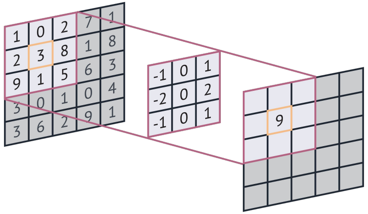
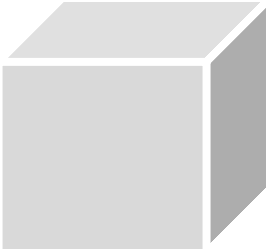
Image segmentation with deep learning

Current state of the art

Presented by Yngve Mardal Moe



Recap on deep learning for image segmentation



3D image segmentation

nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation

Fabian Isensee, Jens Petersen, Andre Klein, David Zimmerer, Paul F. Jaeger, Simon Kohl, Jakob Wasserthal, Gregor Köhler, Tobias Norajitra, Sebastian Wirkert, and Klaus H. Maier-Hein

Division of Medical Image Computing, German Cancer Research Center (DKFZ), Heidelberg, Germany

arXiv:1809.10486v1 [cs.CV] 27 Sep 2018

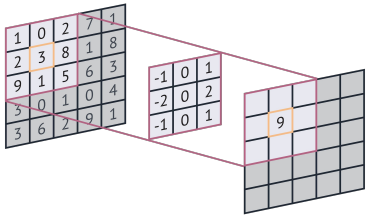
Abstract. The U-Net was presented in 2015. With its straight-forward and successful architecture it quickly evolved to a commonly used benchmark in medical image segmentation. The adaptation of the U-Net to novel problems, however, comprises several degrees of freedom regarding the exact architecture, pre-processing, training and inference. These choices are not independent of each other and substantially impact the overall performance. The present paper introduces the nnU-Net ("no-new-Net"), which refers to a robust and self-adapting framework on the basis of 2D and 3D vanilla U-Nets. We argue the strong case for taking away superfluous bells and whistles of many proposed network designs and instead focus on the remaining aspects that make out the performance and generalizability of a method. We evaluate the nnU-Net in the context of the Medical Segmentation Decathlon challenge, which measures segmentation performance in ten disciplines comprising distinct entities, image modalities, image geometries and dataset sizes, with no manual adjustments between datasets allowed. At the time of manuscript submission, nnU-Net achieves the highest mean dice scores across all classes and seven phase 1 tasks (except class 1 in BrainTumour) in the online leaderboard of the challenge.

Keywords: Semantic Segmentation, Medical Imaging, U-Net

1 Introduction

Medical Image Segmentation is currently dominated by deep convolutional neural networks (CNNs). However, each segmentation benchmark seems to require specialized architectures and training scheme modifications to achieve competitive performance [1,2,3,4,5]. This results in huge amounts of publications in the field that, alongside often limited validation on only few or even just a single dataset, make it increasingly difficult for researchers to identify methods that live up to their promised superiority beyond the limited scenarios they are demonstrated on. The Medical Segmentation Decathlon is intended to specifically address this issue: participants in this challenge are asked to create a segmentation algorithm that generalizes across 10 datasets corresponding to different entities

nnU-Net



Recap on deep learning for image segmentation

3D image segmentation

nnU-Net

In deep learning, we hope to find functions that utilise the structure of our data to perform a given task

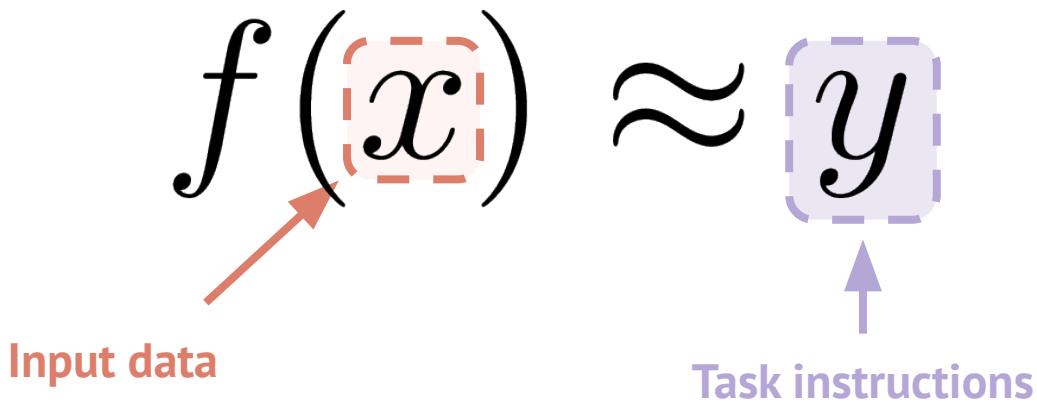
$$f(x) \approx y$$

In deep learning, we hope to find functions that utilise the structure of our data to perform a given task

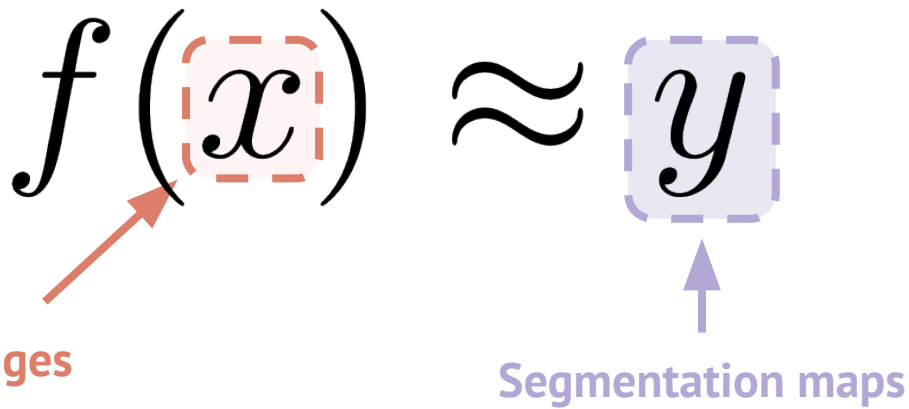
$$f(x) \approx y$$

Input data

In deep learning, we hope to find functions that utilise the structure of our data to perform a given task



In deep learning, we hope to find functions that utilise the structure of our data to perform a given task



To discover this function, we parametrise it by a set of weights and try to find the optimal weights

$$f(x; W) \approx y$$

To discover this function, we parametrise it by a set of weights and try to find the optimal weights

$$f(x; W) \approx y$$

Model weights
(parameters)

Neural networks are described as a cascade of function compositions

$$f(x; W) = \breve{f}_n(\breve{f}_{n-1}(...\breve{f}_2(\breve{f}_1(x; W_1), W_2)..., W_{n-1}), W_n)$$

Neural networks are described as a cascade of function compositions

$$f(x; W) = \breve{f}_n(\breve{f}_{n-1}(...\breve{f}_2(\breve{f}_1(x; W_1), W_2)..., W_{n-1}), W_n)$$

Neural network

Layer functions

Each layer is generally an affine map (linear plus constant) followed by an element-wise nonlinear activation function

$$f(x; W) = \breve{f}_n(\breve{f}_{n-1}(...\breve{f}_2(\breve{f}_1(x; W_1), W_2)..., W_{n-1}), W_n)$$

$$\breve{f}_k(x; A, b) = \phi(Ax + b)$$

Each layer is generally an affine map (linear plus constant) followed by an element-wise nonlinear activation function

$$f(x; W) = \breve{f}_n(\breve{f}_{n-1}(...\breve{f}_2(\breve{f}_1(x; W_1), W_2)..., W_{n-1}), W_n)$$

$$\breve{f}_k(x; A, b) = \phi(Ax + b)$$

Layer weights (W_k above)

Each layer is generally an affine map (linear plus constant) followed by an element-wise nonlinear activation function

$$f(x; W) = \breve{f}_n(\breve{f}_{n-1}(...\breve{f}_2(\breve{f}_1(x; W_1), W_2)..., W_{n-1}), W_n)$$

$$\breve{f}_k(x; A, b) = \phi(Ax + b)$$

Layer weights (W_k above)

Affine map

Each layer is generally an affine map (linear plus constant) followed by an element-wise nonlinear activation function

$$f(x; W) = \breve{f}_n(\breve{f}_{n-1}(...\breve{f}_2(\breve{f}_1(x; W_1), W_2)..., W_{n-1}), W_n)$$

$$\breve{f}_k(x; A, b) = \phi(Ax + b)$$

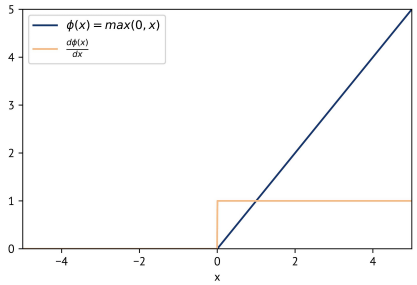
Layer weights (W_k above)

Nonlinear activation function

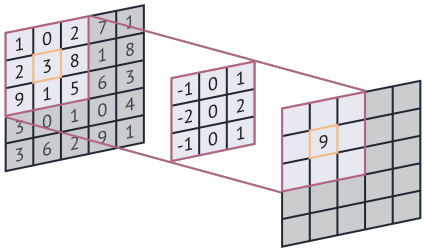
Affine map

This parametrisation gives rise to a large number of choices when designing a neural network

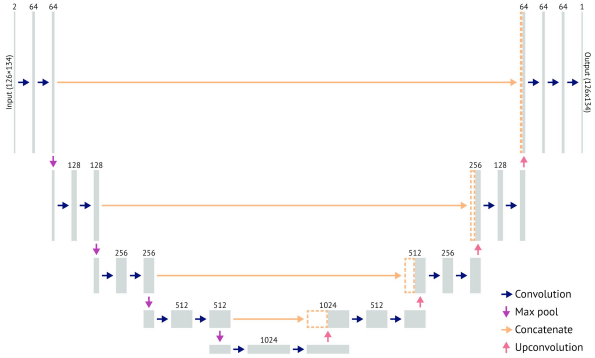
Activation function



Structure of linear map

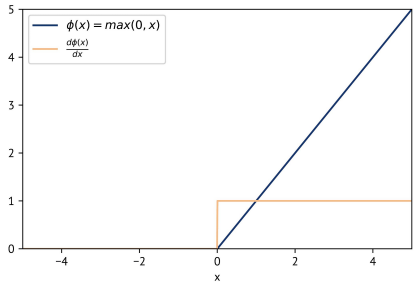


Structure of function composition

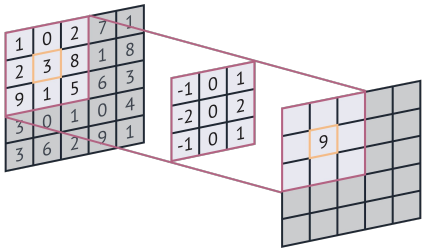


The exact choice of model is called an *architecture*

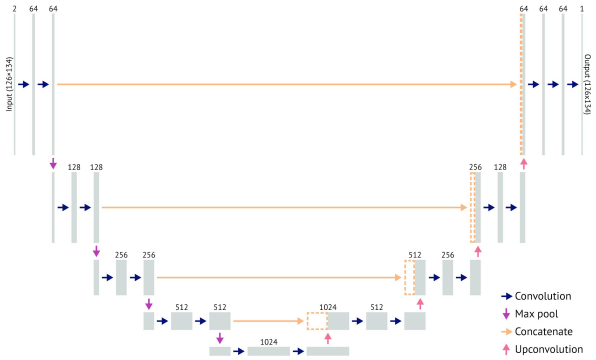
Activation function



Structure of linear map



Structure of function composition



Model architecture

The most commonly used activation function is the ReLU function

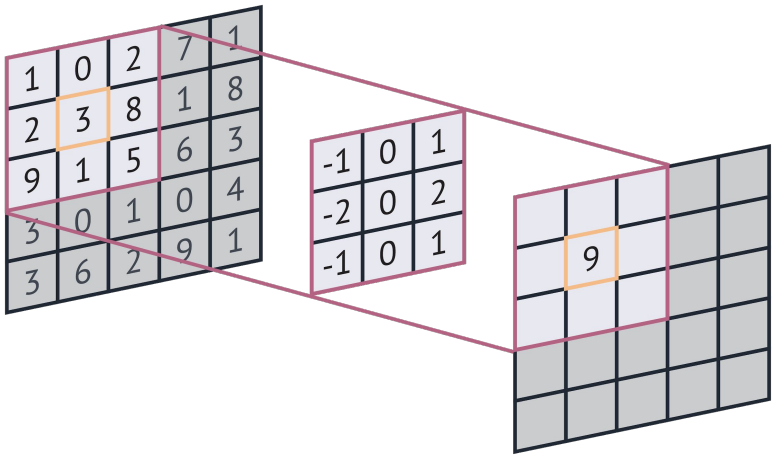
$$\phi(x) = \max(x, 0)$$

However, because of some numerical benefits, the *leaky ReLU* activation function is also often used

