Humboldt-Universität zu Berlin
School of Business and Economics
Chair of Information Systems

Data efficient landmark localization using memory augmented neural networks

Master's thesis

for acquiring the degree of

Master of Science (M. Sc.)
in Information Systems

submitted by: Elias Baumann
Student no.: 587775

Examiners: Prof. Dr. Stefan Lessmann
Dr. Dagmar Kainmueller

Submission Date: 16.11.2020

Contents

1. Abstract

Biomedical image analysis frequently faces the issue of few available samples due to expensive or difficult data collection procedures. While deep learning often focuses on working with large datasets, for automated analysis of biomedical images, methods are required which are able to learn from few samples. In this work, iterative learning of landmarks with a single U-Net is proposed. The iterative task structure includes implicit memory as the prediction from the previous step is used in the next. This work shows that the thereby included recurrent inductive bias can improve landmark localization. In a second step, an explicit memory module in form of a neural turing machine gate is added and compared to attention gates using the same model architecture. By extending a baseline U-Net with a neural turing machine gate or attention gate, localization error can be further decreased both on small datasets with less than 500 training samples and on an even smaller subset of less than 50 samples.

2. Introduction

Landmark localization is a common computer vision problem where locations of predefined keypoints are learned and automatically detected within 2D or 3D images. Apart from allowing quantitative evaluation of image contents, landmark localization has multiple different applications such as pose estimation (e.g. Pfister et al. [41]), facial recognition (e.g. Zhang et al. [60]) and bounding box placement (e.g. Criminisi et al. [13]). In the biomedical imaging domain, landmark localization attempts to automate the manual task of annotating a large number of images which is error prone and time consuming. Landmarks should be localized as accurate as possible, both individually and in relation to other landmarks since results are supposed to be used in decision making in clinical environments and to quantitatively evaluate experiments in the biological domain. Convolutional neural networks have proven to be successful at providing accurate landmark localization results on a number of datasets, however performance decreases when the dataset size decreases [40]. In the biomedical imaging domain, datasets are frequently small, with some containing only 10 annotated images as the process of obtaining samples can be challenging. Furthermore, biological objects frequently contain similar or repeating structures which makes robust predictions difficult. This can also lead to misidentification of landmarks which is detrimental to any automated evaluation [39]. To solve issues with repeating patterns as well as few available samples, localization strategies try to incorporate shape constraints or other prior knowledge in their models so that the machine learning model does not have to search the entire image space but instead already knows some relative placement information [12]. Apart from directly incorporating the complete general shape, an iterative task structure where landmarks have to be predicted one by one or in groups can help the model focus on new locations. Using information about previous landmark locations the model can create an internal shape representation to better predict the remaining landmarks [34]. Furthermore, training a model on a iterative task seems to be beneficial in other adjacent domains

such as semantic segmentation [54].

In this work a U-Net [44] based iterative approach to predicting landmarks on biomedical images is presented. For every step, the network predicts a group of landmarks and then receives its own prediction as input for the next step thereby retaining some information from the past. The hypothesis of this work is that a consistent memory can assist a neural network in preserving previous information and making more informed predictions in following steps. Therefore, a baseline U-Net which receives a single past step as input is evaluated. Then a neural turing machine (NTM) [19] memory component is introduced, so that the network is able to selectively include more information from steps further in the past. Attention gates [38] are also included in the comparison because of potential benefits in selecting appropriate parts of the past step and high similarity in overall structure to the NTM gates. Two different datasets are evaluated: the cephalometric X-ray challenge dataset (Cephal) [52] which has been extensively evaluated on landmark localization and an unpublished dataset containing images of drosophila melanogaster wings with landmarks for automated measurements (Flywing). Given that the task structure puts focus on relative landmark positions, small datasets could particularly benefit from this approach. Thus additional experiments using a small subset of the complete training data are evaluated. In summary, the contributions of this work are:

- Proposal and successful application of an iterative task structure for landmark localization using a single regressor with results on artificially reduced training data
- Evaluation of neural turing machines on a landmark localization task and comparison to attention gates

This work is structured as follows: First, existing related work is introduced and details on methodology are explained. Afterwards, the experimental setup as well as implementation details are presented. Finally, results are evaluated and discussed.

3. Related Work

3.1. Landmark Localization

Landmark localization, also commonly referred to as keypoint localization/detection is a subfield of computer vision which focuses on finding specific pre-defined points (e.g. landmarks) in an image or a 3D volume in order to identify and measure what is visible in the image. It is further differentiated into individual landmark detection and multiple landmark detection. Individual landmark detection reduces the problem to finding only one specific landmark, whereas multiple landmark detection attempts to localize many landmarks. This work will focus on multiple landmark detection for a single instance. Frequently, landmark detection, in particular for medical images, is done using one of the two following strategies: The first strategy revolves around models using global shape information and frequently also re-aligning points of a shape to match the target. These models benefit from low variation between shapes as fewer adjustments are needed for

good results making them viable for medical images as the captured anatomy often does not vary strongly. In particular, early machine learning methods for landmark localization are based on this approach. Active shape models by Cootes et al. [12] try to find examples of a shape within an image by capturing shape variation and realigning to new instances. These shape models are based on landmarks describing the shape and provide average landmark positions with variables controlling the variation within a training set. This method requires an initial guess which can be derived from an existing labeled dataset and images have to be pre-aligned. Liu et al. [34] present a technique combining multiple "single-landmark" detectors iteratively by selecting an appropriate search space within the image. Search spaces are defined by landmarks already detected and the next landmark to be detected is chosen by the lowest size search space. This iteratively reduced area based on the already predicted landmarks reduces the overall computational cost of the individual detector but results always depend on previous detectors. They train multiple detectors for every landmark at different resolutions to refine predictions. The work of Liu et al. [34] includes the core idea of making use of other already predicted landmarks which will be used in iterative experiments as well. However, they use independent classifiers for every iteration and argue that by using this geometric cascade according to Viola and Jones [51] there is no error propagation to later stages. In this work, a similar task structure is applied, however the model for every landmark prediction stays the same. The second strategy relies on markov random fields or hidden markov models to adjust detected landmarks to a shape. Possible centroid locations are generated by different strategies and a machine learning model then selects and refines location proposals to fit a shape prior [16, 14, 15].

A different and recently highly popular approach to image analysis uses deep convolutional neural networks (CNNs). First introduced by LeCun et al. [31] convolutional neural networks have reusable weights and biases applicable across image locations. This neural network was trained using a loss based gradient iteratively updating network weights and biases and could identify handwritten digits. However, only with successful training of deep neural networks [32] and sufficiently advanced computer hardware, CNNs gained traction in image analysis and landmark detection more specifically. Particularly relevant for this is the U-Net convolutional neural network architecture by Ronneberger et al. [44]. They introduce a U-shaped model consisting of a contracting and an expansive path connected by a bottleneck all consisting of convolutional layers. This U-Net architecture is capable of learning image features at multiple resolution scales and combining information for arbitrary image analysis tasks. In their work they use the network for image segmentation, yet the architecture can be adjusted for tasks like landmark localization, bounding box regression or any other image analysis task. While this work focuses on bio-medical images, landmark detection is frequently applied to other research domains such as facial recognition, pose estimation and even object detection. Using deep learning, a number of works have shown great success in predicting faces and poses fast and accurately. Newell et al. [37] develop a stacked hourglass model which is a fully convolutional model consisting of hourglass modules with contracting and expansive paths similar to U-Net. Their goal is to estimate human poses by predicting joint landmarks using individual heatmaps for every landmark with argmax to obtain the

final coordinate. More recent work like Alp Güler et al. [4] utilize landmark localization to predict human poses, but combine the information with mask predictions to create a multi-task problem and jointly refine the final pose. In facial recognition, Zhang et al. [60] predict landmark coordinates directly but rely on additional utility tasks to improve prediction performance. In the same way, Chen et al. [9] predict coordinates directly, but refine their proposals with additional steps. Chen et al. [8] and Liao et al. [33] propose facial recognition specific landmark localization methods which both utilize centroid prediction and refinement of potential candidates. While not all advances from facial recognition can be applied to bio-medical image analysis, the vast number of methods can be adapted to and utilized in the domain.

3.1.1. Heatmap Regression

While works like [8, 33] rely on finding centroids for potential landmark locations, recent work focuses on regressing heatmaps [57, 40, 37, 47] where high values in the heatmap represent likely landmark locations. Usually, such heatmaps are generated by centering a multivariate normal distribution on a landmark coordinate and adjusting the standard deviation as a tuned or learned parameter. This approach can be extended to N-D data. Using heatmaps as labels, one can build a fully convolutional pipeline to do end-to-end training, and the proposed heatmaps can further be refined or simply be used directly as predictions. It is important to note, that while a distribution is used to generate the heatmap, the pixel values do not reflect any sort of probability but instead are only set to make the label less sparse. The alternative method would be to set the pixel which is referenced by the landmark coordinate to e.g. 1 and any other pixel to 0. This would lead to extremely sparse labels e.g. 1 out of $256 * 256 = 65.536$ which machine learning models likely fail to learn or at least provide less accurate results (e.g. [53]). Heatmap regression is an alternative approach to directly regressing landmark coordinates and stands in opposition to classification in works like Donner et al. [14] as continuous values are predicted for all pixels. In the context of deep neural networks, Tompson et al. [47] demonstrate the effectiveness of heatmaps as regression targets for CNNs, altough they separate their images into patches and also focus on human pose estimation instead of bio-medical images. Payer et al. [39] present a CNN based landmark detector for biomedical images. They successfully demonstrate the applicability of a U-Net to the problem of landmark detection by using heatmap regression. They extend a CNN predicting landmark heatmaps with a spatial configuration block, which is a convolutional block predicting relative landmark positions for every landmark. In [40] they improve on their previous work by using a U-Net like architecture for their first stage landmark prediction and change the objective function to include learning the standard deviation parameter σ for the distribution used in heatmap generation. Yang et al. [57] show a similar approach to [40] by using a encoder-decoder U-Net-like architecture to regress heatmaps, but they use the output of every resolution level and apply a loss function which includes a loss term for every level. Afterwards, they concatenate all levels and use it for prediction and an additional refinement step with a markov random field (MRF).

3.2. Data Efficient Learning

Collecting and labeling large amounts of data is expensive and takes a considerable amount of time when done manually [18]. Therefore, methods are needed for problems where little data is available. Traditionally, neural networks struggle when only few samples are available as they are unable to capture generalising information [42]. However, a number of recent works have focused on the problem of limited available training data. Payer et al. [39] consider the issue of limited data in their work and demonstrate some success with their approach by including an additional localization bias. Zhang et al. [59] make limited availability of training data their core problem and present a patch-based two-stage model which relies on pre-aligned images. Their first stage predicts landmark displacements on multiple image patches where not all landmarks are visible. Displacements of further away landmarks are weighted less in the loss function as they are harder to predict. Patches are based on a predefined grid and information of every individual patch is aggregated to jointly predict displacements. The patch-based learning automatically generates a large number of new samples and avoids any arbitrary augmentations which may not benefit the learning process of the model. A patch-based learning strategy also showed success in Hirsch et al. [22] in instance segmentation with low sample size. Another strategy for data efficient learning is few-shot learning: Few-shot learning is an approach where new goals are learned from few examples e.g. with information from other similar tasks (Meta-Learning) or by providing a structure which can adapt rapidly enough. Meta-learning, the process of learning to learn, can be applied if similar enough problems with more data are available. Ravi and Larochelle [43] develop a meta-learning model which uses a Long Short-term Memory (LSTM) neural network to learn an optimization algorithm as well as initialization for the actual classifier. Their work shows potential in learning new classes with few samples, however they perform similar to Vinyals et al. [50] and Koch et al. [28] both using methodology more akin to metric learning to learn from few examples. In a more recent work, Wang et al. [54] propose a recurrent U-Net which iteratively learns segmentation maps which are then combined for the final prediction. They apply their model to a small training set and accurately predict segmentation maps. In a similar fashion, this work uses an iterative training procedure to create a combined prediction for all landmarks.

3.3. Memory Augmented Neural Networks

A potential solution to dealing with few samples is to memorize important information about the task itself or target shape. Santoro et al. [45] propose a one-shot learning model based on memory augmented neural networks (MANN) which are tasked to learn N random classes per episode from the Omniglot dataset [29] with the test data being new random classes not represented in the training data. These memory augmented neural networks rely on an external memory matrix with an attention mechanism to read and write from and to the memory matrix. Santoro et al. [45] adapt the work of Graves et al. [19] and develop an additional similarity metric based on memory areas least used, which is included in the read and write selection. The original work by Graves et al. [19]

expands recurrent neural networks (RNN) to turing completeness by adding a heap which is represented by an external memory matrix and an attention mechanism for memory i/o. They successfully demonstrate the neural turing machine, a fully differentiable model with long term memory and show different simple tasks to prove the effectiveness of their model at typical computer tasks such as sorting values in a vector or repeatedly copying a sequence a specified number of times. Continuing on the idea Graves et al. [20] propose the differentiable neural computer (DNC). Here they address some issues with the neural turing machine, namely how memory is accessed, how information about already edited locations is stored and they add the function to also free up memory locations. However, these benefits also come with a more complex model and more parameters. Weston et al. [55] also show a possible implementation of a memory augmented machine learning model with a manually definable memory selection strategy and an output based on another manually defined function based on memory and input. However, their approach provides less memory modification methods than [19] and [20], yet it matches the core idea. The work of Graves et al. [19] proved to be difficult to train and a number of questions arose regarding memory initialization and modification strategies, but Collier and Beel [11] managed to provide a working implementation which this work is also based on. The difficulty to implement and train the NTM likely contributed to a lack of research using the model, whereas simpler to implement attention mechanisms [49] found high acceptance. More recently, Tran et al. [48] extend Vinyals et al. [50] and propose a matching network augmented with an NTM module [19] and demonstrate promising results on a few-shot image classification task. In another domain, Kim et al. [26] use a NTM for a generative adversarial network to generate video game frames with information from past generated frames for better time consistency.

A core idea of the NTM is its addressing mechanism, which is based on an attention distribution over the memory matrix. As the approach therefore is highly similar to attention, it is important to also regard the attention mechanism as a comparison to the more complex NTM. Attention is a differentiable mechanism which is supposed to emphasize certain parts of an input and de-emphasize other parts, therefore letting a neural network focus on certain areas of data instead of always regarding every input value with the same importance. Attention is used in [50, 19] to find a best matching embedding. It has previously mainly been used in machine translation e.g. [6, 36]. [49] further improved on attention mechanisms and introduced the popular transformer model but also focus on machine translation and natural language. Attention is also used in image analysis with Xu et al. [56] creating image captions using attention to find the most relevant objects in an image which have to be mentioned in the caption.Zhang et al. [58] implement a self-attention mechanism in their self-attention generative adversarial network to improve image synthesis as important parts of the image are more likely to be learned and synthesized. Oktay et al. [38] use an attention gate structure in a U-Net to weight outputs of every U-Net level before upsampling it to the next level in order to better solve an image segmentation task. This work uses the attention gate structure but replaces the gate with a NTM to create a model with long range information storage which iteratively learns landmarks using heatmap regression.

4. Methodology

The architecture in this work consists of multiple concepts which shall be introduced in the following:

- A U-Net baseline which is proven capable of solving a landmark localization task [39] is used as the primary feature extractor,
- The network uses heatmap regression to localize landmarks as successfully demonstrated by [39]. As recent work demonstrated success with an iterative task structure [52, 2], the multiple landmark localization problem is converted to the sequential prediction of landmark subsets.
- The U-Net baseline is extended by a NTM gate akin to Oktay et al. [38] to be able to store long range dependencies on past landmarks in memory and retrieve specific information when necessary.
- The NTM is placed in an encoder-decoder structure to read and write embeddings as there is currently no trivial way to convert the NTM definition by Graves et al. [19] to a fully convolutional model.

This model is compared to an attention gate U-Net [38] and a regular U-Net [44] to validate results.

4.1. U-Net

This section introduces the baseline architecture used throughout all experiments. The U-Net is a fully convolutional neural network (FCN) named after its U-shaped architecture [44]. The main idea behind a model structured like U-Net is to extract features in different resolutions and combine results. Moreover, different numbers of filters are used at different resolution levels as it is argued that there are fewer small patterns and more varying large patterns within a dataset. For example in Flywing, small patterns are contrast borders between pixel values which occur basically everywhere in the image, whereas large patterns might be entire parts of a wing and only appear a few times or not at all in an image. Ronneberger et al. [44] also develop this architecture specifically for low sample biomedical image analysis problems. The U-Net is defined by two paths, a contracting and an expansive path with a bottleneck level to complete the u-shape. The contracting path consists of first a repeated application of convolutional layers with small kernel size and a variant of a rectified linear unit (ReLU) activation function. Then, the output is downsampled using a pooling method (e.g. 2×2 max pooling or average pooling). At the next level, all feature channels are doubled. The number of levels can be defined and should, depending on the problem, be derived by a metric such as the receptive field of the bottleneck convolutional layer. At the lowest defined level, no downsampling is applied and instead the output is fed to the expansive path. This part of the model will be referred to as bottleneck. The expansive path is mirroring the contracting path to an extent. It first upsamples the ouput from the previous level,

concatenates it with to the output of the same level of the contracting path via skip connections and applies several convolutional layers. Therefore at any level, information is also passed between paths. The process is repeated until the top level is reached. As a final step, a 1×1 convolutional layer is applied which creates an output with the number of channels according to the target label. In Ronneberger et al. [44] this last layer creates an output with the number of channels being equal to the number of segmentation classes. In this work the output layer will predict one channel per landmark heatmap. There are two more differences to the original paper: The input of every convolutional layer is padded. As a result, skip connection outputs do not have to be cropped to match the upsampled output from a lower layer. Deconvolutional layers are replaced by nearest-neighbor interpolation upsampling to reduce the parameter count and fit all models into GPU memory while achieving similar performance (see e.g. [7]).

