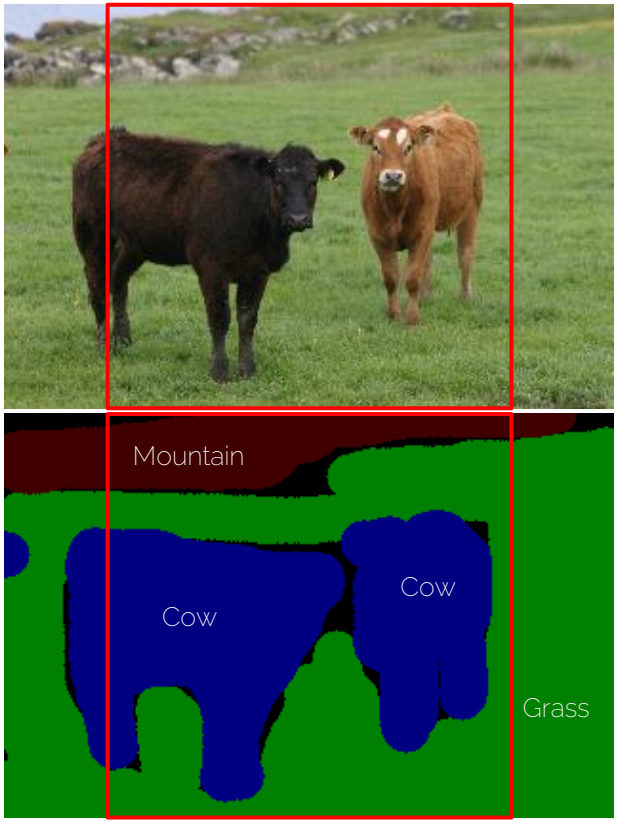
Introduction to Deep Learning (I2DL)

Exercise 10: Semantic Segmentation

Overview

- Semantic Segmentation
  - Loss Function
  - Architecture and Upsampling
- Submission: Semantic Segmentation
  - Deadline: January 12, 2022 15.59

Semantic Segmentation



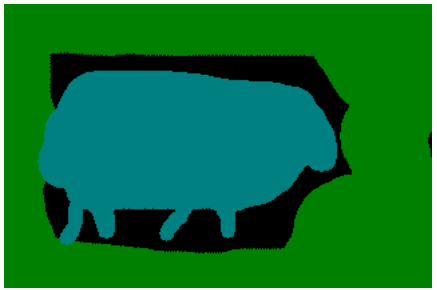
Input:
(3, w, h) RGB image

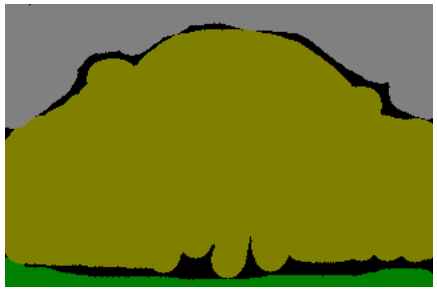
Output:
(23, w, h) segmentation map with scores for every class in every pixel

Semantic Segmentation Labels

object class	R	G	B	Colour
void	0	0	0	
building	128	0	0	
grass	0	128	0	
tree	128	128	0	
cow	0	0	128	
horse	128	0	128	
sheep	0	128	128	
sky	128	128	128	
mountain	64	0	0	

"void" for unlabelled pixels





Metrics: Loss Function

- Averaged per pixel cross-entropy loss

```
for (inputs, targets) in train_data[0:4]:
    inputs, targets = inputs, targets
    outputs = dummy_model(inputs.unsqueeze(0))
    loss = torch.nn.CrossEntropyLoss(ignore_index=-1, reduction='mean')
    losses = loss(outputs,targets.unsqueeze(0))
    print(losses)
```

- **ignore_index** (*int, optional*) – Specifies a target value that is ignored and does not contribute to the input gradient. When `size_average` is `True`, the loss is averaged over non-ignored targets.

