Introduction and Review of nnU-Net

Bernes Lorier Atabonfack

Introduction to nnU-Net (no new Net)

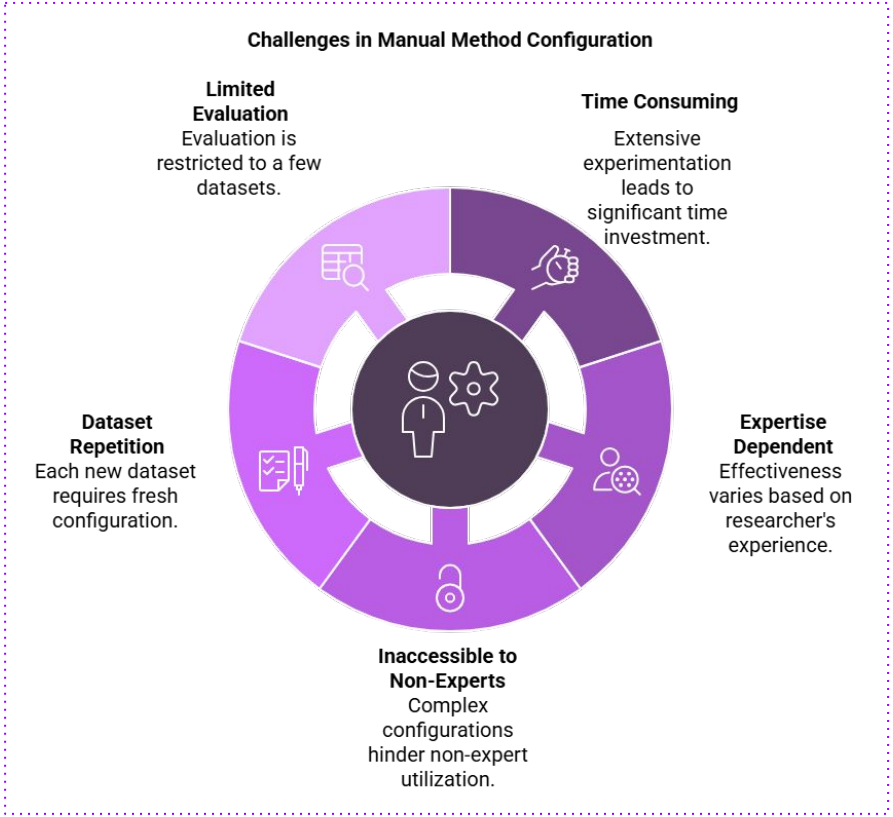
A self Configuration Method for deep learning based image Segmentation