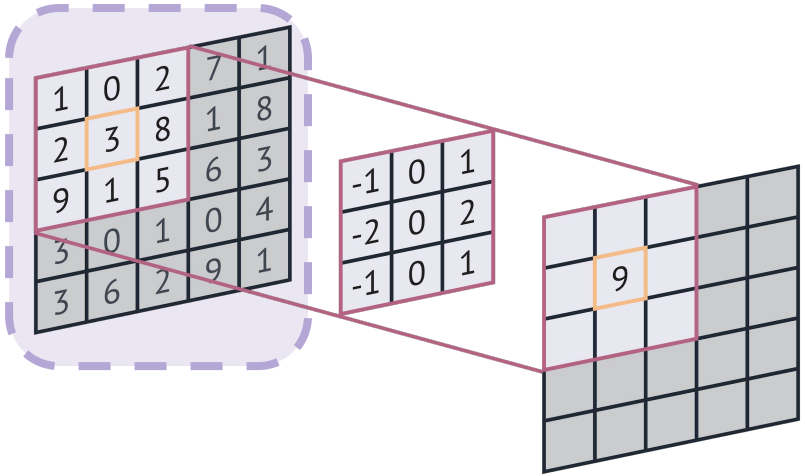
$$\phi(x) = \max(x, 0)$$

$$\phi_a(x) = \max(x, ax)$$

For image segmentation, the linear map is a convolution

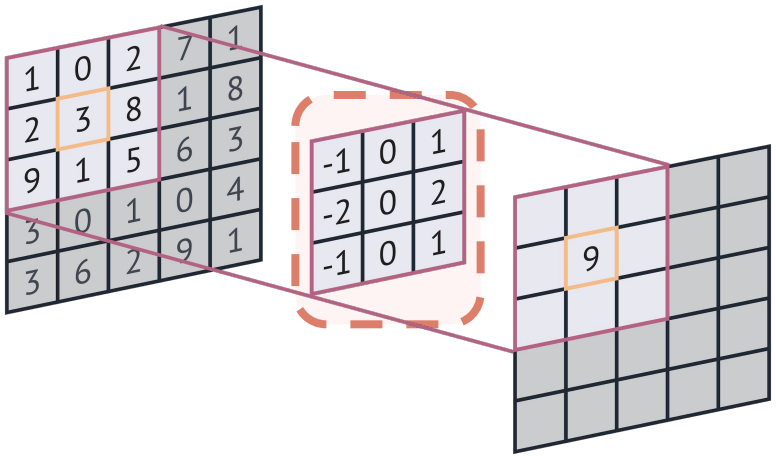


For image segmentation, the linear map is a convolution



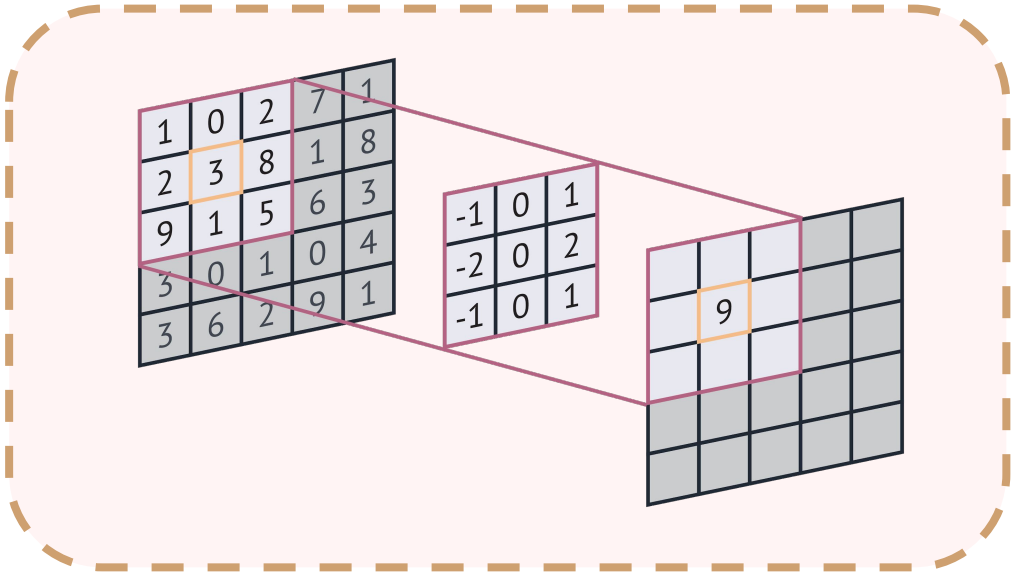
$$\breve{f}_k(x; A, b) = \phi(Ax + b)$$

For image segmentation, the linear map is a convolution



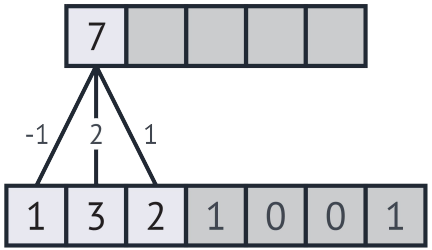
$$\breve{f}_k(x; A, b) = \phi(Ax + b)$$

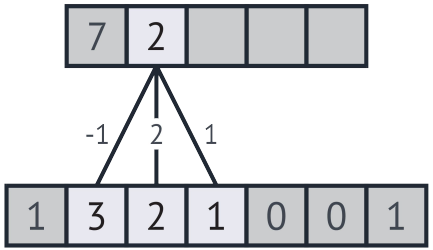
For image segmentation, the linear map is a convolution

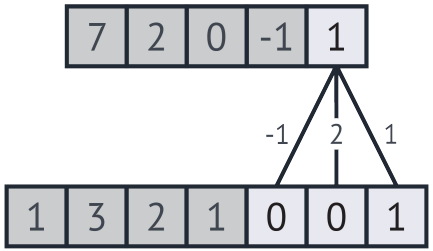


$$\breve{f}_k(x; A, b) = \phi(Ax + b)$$

To illustrate convolution concepts, we will look at 1D convolutions







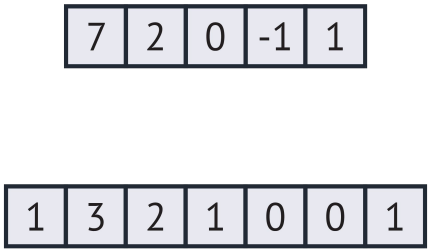
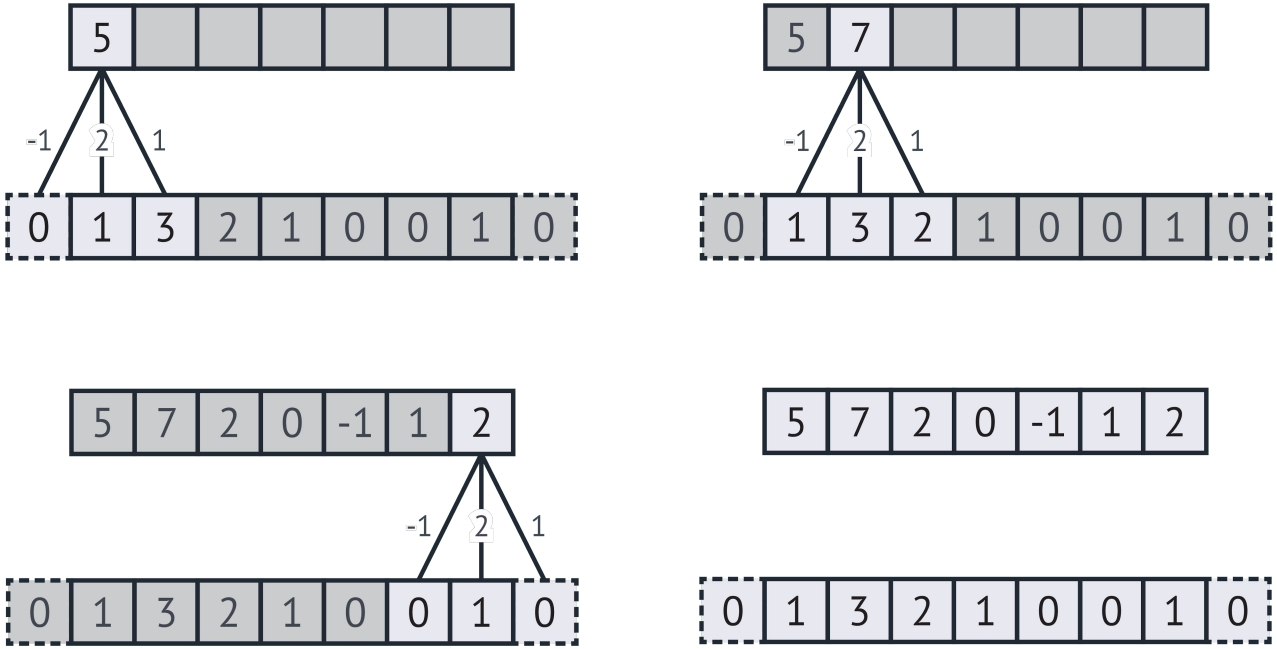
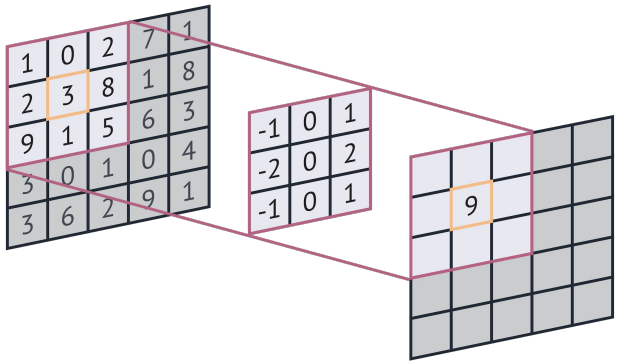
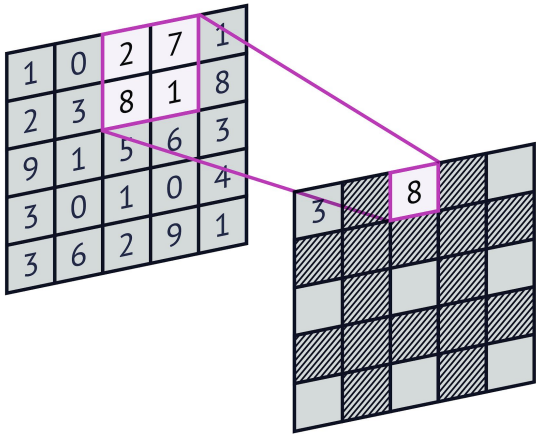


