cAIsar: The Human Locomotion Challenge

UeberNet: A deep learning approach for gait detection

David A. Danhofer
Technical University Munich
Munich, Germany
david.danhofer@tum.de

Paul Groß
Universität Ulm
Ulm, Germany
paul.gross@uni-ulm.de

Philip Meiners
Universität Mannheim
Mannheim, Germany
cmeiners@mail.uni-mannheim.de

Abstract

The automatic detection of the human gait in time series signals is an important stepping stone to developing different types of medical aids for patients with various pathologies. In this paper it is shown, that an enhanced convolutional neural network based on the U-Net architecture, the UeberNet, is suited to reliably and accurately detect steps in time series recordings.

Keywords: gait detection, convolutional neural networks, U-Net

1 Problem Statement

The human gait is a complex mechanism that is subject to alteration by many pathologies that lead quickly to a loss of autonomy and an increased risk of falls. Analyzing and understanding the human gait could lead to applications in early detection and harm prevention for patients [15]. An integral part of understanding the gait itself is being able to detect it in unlabeled time series (TS) sensor data to efficiently generate data for subsequent developments. Previous approaches include the works of Pham et al. [9] and Liu et al. [6].
The aim of this paper is to demonstrate the feasibility of accurately labeling steps in multi-variate TS data recorded from foot-worn inertia measurement units using a state-of-the-art deep learning approach based on Convolutional Neural Networks (CNNs), the UeberNet[1].

2 Data Preprocessing

Two data sets were provided for the challenge. The training data, respectively the “gait data set”, contains all sensor values, meta data (MD) details about the experiment and the patient as well as the step annotations and is publicly available [15]. The evaluation data set provided by the organizer is not publicly available and lacks the annotations. The sets contain a sensor data and a MD file for each run. As the evaluation data was provided as separate files for the left and right foot, the corresponding pairs were joined to match the gait data set.

[1] https://github.com/andfaxle/cAIsar-ai-cup

2.1 Data encoding and formatting

To format the data for usage with the UeberNet, it is required to convert the annotations to a format of the same length as the input TS. The process from encoding the annotations as a binary mask of labels is shown in Fig. 1.

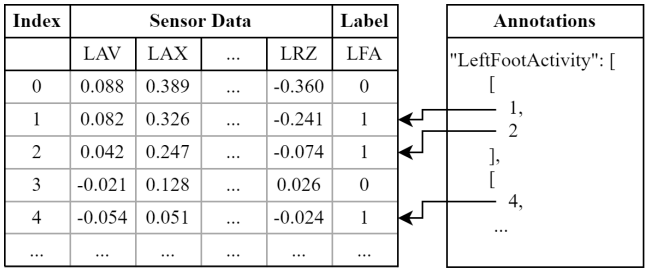

Figure 1. Conversion of the annotations to binary masks

As the TS recorded for different patients vary greatly in length and the UeberNet is not capable of dealing with differing input sizes, every TS was divided into overlapping segments of fixed length. The segment dimensions and overlap can be dynamically adjusted by varying parameters such as the length, the offset and the sample frequency. Additionally, as not every TS has a length, that is a multiple of the segment length, a padding mechanism is provided. Post-prediction the segments were merged back together and the labels converted to the annotation format for evaluation. The MD variables contained in both data sets were translated in an arbitrary order into a linear vector of fourteen variables. Scalar values were not altered, while discrete values were encoded as -1, 0, 1 and mutually exclusive relationships represented as one-hot encoded vectors.

2.2 Training and evaluation data

To validate the performance of the model on unseen data during training, the labeled data set was divided into a training and a validation set. Since the evaluation set provided by the competition host consists of entire TS, the labeled set was split into subsets of entire trials rather than using a time-based split. To ensure no detrimental effects were introduced by the split, a high similarity between the data sets was confirmed by analyzing the distribution of the patients’ characteristics (e.g. age) in either set (s. Fig. 2, Fig. 10).

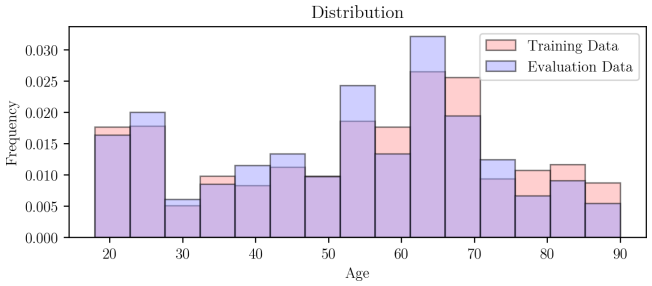

Figure 2. distribution in the training and validation data sets

2.3 Data normalization and augmention

As the different channels of the TS describe different values that naturally vary in their value range, normalization is used to prevent differences in the weighting of the channels during training [16]. For this purpose, the data normalization functions from [2] were applied and analyzed. The best function for the given problem was determined by conducting a cross-validation (CV) each (s. Tab. 1). Based on these results the "sigmoid normalization" (4a) and the Tanh estimator (4b) were found to perform best. The MD on the contrary, was normalized using just one method, a relaxed version of min-max-rescaling and mean-standard-deviation-normalization to map the majority of the values to a range of -1 and 1 and a mean of 0.
The training data is augmented using convolutions, noise, dropout and time warps provided by the python package "tsaug". The parameters for the the augmention were tuned by hyperoptimization and are displayed in figure 11.