4.2. Heatmap Regression

Predicting landmark coordinates requires a convolutional neural network to learn a conversion from image to metric coordinates which adds additional complexity. Using heatmaps, landmark locations can be output directly with high values indicating potential landmark locations. As data is present in coordinate form, coordinates have to be converted to images by creating heatmaps. A landmark on a heatmap is represented by a multivariate normal distribution with its center at the coordinate. The maximum value of the heatmap is located at the landmark coordinates, but given that a normal distribution is not defined as a single point but as a distribution with mean and standard deviation, the heatmap also incorporates increasing values closer to the actual position. This work creates heatmaps using the probability density function of the multivariate normal distribution with a adjustable standard deviation scaling parameter. Thus, pixel values do not represent probabilities and can take values above 1. The scaling parameter for the distribution is learned in [40], but set manually in this work to reduce the number of free parameters in the model. A smaller scaling factor provides more accurate results but also requires longer training for acceptable results. The network is trained by using the sum of square distances loss (SSD) which sums over squared pixel differences between label and prediction (Equation 1). This means, that the network will optimize predictions for every pixel in the output, not just for pixels relevant for the landmark localization task and every pixels loss has the same importance.

$$\mathcal{L}_{ssd} = \sum^{i} \sum^{j} (\tilde{x}_{i,j} - x_{i,j})^2 \tag{1}$$

4.3. Neural Turing Machine

In order to access internal information on past predictions in an iterative landmark learning task, a memory module is useful to allow for specific storage and retrieval of such information when required. Here, a NTM [19] is used because of its higher flexibility in read and write operations than e.g. LSTM [23] and its ability to store large amounts of

data unaltered over time. This work will rely on publicly available NTM implementations by Collier and Beel [11] and snowkylin [46]. A NTM consists of a memory matrix $\mathbf{M}$, a controller C and multiple read and write heads $[h_{r1}, h_{r1}, ..., h_{w1}, h_{r2}, ...]$.

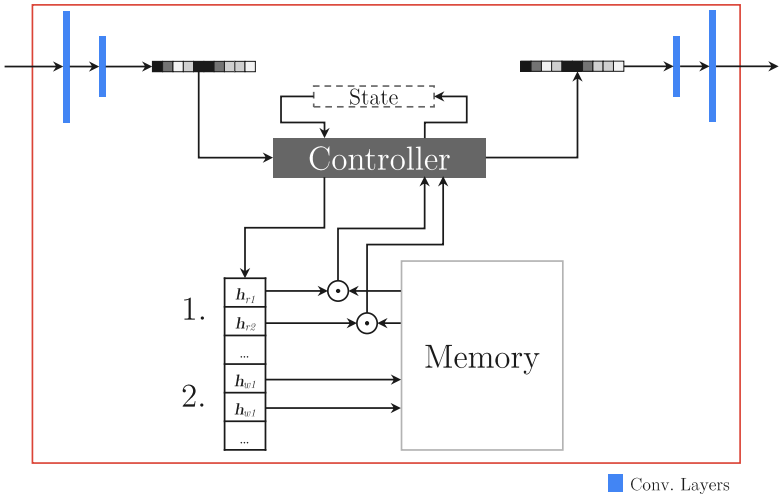


Figure 1: Overview of a NTM with a convolutional encoder encoding inputs into a vector and a convolutional decoder decoding the NTM output. Read operations are executed before write operations.

The memory matrix is a $N \times M$ matrix $\mathbf{M}_t$ where N is the number of memory locations and M is the size of each individual memory vector. A NTM is a sequential model with outputs produced for every time step t: $\mathbf{M}_t, C_t, \mathbf{r}_t = NTM_{t-1}[\mathbf{x}_t, \mathbf{M}_{t-1}, C_{t-1}]$ with $\mathbf{x}$ being an input vector. This also means that the NTM should be provided with sequential input before resetting its memory state, however its behavior as a regularizer or attention module in a non sequential setting is explored as well.

The NTM is fully differentiable, which makes it possible to optimize parameters via gradient descent. However, in its original definition by Graves et al. [19] it requires an input vector instead of an input image and converting the NTM to a model receiving input matrices is not trivial and, depending on the task, does not improve model performance by a large margin [10]. Therefore an appropriate input has to be constructed such that image information can be stored in M. In this work a small convolutional encoder-decoder network encodes an input tensor as vector $\mathbf{x}$ which is then fed into the NTM. The NTM in turn outputs another vector with the same size as the input and the decoder is used to return the NTM output to its original shape. In practice, any encoding / decoding structure that produces a vector is feasible, however for simplicity, convolutional and pooling layers resize the input to a small size which is then flattened and fed into the NTM controller.

For interaction between the NTM's main components, controller and memory bank, there exist a parameterizable number of read and write heads. Read and write heads represent separate interactions with M. The controller receives input and produces output

as well as providing parameters for all read and write heads. Each head $h[\mathbf{k}_t, \beta_t, g_t, \mathbf{s}_t, \gamma_t]$ is defined by a key vector $\mathbf{k}_t$, key strength $\beta_t \geq 0$, interpolation gate $g_t \in [0, 1]$, shift weighting $\mathbf{s}_t$ and a sharpening scalar $\gamma_t \geq 1$. The controller can either be a recurrent neural network such as LSTM or even just a fully connected layer, it only has to be differentiable and has to provide controlling outputs. A recurrent neural network has an advantage over any non recurrent network, as it has an additional internal memory in which it can store previously read vectors from past steps and re-use them in new steps. In this work, a one-layer LSTM will act as the controller network.

4.3.1. Reading and Writing

To be fully differentiable, reading and writing are defined as attention like focus in form of a normalized weighting over all memory rows. Every read head has a weighting vector $\mathbf{w}_t$ over N memory locations which is multiplied with $\mathbf{M}_t$ and summed,

$$\mathbf{r}_t = \sum_i w_t[i]\mathbf{M}_t[i]. \tag{2}$$

After reading, the network also writes to memory by erasing and adding. Each write head has a write vector $\mathbf{w}_t$ for location as well as an erase vector $\mathbf{e}_t$ and add vector $\mathbf{a}_t$. First, the memory contents are erased:

$$\tilde{\mathbf{M}}_{ti} = \mathbf{M}_{t-1}[i]\left(\mathbf{1} - w_t[i]\mathbf{e}_t\right). \tag{3}$$

Memory contents can only be entirely removed if both w_{ti} and $\mathbf{e}_t$ are 0. After decreasing some memory values, the add vector adds information to the memory matrix by scaling $\mathbf{a}_t$ with the weight of each memory location,

$$\mathbf{M}_t[i] = \tilde{\mathbf{M}}_t[i] + w_t[i]\mathbf{a}_t. \tag{4}$$

The weighting vector $\mathbf{w}_t$ is generated using content based addressing and location based addressing. Content based addressing is a simple lookup - given a more or less similar vector, return the most similar in memory. However, when working e.g. with variables, specific locations also need to be addressable regardless of their content which is reflected in location based addressing.

Content Based Addressing: Every read and write head defines a key vector $\mathbf{k}_t$ which is the lookup vector against which to compare every memory location. Then, any differentiable similarity measure can be used to define which locations to attend to. Graves et al. [19] employ cosine similarity:

$$K(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|}. \tag{5}$$

To get a normalized distribution over all memory positions, the softmax function is used with a parameterized base adjusted by the key strength β_t:

$$w_t^c[i] = \frac{\exp(\beta_t K(\mathbf{k}_t, \mathbf{M}_t[i]))}{\sum_j \exp(\beta_t K(\mathbf{k}_t, \mathbf{M}_t[j]))}. \tag{6}$$

This parameterized base allows for a stronger or weaker focus around high values within the distribution. Additionally, each head produces an interpolation gate g_t which is a scalar to adjust how important the content weighting is. Equation 7 shows how content weight is adjusted. A low g_t increases the importance of $\mathbf{w}_{t-1}$, whereas a high g_t increase importance of $\mathbf{w}_t$:

$$\mathbf{w}_t^g = g_t \mathbf{w}_t^c + (1 - g_t)\, \mathbf{w}_{t-1}. \tag{7}$$

Location Based Addressing : The next weighting used for addressing is the shift weighting $\mathbf{s}_t$ allowing read and write heads to move the attention distribution by shifting it using a convolution operation. Shifting is defined by a softmax over a defined allowed shift range e.g. $[-10, 10]$ would have a length 21 shift vector. The shifting is applied as follows:

$$\tilde{w}_{t,i} = \sum_{j=0}^{N-1} w_t^g[j] \mathbf{s}_t[(i-j) mod N], \tag{8}$$

where $[0, N-1]$ are all possible memory locations and the shifting indexing is done using a modulo operator to repeat it for the entire memory size. However, as this operation would not allow for a sharp shift, further sharpening is applied using an additional scalar $\gamma_t \geq 1$ and the following Equation:

$$w_{t,i} = \frac{\tilde{w}_t^{\gamma_t}[i]}{\sum_j \tilde{w}_t^{\gamma_t}[j]}. \tag{9}$$

Combined, memory can be addressed by content, by location and by using information from the access at the last addressing, while being able to disable each addressing mechanism individually. Graves et al. [19] do not show, how they initialize their memory, albeit it strongly influencing model convergence as the NTM has to work with the contents of the initial $\mathbf{M}$ [11]. Collier and Beel [11] introduce three different methods to do initialization and compare them. They compare constant initialization, learned initialization and random initialization:

- Constant initialization sets all values of the memory matrix to the same constant value. This means that at every memory reset, the network can rely on the exact same base values. The constant scheme showed most promising results in the re-implementation by Collier and Beel [11].
- Learned initialization has a network layer that provides initialization for the memory matrix, which could allow for benefiting non-linear initialization schemes, however with additional parameters to learn.
- Random initialization draws random values from a distribution for every memory value. It is possible, that a random initialization regularizes the learning process by injecting noise into the network.

Graves et al. [20] address some shortcomings of this NTM implementations and present the more advanced DNC. However, in their work they show that while having different convergence speeds, both NTM and DNC can achieve good results on the same task. Since the DNC is even more complex than the NTM, a DNC implementation will be left for future work.

4.4. Modifying the U-Net with Memory

As a NTM alone does not suffice as a multi-level feature extractor, a regular U-Net is modified to include multiple NTM gates. Structural inspiration is drawn from attention gates by Oktay et al. [38]. However, NTM gates have an NTM acting as a gate modifying the importance of information on the expansive path by including upsampled information, skip connection and past information.

4.4.1. Attention Gates

A U-Net with attention gates will be used as comparison to the NTM gate model to validate whether a less complex model is sufficient for improvements on a landmark localization task. Attention gates apply additive attention [6] on all upsampled feature maps by creating a single attention matrix based on both the skip connection and upsampled information from the lower U-Net level and multiplying it elementwise. Equation 10 shows how the attention matrix is calculated per pixel. W_x^T and W_g^T are a defined number of 1×1 convolutions applied to the skip connection input g in order to increase or decrease the importance of a single pixel. For the skip connection, a bias term is included in the convolution operations. Both inputs are summed element wise which yields combined pixel importances and a ReLU activation function R is applied as a non-linear function. Finally, a $1 \times 1 \times 1$ convolution ψ^T reduces the tensor to a single matrix to be used as the attention matrix. A sigmoid activation function normalizes the matrix to result in the final attention gate scalar matrix α^l (Equation 11).

$$q_{att}^l = \psi^T(R(W_x^T x_i^l + W_g^T g_i + b_g)) + b_\psi \tag{10}$$

$$\alpha_i^l = \sigma_2(q_{att}^l(x_i^l, g_i; \Theta_{att})) \tag{11}$$

4.4.2. NTM Gates

As the purpose of the attention gate is to reduce overall search space by devaluing some and highlighting other areas of a feature map, it is sensible to implement a NTM U-Net in a similar way. Therefore, the proposed NTM extended U-Net has the same architecture as the attention gate U-Net by Oktay et al. [38], however with the self attention mechanism replaced by a NTM based memory module. The memory module consists of a convolutional encoder-decoder model and the NTM module itself. The encoder receives concatenated input from both skip connection and expansive path and encodes it into a flattened vector. Then the NTM uses this flattened vector as input and produces a relevant output vector. The decoder then upsamples and reshapes this

vector to a gate matrix. The decoded $1 \times N \times N$ gate matrix is then scaled by applying a sigmoid activation function and finally is multiplied element-wise with every filter map of the expansive path tensor. As the encoder-decoder network is only necessary to downsample the input information, depth is reduced and it is dropped at lower levels of the U-Net when filter maps have smaller sizes such as $16x16$. To make use of the NTM module and its memory properties, the landmark localization task should be converted to a iterative task where landmarks are predicted in groups or even individually such that the NTM can provide information from previous steps. For this iterative implementation of a NTM gate U-Net, logits from the previous prediction are fed back into the next step akin to Wang et al. [54].

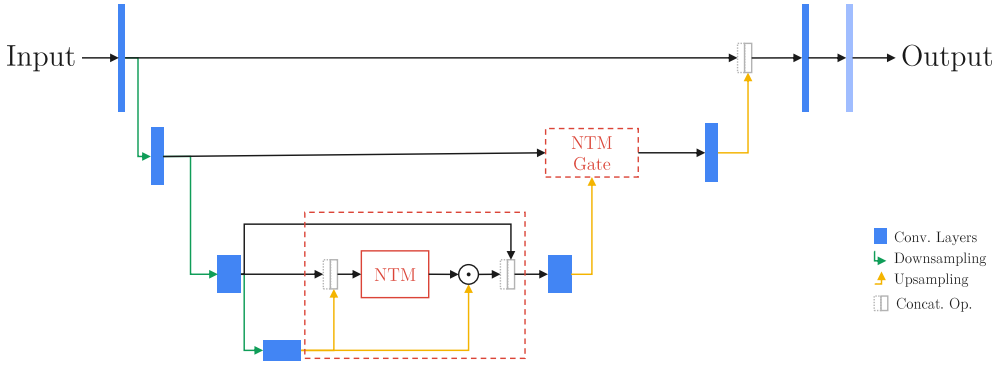


Figure 2: U-Net with NTM Gates. In experiments, the top-most U-Net levels do not have an NTM gate.

5. Experimental Setup

In this section the experiments conducted in order to evaluate the models are introduced. Afterwards, implementation, training and evaluation details are provided.

The NTM gate U-Net model is supposed to iteratively learn groups of landmarks using its memory module and by keeping the prediction from the previous step as input. Validating whether a U-Net gains from an NTM gate requires prior experiments to setup comparison as well as rule out other reasons to why a specific model performs worse or better on an iterative learning task. In this work, a dataset containing fruit fly wings and landmark locations thereof is used (See Section 5.1). At the time of writing, no work has been published on automatic landmark localization using this fly wing blade dataset. Consequently, baseline experiments need to be conducted to compare effects of extended architectures to other models, in particular [40]. Payer et al. [40] also use the public challenge cephalometric X-ray dataset [52] which will be used for model evaluation as well as comparison to other methods. Experiments are structured as follows:

1. The first set of experiments evaluates the performance of all architectures using different amounts of data on simultaneous prediction of all landmarks. The respective training dataset is limited to $66\%, 20\%$ and 6% of its original size. Selected

numbers are chosen arbitrarily to cover multiple data availability scenarios and to include an additional validation set for threefold cross validation. In particular the task with only 6% of data simulates a limited data availability task. To reduce the number of experiments for later tasks, only 6% and 66% will be used for evaluation.

2. The proposed model uses provided landmarks from previous steps to predict other landmarks and may learn additional relative positional information. To evaluate how much of an impact provided landmarks have on prediction performance, an experiment where the baseline U-Net predicts the remaining landmarks is included. $n_l \in [1, 5, 10]$ landmarks are given both for the normal as well as for the limited dataset. If this experiment fails, the model likely does not gain from having information on the first l landmarks and will not gain from an iterative task.

3. Both attention gate U-Net as well as NTM gate U-Net will be assessed on the simultaneous prediction tasks. This experiment validates whether any possible benefits in iterative training with extended architectures stem from iterative training and not increased model complexity.