Image boundaries are often padded by zeros to keep image size constant

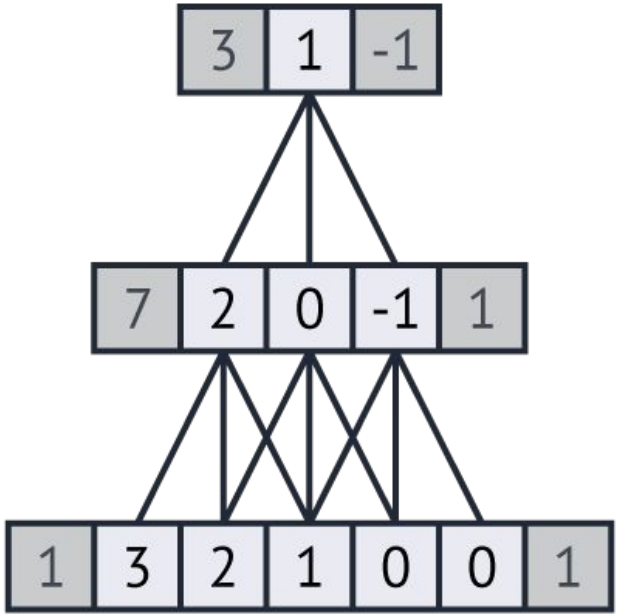


A convolutional neural network is a neural network where all layers can are sliding window operators

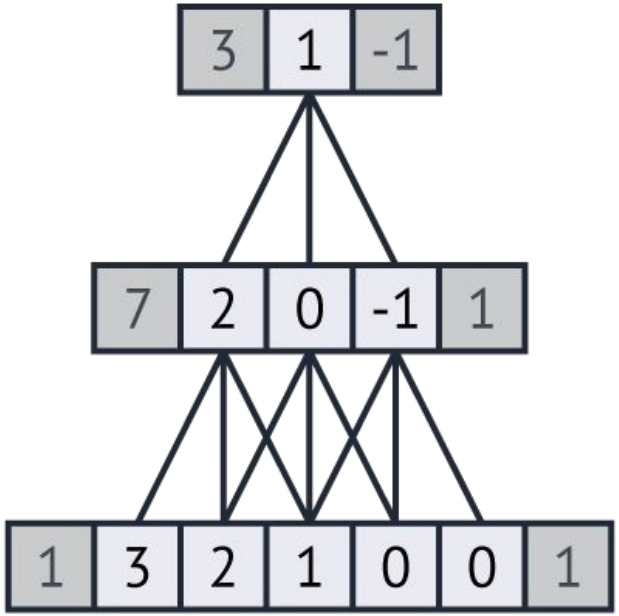




An important concept for convolutional neural networks is *receptive fields*

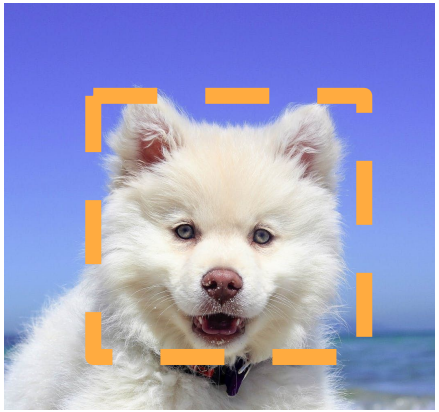


The receptive field of a layer is the size of the input image that affects it



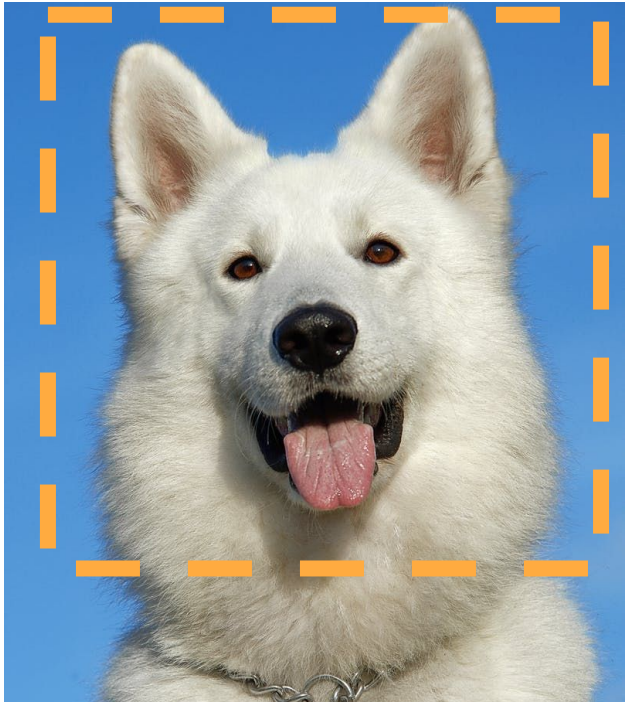
For more info: [Araujo et al. *Computing Receptive Fields of Convolutional Neural Networks*]

The receptive field describes how large features we are able to discover in the input image

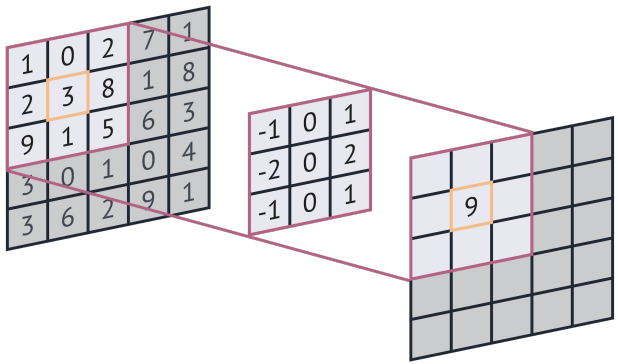


Small receptive field

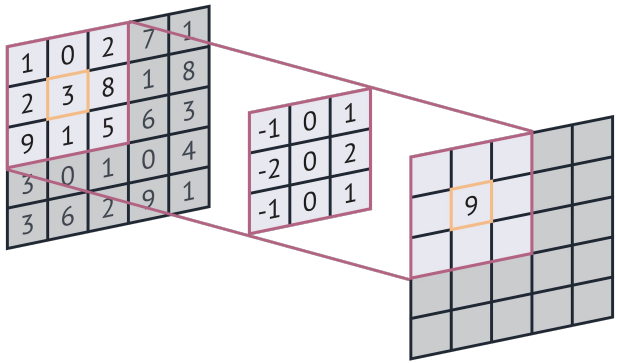
Large receptive field



A single *3×3* convolution has a receptive field of *3×3*

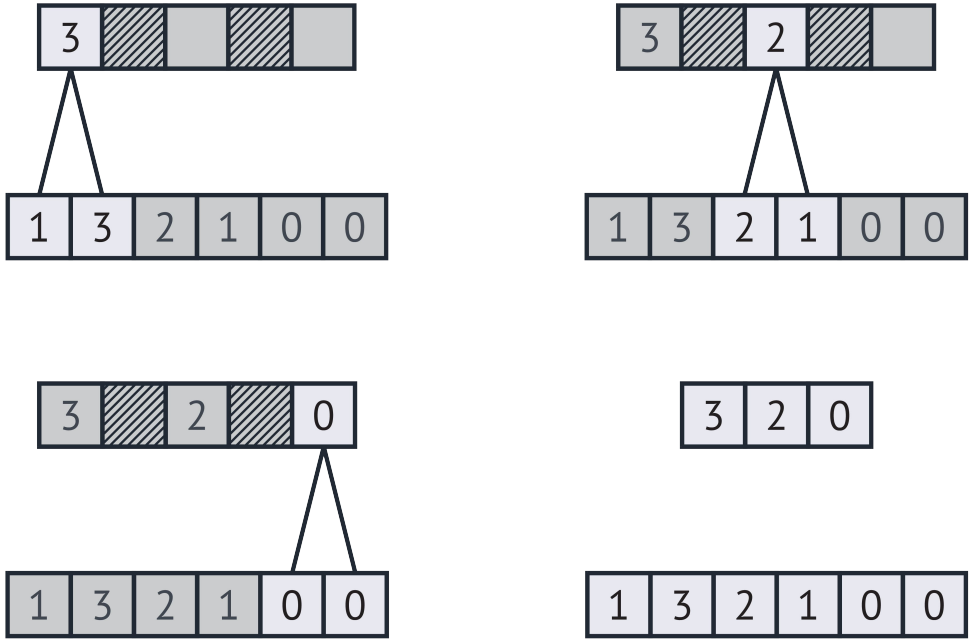


Two stacked *3×3* convolutions has a receptive field of *5×5*

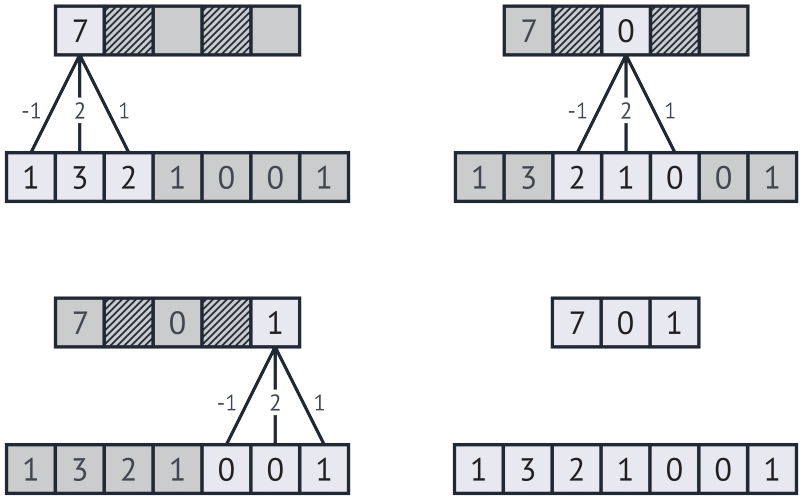


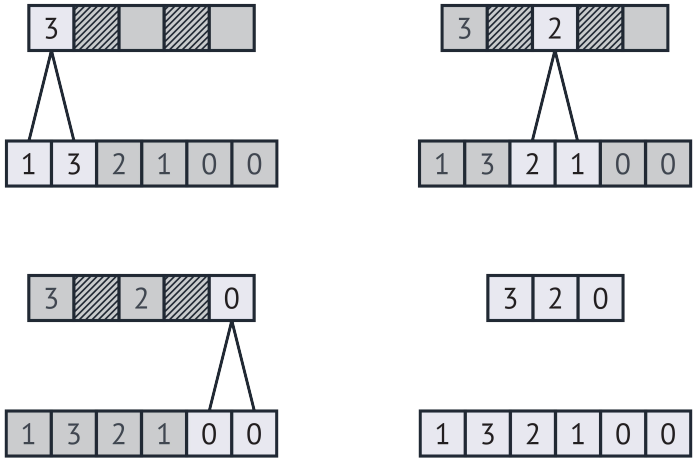
If we just stack convolutions, then the receptive field grows slowly

