Department of Computing

Imperial College of Science, Technology and Medicine

What is an nnU-Net

This is a sumamry of the contents of the file [3] and example of its implementation (from the github [2]).

Author: Anton Zhitomirsky

Contents

1 From [2]

- "Especially in three-dimensional (3D) biomedical imaging, for which dataset properties such as imaging modality, image size, (anisotropic) voxel spacing and class ratio vary drastically, this process can be cumbersome and successful configurations from one dataset rarely translate to another".

1.1 Recipe for getting parameters

Previously, applications have been influenced by "expert-driven configurations". nnU-Net is a "primarily data-driven AutoML approache". Specifically, the paper outlines a recipe that systematizes the configuration process on a task-agnostic level and drastically reduces the search space for empirical design choices when given a new task.

These are quoted below from the paper verbatim

1. Collect design decisions that do not require adaptation between datasets and identify a robust common configuration ('fixed parameters').
2. For as many of the remaining decisions as possible, formulate explicit dependencies between specific dataset properties ('dataset fingerprint') and design choices ('pipeline fingerprint') in the form of heuristic rules to allow for almost-instant adaptation on application ('rule-based parameters').
3. Learn only the remaining decisions empirically from the data ('empirical parameters').

It "automatically configures itself, including preprocessing, network architecture, training and post-processing for any new task in the biomedical domain".

We have a three step recipe to follow to allow for no intervention from the user when it comes to defining custom parameters based on the structure the u-net is analysing:

Design choice	Required input	Automated (fixed, rule-based or empirical) configuration derived by distilling expert knowledge (more details in online methods)
Learning rate	–	Poly learning rate schedule (initial, 0.01)
Loss function	–	Dice and cross-entropy
Architecture template	–	Encoder–decoder with skip-connection ('U-Net-like') and instance normalization, leaky ReLU, deep supervision (topology-adapted in inferred parameters)
Optimizer	–	SGD with Nesterov momentum ($\mu = 0.99$)
Data augmentation	–	Rotations, scaling, Gaussian noise, Gaussian blur, brightness, contrast, simulation of low resolution, gamma correction and mirroring
Training procedure	–	1,000 epochs × 250 minibatches, foreground oversampling
Inference procedure	–	Sliding window with half-patch size overlap, Gaussian patch center weighting
Intensity normalization	Modality, intensity distribution	If CT, global dataset percentile clipping & z score with global foreground mean and s.d. Otherwise, z score with per image mean and s.d.
Image resampling strategy	Distribution of spacings	If anisotropic, in-plane with third-order spline, out-of-plane with nearest neighbor Otherwise, third-order spline
Annotation resampling strategy	Distribution of spacings	Convert to one-hot encoding → If anisotropic, in-plane with linear interpolation, out-of-plane with nearest neighbor Otherwise, linear interpolation
Image target spacing	Distribution of spacings	If anisotropic, lowest resolution axis tenth percentile, other axes median. Otherwise, median spacing for each axis. (computed based on spacings found in training cases)
Network topology, patch size, batch size	Median resampled shape, target spacing, GPU memory limit	Initialize the patch size to median image shape and iteratively reduce it while adapting the network topology accordingly until the network can be trained with a batch size of at least 2 given GPU memory constraints. for details see online methods.
Trigger of 3D U-Net cascade	Median resampled image size, patch size	Yes, if patch size of the 3D full resolution U-Net covers less than 12.5% of the median resampled image shape
Configuration of low-resolution 3D U-Net	Low-res target spacing or image shapes, GPU memory limit	Iteratively increase target spacing while reconfiguring patch size, network topology and batch size (as described above) until the configured patch size covers 25% of the median image shape. For details, see online methods.
Configuration of post-processing	Full set of training data and annotations	Treating all foreground classes as one; does all-but-largest-component-suppression increase cross-validation performance? Yes, apply; reiterate for individual classes No, do not apply; reiterate for individual foreground classes
Ensemble selection	Full set of training data and annotations	From 2D U-Net, 3D U-Net or 3D cascade, choose the best model (or combination of two) according to cross-validation performance

Figure 1: Parameter choices and descriptions

1.2 Fixed Parameters

During development of nnU-Net we identified a robust configuration (that is, certain architecture and training properties) that can simply be used all the time. This includes, for example, nnU-Net's loss function, (most of the) data augmentation strategy and learning rate.

- Architectural template: closely follows U-Net, uses Leaky ReLU instead of ReLU.
- Training schedule: 1000 epochs with 250 mini-batches
- Inference: sliding window

1.3 Rule-based Parameters

Uses the dataset fingerprint to adapt certain segmentation pipeline properties by following hard-coded heuristic rules. For example, the network topology (pooling behavior and depth of the network architecture) are adapted to the patch size; the patch size, network topology and batch size are optimized jointly given some GPU memory constraint.