Metrics: Accuracy

- Only consider pixels which are not „void“

```
def evaluate_model(model):
    test_scores = []
    model.eval()
    for inputs, targets in test_loader:
        inputs, targets = inputs.to(device), targets.to(device)

        outputs = model.forward(inputs)
        _, preds = torch.max(outputs, 1)
        targets_mask = targets >= 0
        test_scores.append(np.mean((preds == targets)[targets_mask].data.cpu().numpy()))


    return np.mean(test_scores)
print("Test accuracy: {:.3f}".format(evaluate_model(dummy_model)))
```

Semantic Segmentation Task

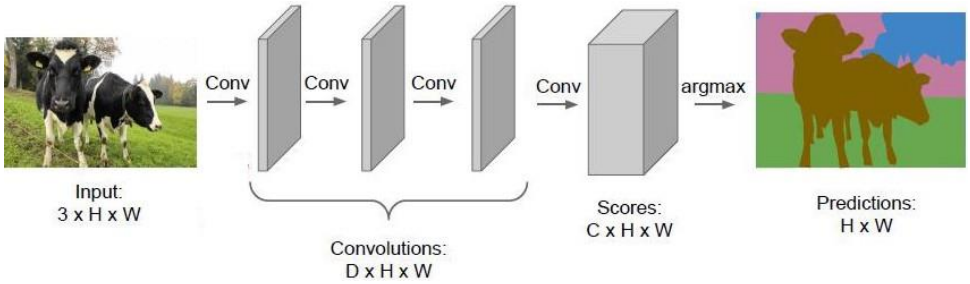
- Input shape: (N, num_channels, H, W)
  Output shape: (N, num_classed, H, W)
- We want to:
  - Maintain dimensionality (H, W)
  - Get features at different spatial resolutions

Edges | Textures | Parts | Objects

Image source: https://distill.pub/2017/feature-visualization/

Naive Solution

- Keep dimensionality constant throughout the network
- Use increasing filter sizes

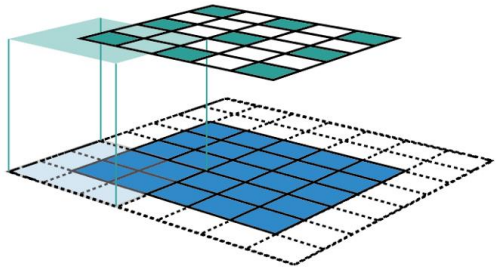


- Problem:
  - Increased memory consumption
    - Filter size would be the same
      e.g., 128 filters a (64x3x3) -> 73k params
    - But we have to save inputs and outputs for every layer
      e.g., 128 filters a (64xWxH) -> millions of params!

Image source: towardsdatascience.com – Divyanshu Mishra

Excursion: Receptive Field (RF)

- Region in input space a fea
  looking at
- E.g., after 2 (5x5) convolutions
  receptive field of 9x9
  (RF after first conv: 5
  RF after second conv: 5+4)



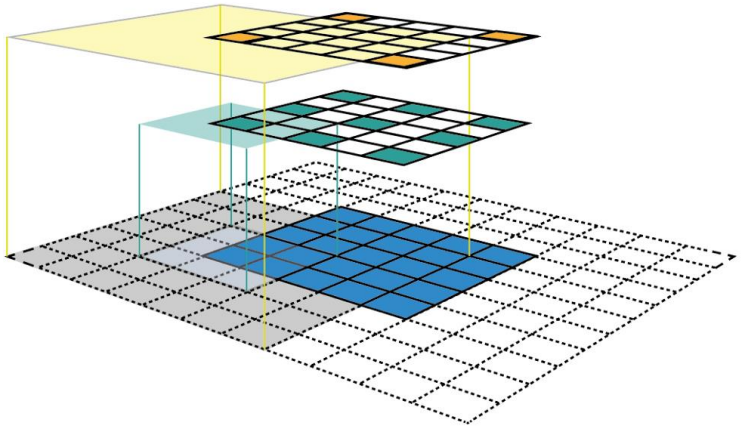
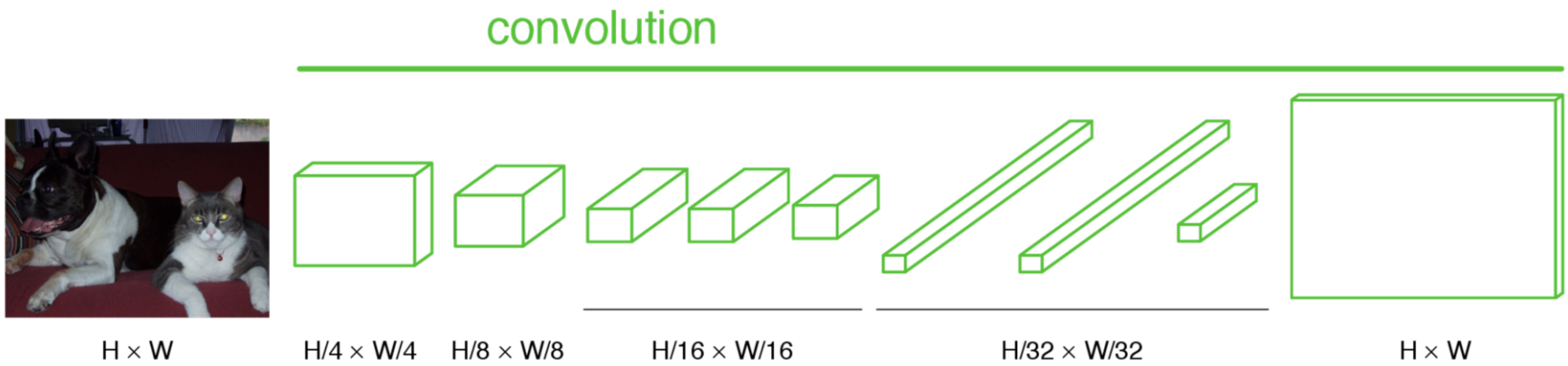