To combat this, we generally use downsampling, or *pooling* layers

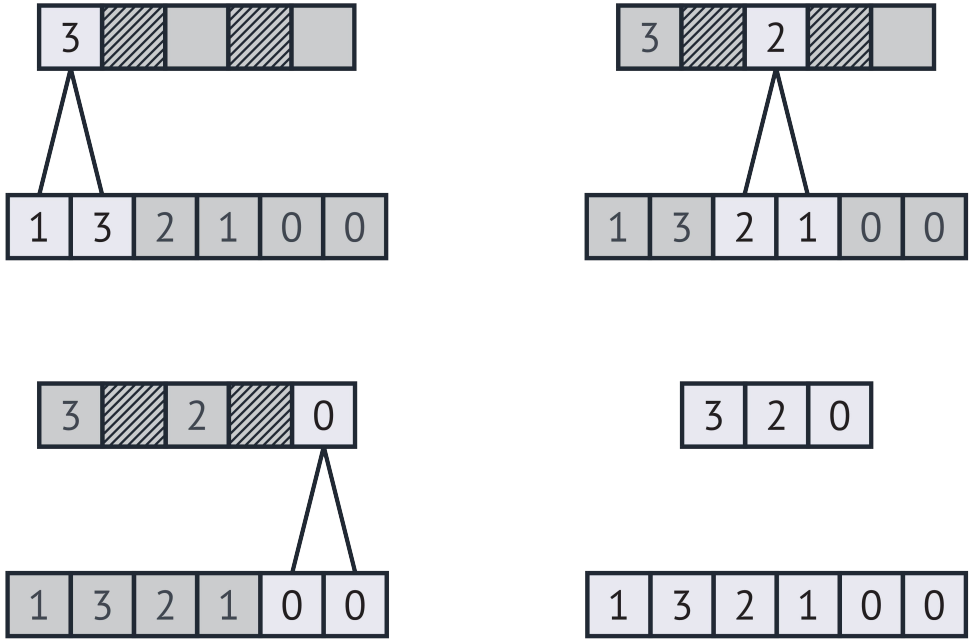


Downsampling is generally either performed via a max-pooling or a strided convolution

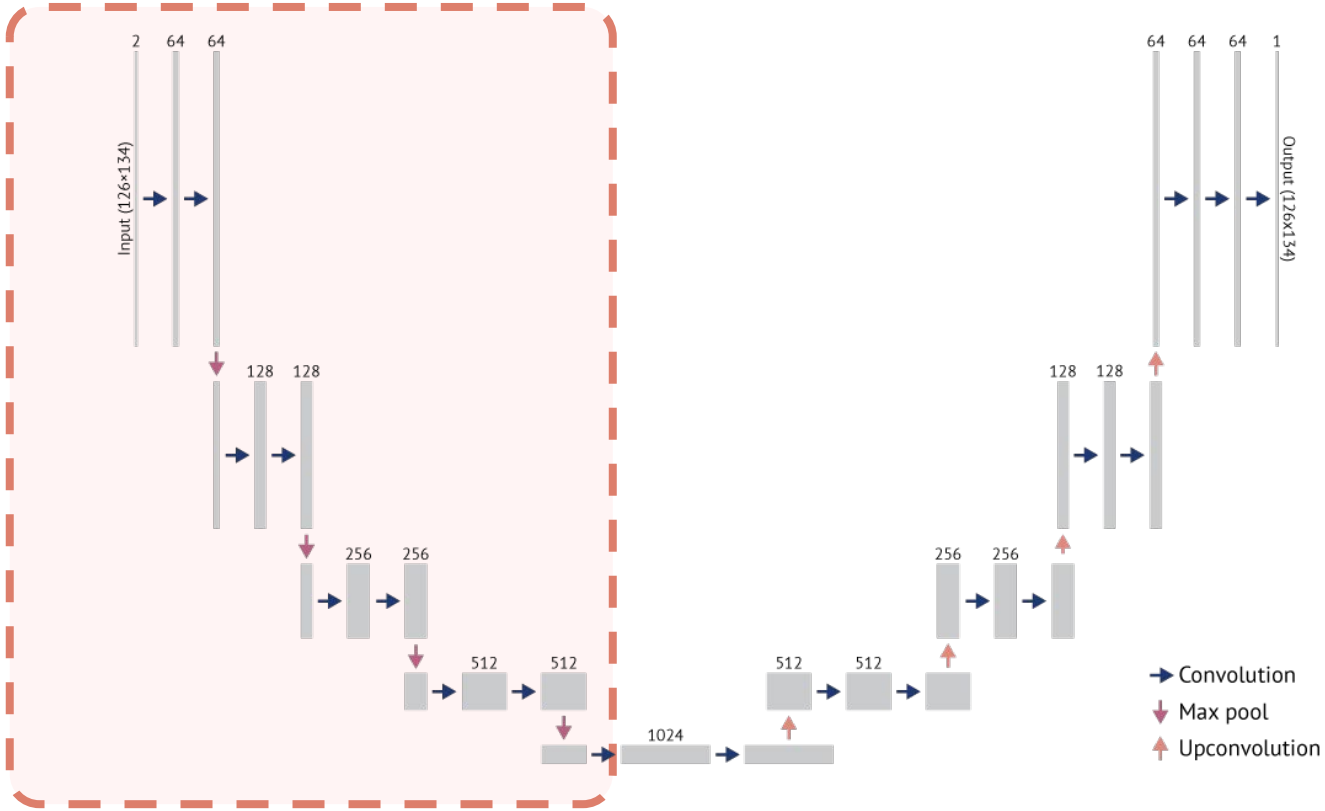




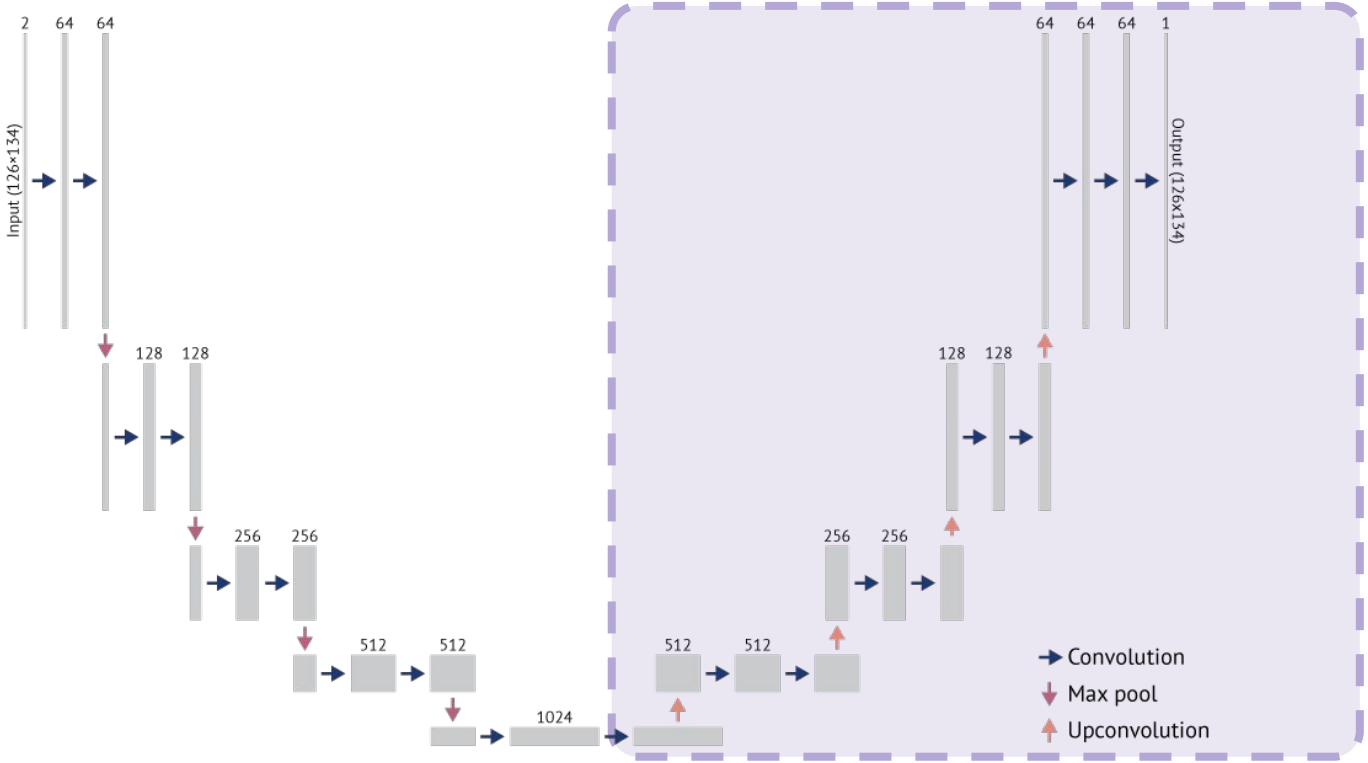
Downsampling reduces the image size, so the output of the network will be smaller than the input



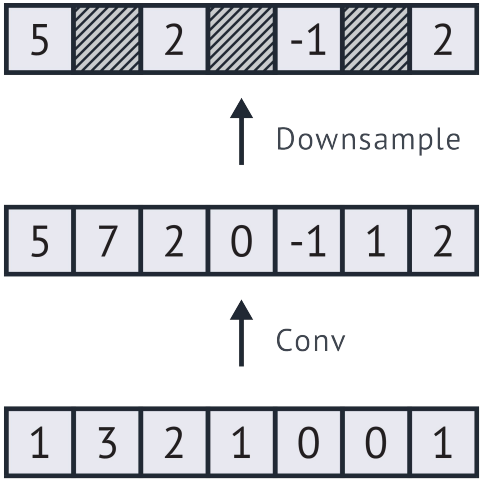
The encoder down-samples the image, allowing us to discover large features in the input image



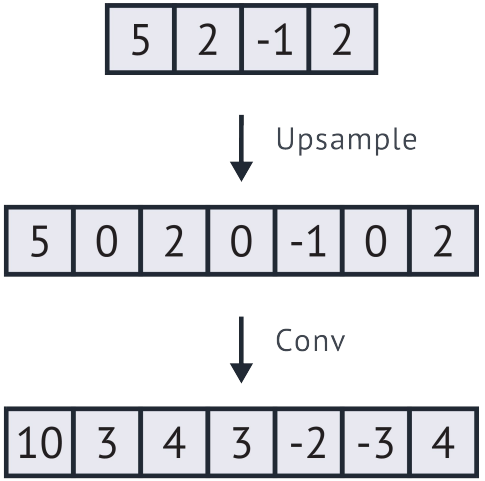
The the decoder upscales the output from the encoder, allowing us to locate where each object is in the image