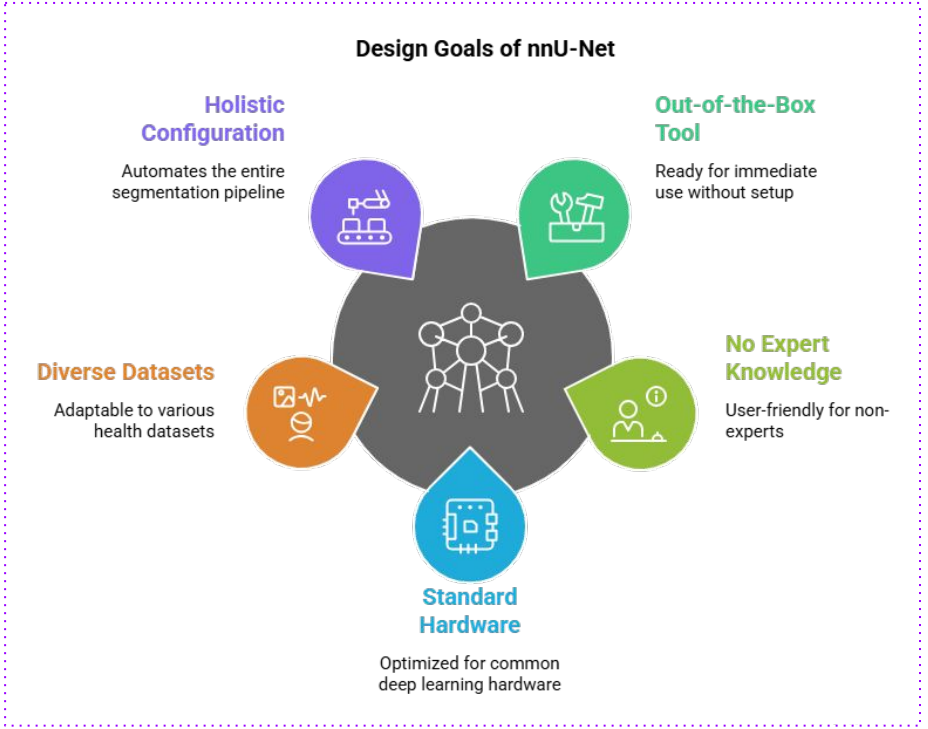
Authors

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

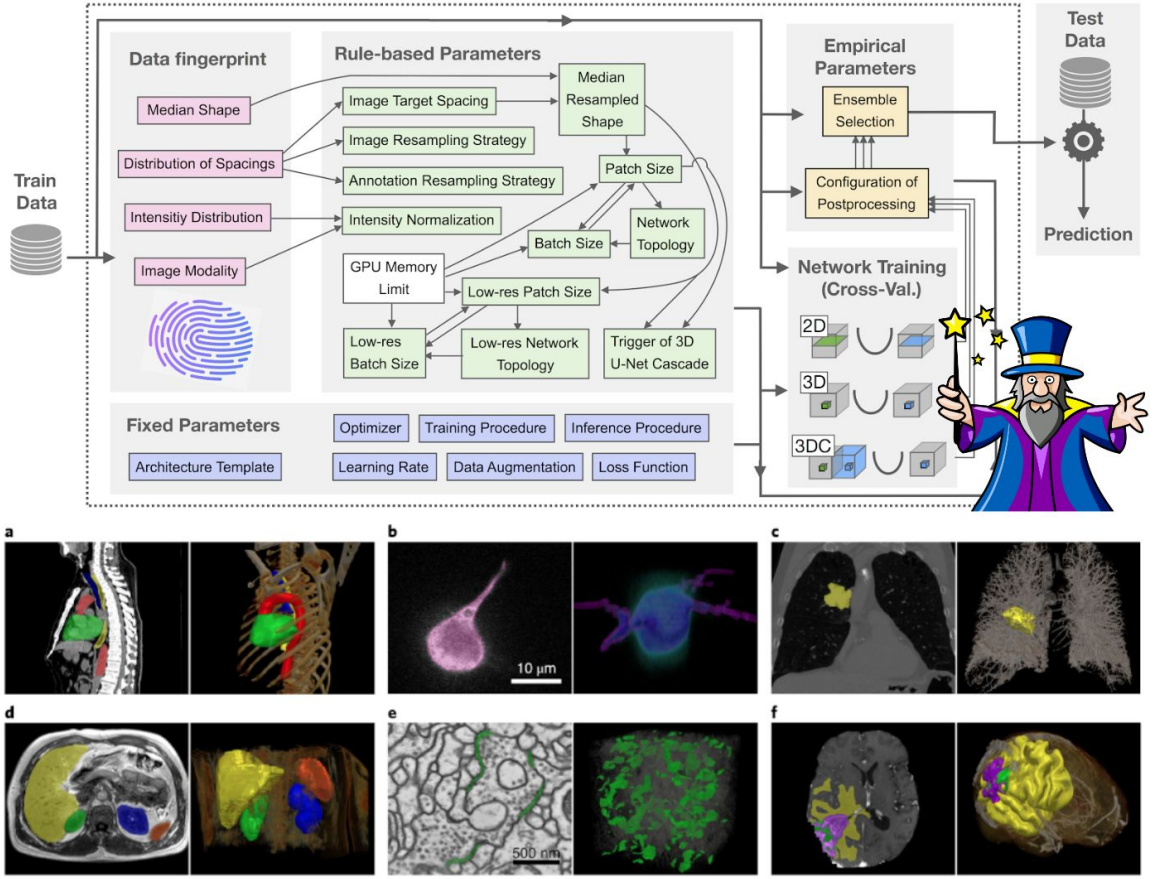
Division of Medical Image Computing, German Cancer Research Center (DKFZ), Heidelberg, Germany

Motivation for nnU-Net and Design Goals

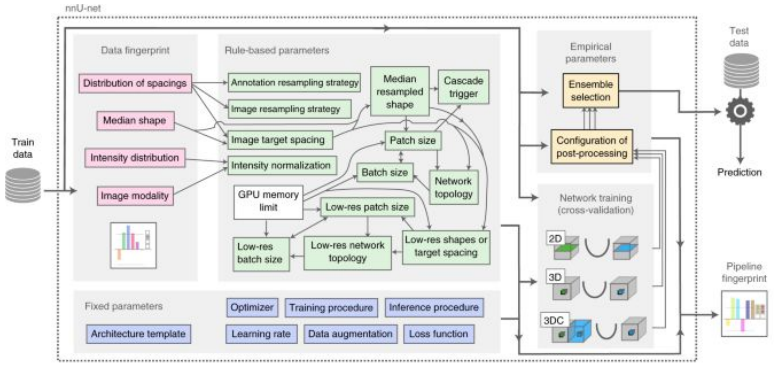




nnU-Net Overview



Fixed Parameters in nnU-Net



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 super-vision (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