Image source: medium.com – Dang Ha The Hien

Coming from Classification

- Use strided convolutions and pooling to increase the receptive field
- Upsample result to input resolution



Better Solution

- Slowly reduce size -> slowly increase size
  - Pooling -> Upsampling
  - Strided convolution -> Transposed convolution
- Combine with normal convolutions, bn, dropout, etc.

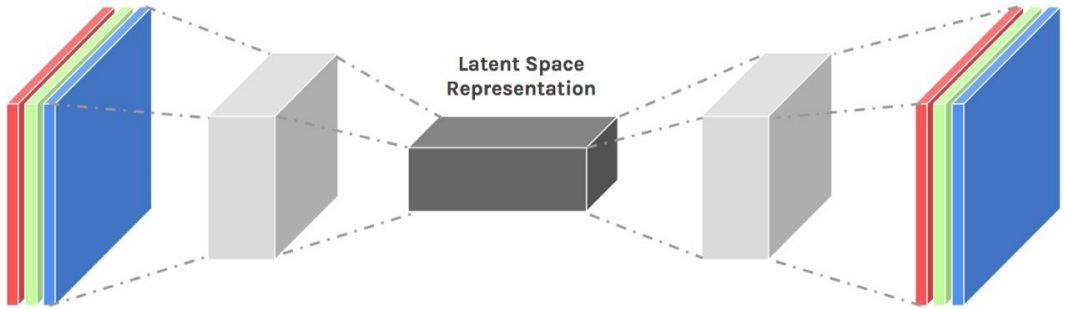
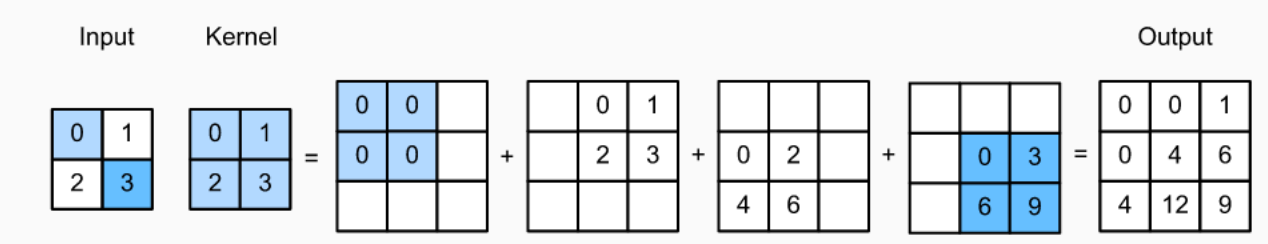


Image source: https://hackernoon.com/autoencoders-deep-learning-bits-1-11731e200694

Transposed Convolutions

- Upsampling with learnable parameters



- Output size computation:
  - Regular conv layer:

$$out = \frac{(in - kernel + 2 * pad)}{stride} + 1$$

  - Transpose convolution for multiples of 2

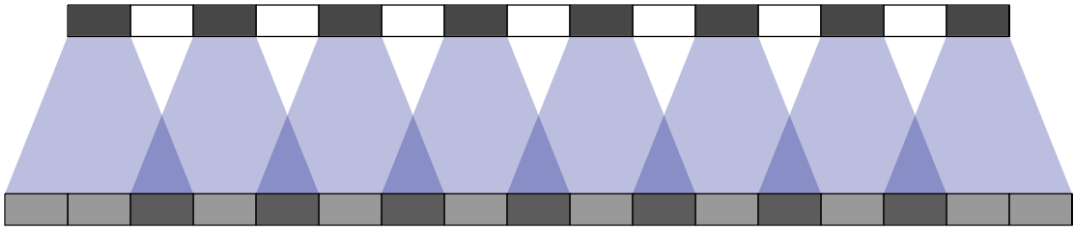
$$out = (in - 1) * stride - 2 * pad + kernel$$

(Transpose computation not relevant for the exam, more info here: https://github.com/vdumoulin/conv_arithmetic)

Image source: towardsdatascience.com – Divyanshu Mishra

Are transpose convolutions superior?