- intensity normalisation
- Resampling
- target spacing
- Adaptation of network topology, patch size and batch size.
- Initialization

- Architectural topology
- Adaptation to GPU memory budget.
- Batch size
- Configuration of the 3D U-Net cascade.

1.4 Empirical Parameters

trial-and-error. For example the selection of the best U-net configuration for the given dataset (2D, 3D full resolution, 3D low resolution, 3D cascade) and the optimization of the postprocessing strategy.

- Ensembling and selection of U-Net configuration(s)
- Post-processing

1.5 Data Fingerprint

- As a first processing step, nnU-Net crops the provided training cases to their non-zero region (improved computational efficiency)
- captures fingerprint like:
  - image size (before and after cropping)
  - image spacing (phyiscal size of voxels)
  - modalities (x-ray, ultrasound, ct, etc...)
  - as well as mean and std of all voxels

2 Summary of [3]

Similarly to the original U-Net architecture they don't change much, but since they are now dealing with 3D data, they consider a pool of basic U-Net architectures: 2D U-Net (2.1), a 3D U-Net (2.2) and a U-Net Cascade (2.3).

2.1 2D U-Net

Contrary to belief that using a 2D network in the context of a 3D network may be counter-intuitive, the paper found that 3D segmentation methods deteriorate in performance if the dataset is anisotropic [3] (when the property varies according to direction [1]).

2.2 3D U-Net

The architecture would typically thrive on contextual information, but we are limited by GPU memory, which allows us to train this architecture only on image patches. This may pose a problem for generating PTV areas because of the large structures, which, in size, may be compared to a Liver which has been cited as 'large images such as Liver, may impede training' [3].

2.3 U-Net Cascade

While the 2D and 3D U-Nets generate segmentations at full resolution, the cascade first generates low resolution segmentations and subsequently refines them.

This addresses the shortcoming in Section 2.2 because we first train the 3D U-Net on downsampled images (which compresses the amount of contextual information you need) (stage 1) and then upsamples to the original voxel spacing and passed as additional (one hot encoded) input channels to a second 3D U-Net, which is trained on patches at full resolution (stage 2).

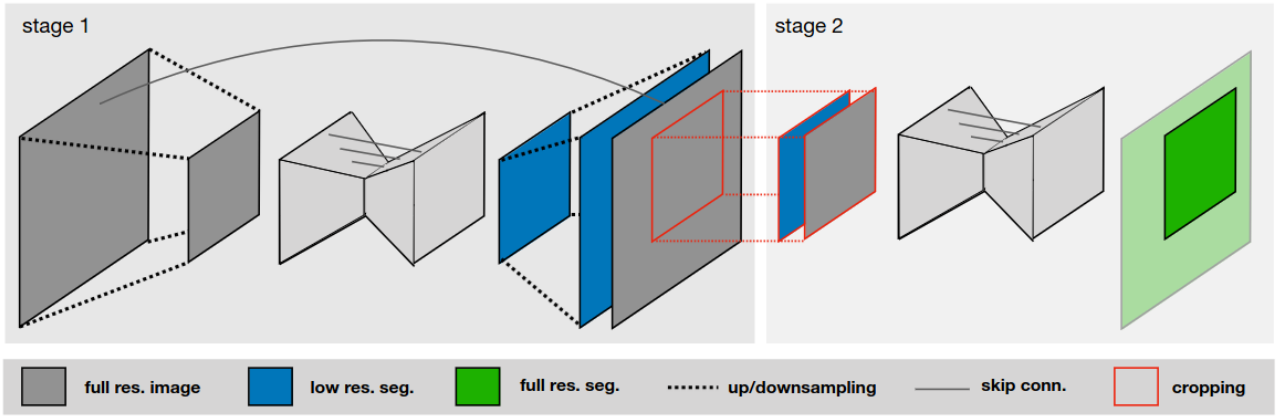

Fig. 1. U-Net Cascade (on applicable datasets only). Stage 1 (left): a 3D U-Net processes downsampled data, the resulting segmentation maps are upsampled to the original resolution. Stage 2 (right): these segmentations are concatenated as one-hot encodings to the full resolution data and refined by a second 3D U-Net.

References

[1] URL: https://sciencenotes.org/isotropic-vs-anisotropic-definition-and-examples/.

[2] Fabian Isensee et al. “nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation”. In: (2021). URL: https://www.nature.com/articles/s41592-020-01008-z.

[3] Fabian Isensee et al. “nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation”. In: (2018). URL: https://arxiv.org/pdf/1809.10486.pdf.