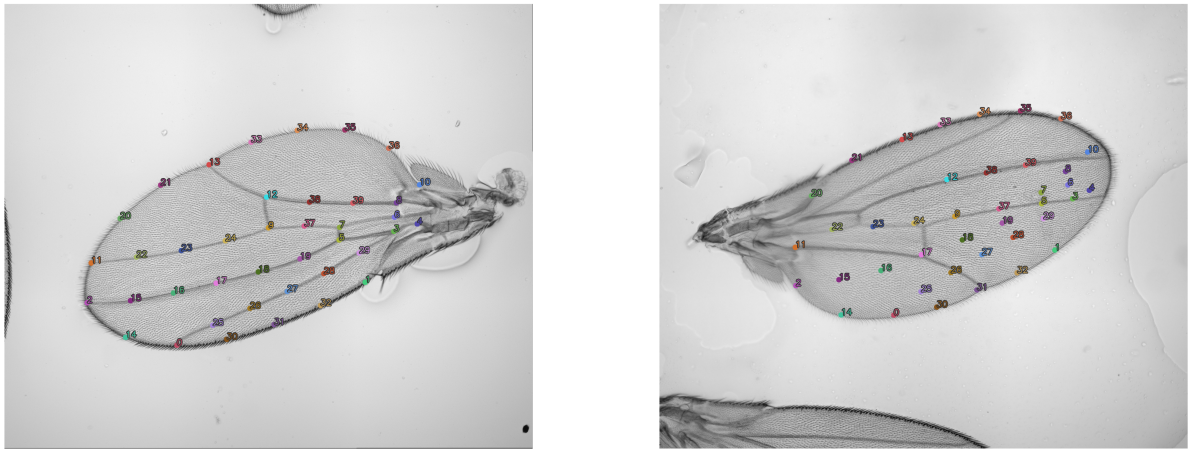
4. The proposed NTM U-Net, the attention gate U-Net and a baseline U-Net are evaluated on the iterative learning task with different sequence lengths $s_l \in [5, 10]$.

To provide good results on each task while limiting the amounts of experiments necessary for this work, the baseline U-Net is tuned on the first task using 100% of the data. While results could be improved by tuning the U-Net again on the iterative task, the necessary computational time does not justify the gained performance. For the NTM, reasonable parameters based on input size are chosen to avoid further parameter search. Attention gate parameters are matched according to the U-Net level they are in. Results are always reported on a holdout test-set which stays the same for all experiments.

5.1. Data

The following section introduces the two datasets used in this work. Both datasets consist of ordered landmarks which always refer to the same location of the object. The first dataset, referred to as Flywing, consists of 712 light sheet microscopy images of fruit fly (*D. Melanogaster*) wings with 39-40 annotated landmarks [30]. Given that the last landmark is missing in some examples, landmark heatmaps of all samples are padded to have 40 landmarks using zero padding. Evaluation on the last landmark is therefore not entirely possible, however experiments later will show that the network learns to always predict the last landmark even with it missing in some training samples. The dataset consists of 3840×3234 greyscale images in TIFF format with one pixel corresponding to $0.645796 \mu m^2$, converted and resized to 256×256 8bit JPEGs to save space and allow for easier and faster work with TensorFlow. It is not publicly available but examples can be found in Figure 3a. Images depict wings in random orientations and with slight shape differences, however the general shape variation is low. A number of images also show parts of other wings and some images contain air or liquid bubbles. 14 Landmarks are

placed manually on intersecting veins as well as endpoints. The remaining 26 landmarks are positioned using regular intervals on splines between two landmarks matching e.g. a vein on the wingblade. The data is pre-split into 250. randomly selected test samples and 462 training samples to avoid any leakage and for better comparison to the second dataset. Figure 3a shows two examples from the dataset, one from the training set and one from the test set with annotated and enumerated landmarks. In addition, some examples are incorrectly labeled (see Figure 3b) but will be kept in the dataset to avoid re-evaluating and labeling all 712 samples.



(a) Example from training set

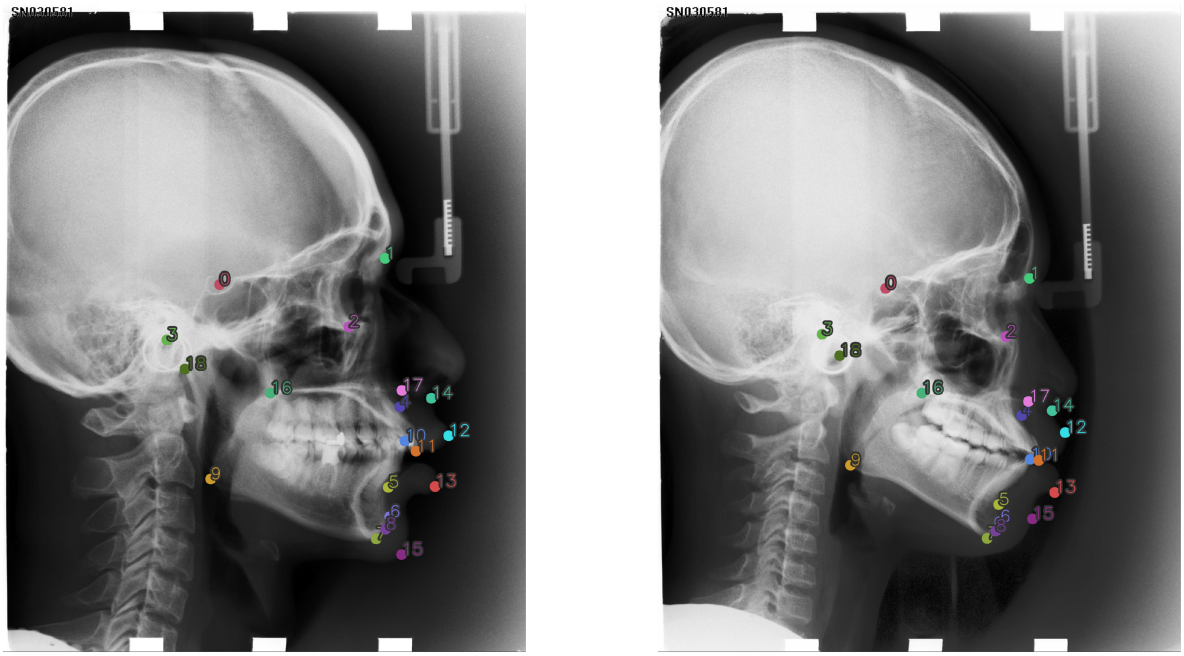
(b) Wrongly labeled sample

Figure 3: Samples from Flywing: Correctly labeled sample on the left, incorrectly labeled sample on the right

The dataset shall be briefly analyzed in terms of differences between train and test set. Mean and standard deviation of relative distances between all landmarks are calculated. Using the difference of mean relative landmark distances between training and test set, some information about potential shape differences regardless of object orientation and location can be inferred. For Flywing, no large differences are observed apart from the last landmark of which the high values emerge from the padded landmark. For empty landmark heatmaps, landmark coordinates default to $0, 0$. For the complete matrix, see Appendix A.1.1. The standard deviation of relative landmark distances within the dataset can be considered a measure of overall shape variability (see Appendix A.1.2). For some landmarks, high values can be observed, but these values likely stem from mislabeled examples and actual shape variation is relatively low.

The second dataset, Cephal, is a public challenge dataset from the ISBI 2015 Cephalometric X-ray Image Analysis Challenge [52] which contains X-ray images of human heads with 19 annotated landmarks around denture and jaw from the side. Original images are 1935×2400 24bit grayscale bitmaps in which one pixel covers $1mm^2$. They are converted to $256x256$ 8bit JPEGs to be able to use the same pipeline for both datasets. Figure 4 shows some examples from Cephal. Skull orientation is fixed but skull shapes show some variation. Landmarks are not all positioned on clear geometric shapes and

some landmarks, particularly at chin and front of the head are within few pixels of one another. For train-test split, the later published challenge splitting into 150 training samples and 250 test samples is used. Any results in this work can therefore also be compared to other published results, e.g. [40]. Both datasets provide pixel coordinates for each landmark which are converted to per-landmark heatmaps as labels using tensorflows multivariate normal distribution implementation and its probability density function and a standard deviation of $\sigma = 2$. Cephal is extended by one empty landmark e.g. the heatmap contains only zeroes to allow for iterative experiments where the number of predicted landmarks are split into equal numbers. Figure 4 shows examples from both training and test set with enumerated landmarks annotated.



(a) Training set sample

(b) Test set sample

Figure 4: Annotated samples from Cephal

The cephalometric X-ray dataset is analyzed using the same metric as for Flywing. Landmarks 5 and 15 show large mean differences for train and test set compared to the remaining variables. [52] report that landmarks were manually labelled and reviewed by two experts with ground truth being the mean markup between the individual labels by the experts. Therefore, the variation for landmarks 5 and 15 is likely based on different labelling methods among experts. Considering the standard deviation of relative landmark distances, there is an overall high variation for all landmarks apart from the three chin landmarks indicating high shape variability in the cephalograms despite its fixed orientation (Appendix A.1.3).

5.1.1. Augmentation

Each dataset is augmented using imgaug [24] to artificially increase the dataset size and train a more robust model. For consistent transformations with coordinates, landmarks are converted to heatmaps and stacked along the color channel of the corresponding image. Image transformations such as cropping and padding are then applied on the entire stack at once. As datasets differ in shape and orientation variation, different transformations are applied to the datasets. Contrast or color augmentations are not included as there is no observable variation in terms of color and contrast. Both datasets are augmented with elastic transform, cropping and affine transformation which includes scaling, translating, rotating and shearing of an image. Images of Flywing additionally can be flipped both vertically and horizontally. Detailed information on augmentation parameters can be obtained in Appendix A.3. Some of the applied augmentation methods remove landmarks close to image borders. Consequently, the network will sometimes not receive a correct loss on some landmarks but some robustness for new images with cropped out landmarks will be learned. In a constructed dataset such as Cephal this will lead to no performance gain, but will improve performance on real-world datasets such as Flywing. It is important to note, that augmentation transformations may bring unnecessary complexity into the dataset [59], which should be taken into consideration when evaluating the results of this work. Datasets in all experiments are aggressively augmented to increase its size to 400% of its original size except for the strongly reduced experiments where the selected 6% are augmented to be of 120% size of the original dataset.

5.2. Implementation Details

All models are implemented using TensorFlow 2.1 [1]. For the initial set of simultaneous prediction baseline experiments, a small parameter search is conducted to find a suitable U-Net architecture by comparing different U-Net depths, starting filter numbers and filter sizes. Generally, some of these parameters could be set by considering the receptive field of the bottleneck convolutional operation, which is supposed to include the entire object. However, as for both datasets used in this work, the object covers the entire image requiring a large receptive field which would require either a larger U-Net or more downsampling per level. Therefore, suitable parameters are selected and the baseline U-Net is tuned separately for both datasets. Tuning results can be found in Appendix Table A.2. Larger models with 7 or more levels could not be considered because of limited GPU memory. Moreover, larger models require significantly longer training time and therefore energy while not improving results by a large margin. The following U-Net parameters are selected for all experiments:

At every level a stack of three convolutional layers with leaky ReLU as activation function is applied. At the end of each U-Net a 1×1 convolutional layer with no activation function converts the last U-Net output into the target labels. Similar to Payer et al. [39], dropout with 0.2 dropout rate is added to the lowest two levels. Attention gates use the same number of filters as the current U-Net level they are applied at except for the last

	Flywing	Cephal
Levels	6	5
Initial Filters	40	40
Kernel size	3	5
Filter increase factor	2	2

Table 1: U-Net parameters for both datasets. A filter increase factor of 2 indicates that number of filters are doubled at every U-Net level.

layer in Flywing, where the number of filters are halved to avoid running out of memory. For the NTM gates, every encoder-decoder network is defined such that the current input filter maps are resized to 16×16 for the NTM controller input. Each convolutional layer in the encoder has the same number of filters as the corresponding U-Net level convolutional layers. The exception being 1×1 convolutional layers for controller input and final decoder output which only have 1 filter. To train the model, ADAM [27] optimizer with $1e^{-4}$ learning rate is applied with additional exponential learning rate decay of .99 every 1000 training steps to reduce loss volatility in later training iterations. Iterative experiments frequently converge to NaN or have a large number of loss spikes during learning and are therefore ran with $5e^{-5}$ learning rate. Models are trained with batch size 4 on 4 NVIDIA Tesla V100 GPUs for 30000 steps. Training a model takes between 3 and 15 hours per cross validation fold depending on model size and experiment.

5.3. Evaluation Metrics

Results are evaluated using several metrics: All models are evaluated on the sum of squared distance loss, which evaluates the entire image prediction pixel-wise. The loss by itself is not a good evaluation criterion, but comparing losses of multiple models gives a good indication on how much noise is present in final predictions. To refine the evaluation, several metrics based on landmark positions are used. The argmax function is used to extract landmark coordinates from predictions on which metrics are applied. The distance between predicted and ground truth landmarks is evaluated individually per landmark using euclidian distance. Results are reported as point-to-point error (PE) on the rescaled image (256×256) in pixels (PE_{px}) as well as scaled based on both width and height of the original image to report millimeter point-to-point error PE_{mm}. Furthermore, it is validated whether a landmark is outside of multiple defined PE margins to report the percentage of outliers ($\%O$). Margins are set to 2px,4px and 10px and 2,4 and 10mm respectively allowing for a direct comparison to results of other published works [52, 40]. Finally, it is evaluated whether a landmark is closest to its ground truth counterpart which becomes particularly interesting if landmarks are close together and also shows whether a network is capable of separating landmarks. The metric is calculated by computing distances between all landmarks and checking whether the lowest distance is on the diagonal of the resulting distance matrix. Glocker et al. [17] refer to this metric as percentage of correctly identified landmarks ($\%ID$). Mean and standard deviation

for all experimental results are always reported and standard deviations across multiple results are combined using an adapted algorithm based on Higgins et al. [21].

6. Results

In this section, results for all experiments on Flywing and Cephal are presented.

6.1. Flywing

The Flywing dataset is larger compared to Cephal and has landmarks positioned on edges and patterns in the images. The search space for these landmarks can already be reduced by a sobel filter which would highlight contrast boundaries and therefore wing boundaries. However, as some landmarks are interpolations of a spline between other points, the actual position of such landmarks has no meaning in the image context but only as a representation of the spline. As the 26 spline landmarks are directly based on the location of the 14 first landmarks, this dataset has an intrinsic two step task structure which is expected to be observed in the iterative learning task.

6.1.1. Baseline Evaluation

Baseline results show that a basic U-Net can predict landmarks with 1.91px mean PE_{px} cross validated on 66% training data with 99.3% identification rate and 13% O_{px} with $r = 2$. Results on the dowscaled image are scaled up to original image size and adjusted according to image resolution. For Flywing, μm will be reported instead of mm as the entire fly wing only spans 2mm. Reducing training dataset size does have a strong impact on all metrics. The outlier rate at $r = 10$ is doubled regardless of scale, mean PE_{px} increases by .33px and 1.12px with 20% and 6% training data and $\%O_{px}$ at $r = 4$ is quadrupled. 20% training data does not differ as much from 66% as 6%. Table 2 summarizes results from the initial baseline task of simultaneously predicting all 40 landmarks.

% Data	PE_{px}	$\%O_{px}$			$PE_{\mu m}$	$\%O_{\mu m}$		$\%ID$
	Mean ± SD	$r = 2$	$r = 4$	$r = 10$	Mean ± SD	$r = 2$	$r = 10$	
66	1.91 ± 5.86	13.0%	2.7%	0.7%	17.35 ± 52.83	71.8%	33.0%	99.3%
20	2.24 ± 6.28	20.0%	3.9%	0.8%	20.41 ± 56.61	80.3%	44.6%	99.2%
6	3.03 ± 6.47	38.4%	11.7%	1.4%	27.67 ± 58.18	89.7%	63.7%	98.9%

Table 2: Evaluation metrics for Baseline experiments

With 6% training set size, the model fits to the training data having a lower training loss (7280 vs 10825) than 66%, but then does not generalize well with comparatively high validation loss (22599). All models are trained for 30000 iterations to be able to compare results, but the 6% model in particular has a noticeable negative slope in the last few training iterations (see Figure A.4.1). For 66% training data, the training loss is

higher than validation loss. Only for the final training steps the training loss is lower than validation loss with a difference of 1025 on the last training step. For less training data, the validation loss decreases slowly over time with a difference of 15293 at the last training step.

Considering the individual landmarks PE_{px}, all training set sizes above 6% have similar performance across all landmarks with equally slight increases in error across all landmarks. Only when reducing the data even further to 6% predictions have increased PE_{px} with a large number of landmarks having around 2px mean difference to the other models. Patterns can be observed where multiple landmarks in a row have similar and error and linear increase or decrease in coordinate distance e.g. landmarks 14 to 19 or 20 to 24.