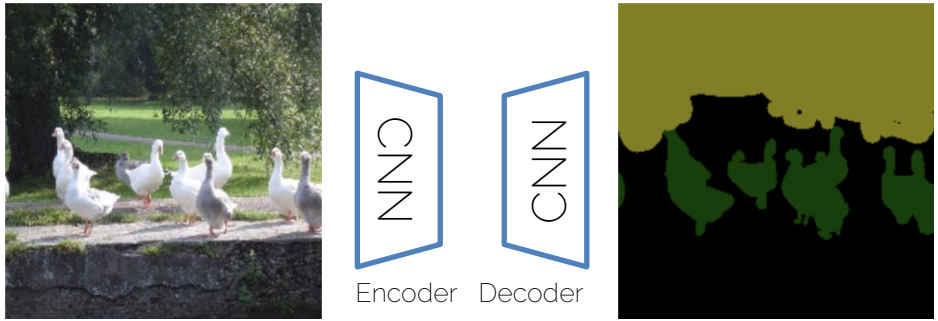
- Short answer: no, not always
- Long answer: possible checkerboard artifacts for general image generation, see https://distill.pub/2016/deconv-checkerboard/



- My personal go-to:
  - Regular upsampling, followed by a convolution layer

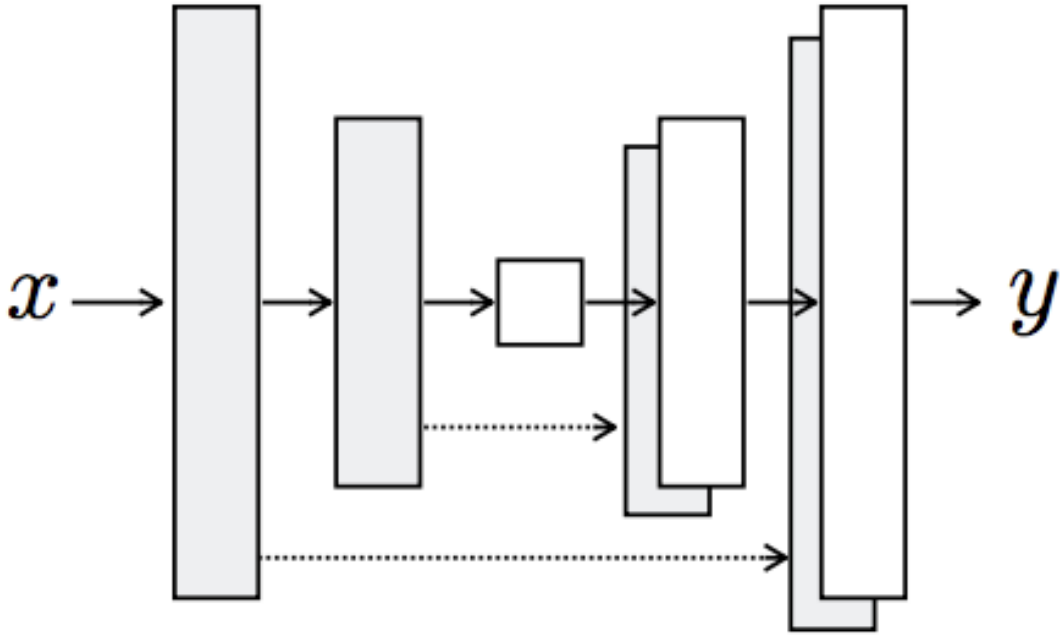
How to compete/get results quickly?

- Transfer Learning!