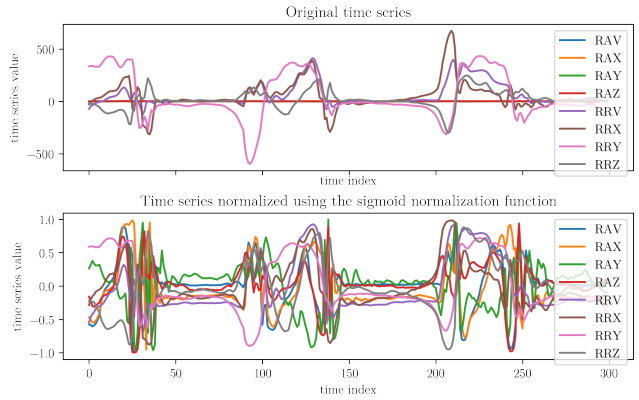

Figure 3. Original and normalized TS with eight channels in comparison.

2.4 Practical aspects

The data generation for model training and evaluation is realized by the *BatchGenerator*-class, that inherits from the *keras.Sequence*-class. The BatchGenerator loads the CSV- and JSON-files from storage and dynamically generates the training and validation batches. It can be parameterized to alter the usage of MD, the normalization modes, the sampling frequency etc. and provides functionalities such as multiprocessing and lazy loading of data to ensure scalability.

3 UeberNet

The choice of the model is based on the idea introduced by Wen and Keyes [16] that TS anomaly detection shares many aspects with image segmentation. The problem of detecting and labeling steps in a TS can therefore be formulated as follows: The TS constitutes a multi-channel 1-dimensional image, in which the steps appear as different segments of anomalous behavior. A **U-Net**-like network [16] allows for predicting a multi-channel mask of probabilities on the TS, that describes the likelihood of one "pixel" of the TS belonging to one or several anomaly classes, i.e. the left and right footstep.
The proposal of Wen and Keyes [16] is an adaption of the U-Net by Ronneberger et al. [10] for TS, that was originally developed for medical image segmentation. Although little research has been put towards furthering the TS specific application case of the U-Net, lots of different variants have been developed for the image segmentation use-case [12]. All these variants are usually based on the original U-Net and introduce an additional architectural mechanism to the network structure that outperforms the unmodified U-net.

In order to find the best possible model to detect steps in TS data the original TS U-Net [16] was gradually enriched with additional architectural features that can be evaluated individually and collectively during hypertuning forming the UeberNet (s. Fig. 4). Since all the following features were originally developed for use with images they were adapted to work in the TS context of the task.
The UeberNet differentiates between two level of architectural decisions. The *base-architecture* determines the structure of the network, i.e. the number and placement of the nodes, and must be either **U-Net** [16] or **U-Net++**, following the work by Zhou et al. [18]. Additional *sub-architectures* are all optional, compatible in any combination and affect the behavior of the network on a node-level. The four sub-architectures are the **Attention** gate [8], the **Dense** structure [5], the refined **Inception** module [13] and the **Residual** principle [4].

The U-Net itself (thick lines in Fig. 4) consists of three major elements: the encoder, the decoder and the skip connections. The encoding path (black), also referred to as backbone, encodes the original multi-channel TS using an alternating sequence of $i_{\max}$ convolutional nodes and down-sampling operations. The length of the TS representation decreases fourfold for each "node layer" i of the U-Net, whereas the depth,