Learning rate	PolyLR Schedule (Init: 0.01)
Loss function	Dice + Cross-Entropy
Architecture template	Encoder-decoder with skip-conn. ("U-Net-like") and: Instance norm., Leaky ReLU, deep super-vision (topology adapted in Inferred Parameters)
Optimizer	SGD with Nesterov Momentum (=0.99)
Data augmentation	Rotations, scaling, Gaussian noise, Gaussian blur, brightness, contrast, simulation of low resolution, Gamma and mirroring
Training procedure	1000 epochs x 250 minibatches, foreground oversampling
Inference procedure	Sliding window with half patch size overlap, Gaussian patch center weighting

Data Preprocessing

Cropping

All data is cropped to nonzero regions to reduce size and computational load.

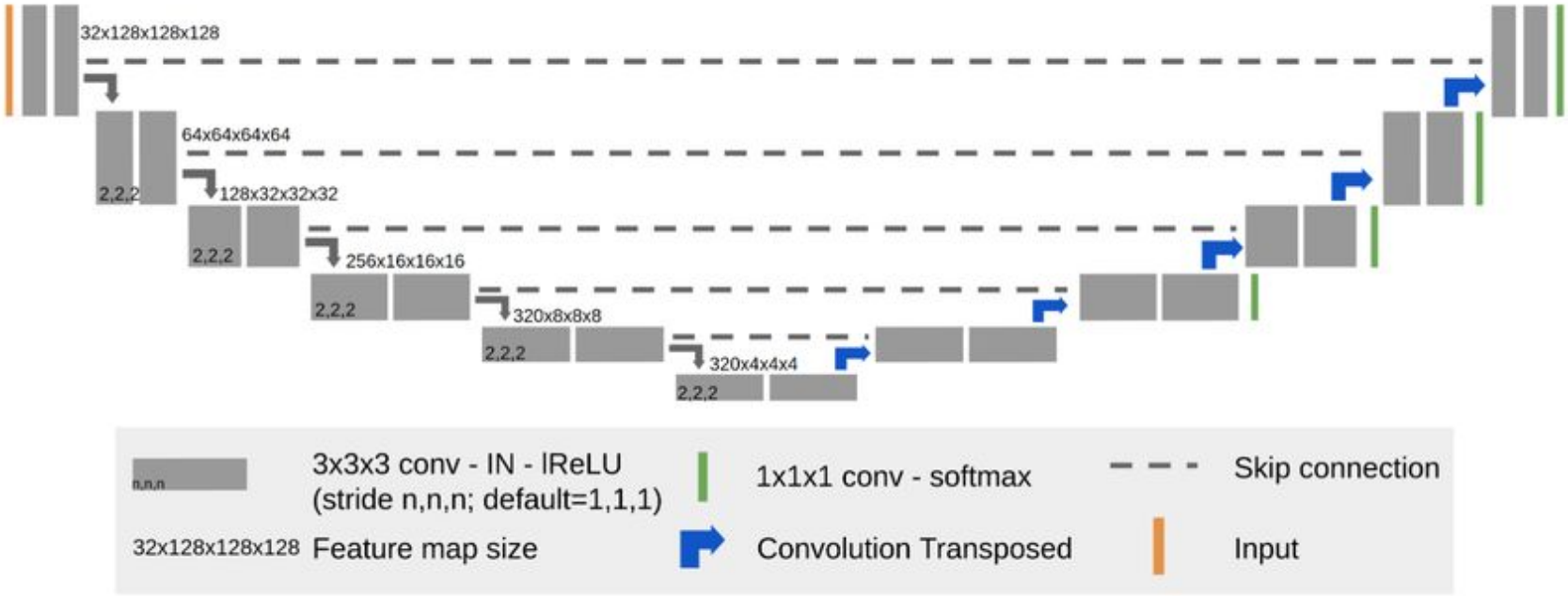
Resampling

All images are resampled to the dataset's median voxel spacing to ensure consistent spatial representation, with U-Net Cascade used for very large or anisotropic datasets by further downsampling.

Normalization

In nnU-Net, CT images are normalized globally based on training set statistics with percentile clipping and z-score normalization, while MRI and other modalities are normalized individually per patient using z-score normalization.