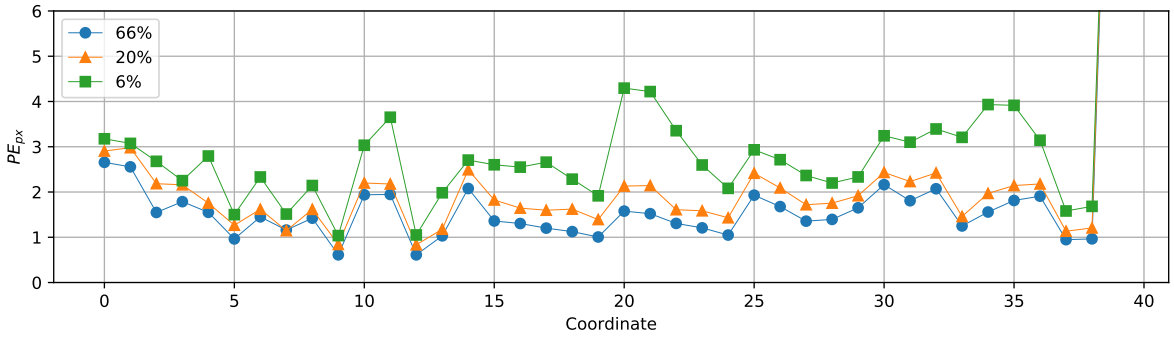


Figure 5: Per landmark coordinate distance for different amounts of training data

6.1.2. Providing Landmarks as Input

For the following experiments, only 66% and 6% training set sizes are used to observe differences on small versus large training sets. When providing the first n ground truth landmarks as input, results are dependent on which landmarks are evaluated. They are adjusted to only contain information on the remaining 30 landmarks which are never used as input (see Table 3). This ensures that metrics contain the same information for all experiments. With only one anchor point and 66% training data, mean PE_{px} decreases by .26px and 3.3% less $\%O_{px}$ at $r = 2$. Providing more than one landmark reduces mean PE_{px} further to 1.43px and 1.34px respectively. Using 6% training data and including one landmark as input lowers mean PE_{px} by .25px compared to baseline including a .25px lower standard deviation. 5 and 10 included landmarks have a stronger impact with 2.28px and 1.95px mean PE_{px}. The model receiving 10 landmarks on 6% training data predicts landmarks .07px closer to ground truth than the baseline without landmarks on the large training dataset yet with higher $\%O_{px}$ for $r \in [2, 4]$. On 6% training data $\%ID$ increases by .7% when increasing the number of input landmarks matching $\%ID$ rates for 66% training data.

%Data	n_l	PE_{px}	$\%O_{px}$			$PE_{\mu m}$	$\%O_{\mu m}$		$\%ID$
		Mean ± SD	$r = 2$	$r = 4$	$r = 10$	Mean ± SD	$r = 2$	$r = 10$	
66	0	2.02 ± 6.32	13.4%	2.5%	0.8%	18.31 ± 56.74	72.8%	34.3%	99.2%
	1	1.76 ± 6.28	10.1%	1.6%	0.7%	15.97 ± 55.84	70.0%	30.0%	99.3%
	5	1.43 ± 5.60	6.1%	1.0%	0.5%	12.82 ± 49.64	63.5%	22.8%	99.5%
	10	1.34 ± 5.73	4.8%	0.9%	0.5%	12.06 ± 50.87	58.0%	18.6%	99.5%
6	0	3.29 ± 6.92	43.1%	13.8%	1.6%	30.06 ± 62.04	90.9%	67.5%	98.8%
	1	3.04 ± 6.67	38.6%	12.2%	1.7%	27.76 ± 59.35	88.9%	63.0%	98.9%
	5	2.28 ± 5.72	28.2%	7.0%	0.9%	20.77 ± 50.76	83.8%	52.1%	99.4%
	10	1.95 ± 5.64	18.5%	4.0%	0.7%	17.61 ± 49.96	79.6%	43.0%	99.5%

Table 3: Comparison between different numbers of provided keypoints. Metrics are only computed on the last 30 landmarks, such that results are comparable within this experiment.

6.1.3. NTM and Attention Baseline

To validate any findings on the iterative task using extended architectures, both NTM gate as well as the attention gate are evaluated on the simultaneous prediction task. Training the attention gate model for 30000 iterations using 66% yields a .1px higher mean coordinate distance than the baseline counterpart (see Figure 4). However, evaluating training and validation loss, a high negative slope at 30000 iterations for both training set sizes can be observed (see Appendix A.4.2). The 66% attention gate model also visibly improves on landmark 0,1 and 10 but is unable to predict landmarks as close to ground truth as baseline for spline landmarks (Appendix A.4.3). The NTM gate model has a high mean point-to-point error with 5.79px and a standard deviation of 17.3px but still a comparatively low $\%O_{px}$ of 27.1% at $r = 2px$, both of which are twice as high as the other models on the same task. On 6% training data, the attention model and NTM gate model have a 1.87px and 2.87px higher mean PE_{px} than U-Net baseline and around 50% outlier rate at $r = 2px$.

%Data	Model	PE_{px}	$\%O_{px}$			PE_{mm}	$\%O_{\mu m}$		$\%ID$
		Mean ± SD	$r = 2$	$r = 4$	$r = 10$	Mean ± SD	$r = 2$	$r = 10$	
66	Attention	1.99 ± 5.88	16.0%	3.1%	0.8%	18.12 ± 52.84	74.7%	37.6%	99.3%
	NTM	5.79 ± 17.30	27.1%	10.9%	6.0%	53.93 ± 162.69	79.8%	47.9%	93.5%
6	Attention	4.90 ± 8.23	48.7%	25.5%	6.2%	45.20 ± 75.44	90.7%	68.9%	95.8%
	NTM	5.90 ± 9.36	52.8%	30.6%	8.4%	54.17 ± 85.43	91.8%	71.5%	94.8%

Table 4: Results for baseline simultaneous model with NTM or Attention Gate

Both NTM experiments predict worse results than baseline with large differences in loss and coordinate distances. However, similar to attention gates, the NTM gate loss does not plateau at 30000 iterations for both 66% and 6% training data (see Appendix A.4.4).

6.1.4. Iterative Training and Prediction

Similar to [54], results show that learning iteratively can improve model performance. The baseline U-Net without any recurrent- or memory module predicts more accurate landmark locations than its simultaneous counterpart with improvements. On 66% the baseline U-Net reaches a 1.76px mean PE_{px} with $s_l = 5$, .07px lower than with $s_l = 10$. The $s_l = 5$ U-Net also has the lowest overall outlier rate for $r \in [2, 4]px$ and is on par with other models for $r = 10px$. Upscaled to original image dimensions, the mean $PE_{\mu m}$ is $1.32\mu m$ lower than simultaneous baseline and 4.4% more landmarks are predicted within $2\mu m$. For 6% training data, the $s_l = 5$ U-Net again predicts with the lowest PE_{px} among all compared models, .38px lower than the $s_l = 10$ U-Net and accordingly with the lowest outlier rate at $r = 2px$ of 29.6px. $\%O_{px}$ for $s_l = 10$ is only .2% lower than $s_l = 5$. Results for all models are summarized in Table 5.

%Data	Model	s_l	PE_{px}	$\%O_{px}$			$PE_{\mu m}$	$\%O_{\mu m}$		$\%ID$
			Mean ± SD	$r = 2$	$r = 4$	$r = 10$	Mean ± SD	$r = 2$	$r = 10$	
66	U-Net	10	1.83 ± 6.00	9.9%	2.2%	0.8%	16.76 ± 54.24	68.2%	28.6%	99.2%
	Attn.		1.84 ± 5.89	11.4%	2.2%	0.7%	16.79 ± 52.99	71.2%	31.4%	99.3%
	NTM		2.03 ± 6.07	16.7%	3.2%	0.7%	18.52 ± 54.72	76.7%	39.6%	99.3%
	U-Net	5	$\mathbf{1.76 \pm 6.00}$	**9.7%**	**2.1%**	0.7%	$\mathbf{16.03 \pm 53.70}$	**67.4%**	**28.1%**	99.3%
	Attn.		1.89 ± 6.32	12.6%	2.4%	0.7%	17.17 ± 56.68	72.3%	34.2%	99.3%
	NTM		2.03 ± 6.28	17.1%	3.3%	0.7%	18.54 ± 56.32	75.7%	39.3%	99.3%
6	U-Net	10	3.04 ± 6.99	29.8%	**7.3%**	1.6%	27.76 ± 63.17	86.5%	56.1%	98.6%
	Attn.		3.54 ± 6.95	40.3%	16.1%	2.9%	32.46 ± 62.95	89.2%	63.8%	98.0%
	NTM		3.47 ± 6.72	41.0%	16.3%	2.0%	49.42 ± 94.38	87.3%	87.3%	98.5%
	U-Net	5	$\mathbf{2.66 \pm 6.42}$	**29.6%**	7.5%	**1.2%**	$\mathbf{24.24 \pm 57.59}$	**86.3%**	**55.5%**	**98.9%**
	Attn.		3.88 ± 7.99	37.7%	13.7%	3.3%	35.51 ± 72.77	88.8%	61.4%	97.0%
	NTM		3.06 ± 6.15	39.6%	14.6%	1.5%	43.50 ± 85.86	88.9%	88.9%	99.0%

Table 5: Comparison between iterative models with different s_l on both 66% and 6% Flywing training data

The individual outlier rate for $r = 2px$ with 66% training data (Appendix A.4.5) is 0.3% lower on average compared to baseline for the first 14 landmarks, but 4.9% lower for the constructed 26 landmarks. The same effect can be observed for 6% training data (Appendix A.4.5) where the mean difference for the first 14 landmarks is 2.3% but 12.2% on the remaining 26.

Attention: As in the previous experiment, attention gates do not support the U-Net in the landmark localization task. On 66% training data, the negative effect of including attention gates is weaker. While the PE_{px} for attention gates with $s_l = 10$ is 0.01px higher, but with a .11px lower standard deviation than baseline, it has a 1.5% higher $\%O_{px}$ for $r = 2$. On 6% training data, attention gates have .5px and 1.22px higher PE_{px} compared to the baseline U-Net on the same task. On 66% training data, the loss slope for both validation and training loss is still steeper than for baseline at 30000 iterations (see Appendix A.4.7). Therefore, additional training could improve point-to-point error to lower than baseline. Differences on Landmarks 0-13 are small, but the attention model

with $s_l = 10$ has .02px lower mean PE_{px}. Only on the constructed landmarks (14-39), the mean PE_{px} for attention models is .2px and .14px higher compared to baseline (1.91px).

NTM: NTM extended models have the highest point-to-point error on 66% training data and consequently also the highest outlier rate for $r_{px} \in [2, 4]$. $\%ID$ is not worse than any other model which is consistent with differences in outliers within small radii. The comparatively best performing NTM uses 6% training data and $s_l = 5$ and has only .02px higher PE_{px} but 9.8% higher outlier rate at $r = 2px$ than baseline U-Net with $s_l = 10$. When splitting results into manually placed and constructed landmarks, results match what could be observed with attention gates. Comparing $s_l = 5$ U-Net and NTM gate U-Net, the NTM has 2.21px mean PE_{px} compared to 2.23px for baseline on the manual landmarks but 2.92px compared to 2.29px on constructed ones. $\%O_{px}$ for $r = 2$ is 1% higher for the NTM model on the first 14 landmarks, but then 15% higher for the 26 remaining (see Appendix A.4.6,A.4.6). As with attention gates, NTM gates also require longer than 30000 iterations for convergence as the loss slope is steep compared to baseline U-Net loss at 30000 iterations (see Appendix A.4.8).

6.2. Cephal

Landmarks in a lateral cephalogram are constructed by utilizing some manually localized landmarks to find others. Angles and distances between landmarks are used for orthodontic analysis which makes accurate localization particularly important [5].

6.2.1. Baseline Evaluation

The baseline model is able to predict landmarks with a mean PE_{mm} of 2.19mm and 39.8% O_{mm} and 13.4% O_{mm} for $r \in [2, 4]$. Despite using 50 samples less than Payer et al. [40], results are considerably worse with them achieving 26.67% and 10.25% $\%O_{mm}$ for $r \in [2, 4]$ on the same test set. Reducing the amount of training data makes accurate predictions more difficult with mean PE_{mm} increasing by 0.22mm and 1.17mm as well as $\%O_{mm}$ ($r = 2$) increasing by 3.8% and 13.9% respectively. $\%ID$ also decreases by 1.2% and 4.9%. All results for this experiment can be found in Table 6. Analyzing the training loss and validation loss for 66% and 6%, a large difference between training and validation loss can be observed e.g. 14066 absolute difference for 6%. Both models overfit to training data with mean validation loss having its minimum around 10000 training iterations and increasing afterwards. With less available data, the model overfits more. Loss plots can be found in Appendix A.5.1.

6.2.2. Providing Landmarks as Input

When providing the model with ground truth landmarks, less outliers and a lower point-to-point error for the remaining landmarks are expected as the existing landmarks can be used as anchor points. In this experiment this expectation holds, however with less of an impact. All results are shown in Table 7. Using 66% training data, $\%O_{px}$ improves by .3

% Data	PE_{px}	$\%O_{px}$			PE_{mm}	$\%O_{mm}$			$\%ID$
	Mean ± SD	$r=2$	$r=4$	$r=10$	Mean ± SD	$r=2$	$r=4$	$r=10$	
66	2.58 ± 1.77	45.4%	17.9%	2.3%	2.19 ± 1.48	39.8%	13.4%	1.2%	95.9%
20	2.84 ± 2.11	49.7%	20.4%	2.5%	2.41 ± 1.78	43.6%	15.5%	1.5%	94.7%
6	3.94 ± 4.00	60.4%	29.3%	5.2%	3.36 ± 3.44	53.7%	23.3%	3.5%	91.0%
Payer et al. [40]	-	-	-	-	-	26.67%	10.25%	-	-

Table 6: Evaluation metrics for Baseline experiments

%and 1.1% for $r = 2$ but for both $r = 4$ and $r = 10$ values increase over baseline with zero input landmarks. With 10 ground truth landmarks, all metrics improve except for $\%O$ with $r = 10$ which stays the same as baseline. $\%ID$ however increases with 1 and 5 input landmarks, but decreases by .9% with 10 landmarks. Using 6% training data, the model does improve with input ground truth landmarks, yet providing more than one landmark does not further improve metrics for the last 10 landmarks. For one input ground truth landmark mean PE_{px} decreases by 0.61px and $\%O_{px}$ decreases by 2.7%, 2.8% and 2% for $r \in [2, 4, 10]$. To further investigate these results, individual per landmark PE_{px} is analyzed with results for all landmarks on 6% training data (Appendix A.5.2). Landmarks 10, 11 and 18 have a higher point-to-point error with more input landmarks with +0.67px, +0.17px and +.17px respectively for 5 input landmarks compared to zero input landmarks. A decrease in error can be observed for landmarks 12-15 with -0.33, -0.43, -0.49 and -0.58. One input landmark yields the lowest PE_{px} across the last 10 landmarks and strongly improves point-to-point error in the first 9 landmarks (-0.695px). Providing landmarks also yields a 3.2px lower PE_{px} for landmark 9.

%Data	n_l	PE_{px}	$\%O_{px}$			PE_{mm}	$\%O_{mm}$			$\%ID$
		Mean ± SD	$r=2$	$r=4$	$r=10$	Mean ± SD	$r=2$	$r=4$	$r=10$	
66	0	2.72 ± 1.92	47.2%	18.7%	3.6%	2.30 ± 1.60	41.6%	13.7%	1.9%	96.0%
	1	2.72 ± 1.82	46.9%	19.2%	3.8%	2.30 ± 1.50	40.9%	14.4%	1.9%	96.6%
	5	2.72 ± 1.99	46.1%	18.8%	3.7%	2.31 ± 1.67	40.3%	14.7%	2.0%	96.3%
	10	2.55 ± 1.81	42.7%	16.2%	3.6%	2.15 ± 1.48	36.4%	12.0%	1.9%	95.1%
6	0	4.18 ± 4.27	61.3%	30.9%	7.2%	3.59 ± 3.75	54.9%	25.5%	5.1%	91.4%
	1	3.57 ± 2.91	58.6%	28.1%	5.2%	3.07 ± 2.57	52.4%	22.6%	3.4%	92.4%
	5	3.75 ± 3.16	60.2%	29.4%	5.7%	3.23 ± 2.78	54.0%	23.9%	3.8%	89.8%
	10	3.57 ± 2.64	58.7%	28.6%	5.6%	3.05 ± 2.27	52.6%	22.5%	4.0%	87.6%

Table 7: Comparison between different numbers of provided keypoints. Metrics are only computed on the last 9 landmarks, such that results are comparable

6.2.3. NTM and Attention Baseline

In contrast to Flywing, the attention gate model does converge to better results than baseline within 30000 iterations. Both models improve on the baseline. Using 66% training data, the NTM gate model has the lowest overall PE_{px} with 2.52px mean despite not using the write operation at all and the lowest outlier rates with 41.9% at 2px or 36.7% at 2mm. The Attention model also improves on Baseline by mean 0.04px, 0.02px

higher than the NTM but with a .14px lower standard deviation. On less data, the attention gates predict with 3.44px mean point-to-point error, .65px lower than the NTM model on the same reduced dataset with less outliers across all radii. All extended baseline results can be found in Table 8.