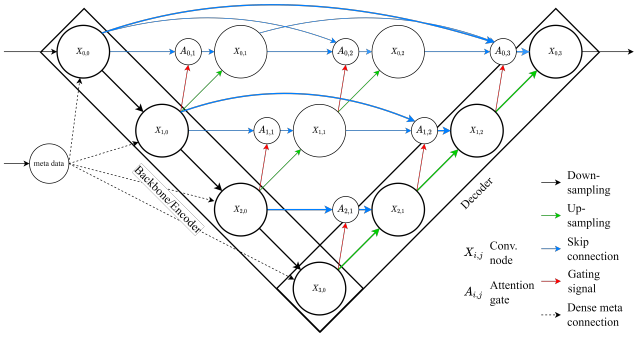

Figure 4. UeberNet

i.e. the number of channels/feature maps in the representation increases two-fold. The decoding path (green) inverts these operations by upsampling the layers' output by a factor of four and applying convolutions with a halving number of filters expanding the representation again while increasing its shallowness. This encoder-decoder-architecture becomes the U-Net by means of the skip connections (blue) that propagate contextual information in form of feature maps from previous layers to posterior layers of the same dimensions greatly increasing the performance on predictions.
To generalize on the complexity of the TS U-net [16], the UeberNet approach allows for any $i_{\max} \geq 2$ to be chosen as the number of nodes in the backbone. Additionally, the base-filter count f and the number of consecutive convolutions per convolutional node n may be altered (s. Fig. 5).

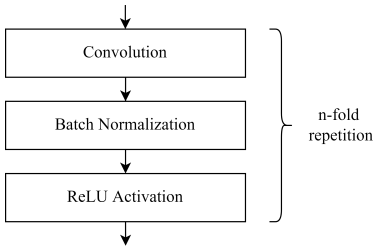

Figure 5. Generalized convolutional node $X_{i,j}$ that contains n convolutional modules with $2^i f$ filters each

This standard U-Net can be improved upon by introducing additional convolutional nodes along the skip-connections leading to the **U-Net++** [18]. The nested structure improves the flow of information from left to right. Each additional node $X_{i,j}$ takes the upsampled input from node $X_{i+1,j-1}$ as well as the output from all previous nodes on the same layer i and concatenates the filter maps before performing the operations of the regular convolutional node. Choosing $n = 1$ for the UeberNet, the following equation (1) [18] shows the different outputs of each node, where $\mathcal{H}$ denotes the convolution, $\mathcal{U}$ the upsampling operation and [] the concatenation.

$$X_{i,j} = \begin{cases} \mathcal{H}(X_{i-1,j}) & j = 0 \\ \mathcal{H}([[X_{i,k}]_{k=0}^{j-1}, \mathcal{U}(X_{i+1,j-1})]) & j > 0 \end{cases} \quad (1)$$

The additive **Attention** [8] gate for U-net-based networks introduces the notion that different segments of the output of the encoding path are of different relevance when decoding the information. To decide on the importance of different "pixels" a gating signal g is required. For a node $X_{i,j}$ on the decoding path of the U-Net this is the output of the node $X_{i+1,j-1}$ (s. Fig. 4, red). Fig. 6 shows the attention gate adapted for a 1-dimensional channel-last setting, showing the different dimensions of the tensors. Due to the difference in the channel dimensions, both inputs are first transformed linearly to an intermediate dimension $\frac{l_x}{2} \times 2^i \phi$ with $\phi \in [1, 2]$. Resampling the calculated value using bi-linear interpolation allows for resizing the gating vector α to rescale the value signal by calculating $\hat{X} = \alpha X$.

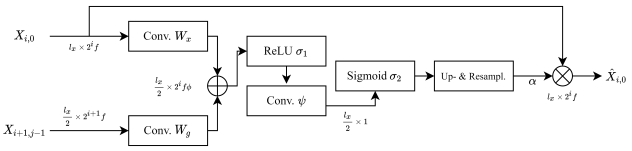

Figure 6. Adapted attention gate

$$\alpha = \sigma_2(\psi^T + \sigma_1(W_x^T X_{i,0} + W_g^T X_{i+1,j-1} + b_g) + b_\psi) \quad (2)$$

Introducing additional connections in between convolutions to form a **Dense** [5] U-Net increases the resolution, the network is capable of handling, and improves performance by providing every posterior layer with contextual information from all previous layers within the same node. Considering a node $X_{i,j}$ with $n = 5$, the node consists of 5 modules, i.e. a convolutional or inception module. The following equation describes the dense logic by defining the output of each module with $k = 1, ..., n$.

$$X_{i,j,k} = \mathcal{H}([X_{i,j,m}]_{m=0}^{k}) \quad (3)$$

The **Inception** [13] mechanism allows for constructing deeper CNNs, while simultaneously preventing overfitting, retaining efficiency and allowing for differently sized feature structures. The inception mechanism used for the UeberNet is a 1-dimensional adaption of the refined GoogLeNet-inception module by Szegedy et al. [13], that can be seen in Fig. 7. To generalize the concept however, the kernel/pool size k for the different operations can be chosen freely. This allows for easier adaption to the TS-context of the context that might favor larger kernels to represent temporal relationships. If used in the UeberNet the inception module replaces one block of operations as seen in Fig. 5.
The deep **Residual** [4] mechanism aims at improving the