Winning Architecture for BraTS 2020 Segmentation Challenge



BraTS Specific Configuration

Region based Training

They focus on directly optimizing the three tumor subregions; whole tumor, tumor core, and enhancing tumor, to improve segmentation performance.

Post Processing

They improve rankings by removing small predicted enhancing tumors based on a threshold, replacing them with necrosis to optimize challenge performance.

Batch Norn & Batch Dice

Data Augmentation Relative to baseline nnUNet

- increase the probability of applying rotation and scaling from 0.2 to 0.3.
- increase the scale range from (0.85, 1.25) to (0.65, 1.6)
- select a scaling factor for each axis individually
- use elastic deformation with a probability of 0.3
- use additive brightness augmentation with a probability of 0.3
- increase the aggressiveness of the Gamma augmentation

nnU-Net Model Review

Environment Setup

- Create a conda or a python virtual environment
- Install the latest PyTorch version compatible you environment
- Install nnU-Net (as a standard baseline or an Integrative framework)
- Install hidden layer to be able to generate plots
- Create data folders
- Set environment variables

Data folder structure

nnUNet_raw

```
nnUNet_raw/Dataset001_NAME1
├── dataset.json
├── imagesTr
│   ├── ...
├── imagesTs
│   ├── ...
└── labelsTr
    ├── ...
nnUNet_raw/Dataset002_NAME2
├── dataset.json
├── imagesTr
│   ├── ...
├── imagesTs
│   ├── ...
└── labelsTr
    ├── ...
```

nnUNet_preprocessed

Folder storing preprocessed data and from which data is read during training

nnUNet_results

Folder for saving saved weights and pretrained models if they are downloaded

Frequently Used Commands

Environment Setup

- **Create virtualenv:**
  conda create --name myenv <python-version(optional)> /python -m venv myenv]
- **Install PyTorch:**
  conda/pip install torch torchvision
- **Installing nnUNet:**
  pip install nnunetv2 OR
  git clone https://github.com/MIC-DKFZ/nnUNet.git
  cd nnUNet
  pip install -e .
- **Installing Hidden Layer:**
  pip install --upgrade git+https://github.com/FabianIsensee/hiddenlayer.git
- **Exporting folder paths:**
  export nnUNet_raw="/path/to/nnUNet_raw"
  export nnUNet_preprocessed="/path/to/nnUNet_preprocessed"
  export nnUNet_results="/path/to/nnUNet_results"

Model Review Continues

Training

- Rename data to nnU-Net format
- Prepare a json creator compatible with nnU-Net
- Convert dataset to nnUNet format
- Preprocess the data
- Run the training.

- **Create a Json metadata file:**
  Select the python script closest to your dataset in the “dataset_conversion” folder and modify it to suit your dataset.
  Run the modified script to create the json file.

- **Preprocess Data:**
  nnUNetv2_plan_and_preprocess -d DATASET_ID --verify_dataset_integrity -np 1

- **Training:**
  nnUNetv2_train DATASET_ID <2d/3d_fullres/3d_cascade_fullres> <1->5/all> -tr nnUNetTrainer_250epochs
  See different training configurations HERE

Model Review Continues

Inference & Evaluation

- Create an output folder
- Use the inference code from the Github page
- Calculate the metrics

Test Folder structure

```
imagesTs/
├── BraTS-SSA-XXXXX-000_0000.nii.gz # T1n
├── BraTS-SSA-XXXXX-000_0001.nii.gz # T1c
├── BraTS-SSA-XXXXX-000_0002.nii.gz # T2w
└── BraTS-SSA-XXXXX-000_0003.nii.gz # T2f
```

imagesTs folder structure

- **Inference:**

```
nnUNetv2_predict -i path/to/imagesTs -o
nnUNet_dirs/nnUNet_raw/nnUNet_tests/ -d DATASET_ID -c
<2d/3d_fullres/3d_cascade_fullres> -tr nnUNetTrainer_250epochs -f
<1->5/all>
```

- **Metric computation:**

```
nnUNetv2_evaluate_folder /path/to/ground truth/ /path/to/predictions/
-djfile path/to/dataset.json -pfile path/to/plans.json
```

Resources

- YouTube Video of nnU-Net overview: https://youtu.be/3po8qVzz5Tc

- nnU-Net GitHub: https://github.com/MIC-DKFZ/nnUNet/tree/master

- Article on Setting up nnU-Net: https://towardsai.net/p/l/how-i-use-nnunet-for-medical-image-segmentation-a-comprehensive-guide
- nnU-Net Paper: https://arxiv.org/abs/1809.10486
- nnUNet for BraTS paper 2020: https://arxiv.org/abs/2011.00848

Thank you for attending!