f}_j$
2	0
1	2
-2	-1

$$K(\boldsymbol{f}_i, \boldsymbol{f}_j) = \exp\left(-\frac{1}{2}(\boldsymbol{f}_i - \boldsymbol{f}_j)^\top(\boldsymbol{f}_i - \boldsymbol{f}_j)\right)$$

Pytorch Unfold

-5	3	2	-5	3
4	3	2	1	-3
1	0	3	3	5
-2	0	1	4	4
5	6	7	9	-1

Image **5x5**

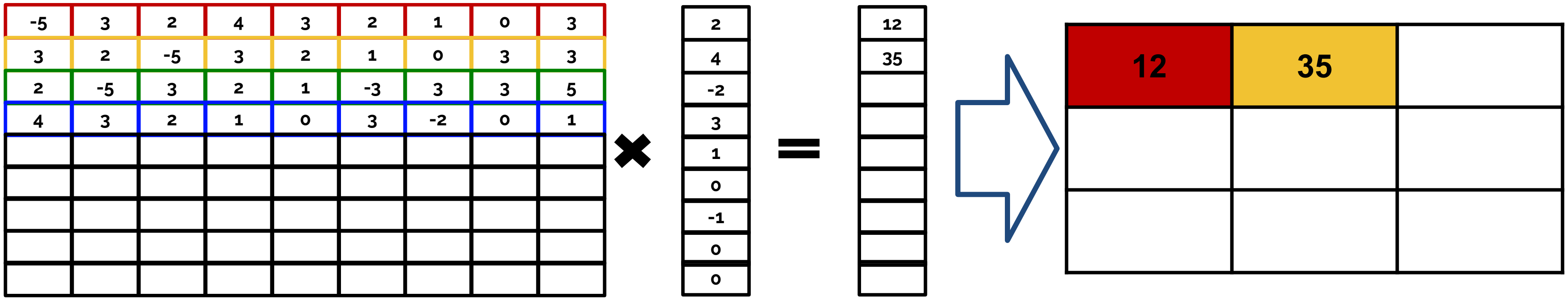
-5	3	2	4	3	2	1	0	3
3	2	-5	3	2	1	0	3	3
2	-5	3	2	1	-3	3	3	5
4	3	2	1	0	3	-2	0	1

Weight **3x3**

2	4	-2
3	1	0
-1	0	0

2	4	-2	3	1	0	-1	0	0

Pytorch Unfold



- unfolding allows to rewrite the convolution operation as a matrix vector multiplication
- reshaping afterward is necessary

Pytorch Unfold - weighting kernel

-5	3	2	4	3	2	1	0	3
3	2	-5	3	2	1	0	3	3
2	-5	3	2	1	-3	3	3	5
4	3	2	1	0	3	-2	0	1

✖ element-wise ✖

1	2	-1	-1	0	0	0	1	3
2	1	-2	0	-1	1	1	0	0

2
4
-2
3
1
0
-1
0
0

=

6
-1

6	-1	

Links

- Test server: https://cv3dst.cvai.cit.tum.de/login
- If you have trouble registering https://forms.gle/yZkZiDiyHxWuNqQG7
- Data for Exercise 04: https://vision.in.tum.de/webshare/g/cv3dst/exercise_04.zip

Links for the individual datasets

- MOT https://vision.in.tum.de/webshare/g/cv3dst/datasets/MOT16.zip
- market https://vision.in.tum.de/webshare/g/cv3dst/datasets/market.zip
- obj_seg https://vision.in.tum.de/webshare/g/cv3dst/datasets/obj_seg.zip
- reid_gnn https://vision.in.tum.de/webshare/g/cv3dst/datasets/reid_gnn.zip