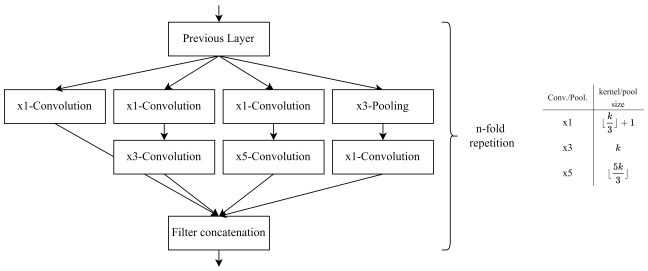

Figure 7. Inception module [13] adapted for TS with a variable base-kernel size k

performance and learning process of deep neural networks by addressing the degradation problem. For this purpose additional connections are introduced that skip consecutive convolutional/activation blocks and are added to the output of these skipped layers. To incorporate this idea into UeberNet (with $n \geq 2$) these additional skip-connections are introduced on a node-level, i.e. pairs or triplets of convolution/inception modules within one node $X_{i,j}$ are bypassed (s. Fig. 8) and subsequently the outputs added.

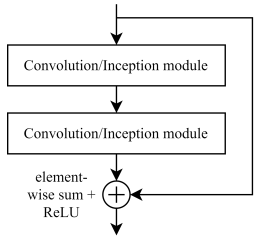

Figure 8. Logic of deep residual CNNs [4]

As no suitable previous work was found on including metadata in CNNs, a novel concept was developed. The MD is fed to the network by expanding the MD vector by means of a dense layer with as many neurons as the respective size of the TS for that node level and subsequently treated as an additional channel. Dropout layers are added after any node in the network and specifically before and after the MD expansion.

The model was trained on all 16 sensor channels on segments of length 1024 predicting both foot activations at the same time using binary crossentropy, dice [17] and tversky loss [11] with varying values for β. The neural network was programmed using tensorflow (tf) v2.8.0 [14] and the Keras API [3]. Custom layers, e.g. the attention gate, and frequently reused layer groups, e.g. the inception module, are realized in separate classes inheriting from *keras.Layer*. The custom loss functions were programmed using tf-operations.

4 Results

There are two elementary components to hypertuning: Selecting the different parameter combinations in a meaningful manner and allocating resources to the different trials to maximize output. To realize these two concepts the library ray [7] was used. To conduct the search of the hyperparameter space choosing new trials based on past experiences hyperopt [1] was employed, which supports discrete and conditional search spaces. The modified F1-score, provided by the competition host calculated on the unseen validation data was selected as target metric. Additional metrics, such as precision and recall, were monitored. The resource allocation to different trials was performed using Asynchronous Successive Halving Algorithm (ASHA), an algorithm that allows for parallel scheduling and early-stopping of trials. The best findings were then selected to be validated using a 5-fold CV with random splits of 80% training and 20% validation data.

The best selection of parameters are included in the appendix and the repository. Some combinations of hyperparameters, especially those, that contained inception as a sub-architecture could be shown to outperform other combinations during hypertuning, but proved to be highly unstable during CV. This can be explained due to the high amount of additional layers and difficulties with weight initialization.

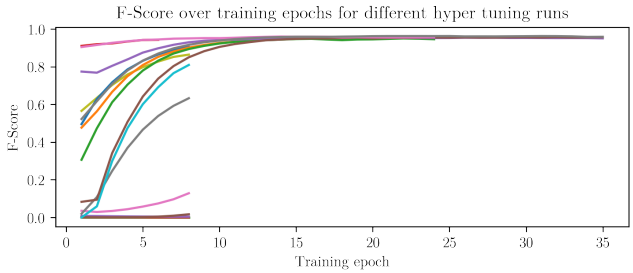

Figure 9. Performance of 16 of 912 runs during hyper optimization.

By analyzing the predictions of the model, recurring error patterns such as splits in steps were identified. This was corrected by post-processing the predictions in which two separate steps predicted by the model were merged if they were below average in size and relative distance to one another. The score on the validation data was improved by about 0.7 percentage points.

5 Future Work

The expansive hyperparameter space, that allows for configuring almost every property of every layer per node separately was not searched fully, but rather the searched properties were applied globally. Future optimizations could target subsets of nodes in the network separately, or analyze the effects of altering yet to be changed parameters, to achieve

even better results. Considering the broader scope of the network architecture additional research regarding the MD could be conducted.

In regards to future applications of the findings it must be noted, that being able to recognize the duration and frequency of steps is only a first step on the way to analyzing the human gait to e.g. develop early warning systems for neuro-degenerative pathologies. Further algorithms and methods can build on greater amounts of data, that can accurately be labeled using the UeberNet rather than manual annotations by experts. This opens up the possibility of e.g. developing medical aid systems based on smartphone sensor data.