This upscaling is normally done via a transposed strided convolution

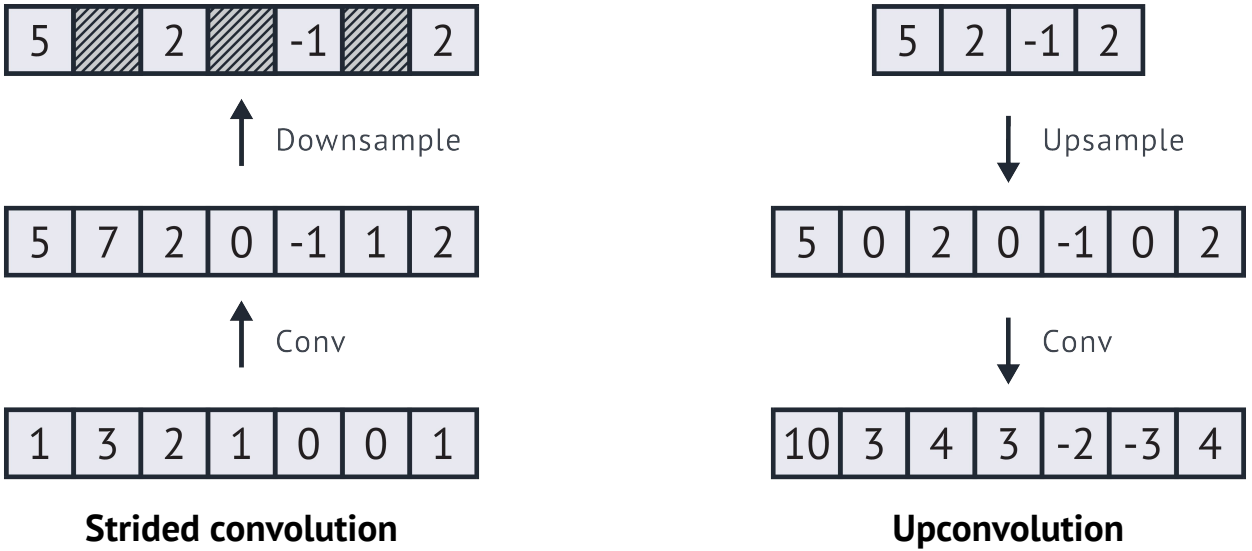


Strided convolution



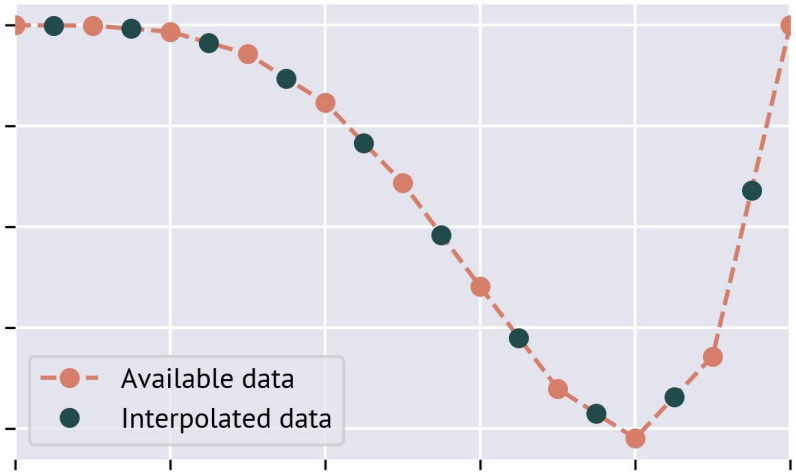
Transposed strided convolution
“Upconvolution”
“Deconvolution”
[...]

Upconvolution can lead to checkerboard artefacts, so some authors prefer to use interpolation instead

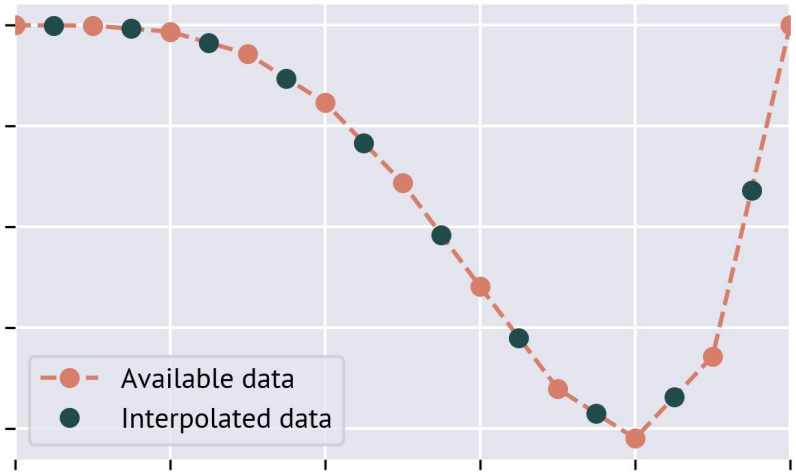


[Augustus Odena et al. - Deconvolution and Checkerboard Artifacts]

If you image has odd dimensions before downscaling, then a linear interpolation is needed



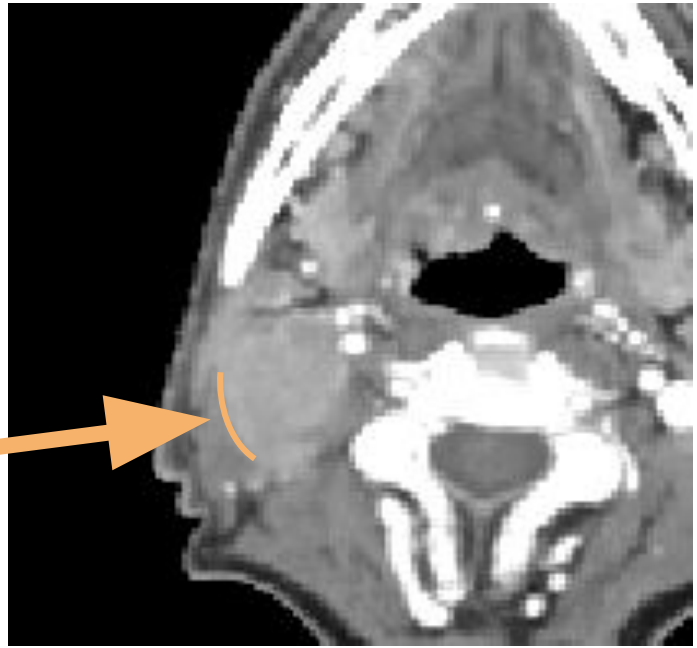
If you image has odd dimensions before downscaling, then a linear interpolation is needed



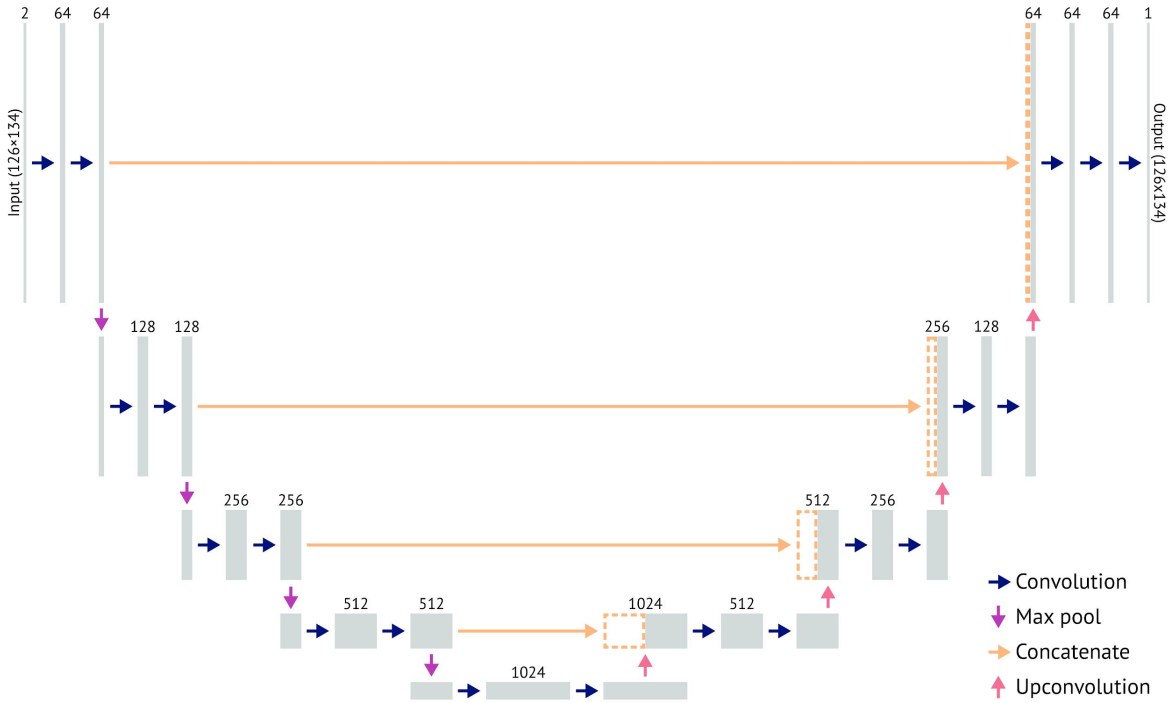
Cubic interpolation cannot run on GPU (at least in TensorFlow)

Unfortunately, the downsampling procedure discards high-frequency information

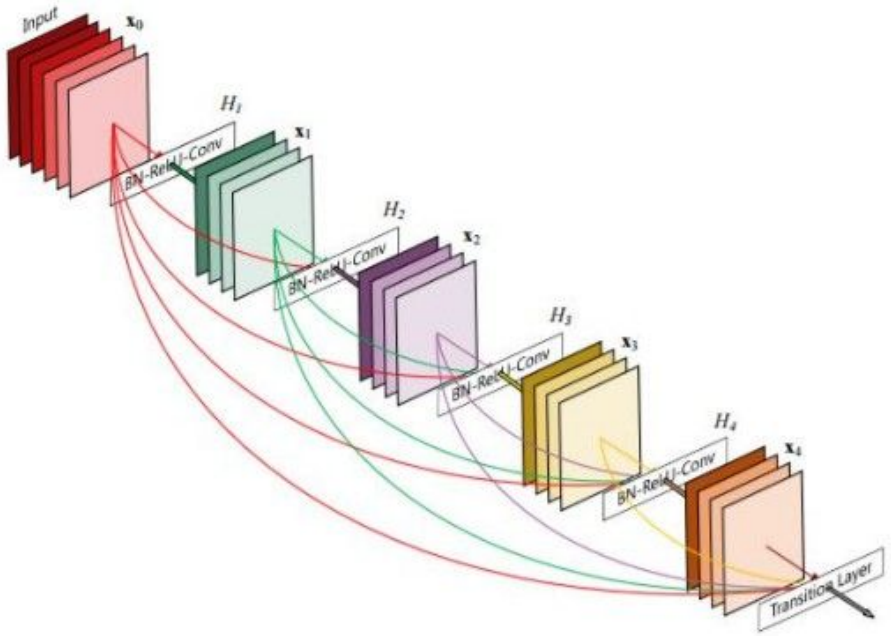
Where is the tumour edge?



The U-Net and SegNet architectures were designed to preserve this information



Other architectures also exist, however, U-Net is regarded as a good choice



Now that we have decided on an architecture, how do we fit these models?