%Data	Model	PE_{px}	$\%O_{px}$			PE_{mm}	$\%O_{mm}$			$\%ID$
		Mean ± SD	$r=2$	$r=4$	$r=10$	Mean ± SD	$r=2$	$r=4$	$r=10$	
66	Attention	2.54 ± 1.71	42.4%	18.0%	2.4%	2.15 ± 1.43	37.0%	13.9%	1.3%	96.4%
	NTM	2.52 ± 1.85	41.9%	17.6%	2.3%	2.13 ± 1.55	36.7%	13.5%	1.3%	96.6%
6	Attention	3.44 ± 2.91	53.9%	25.8%	4.2%	2.91 ± 2.45	47.3%	20.3%	2.6%	92.9%
	NTM	4.09 ± 4.32	58.1%	29.8%	6.1%	3.48 ± 3.76	51.6%	24.5%	4.3%	90.3%

Table 8: esults for baseline simultaneous model with NTM or Attention Gate

6.2.4. Iterative Training and Prediction

Using the baseline U-Net on the iterative task does not benefit any observed metric. On 66% training data, the U-Net iterative baseline matches simultaneous task performance on $s_l = 5$ but has a .03px lower mean PE_{px} with $s_l = 10$. On 6%, the iterative baseline has a higher point-to-point error across sequence lengths with the highest overall outlier rates across all models. Considering the individual point-to-point error, both iterative models have a lower error on the first two landmarks, match performance on the next 3 but then do not manage to predict the last ten landmarks with 5.50px mean PE_{px} for $s_l = 10$ and 7.83px mean PE_{px} for $s_l = 5$.

%Data	Model	s_l	PE_{px}	$\%O_{px}$			PE_{mm}	$\%O_{mm}$			$\%ID$
			Mean ± SD	$r=2$	$r=4$	$r=10$	Mean ± SD	$r=2$	$r=4$	$r=10$	
66	U-Net	10	2.61 ± 1.74	45.6%	18.8%	2.3%	2.21 ± 1.46	39.8%	14.0%	1.3%	95.9%
	Attn.		2.54 ± 1.70	42.9%	18.3%	2.4%	2.14 ± 1.42	37.6%	14.1%	1.2%	**96.2%**
	NTM		2.54 ± 1.73	43.1%	18.1%	2.3%	2.15 ± 1.43	37.9%	13.7%	**1.1%**	95.8%
	U-Net	5	2.58 ± 1.89	44.9%	17.8%	**2.2%**	2.18 ± 1.58	39.0%	**13.3%**	1.3%	95.7%
	Attn.		$\mathbf{2.49 \pm 1.76}$	**41.5%**	**17.7%**	2.4%	$\mathbf{2.11 \pm 1.47}$	**36.4%**	13.5%	1.3%	**96.2%**
	NTM		2.56 ± 1.69	43.1%	18.3%	2.5%	2.17 ± 1.41	38.1%	14.1%	1.3%	95.3%
6	U-Net	10	4.72 ± 5.69	60.8%	30.6%	6.1%	4.04 ± 4.95	54.8%	24.6%	4.6%	88.7%
	Attn.		$\mathbf{3.42 \pm 2.51}$	**57.1%**	**27.7%**	**4.1%**	$\mathbf{2.92 \pm 2.11}$	**51.3%**	**22.2%**	**2.6%**	**92.2%**
	NTM		3.85 ± 2.87	61.5%	32.1%	6.8%	3.30 ± 2.44	55.3%	26.4%	4.7%	89.8%
	U-Net	5	6.19 ± 8.69	60.8%	30.0%	7.9%	5.28 ± 7.52	54.8%	23.9%	6.5%	87.5%
	Attn.		4.92 ± 6.46	57.3%	30.3%	7.1%	4.21 ± 5.61	51.4%	24.9%	5.1%	88.6%
	NTM		3.76 ± 3.46	58.9%	30.2%	5.3%	3.20 ± 2.97	53.0%	24.2%	3.3%	90.5%

Table 9: Comparison between iterative models with different s_l on both 66% and 6% Cephal training data

Attention: On 66% training data, attenion-gate U-Nets predict with the lowest overall point-to-point error among the evaluated models. The $s_l = 5$ attention model predicts with 2.49px mean PE_{px} which is .05px lower than the simultaneous attention U-Net and .09px lower than simultaneous baseline U-Net. On the large training set, a lower

sequence length also corresponds to lower PE_{px} (-.05px) and lower $\%O_{px}$ (-1.4%, -.6%, +0%). Neither of the iterative attention models overfits with validation loss decreasing and finally plateauing at the end of the training period. On 6% training data, attention gates with $s_l = 10$ reach the lowest PE_{px} of 3.42px for all models. However, outlier rates for $r \in [2, 4]$ are 3.2% and 1.9% higher than the same model on the simultaneous task. With $s_l = 5$, attention gates converge to a loss of 15820 on test data compared to 15640 with $s_l = 10$ but predict with 1.3px higher PE_{px}. In the same way, $\%O_{px}$ at $r = 2$ is only .2% higher, but then 2.6% and 3% higher at $r \in [4, 10]$.

NTM: On 66% training data the NTM with $s_l = 10$ achieves 1.1% O_{mm} at $r = 10$ which is the lowest of all compared models. Other than that, NTM gates seem to generate worse predictions compared to attention gates across all metrics. Compared to the simultaneous baseline, NTM gates improve the PE_{px} by .04px and .02px and the $\%O_{px}$ on $r = 2$ by 2.3% but have a higher outlier rate for radii above 2. While the $s_l = 5$ model has the worse metrics on 66% training data, the opposite is true for 6%. Here, the $s_l = 5$ NTM has a .09px lower mean PE_{px} and 2.6%, 1.9% and 1.5% lower $\%O_{px}$ than $s_l = 10$. None of the NTM models overfit, however the validation loss on 6% training data is the only observed loss to first increase and then decrease. (see Appendix A.5.4).

The evaluated models yield different results across datasets. Implications, reasons and possible future research directions will be discussed in the next section.

7. Discussion

Results show that implicit memory in form of an iterative learning task or explicit memory in form of a NTM module can improve landmark localization across datasets. However, results are different for the analyzed datasets. On Flywing, the baseline U-Net on 66% training data is sufficient for a low point-to-point error and a 99% identification rate and while the outlier rate at $r = 2px$ decreases strongly with less data, a U-Net is still able to produce reasonable results. Despite including wrongly labeled images and having a missing landmark, the model predicts wrongly labled examples correctly and includes the missing landmark at the correct location in all predictions. The U-Net also detects the positional correlations of spline landmarks as observable in almost linear PE patterns. Including ground truth landmarks as network input has a high impact on localization accuracy of the remaining landmarks which is probably also due to the dataset consisting of manual annotations for the first 14 landmarks of which up to 10 are provided and the remaining 26 being directly based on the position of the first 14. Moreover, as mean PE_{px} on 6% training data with 10 provided input landmarks is lower than 66% training data with zero input landmarks, having information on those first n landmarks is key for an overall accurate localization in Flywing. It might be valuable to re-run and compare models only on the manually placed landmarks on Flywing for more generally applicable results. NTM gates and attention gates do not benefit the U-Net when predicting all landmarks at once, at least when setting the networks to only run for 30000 steps. Therefore, any performance gains in the iterative task would be due to the additional

memory and attention mechanisms. As observed in the iterative experiment, the U-Net without any additional gates performs best, particularly with a small sequence length. The implicit memory of keeping predictions from the last timestep as new input seems to be sufficient for a more accurate localization and does not hinder convergence despite inputting inaccurate landmark locations during training. Further experiments should be conducted with iterative training without using predictions from $t-1$ but instead with teacher forcing where ground truth landmarks are put as input during training. This allows the network to utilize the input early in the training without converging to results where the additional information is discarded. Neither attention gates nor NTM gates outperform the U-Net baseline on the iterative task when considering all landmarks. However, when only analyzing the manually placed landmarks, the attention gate does in fact predict with lower point-to-point error and the NTM gate matches the U-Net while both models did not converge to a validation loss plateau at 30000 steps.

When inspecting the output maps generated by the attention gates, some interesting patterns can be observed (see Appendix A.4.9). In the first two rows, white rings show up on landmark positions indicating that a previous landmark position is erased as a potential candidate. At the same time, the halos would increase the weight of values directly around the old landmark. In lower resolution levels, this effect is no longer observable. Areas with high values are not on target landmark locations but highlight other areas of the wing. Finally, on the higher resolution gate outputs, image borders have higher values than adjacent pixels which could indicate an effect Kayhan and Gemert [25] describe as boundary effect. The model might use image boundaries to learn positional encodings for landmarks which would lead to better results on images where landmarks are at the same position but worse for e.g. wings in other orientations. The same cannot be observed for NTM (see Appendix A.4.10). Here gate maps show mostly artifacts or zero values for all pixels. This could indicate that the NTM gate is learned to be ignored in the training process. Activity regularization on the gate output or additional tasks could potentially lead to the NTM being included in the localization task.

On Cephal, the trained baseline U-Net does not match results of Payer et al. [40] suggesting that some choices for the baseline should be improved to compare results of this work to state-of-the-art. However, differences between evaluated models still remain and improvements are valid nonetheless. Reducing the training set to 10 samples has a high impact on point-to-point error and doubles the outlier rate at $r = 10$. All models are unable to accurately localize landmark 15 and have difficulties to localize 5 and 9 which validates the findings from data exploration (Appendix A.1.3, A.1.4). Compared to Flywing, models on 66% of Cephal reduce localization error on some landmarks by including ground truth landmarks as input. On 6% training data, including landmarks does strongly lower PE indicating that in particular for very limited training data, providing landmarks as input could improve predictions. For the validation experiment, NTM- and attention gates are trained on the simultaneous task and show a decrease in point-to-point error, even with the NTMs write heads never writing to memory. Results for the NTM could originate from not overfitting to training data, whereas baseline does after 10000 training iterations. Particularly evaluation on Cephal should be re-done by using some early stopping mechanism to evaluate all models with lowest validation

loss. Contrary to Flywing, the baseline U-Net does not have lower PE on the iterative task. However, both extended architectures can gain from iterative landmark learning. attention gates have the best overall results on iterative training regardless of sample size NTM gates do not match simultaneous performance on 66% training data, indicating that the memory component is not used as expected. On 6% training data with $s_L = 5$, the NTM has a low outlier rate at $r = 10$ implying that while not predicting as close to ground truth, it is able to retain some shape information. Evaluating the gate output maps on Cephal, large areas of the map have low values decreasing the effect of pixels on the overall prediction. Comparing attention maps and NTM gate output, overall shapes are similar highlighting the same areas. However, while attention maps directly include positions of past landmarks, the NTM does not include them but instead only increases importance for a large connected area. These results show that given the same overall architecture, both models produce similar overall solutions for the gates. Again, image borders are consistently considered important showing that the task might in part be solved by using border based positional encodings (See Appendix A.5.5).

Overall, the effect of an iterative training has a stronger impact on Flywing compared to Cephal. Converting the landmark localization task to an iterative task introduces a recurrent inductive bias [2] which could explain the improved localization as it forces the network to consider other landmarks as decision support for new predictions. For more conclusive results, a number of further investigations should be conducted: The order of landmarks is likely detrimental for iterative learning. Grouping landmarks and re-ordering could change results both on Flywing and Cephal. In particular for Flywing, providing the start and endpoint of the target spline landmarks should improve localization. Evaluations for all models should be re-done by training models until the validation loss reaches a plateau. Models require different learning rates to even converge, making a fixed number of training iterations infeasible for proper comparison. In other works, NTMs seem to require more training iterations until they converge [19, 45, 11]. In this work, NTMs do not converge with low learning rates, which could be due to a lack of a well defined task. A loss function with matching signal and/or auxiliary tasks could improve task definition and NTM convergence. In the same way, altering the task by including ground truth labels instead of model predictions during training could help in early training steps. Finally, the NTM parameters are chosen arbitrarily and could be investigated separately to find a set that properly fits the task. The baseline U-Net could also be further improved as neither baseline includes the entire object in its receptive field. As this works baseline U-Net results are far from the localization U-Net results of Payer et al. [40], the training pipeline should be re-evaluated and their loss function should be included to match state-of-the-art results. Furthermore, the approach should be validated on datasets with high numbers of landmarks, where sequential learning could have a different impact and learning order could be more influential. Outside of validating findings, future work could investigate recurrent CNNs or LSTMs as memory components and compare them to results with NTM gates. The effect of including an encoder-decoder network should also be evaluated separately, as recent work demonstrates the effectiveness of pre-training models using only autoencoders [3]. Then, as image borders are consistently considered important in all models, the impact of the boundary

effect [25] should be further analyzed by e.g. including CoordConv layers [35]. Such layers are capable of modifying the degree of translation invariance and thereby mitigating or increasing the boundary effect. Finally, as this works NTM-based architecture is directly based on attention gates, it serves the same purpose. There are a number of other potential architectures with a U-Net to use a consistent memory in which could be explored (e.g. [26]).

8. Conclusion

In this work the effect of including memory on landmark localization was investigated. Conducted experiments show that an iterative task can improve localization by providing implicit memory in form of previous predictions as input. Yet, results are inconclusive, as the impact is dataset dependent. Explicit memory in form of an NTM can further lower point-to-point error but using an attention architecture is more straightforward, faster to train and yields better results on manually placed landmarks. Importantly, experiments also showed that models including an additional recurrent inductive bias can learn accurate landmark locations from a very small training set.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, pages 265–283, 2016. ISBN 9781931971331. URL `https://tensorflow.org`.

[2] Samira Abnar, Mostafa Dehghani, and Willem Zuidema. Transferring inductive biases through knowledge distillation. *arXiv preprint arXiv:2006.00555*, 2020.

[3] Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

[4] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

[5] Athanasios E Athanasiou. *Orthodontic cephalometry*. Mosby-Wolfe, 1995.

[6] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[7] Kaili Cao and Xiaoli Zhang. An improved Res-UNet model for tree species classification using airborne high-resolution images. *Remote Sensing*, 12(7): 1128, 4 2020. ISSN 20724292. doi: 10.3390/rs12071128. URL `https://www.mdpi.com/2072-4292/12/7/1128`.

[8] Hao Chen, Chiyao Shen, Jing Qin, Dong Ni, Lin Shi, Jack CY Cheng, and Pheng-Ann Heng. Automatic localization and identification of vertebrae in spine ct via a joint learning model with deep neural networks. In *International conference on medical image computing and computer-assisted intervention*, pages 515–522. Springer, 2015.

[9] Xi Chen, Erjin Zhou, Yuchen Mo, Jiancheng Liu, and Zhimin Cao. Delving Deep into Coarse-to-Fine Framework for Facial Landmark Localization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, volume 2017-July, pages 2088–2095, 2017. ISBN 9781538607336. doi: 10.1109/CVPRW.2017.260.

[10] Jen Tzung Chien and Kai Wei Tsou. Convolutional neural turing machine for speech separation. In *2018 11th International Symposium on Chinese Spoken Language Processing, ISCSLP 2018 - Proceedings*, pages 81–85. Institute of Electrical and Electronics Engineers Inc., 7 2018. ISBN 9781538656273. doi: 10.1109/ISCSLP.2018.8706621.

[11] Mark Collier and Joeran Beel. Implementing neural turing machines. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11141 LNCS, pages 94–104. Springer Verlag, 2018. ISBN 9783030014230. doi: 10.1007/978-3-030-01424-7{_}10.

[12] Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[13] Antonio Criminisi, Jamie Shotton, Duncan Robertson, and Ender Konukoglu. Regression forests for efficient anatomy detection and localization in ct studies. In *International MICCAI Workshop on Medical Computer Vision*, pages 106–117. Springer, 2010.

[14] René Donner, Bjoern H. Menze, Horst Bischof, and Georg Langs. Global localization of 3D anatomical structures by pre-filtered Hough Forests and discrete optimization. *Medical Image Analysis*, 17(8):1304–1314, 12 2013. ISSN 13618415. doi: 10.1016/j.media.2013.02.004.

[15] Thomas Ebner, Darko Stern, Rene Donner, Horst Bischof, and Martin Urschler. Towards automatic bone age estimation from MRI: localization of 3D anatomical landmarks. *Medical image computing and computer-assisted intervention : MICCAI*

... International Conference on Medical Image Computing and Computer-Assisted Intervention, 17:421–428, 2014.

[16] Ben Glocker, Johannes Feulner, Antonio Criminisi, David R Haynor, and Ender Konukoglu. Automatic localization and identification of vertebrae in arbitrary field-of-view ct scans. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 590–598. Springer, 2012.

[17] Ben Glocker, Darko Zikic, Ender Konukoglu, David R Haynor, and Antonio Criminisi. Vertebrae localization in pathological spine ct via dense classification from sparse annotations. In *International conference on medical image computing and computer-assisted intervention*, pages 262–270. Springer, 2013.