References

[1] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2015. Algorithms for Hyper-Parameter Optimization. *Computational Science & Discovery* 8, 1 (2015), 9.

[2] Samit Bhanja and Abhishek Das. 2018. Impact of data normalization on deep neural network for time series forecasting. *arXiv preprint arXiv:1812.05519* (2018).

[3] Chollet, François and others. 2015. Keras.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Las Vegas, NV, USA, 770–778. https://doi.org/10.1109/CVPR.2016.90

[5] Martin Kolarik, Radim Burget, Vaclav Uher, and Lukas Povoda. 2019. Superresolution of MRI Brain Images Using Unbalanced 3D Dense-U-Net Network. In *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, Budapest, Hungary, 643–646. https://doi.org/10.1109/TSP.2019.8768829

[6] Xin Liu, Ning Li, Geng Xu, and Yonggang Zhang. 2021. A Novel Robust Step Detection Algorithm for Foot-Mounted IMU. *IEEE Sensors Journal* 21, 4 (Feb. 2021), 5331–5339. https://doi.org/10.1109/JSEN.2020.3030771

[7] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. 2018. Ray: A Distributed Framework for Emerging AI Applications. arXiv:1712.05889 [cs, stat]

[8] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y. Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. 2018. Attention U-Net: Learning Where to Look for the Pancreas. *arXiv:1804.03999 [cs]* (May 2018). arXiv:1804.03999 [cs]

[9] Minh H Pham, Morad Elshehabi, Linda Haertner, Silvia Del Din, Karin Srulijes, Tanja Heger, Matthis Synofzik, Markus A Hobert, Gert S Faber, Clint Hansen, et al. 2017. Validation of a step detection algorithm during straight walking and turning in patients with Parkinson's disease and older adults using an inertial measurement unit at the lower back. *Frontiers in neurology* 8 (2017), 457.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs]* (May 2015). arXiv:1505.04597 [cs]

[11] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. 2017. Tversky Loss Function for Image Segmentation Using 3D Fully Convolutional Deep Networks. In *Machine Learning in Medical Imaging*, Qian Wang, Yinghuan Shi, Heung-Il Suk, and Kenji Suzuki (Eds.). Vol. 10541. Springer International Publishing, Cham, 379–387. https://doi.org/10.1007/978-3-319-67389-9_44

[12] Nahian Siddique, Sidike Paheding, Colin P. Elkin, and Vijay Devabhaktuni. 2021. U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications. *IEEE Access* 9 (2021), 82031–82057. https://doi.org/10.1109/ACCESS.2021.3086020

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper with Convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Boston, MA, USA, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

[14] TensorFlow Developers. 2022. TensorFlow. Zenodo. https://doi.org/10.5281/ZENODO.4724125

[15] Charles Truong, Rémi Barrois-Müller, Thomas Moreau, Clément Provost, Aliénor Vienne-Jumeau, Albane Moreau, Pierre-Paul Vidal, Nicolas Vayatis, Stéphane Buffat, Alain Yelnik, Damien Ricard, and Laurent Oudre. 2019. A Data Set for the Study of Human Locomotion with Inertial Measurements Units. *Image Processing On Line* 9 (Nov. 2019), 381–390. https://doi.org/10.5201/ipol.2019.265

[16] Tailai Wen and Roy Keyes. 2019. Time Series Anomaly Detection Using Convolutional Neural Networks and Transfer Learning. *arXiv:1905.13628 [cs, stat]* (May 2019). arXiv:1905.13628 [cs, stat]

[17] Rongjian Zhao, Buyue Qian, Xianli Zhang, Yang Li, Rong Wei, Yang Liu, and Yinggang Pan. 2020. Rethinking Dice Loss for Medical Image Segmentation. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, Sorrento, Italy, 851–860. https://doi.org/10.1109/ICDM50108.2020.00094

[18] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. 2018. UNet++: A Nested U-Net Architecture for Medical Image Segmentation. *arXiv:1807.10165 [cs, eess, stat]* (July 2018). arXiv:1807.10165 [cs, eess, stat]

A Normalization functions

The following normalization functions were used as hyperparameters: Sigmoid normalization (4a), tanh estimator (4b) and min-max normalization (4c).

$$x_{\text{sig}}(x) = \frac{1}{1 - e^{-x}} \tag{4a}$$

$$x_{\tanh}(x) = 0.5[\tanh[\frac{0.01(x - \mu)}{\sigma}] + 1] \tag{4b}$$

$$x_{\text{min-max}}(x) = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{4c}$$

where:

σ = standard deviation of x

μ = mean value of x

B Meta data

A detailed review of the distribution of meta data variables across the training and evaluation set is shown in Fig. 10.