We formulate a loss function that measures the severity of our prediction errors

$$L(W;\mathcal{T}) = \sum_{(x^{(i)},y^{(i)})\in\mathcal{T}} l(f(x^{(i)};W),y^{(i)})$$

Training example

Training set

Loss function

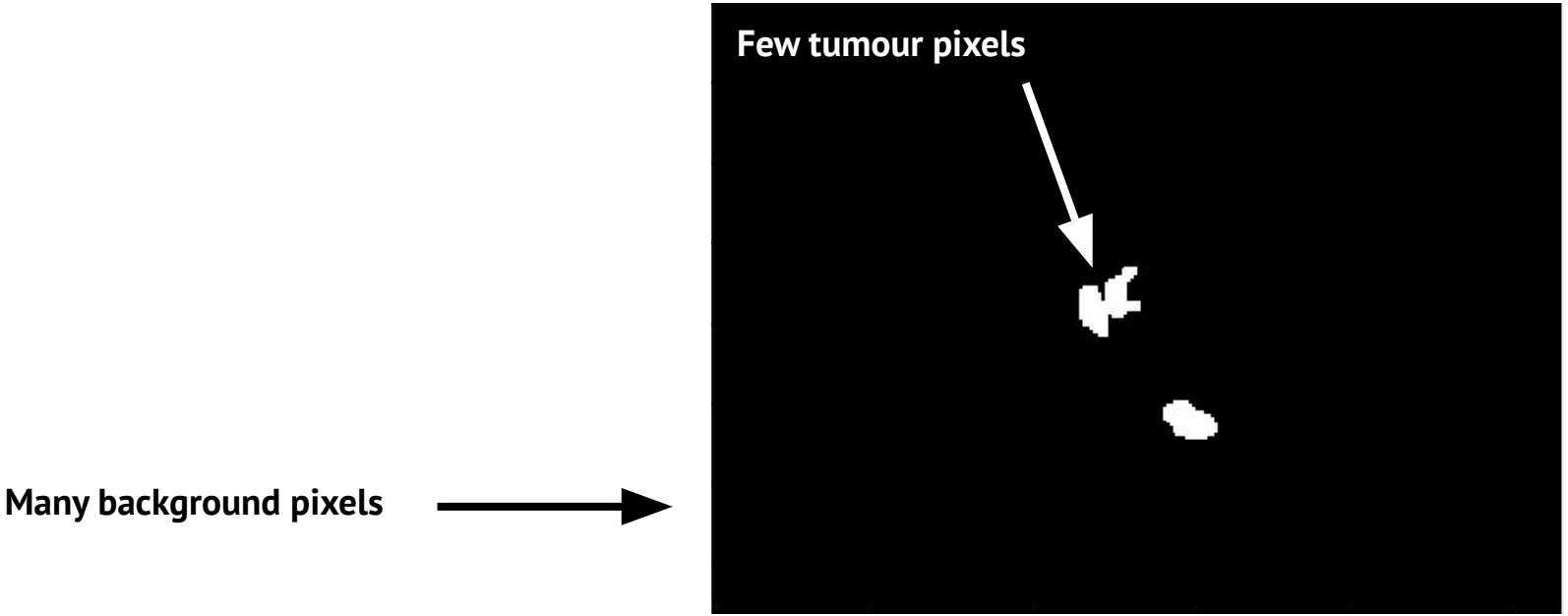
For classification problems, we often use the cross entropy error

$$l_{\mathrm{CE}}(f(x^{(i)};W),y^{(i)}) = -\sum_{p}\left[y_p^{(i)} log\left(f\left(x_p^{(i)}\right)\right) + \left(1-y_p^{(i)}\right) log\left(1-f\left(x_p^{(i)}\right)\right)\right]$$

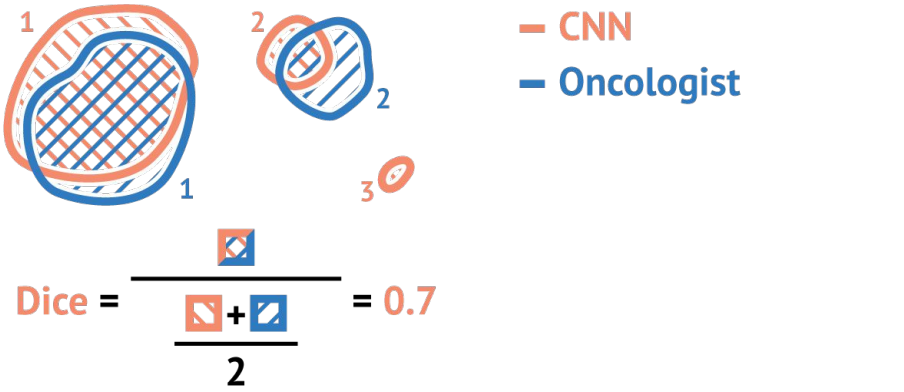
True pixel class

Predicted pixel class

However, this loss function puts too much weight into correctly classifying background pixels

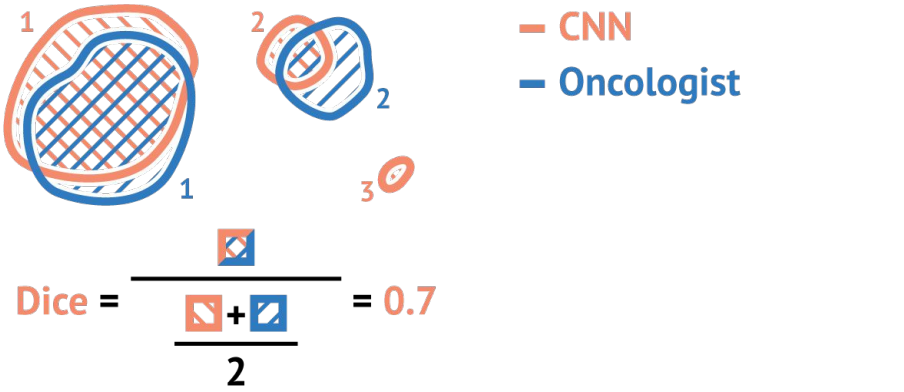


Therefore, we often use the Dice loss instead



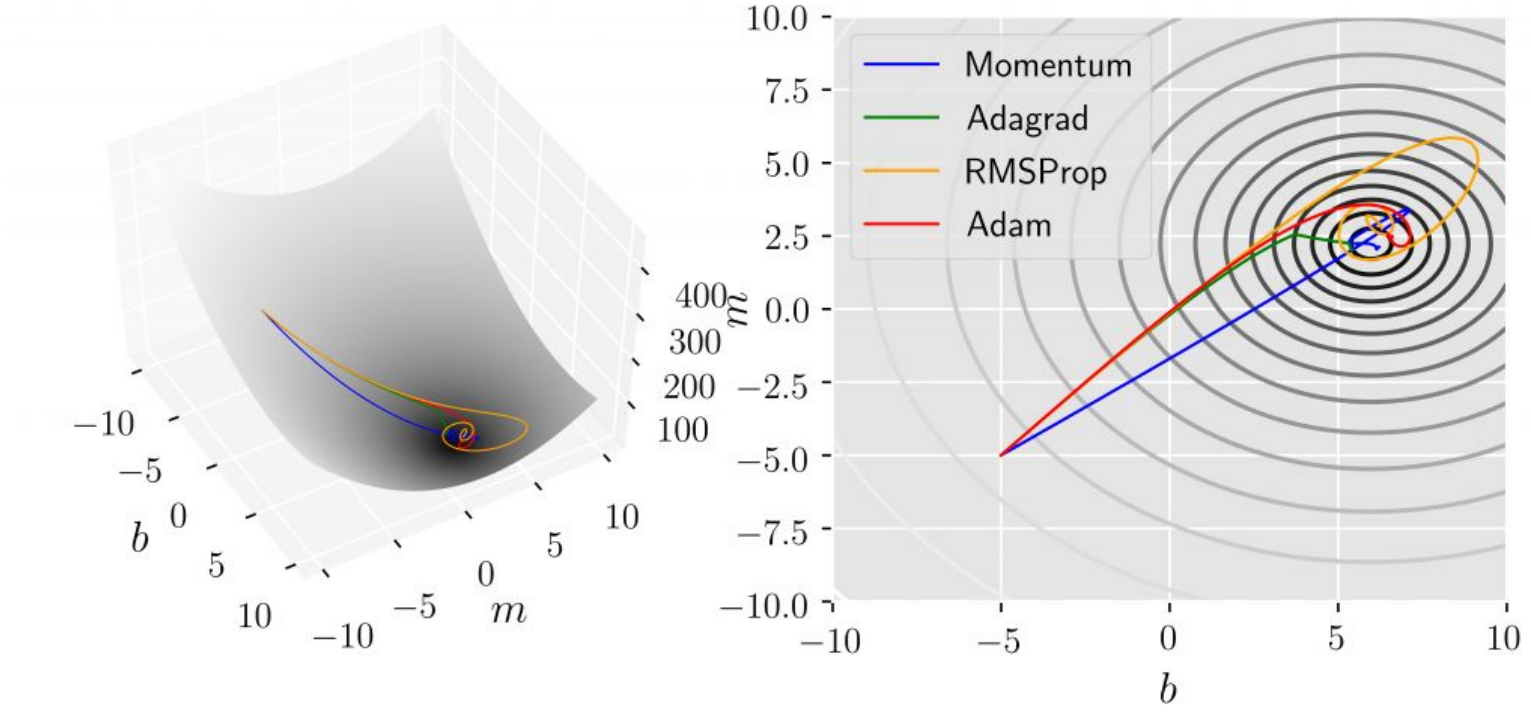
$$l_{\text{dice}}(f(x^{(i)}; W), y^{(i)}) = \frac{2\sum_p \left[f(x_p^{(i)}) y_p^{(i)}\right]}{\sum_p f(x_p^{(i)}) + \sum_p y_p^{(i)}}$$

Therefore, we often use the Dice loss instead



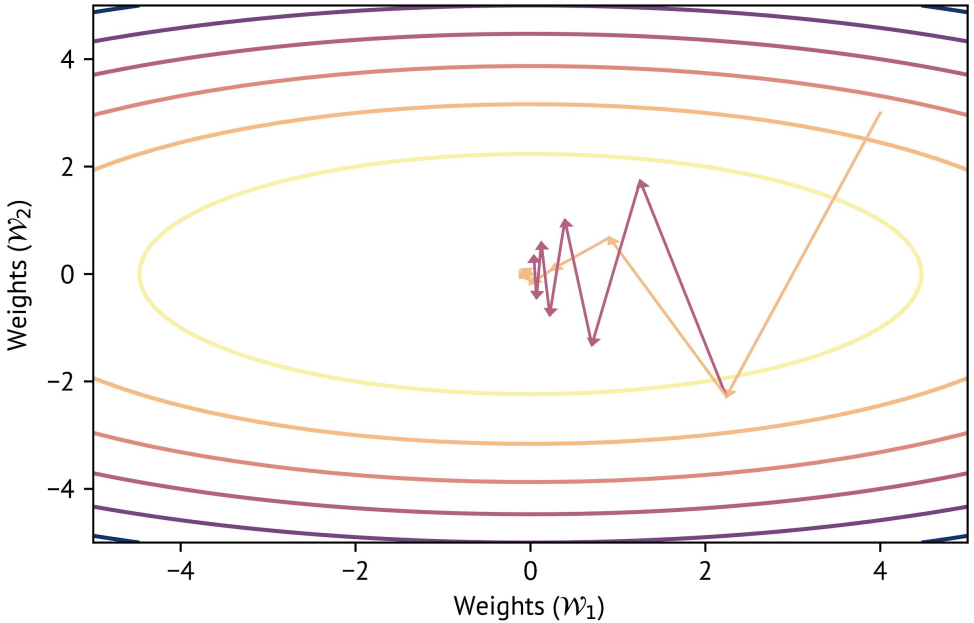
$$l_{\text{dice}}(f(x^{(i)}; W), y^{(i)}) = \frac{2\sum_p \left[f(x_p^{(i)}) y_p^{(i)}\right]}{\sum_p f(x_p^{(i)}) + \sum_p y_p^{(i)}}$$