[18] V Grau, M Alcañiz, M C Juan, C. Monserrat, and C Knoll. Automatic localization of cephalometric landmarks. *Journal of Biomedical Informatics*, 34(3):146–156, 2001. ISSN 15320464. doi: 10.1006/jbin.2001.1014.

[19] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[20] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[21] Julian PT Higgins, Tianjing Li, and Jonathan J Deeks. Choosing effect measures and computing estimates of effect. *Cochrane Handbook for Systematic Reviews of Interventions*, pages 143–176, 2019.

[22] Peter Hirsch, Lisa Mais, and Dagmar Kainmueller. Patchperpix for instance segmentation. *arXiv preprint arXiv:2001.07626*, 2020.

[23] Sepp Hochreiter and Jürgen Schmidhuber. LSTM can solve hard long time lag problems. In *Advances in Neural Information Processing Systems*, pages 473–479, 1997. ISBN 0262100657.

[24] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, and Others. imgaug, 2020. URL `https://github.com/aleju/imgaug`.

[25] Osman Semih Kayhan and Jan C van Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14274–14285, 2020.

[26] Seung Wook Kim, Yuhao Zhou, Jonah Philion, Antonio Torralba, and Sanja Fidler. Learning to simulate dynamic environments with gamegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[27] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[28] Gregory Koch, Richard S Zemel, and Ruslan Salakhutdinov. Siamese Neural Networks for One-shot Image Recognition. *ICML - Deep Learning Workshop*, 2015. ISSN 19454589.

[29] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 10959203. doi: 10.1126/science.aab3050. URL `www.sciencemag.org`.

[30] Yann Le Poul, Yaqun Xin, Liucong Ling, Bettina Mühling, Rita Jaenichen, David Hörl, David Bunk, Hartmann Harz, Heinrich Leonhardt, Yingfei Wang, Elena Osipova, Mariam Museridze, Deepak Dharmadhikari, Eamonn Murphy, Remo Rohs, Stephan Preibisch, Benjamin Prud'homme, and Nicolas Gompel. Deciphering the regulatory logic of a drosophila enhancer through systematic sequence mutagenesis and quantitative image analysis. *bioRxiv*, 2020. doi: 10.1101/2020.06.24.169748. URL `https://www.biorxiv.org/content/early/2020/06/25/2020.06.24.169748`.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998. ISSN 00189219. doi: 10.1109/5.726791.

[32] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning, 5 2015. ISSN 14764687. URL `https://www.nature.com/articles/nature14539`.

[33] Haofu Liao, Addisu Mesfin, and Jiebo Luo. Joint vertebrae identification and localization in spinal CT images by combining short- and long-range contextual information. *IEEE Transactions on Medical Imaging*, 37(5):1266–1275, 2018. ISSN 1558254X. doi: 10.1109/TMI.2018.2798293. URL `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8269371`.

[34] David Liu, Kevin S. Zhou, Dominik Bernhardt, and Dorin Comaniciu. Search strategies for multiple landmark detection by submodular maximization. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2831–2838, 2010. ISBN 9781424469840. doi: 10.1109/CVPR.2010.5540016.

[35] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the CoordConv solution. In *Advances in Neural Information Processing*

Systems, volume 2018-Decem, pages 9605–9616, 2018. URL `https://github.com/uber-research/coordconv`.

[36] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[37] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[38] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

[39] Christian Payer, Darko Štern, Horst Bischof, and Martin Urschler. Regressing heatmaps for multiple landmark localization using cnns. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 230–238. Springer, 2016.

[40] Christian Payer, Darko Štern, Horst Bischof, and Martin Urschler. Integrating spatial configuration into heatmap regression based cnns for landmark localization. *Medical image analysis*, 54:207–219, 2019.

[41] Tomas Pfister, James Charles, and Andrew Zisserman. Flowing convnets for human pose estimation in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pages 1913–1921, 2015. ISBN 9781467383912. doi: 10.1109/ICCV.2015.222. URL `http://www.robots.ox.ac.uk/`.

[42] Šarūnas Raudys and Anil K Jain. Small sample size problems in designing artificial neural networks. In *Machine Intelligence and Pattern Recognition*, volume 11, pages 33–50. Elsevier, 1991.

[43] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[45] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-Learning with Memory-Augmented Neural Networks. In *33rd International Conference on Machine Learning, ICML 2016*, volume 4, pages 2740–2751, 2016. ISBN 9781510829008.

[46] snowkylin. TensorFlow implementation of Neural Turing Machines (NTM), with its application on one-shot learning (MANN), 2020. URL https://github.com/snowkylin/ntm.

[47] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014.

[48] Kien Tran, Hiroshi Sato, and Masao Kubo. Memory augmented matching networks for few-shot learnings. *International Journal of Machine Learning and Computing*, 9(6):743–748, 2019. ISSN 20103700. doi: 10.18178/ijmlc.2019.9.6.867. URL http://www.ijmlc.org/vol9/867-AM0018.pdf.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 5999–6009, 2017.

[50] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3637–3645, 2016.

[51] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[52] Ching-Wei Wang, Cheng-Ta Huang, Jia-Hong Lee, Chung-Hsing Li, Sheng-Wei Chang, Ming-Jhih Siao, Tat-Ming Lai, Bulat Ibragimov, Tomaž Vrtovec, Olaf Ronneberger, et al. A benchmark for comparison of dental radiography analysis algorithms. *Medical image analysis*, 31:63–76, 2016.

[53] Sheng Wang, Jian Peng, Jianzhu Ma, and Jinbo Xu. Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields. *Scientific Reports*, 6(1):1–11, 1 2016. ISSN 20452322. doi: 10.1038/srep18962. URL www.nature.com/scientificreports.

[54] Wei Wang, Kaicheng Yu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Recurrent U-Net for Resource-Constrained Segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2142–2151, 2019. URL https://github.com/.

[55] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[56] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *32nd International*

Conference on Machine Learning, ICML 2015, volume 3, pages 2048–2057, 2015. ISBN 9781510810587.

[57] Dong Yang, Tao Xiong, and Daguang Xu. Automatic vertebra labeling in large-scale medical images using deep image-to-image network with message passing and sparsity regularization. In *Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics*, pages 179–197. Springer, 2019.

[58] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pages 12744–12753, 2019. ISBN 9781510886988. URL `https://github.com/`.

[59] Jun Zhang, Mingxia Liu, and Dinggang Shen. Detecting Anatomical Landmarks from Limited Medical Imaging Data Using Two-Stage Task-Oriented Deep Neural Networks. *IEEE Transactions on Image Processing*, 26(10):4753–4764, 10 2017. ISSN 10577149. doi: 10.1109/TIP.2017.2721106.

[60] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pages 94–108. Springer, 2014.

A. Appendix

A.1. Data Analysis

A.1.1. Flywing Train/Test Mean Location Differences

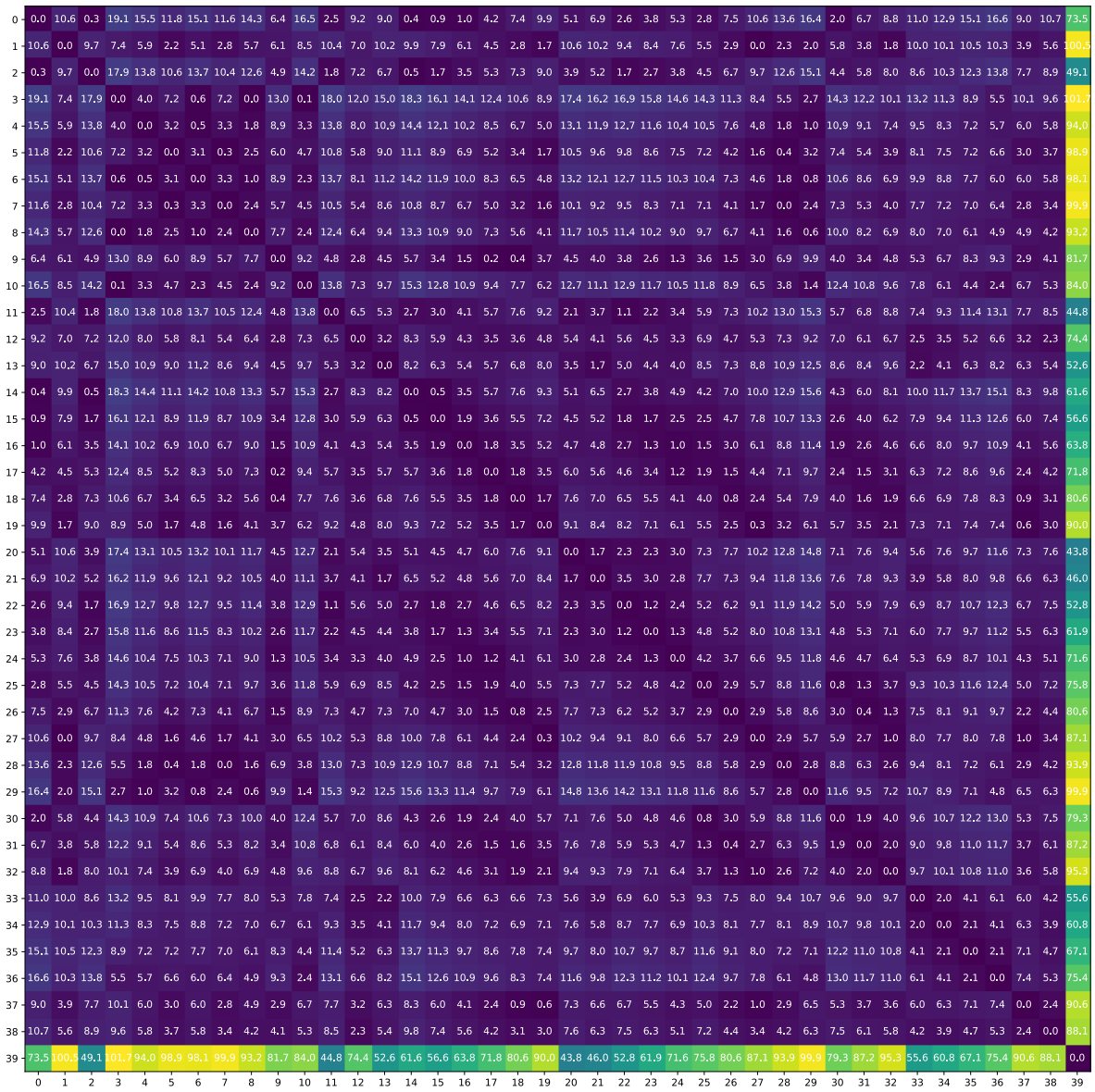


Figure 6: Absolute differences between mean relative landmark distances of train and test set for Flywing data

A.1.2. Flywing Train Location Standard Deviation

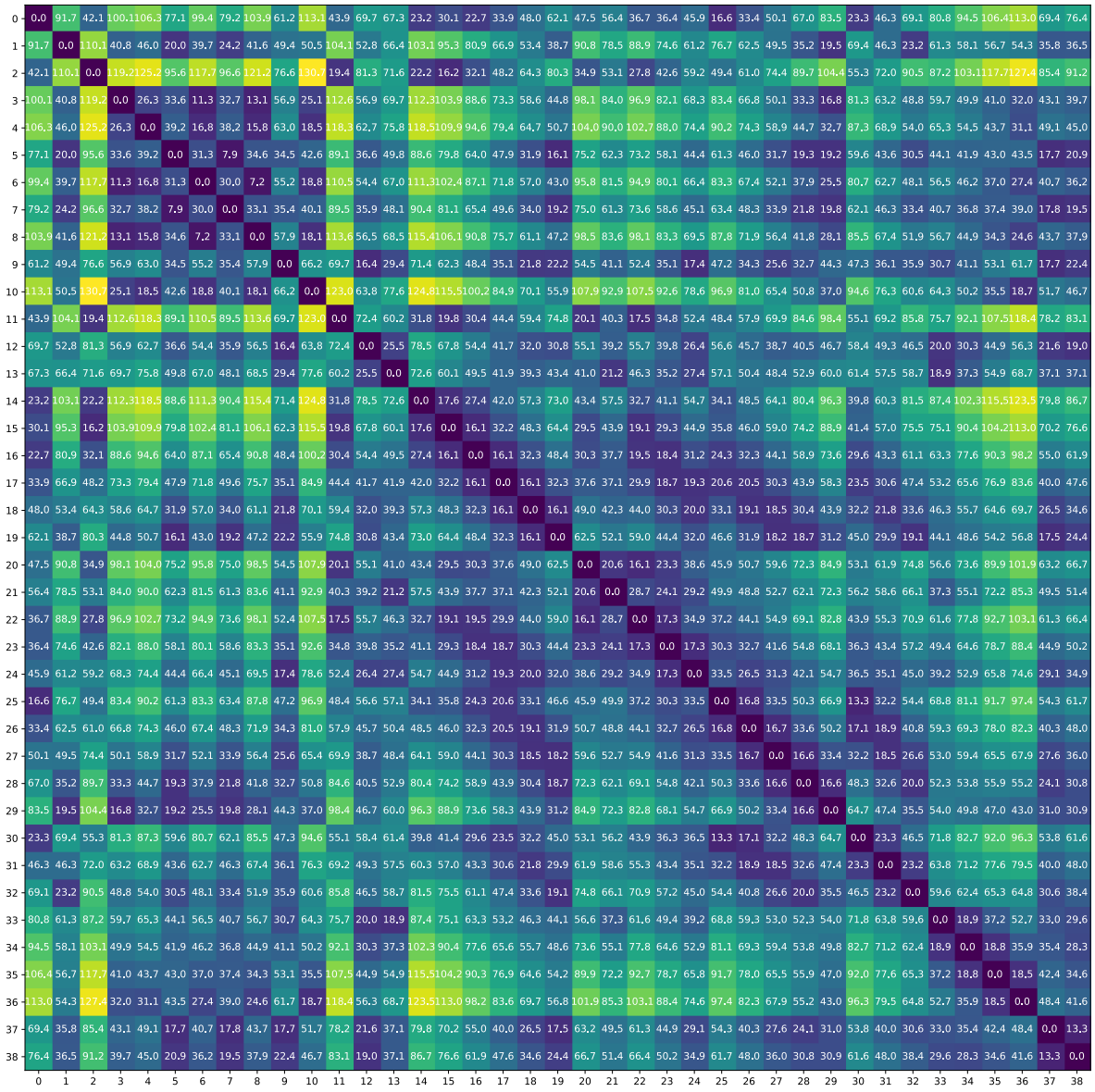


Figure 7: Standard deviation of relative landmark distances for Flywing training set

A.1.3. Cephal Train/Test Mean Location Differences

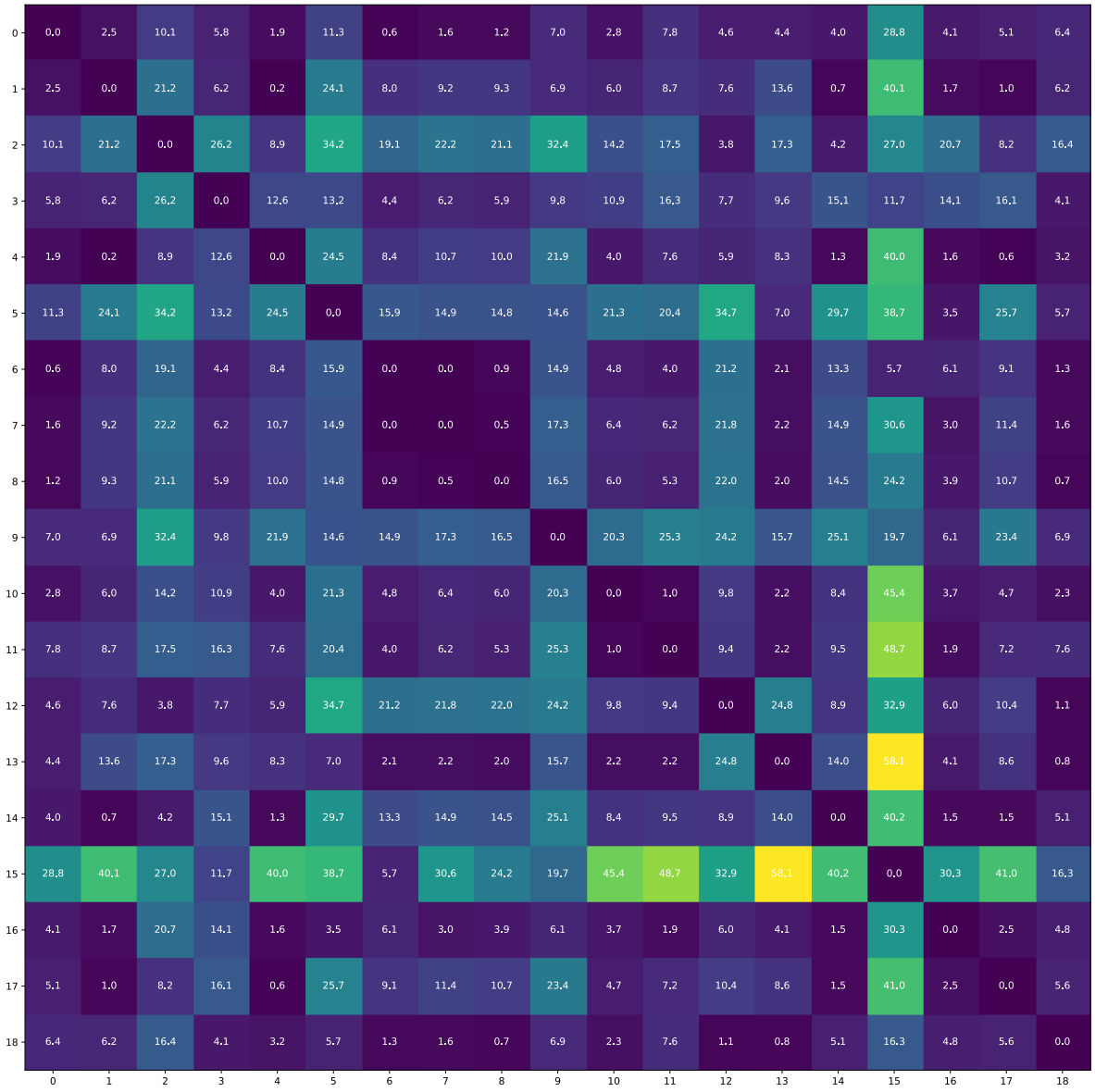


Figure 8: Absolute differences between mean relative landmark distances of train and test set for Cephal data. The 400_senior set is used in all experiments in this work.