C Hyperoptimization

The performance of runs across all tested hypervariables is shown in Fig. 11 and 12, where the results of all runs is compared to the best fifth. A detailed overview including the value ranges of the hypervariables and the best found configuration for the UeberNet are explained in Tab. 2. The hypervariables relevant to the general training procedur can be found in Tab. 3.

Score	Original	Min-Max Norm.	Tanh Estimators	Standard Score	Median Norm.	Sigmoid Norm.	Decimal Scaling
Average	0.9441	0.9481	0.9482	0.9453	0.9442	0.9481	0.9482
Standard deviation	0.0083	0.0047	0.0059	0.0072	0.0085	0.0047	0.0059

Table 1. Cross validation result with different normalization settings.

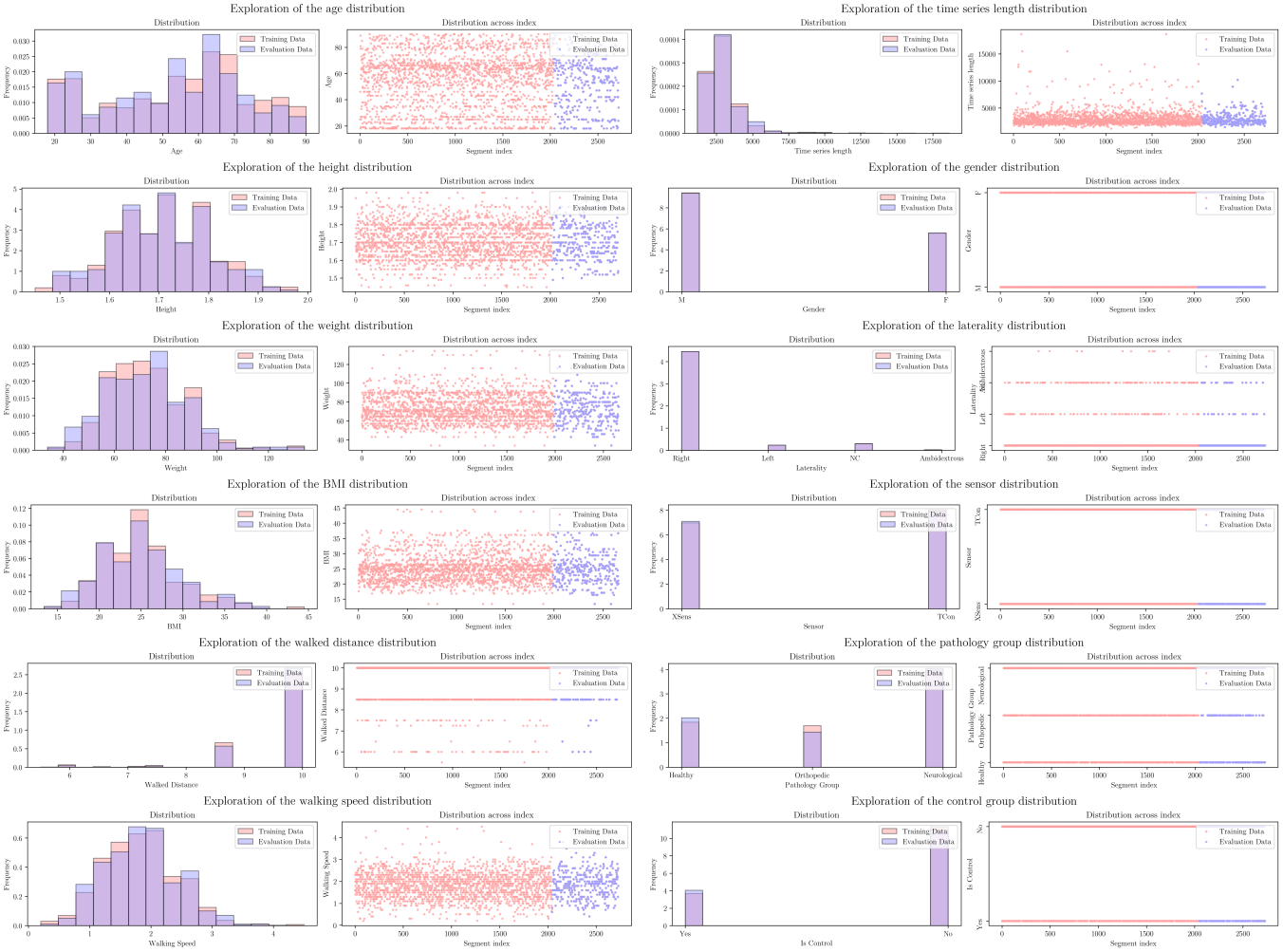

Figure 10. Metadata exploration for determination of training and validation data set.