First minimise the loss using an adaptive momentum method such as Adam

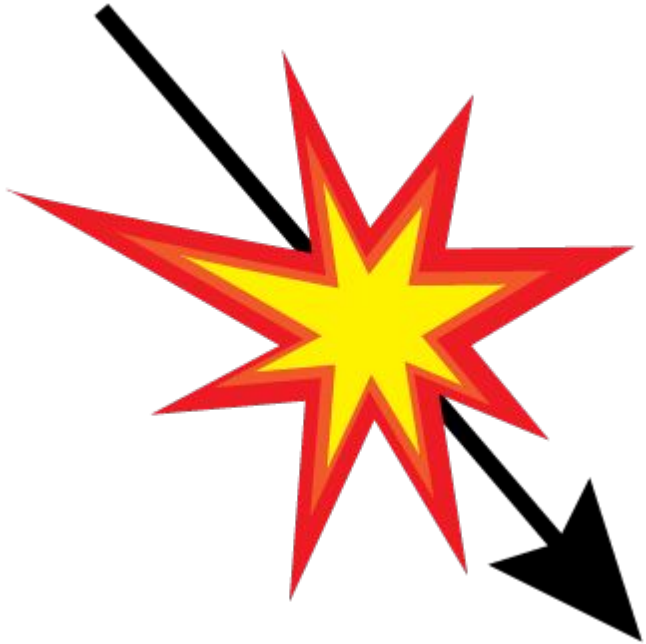


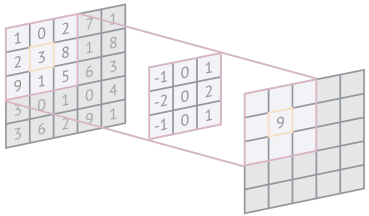
[Figure by Sebastian Hell]

If the generalisation error is large, try using momentum SGD optimiser instead of an adaptive momentum method

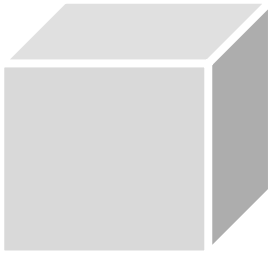


And always remember to normalise your layers to avoid exploding gradients!





Recap on deep learning for image segmentation



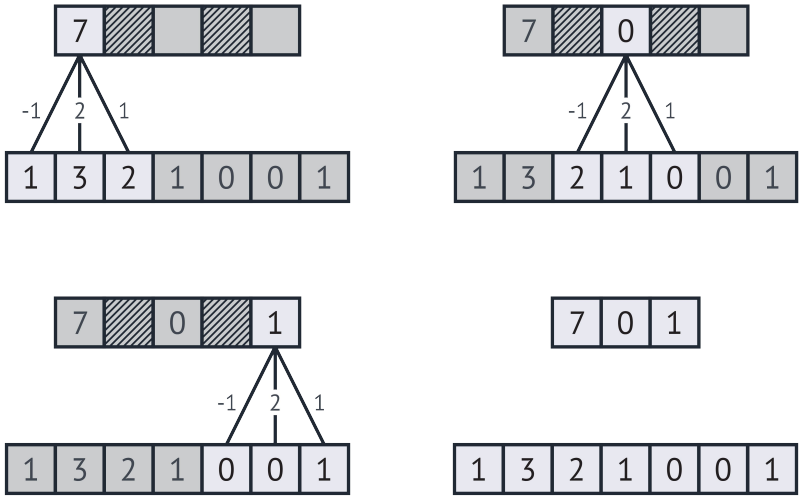
3D image segmentation

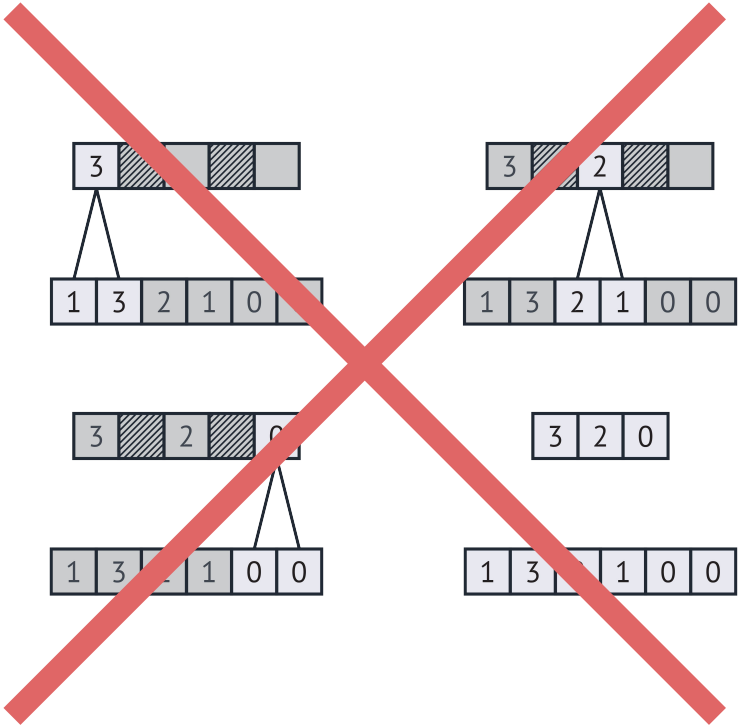
nnU-Net

With 3D images, the memory footprint grows significantly

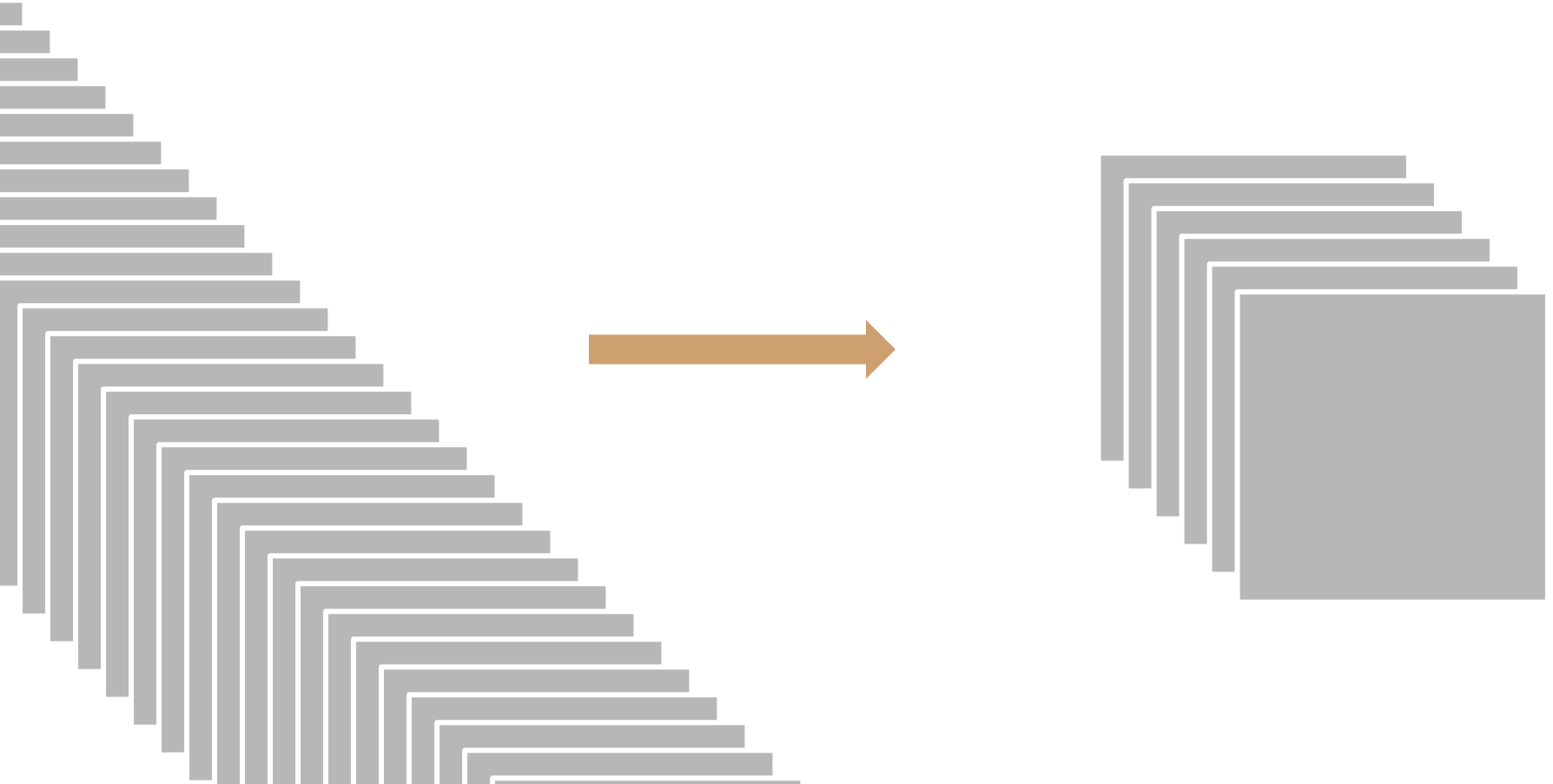
Therefore, we need to use a shallower network than in 3D to fit the model in memory

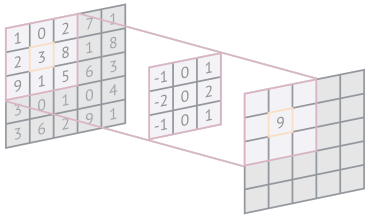
You should consider strided convolutions for 3D instead of max-pooling to reduce memory footprint





You also need to reduce the number of images per batch





Recap on deep learning for image segmentation

3D image segmentation

nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation

arXiv:1809.10486v1 [cs.CV] 27 Sep 2018

nnU-Net

nnU-Net: Self-adapting Frameworkfor U-Net-Based Medical Image Segmentation

nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation

Fabian Isensee, Jens Petersen, Andre Klein, David Zimmerer, Paul F. Jaeger, Simon Kohl, Jakob Wasserthal, Gregor Köhler, Tobias Norajitra, Sebastian Wirkert, and Klaus H. Maier-Hein

Division of Medical Image Computing, German Cancer Research Center (DKFZ), Heidelberg, Germany

Abstract. The U-Net was presented in 2015. With its straight-forward and successful architecture it quickly evolved to a commonly used benchmark in medical image segmentation. The adaptation of the U-Net to novel problems, however, comprises several degrees of freedom regarding the exact architecture, pre-processing, training and inference. These choices are not independent of each other and substantially impact the overall performance. The present paper introduces the nnU-Net ('no-new-Net'), which refers to a robust and self-adapting framework on the basis of 2D and 3D vanilla U-Nets. We argue the strong case for taking away superfluous bells and whistles of many proposed network design and instead focus on the remaining aspects that make out the performance and generalizability of a method. We evaluate the nnU-Net in the context of the Medical Segmentation Decathlon challenge, which measures segmentation performance in ten disciplines comprising distinct entities, image modalities, image geometries and dataset sizes, with no manual adjustments between datasets allowed. At the time of manuscript submission, nnU-Net achieves the highest mean dice scores across all classes and seven phase 1 tasks (except class 1 in BrainTumour) in the online leaderboard of the challenge.