A.1.4. Cephal Train Location Standard Deviation

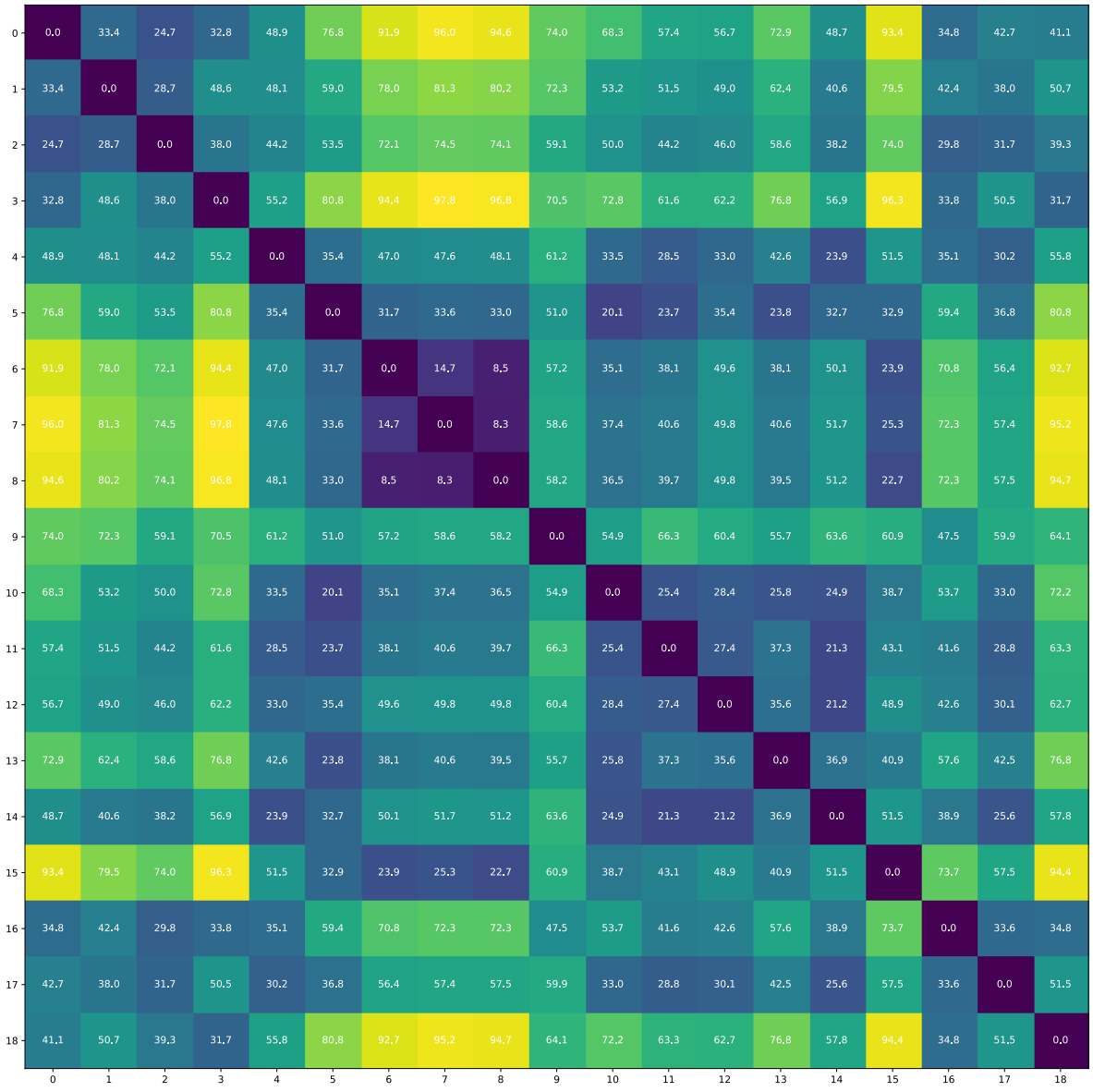


Figure 9: Standard deviation of relative landmark distances for Cephal training set.

A.2. Tuning Results

Dataset	levels	filters	filter size	Train loss	Test loss	PE_{px}	$\%ID$	Runtime (s)
Flywing	4	40	3	17885.938 +- 4798.729	11834.93 +- 3105.77	4.10 +- 12.11	0.96 +- 0.17	7456
			5	12594.150 +- 4665.822	8658.59 +- 3377.02	3.50 +- 13.19	0.96 +- 0.15	9183
	5	40	3	13714.851 +- 4975.420	8600.61 +- 3419.93	1.81 +- 6.08	**0.99 +- 0.04**	8020
			5	11559.031 +- 3893.704	9338.66 +- 3187.36	4.99 +- 17.62	0.92 +- 0.26	11560
	6	40	3	11852.613 +- 4640.965	**7868.16 +- 3486.68**	**1.75 +- 5.99**	**0.99 +- 0.04**	10863
			5	**11301.656 +- 4463.174**	8428.62 +- 2987.71	3.75 +- 13.83	0.94 +- 0.22	20759
Cephal	4	20	3	3639.155 +- 1324.989	13373.08 +- 1909.24	2.59 +- 1.72	**0.97 +- 0.09**	3846
			5	2087.676 +- 1210.332	13389.17 +- 1663.63	2.53 +- 1.54	**0.97 +- 0.09**	4294
		40	3	2034.603 +- 1164.412	12753.77 +- 1490.34	2.50 +- 1.77	**0.97 +- 0.09**	4128
			5	1501.736 +- 1031.009	12487.33 +- 1479.94	2.45 +- 1.90	**0.97 +- 0.09**	5725
	5	20	3	2621.050 +- 1294.343	14287.58 +- 1905.08	2.67 +- 1.58	0.96 +- 0.10	4006
			5	1914.839 +- 1260.151	13527.56 +- 1823.90	2.54 +- 1.51	0.96 +- 0.11	4842
		40	3	1643.077 +- 1082.281	12954.85 +- 1584.84	2.47 +- 1.56	0.96 +- 0.10	4671
			5	**1405.925 +- 997.698**	**12404.03 +- 1948.36**	**2.38 +- 1.60**	**0.97 +- 0.09**	8079
	6	20	3	2150.742 +- 1183.890	14735.74 +- 2128.61	2.69 +- 1.88	0.96 +- 0.10	4543
			5	1742.483 +- 1057.303	13471.34 +- 1506.67	2.54 +- 1.66	**0.97 +- 0.09**	7073
		40	3	1521.513 +- 991.182	13198.05 +- 1624.71	2.48 +- 1.55	0.96 +- 0.10	7449
			5	1499.532 +- 1001.126	12555.24 +- 1514.34	2.43 +- 1.78	**0.97 +- 0.09**	17158

Table 10: Tuning results using full training set with different parameters. All models are trained for 30000 iterations.

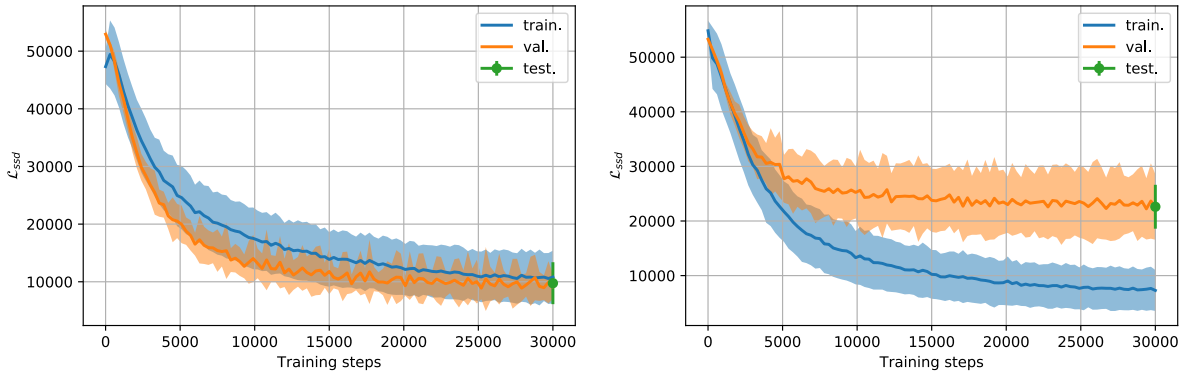
A.3. Applied Augmentations

Augmentation		Flywing	Cephal	Chance
Crop and Pad (constant 1e-5)		(-25%, 25%)	(-10%, 10%)	50%
Affine Transform	scale	(0.8, 1.2)	(0.9, 1.1)	50%
	translate	(-20%, 20%)	(-10%, 10%)	
	rotate	(-45° , 45°)	(-5° , 5°)	
	shear	(-16° , 16°)	(-5° , 5°)	
Elastic transform	alpha	(0.0, 40.0)	(0.0, 20.0)	50%
	sigma	(4.0, 8.0)	(2.0, 4.0)	
Flip left right		y	n	30%
Flip up down		y	n	30%

Table 11: Applied augmentations to the individual datasets using the imgaug library [24] with random chance for each transformation.

A.4. Flywing Results

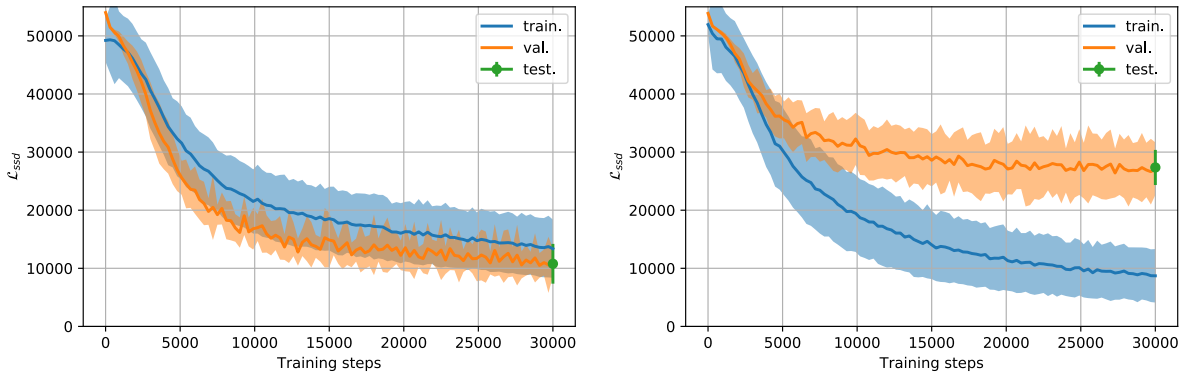
A.4.1. Baseline Simultaneous Training Loss



(a) 66% Training data (b) 6% Training data

Figure 10: Loss over training iterations for Flywing baseline simultaneous prediction

A.4.2. Attention Simultaneous Training Loss



(a) 66% Training data (b) 6% Training data

Figure 11: Loss over training steps for attention gate U-Net on simultaneous prediction task with Flywing

A.4.3. Simultaneous Training Baseline vs. NTM vs. Attention gate Point-to-Point Error

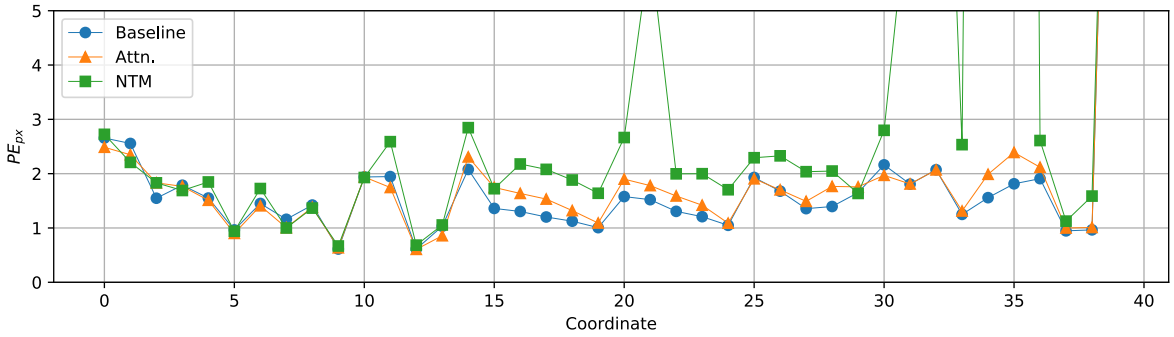
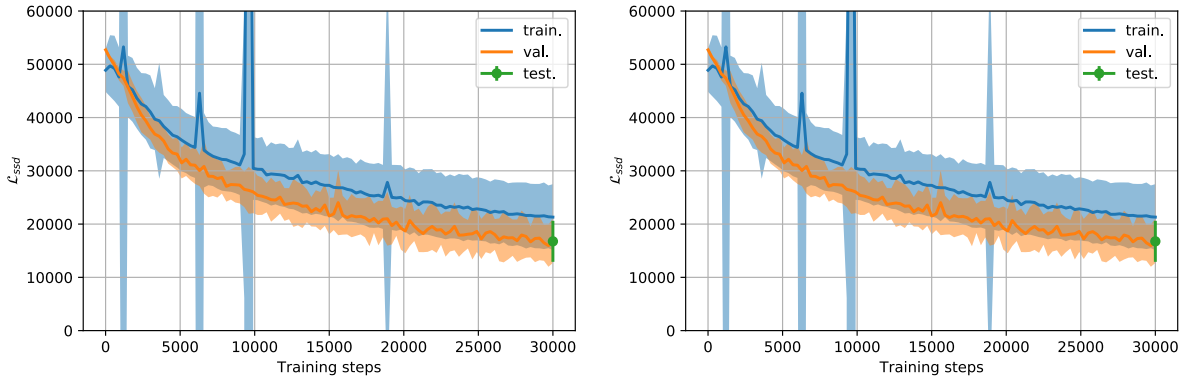


Figure 12: PE_{px} on simultaneous training task, 66% training data. Compared are Baseline against NTM and attention gate.