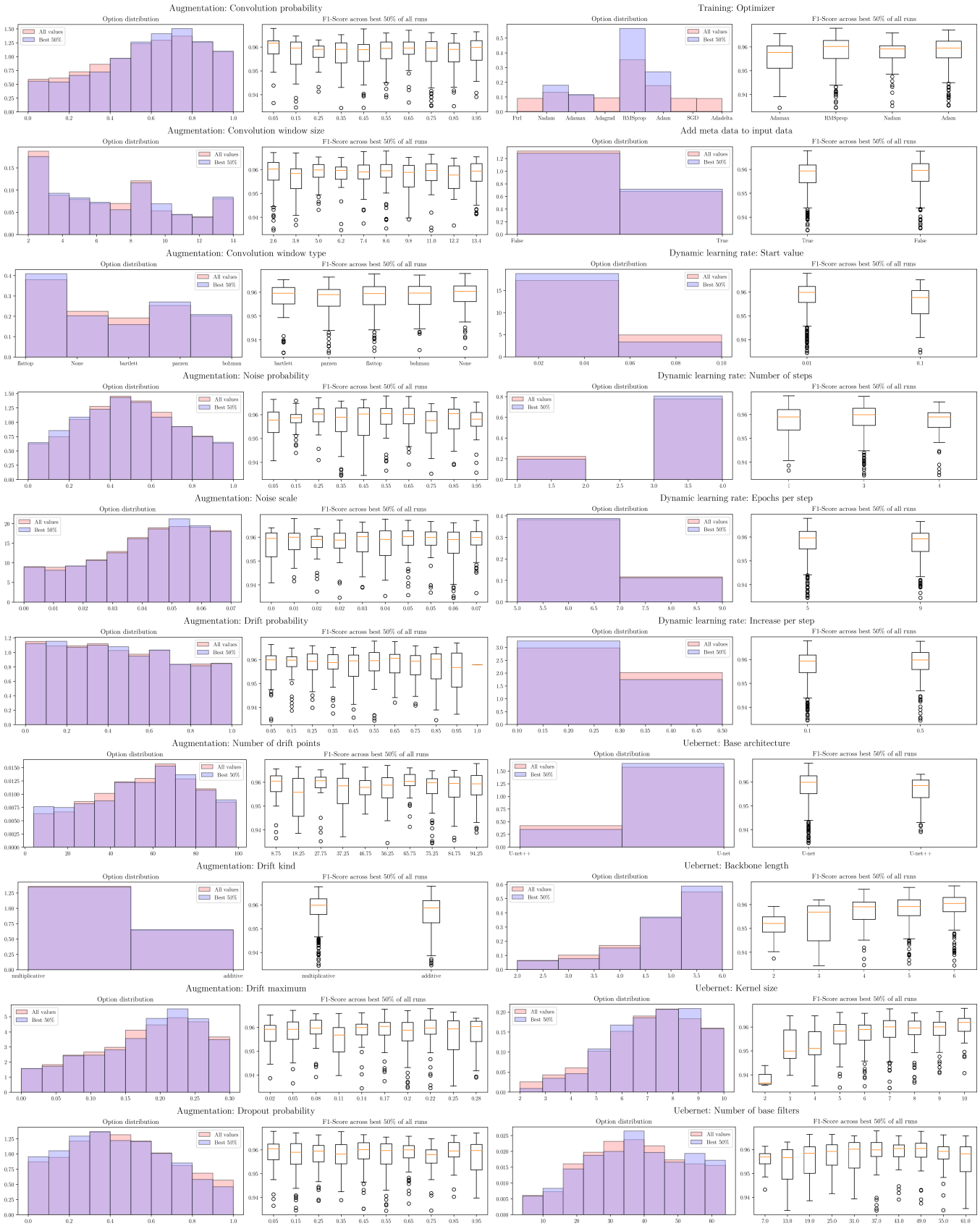

Figure 11. Evaluation of different hyperparameters (1)