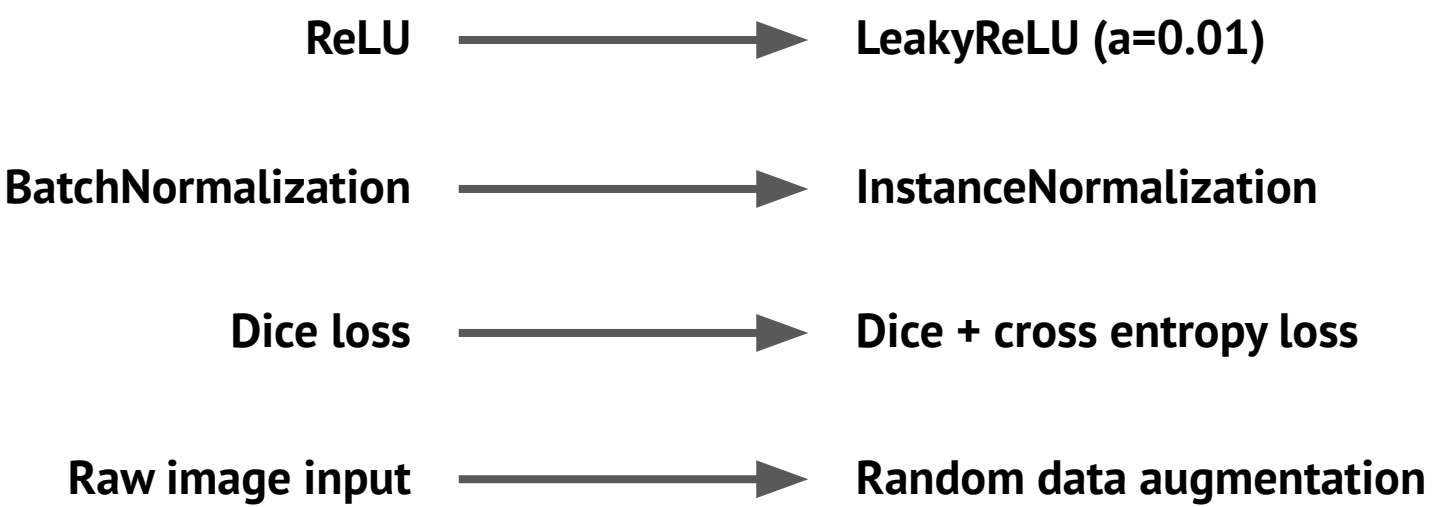
Keywords: Semantic Segmentation, Medical Imaging, U-Net

1 Introduction

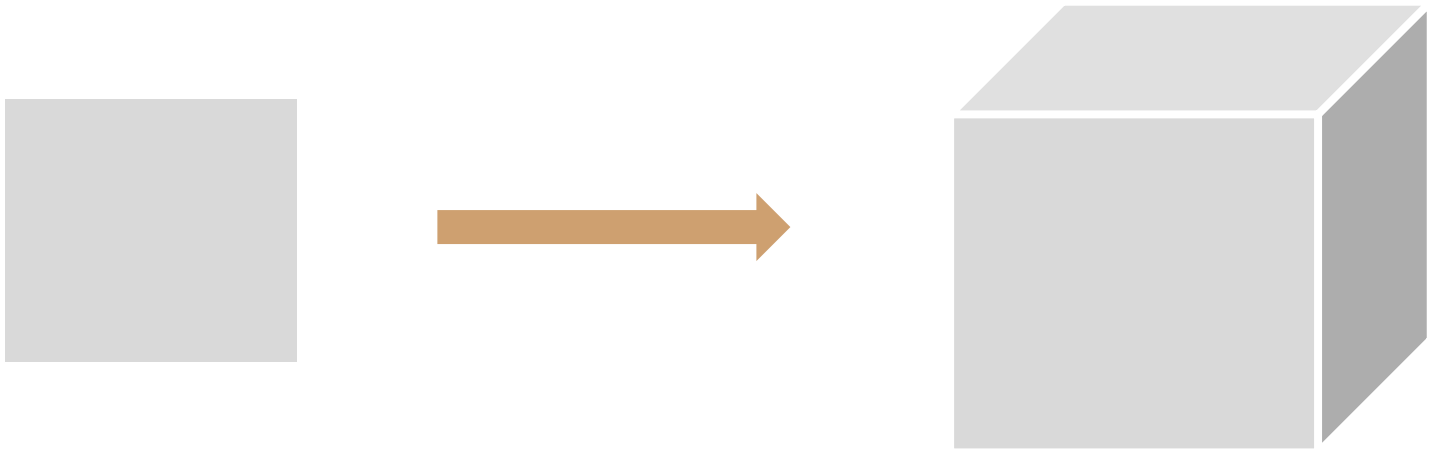
Medical Image Segmentation is currently dominated by deep convolutional neural networks (CNNs). However, each segmentation benchmark seems to require specialized architectures and training scheme modifications to achieve competitive performance [1,2,3,4,5]. This results in huge amounts of publications in the field that, alongside often limited validation on only few or even just a single dataset, make it increasingly difficult for researchers to identify methods that live up to their promised superiority beyond the limited scenarios they are demonstrated on. The Medical Segmentation Decathlon is intended to specifically address this issue: participants in this challenge are asked to create a segmentation algorithm that generalizes across 10 datasets corresponding to different entities

https://arxiv.org/pdf/1809.10486.pdf

The idea is that the U-Net architecture is good enough, only small tweaks to the original architecture is needed



For 3D images, they replaced 2D convolutions with 3D convolutions



The depth of the models in the nnU-Net architecture is adaptive to the dataset size

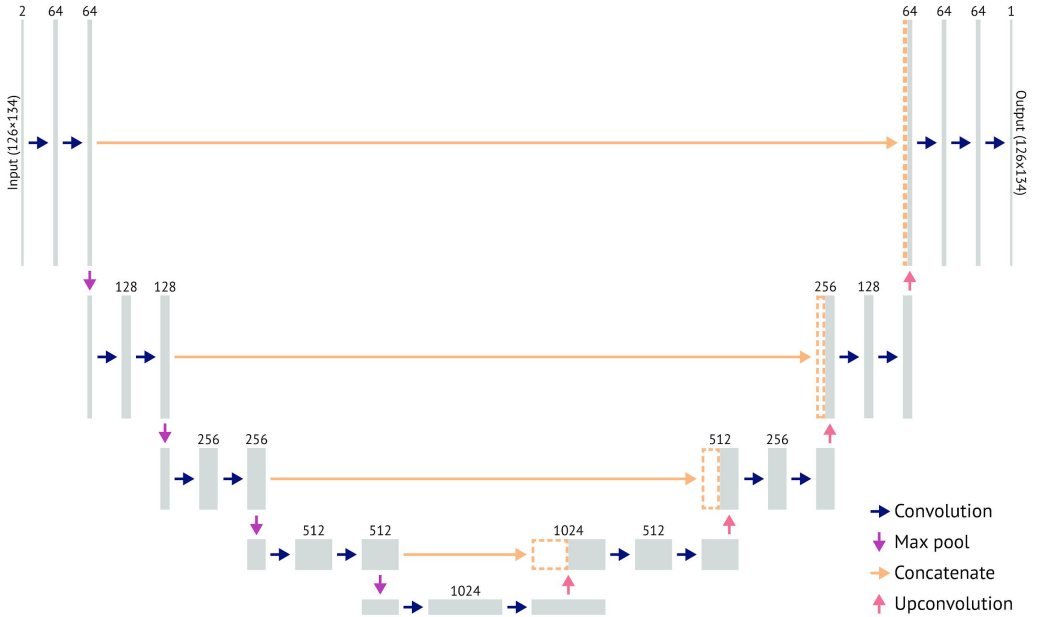
Architecture specification:

- Two convolutions between each up/down-sample operation
- Downsample until image has size 8 in each dimension

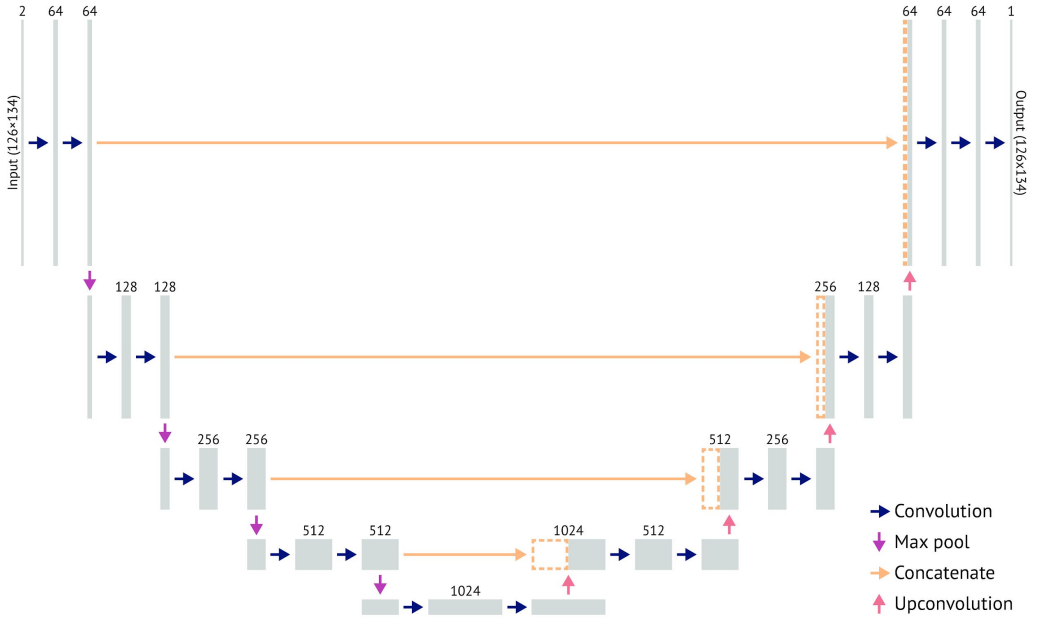
In a later version of the preprint, they show that there is generally not a large effect of any hyperparameter choice

	BraTS	Liver lowres	Liver fullres	Hippocampus	Prostate	Lung nodule	Pancreas
Vanilla nnU-Net	0.72	0.79	0.78	0.89	0.77	0.65	0.65
Batch norm instead of Inst. norm	1.0%	-0.1%	2.9%	-0.1%	-1.3%	-14.2%	-3.7%
No feature map normalization	1.1%	-4.6%	-22.8%	-0.2%	-4.2%	3.0%	-100.0%
ReLU instead of LeakyReLU	0.6%	0.0%	1.0%	-0.1%	-0.2%	-0.4%	0.5%
No data augmentation	-0.8%	-4.9%	1.5%	-1.5%	-0.4%	4.2%	-11.3%
Only cross-entropy loss	-0.6%	-12.0%	-6.3%	0.0%	-1.4%	-25.4%	-8.8%
Only dice loss	0.9%	-2.5%	-10.1%	-0.3%	-3.0%	-11.5%	1.6%

In conclusion, don’t use anything fancier than U-Net for segmentation tasks



In conclusion, don’t use anything fancier than U-Net for segmentation tasks



Except possibly using CRFs as a post-processing step