- Possible solutions
  - "The Oldschool"
    - Take pretrained Encoder, set up decoder and only train decoder
    - Encoder candidates: AlexNet, MobileNets
  - "The Lazy"
    - Take a fully pretrained network and adjust outputs

Advanced architecture: U-Net & Skip Connections



Note on teaching survey

- Thanks to 86 students who took the time to fill it!
- Interaction with the staff – office hours!
- Piazza vs. Moodle

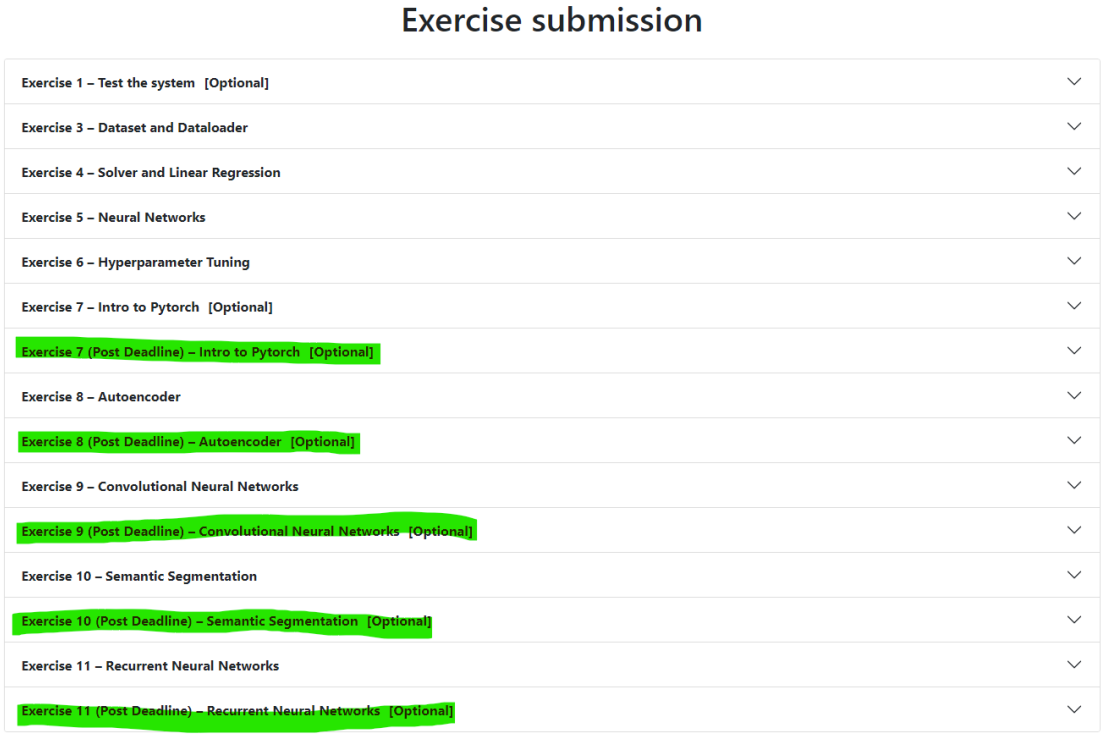
- Ich mag Piazza nicht, ist doof
- Piazza anstatt Moodle für Fragen

- Ohne Piazza sind einige Inhalte der Hausaufgaben deutlich schwieriger zu lösen

"Without piazza some of the exercise contents are considerably harder to solve"

Note on teaching survey

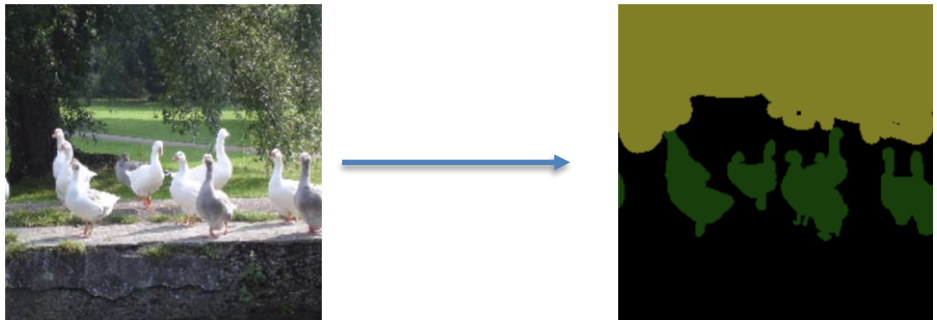
- Perhaps the submission portal for the exercises in the second half of the lecture can remain open so that we can submit our code purely for testing our solutions against the test data in the submission portal, and not for getting the grade. This is because fhe material and exercise difficulty steps up with the introduction of pytorch, due to which some of us fall behind. While we may lose out on the bonus, it would be great for us to still be able to submit our solutions to test ourselves.



These don't count for the bonus!

Summary

- Monday January 10, 2022: Watch Lecture 11: RNN
- Wednesday January 12, 2022: Submit exercise
- Thursday January 13, 2022: Tutorial 11
- Your model's accuracy is all that counts!



Good luck & see you next week