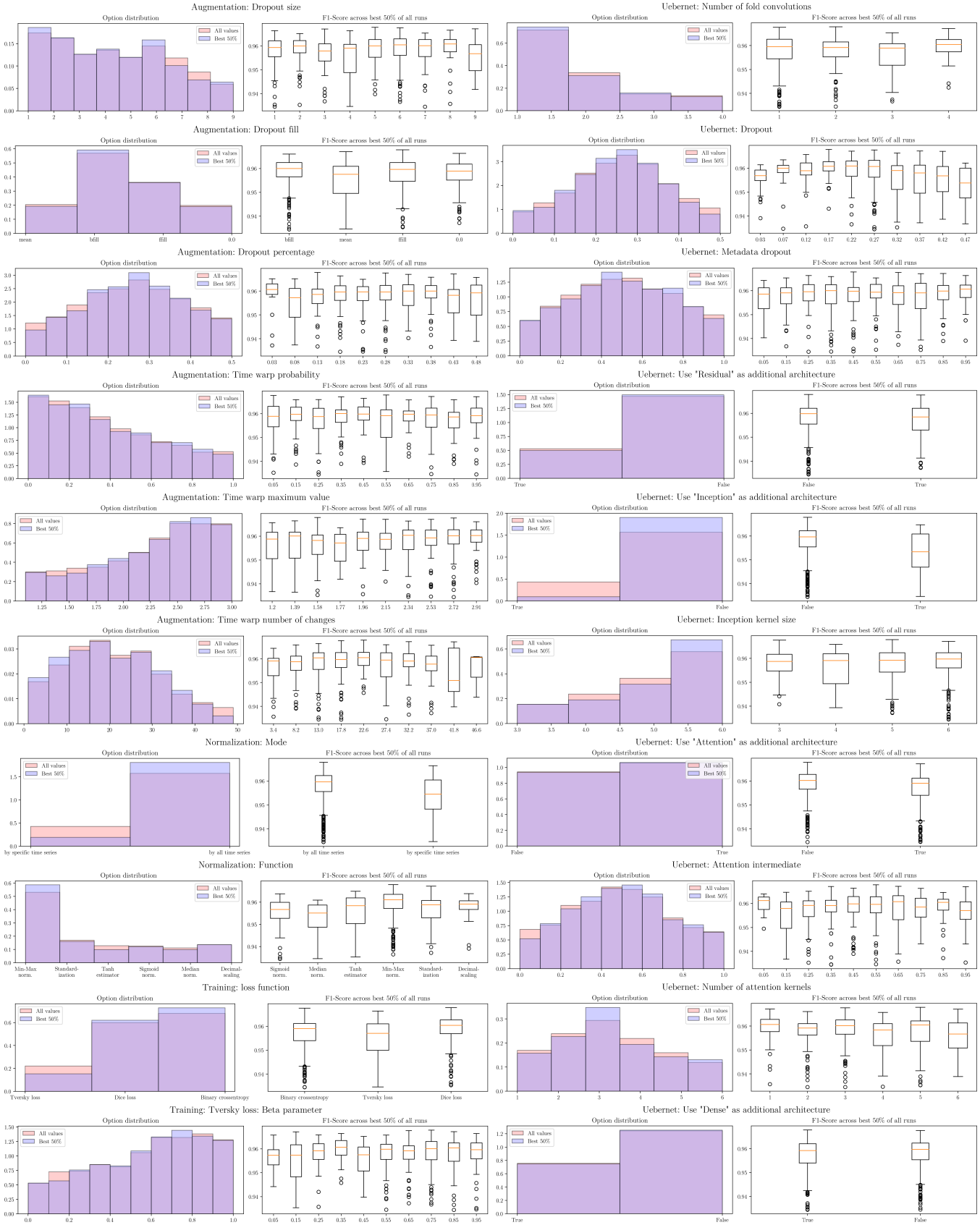

Figure 12. Evaluation of different hyperparameters (2)

parameter	description	value range	best
base architecture	number and placement of nodes	{U-Net, U-Net ++}	U-Net
backbone length	dimension of network by number nodes	{2, ..., 6}	5
kernel size	kernel size of all convolutions in the network	{2, ..., 10}	9
base filter count f	determines the number of filters of the convolutions per layer	{4, ..., 64}	29
n-fold convolutions	number of successive modules per node	[1, ..., 5]	1
dropout	dropout rate of all post-node dropout layers during training	[0, 1]	0.34
sub-architectures	sub-architectures in use	$\subseteq$ {attention, dense, inception, residual}	{dense, attention}
inception kernel-size k	kernel dimension in the inception module	{3, ..., 6}	6
attention intermediate ϕ	factor for intermediate dimension used by the attention gate	[1, 2]	0.63
attention kernel size	size of the resampling kernel of the attention gate	{1, ... , 6}	5
meta data	whether to use MD in the network	{False, True}	True

Table 2. Explored Hyperparameters for the UeberNet-architecture

parameter	description	value range	best
loss function	loss function used for training	{BinaryCrossentropy, dice loss, tversky loss}	tversky loss
tversky β	value of β for the tversky loss with $\alpha = 1 - \beta$	[0, 1]	0.91
optimizer	optimizer used for training	{Adadelta, Adagrad, Adam, Adamax, Ftrl, Nadam, RMSprop, SGD}	Adamax

Table 3. Explored Hyperparameters for the training procedure