A.4.4. NTM Simultaneous Training Loss



(a) 66% Training data

(b) 6% Training data

Figure 13: Loss over training steps for NTM gate U-Net on simultaneous prediction task with Flywing

A.4.5. Iterative Training vs. Simultaneous Training Baseline Point-to-Point Error

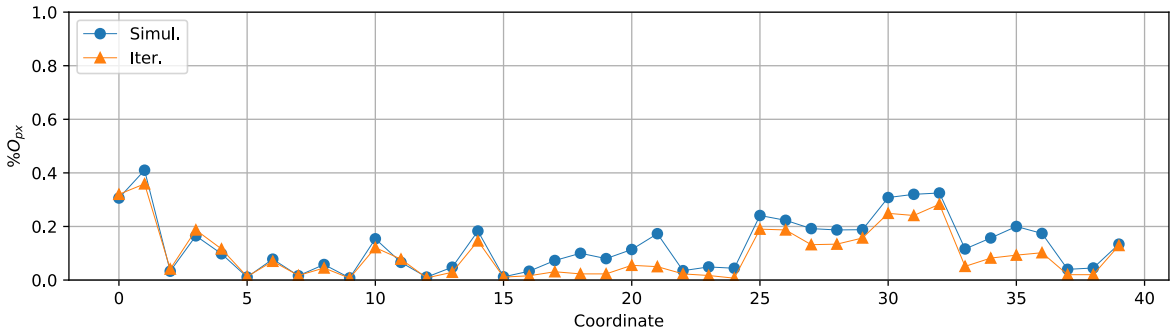


Figure 14: Per landmark $\%O_{px}$ for $r = 2px$ on 66% training data using Flywing

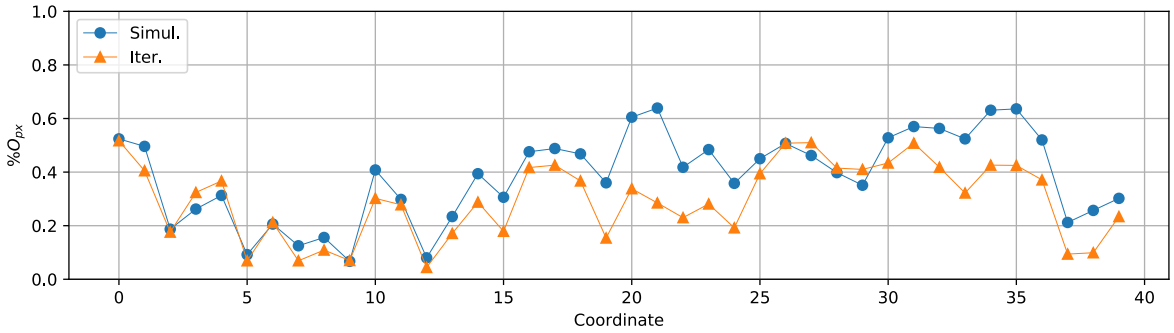


Figure 15: Per landmark $\%O_{px}$ for $r = 2px$ on 6% training data using Flywing

A.4.6. Iterative Training Baseline vs. NTM

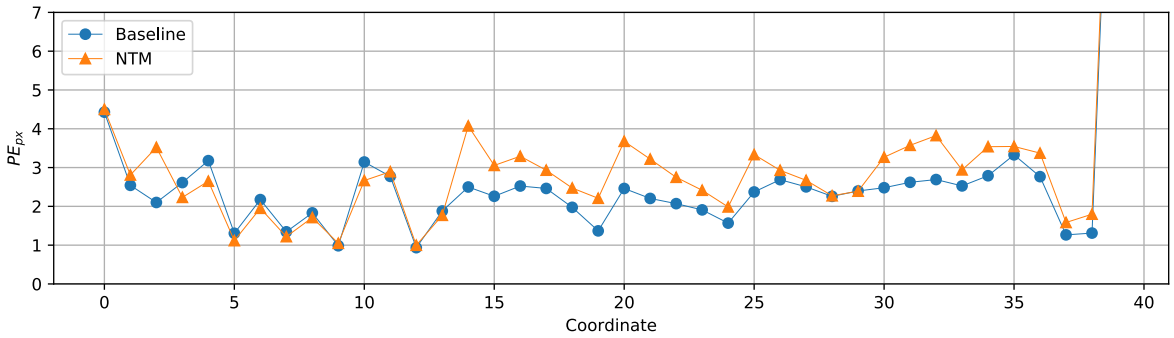


Figure 16: PE_{px} on 6% training data. Baseline U-Net with $s_l = 5$ vs. NTM gates with $s_l = 5$

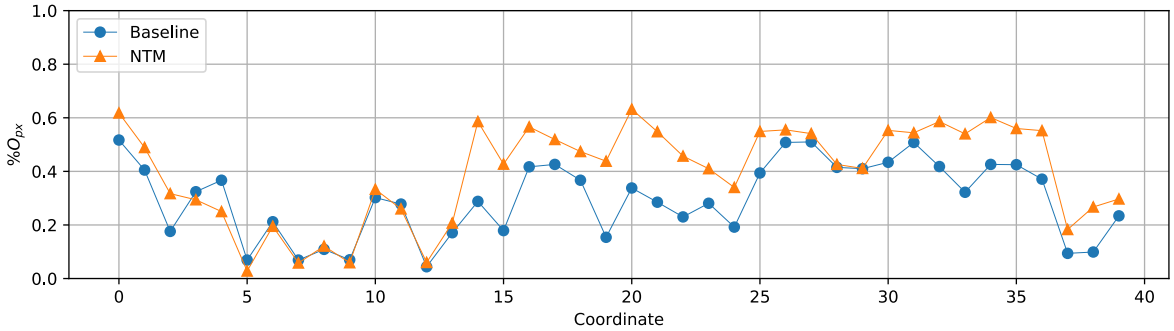
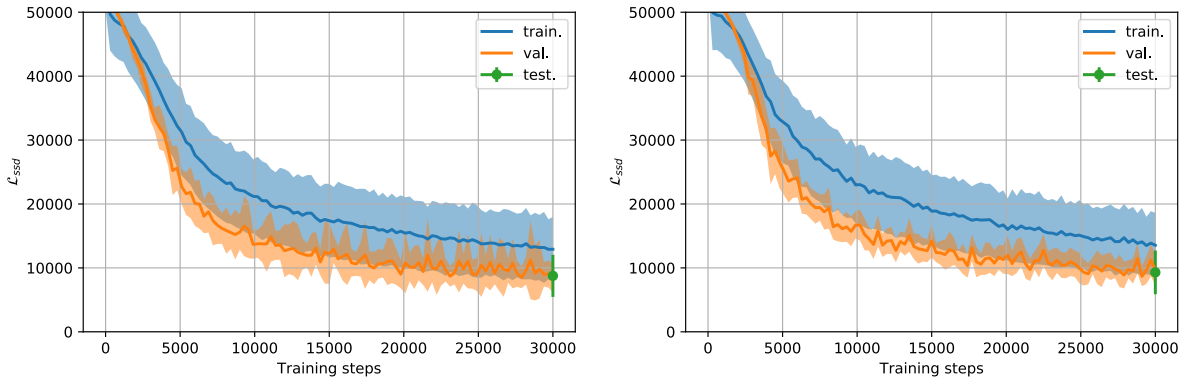


Figure 17: $\%O_{px}$ on 6% training data. Baseline U-Net with $s_l = 5$ vs. NTM gates with $s_l = 5$

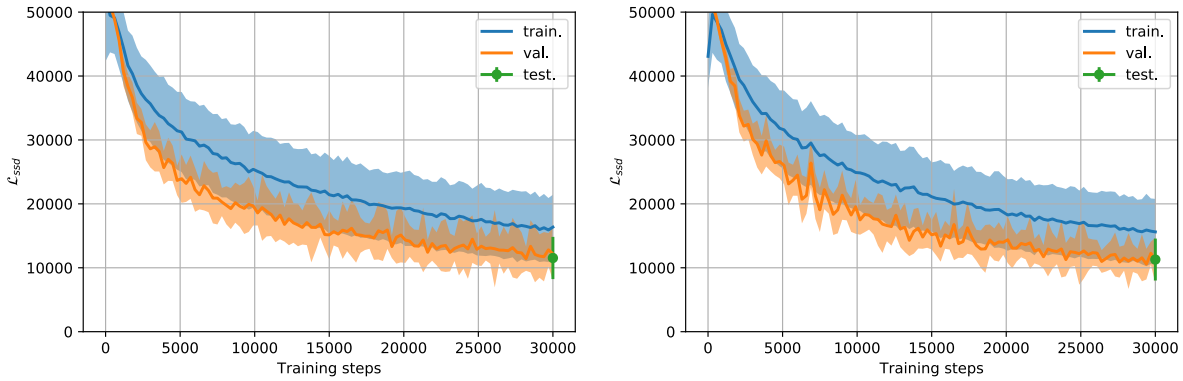
A.4.7. Attention Iterative Training Loss



(a) $s_l = 10$ (b) $s_l = 5$

Figure 18: Loss over training steps for attention gate U-Net on iterative task with 66% Flywing

A.4.8. NTM Iterative Training Loss



(a) $s_l = 10$ (b) $s_l = 5$

Figure 19: Loss over training steps for NTM gate U-Net on iterative task with 66% Flywing

A.4.9. Iterative Training Attention Gate Output Maps

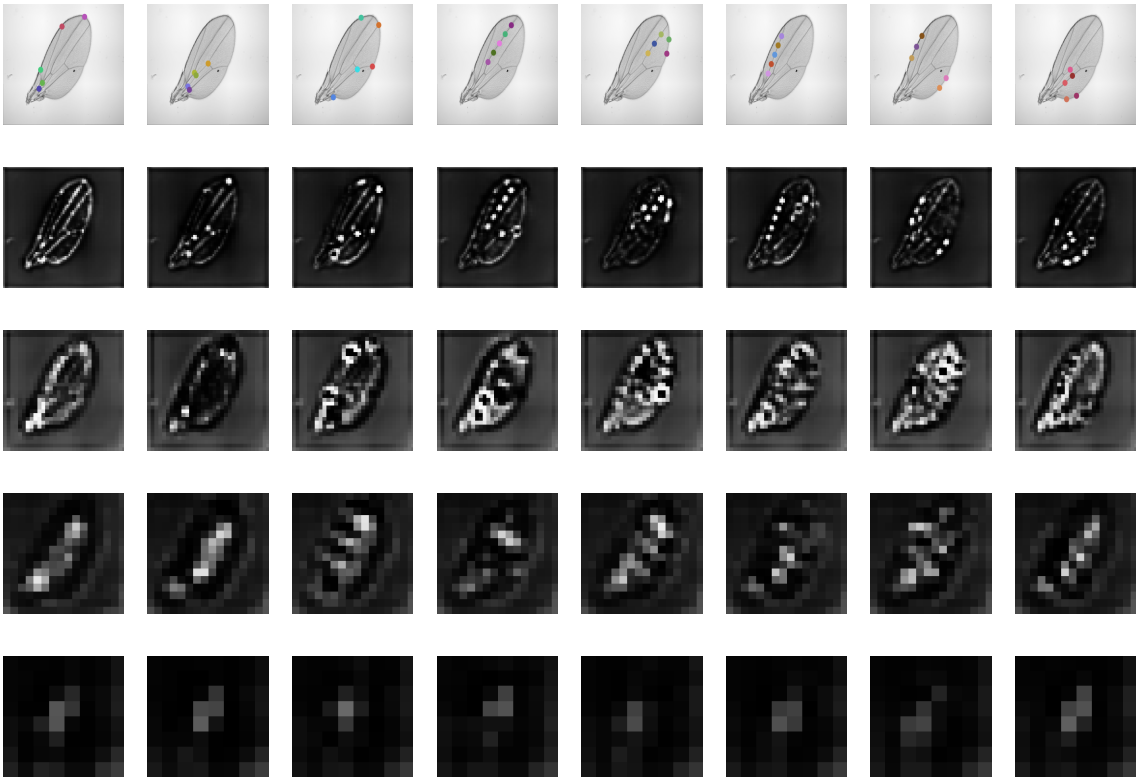


Figure 20: Attention gate output maps for individual levels at all sequence steps. Flywing, 66% training data.

A.4.10. Iterative Training NTM Gate Output Maps

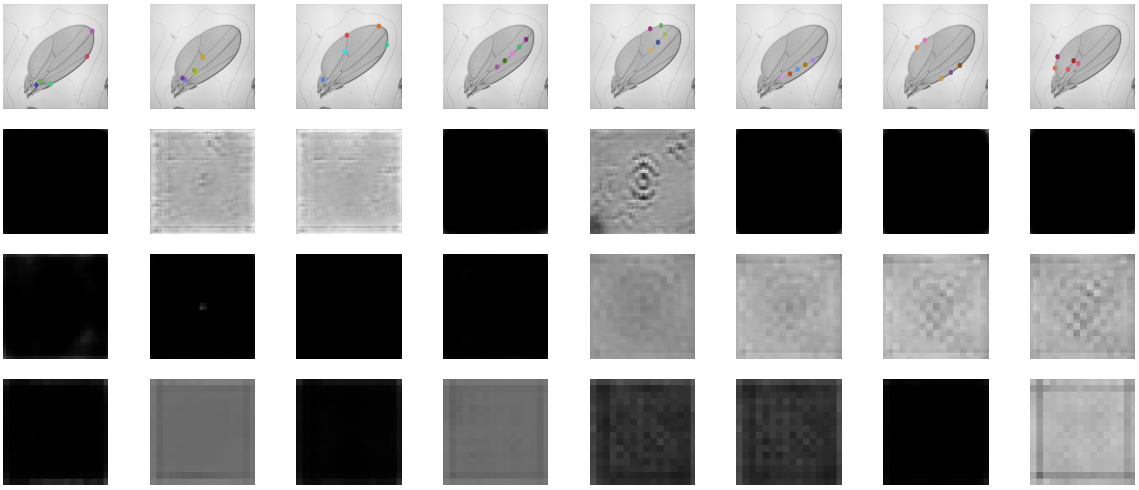
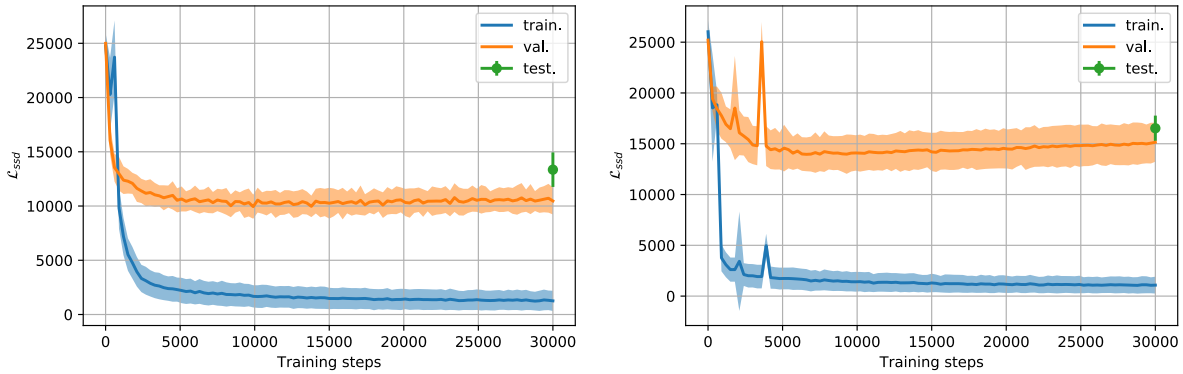


Figure 21: NTM gate output maps for individual levels at all sequence steps. Flywing, 66% training data. FOr NTM levels chosen are : 3,4,5 of 1-6

A.5. Cephal Results

A.5.1. Baseline Simultaneous Training Loss



(a) 66% Training data

(b) 6% Training data

Figure 22: Loss over training steps for baseline U-Net on simultaneous prediction task with Cephal

A.5.2. Provided Input Landmarks Point-to-Point Error

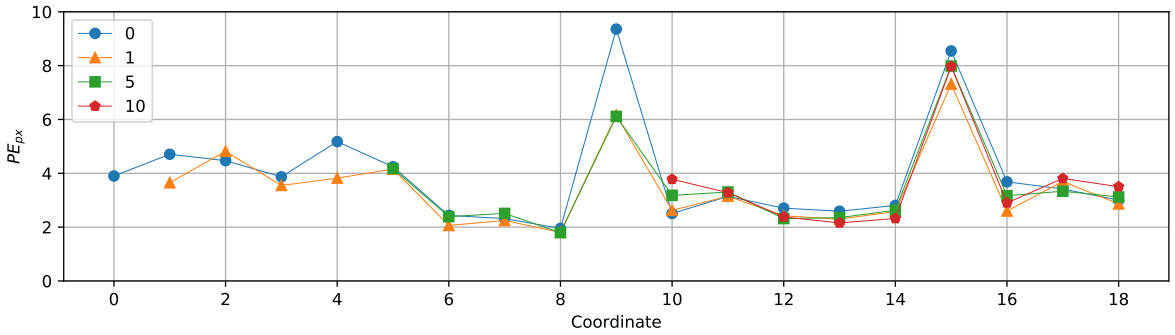


Figure 23: Per landmark coordinate distance for different numbers of provided landmarks using only simultaneous prediction with U-Net

A.5.3. Iterative Training vs. Simultaneous Training Baseline Point-to-Point Error

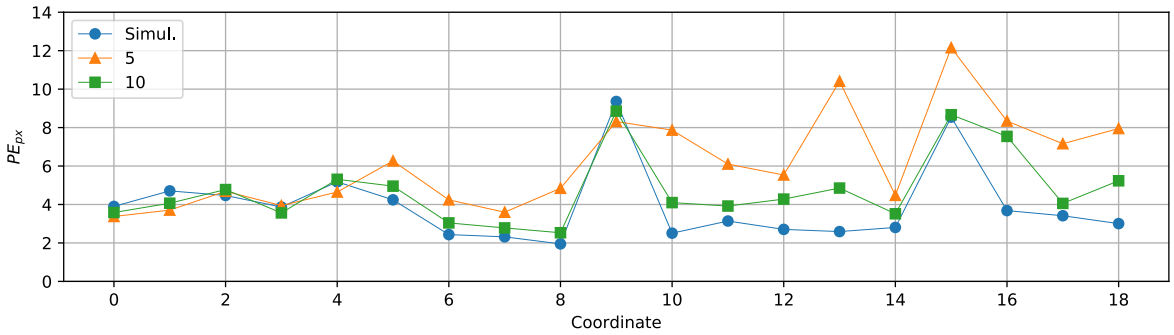
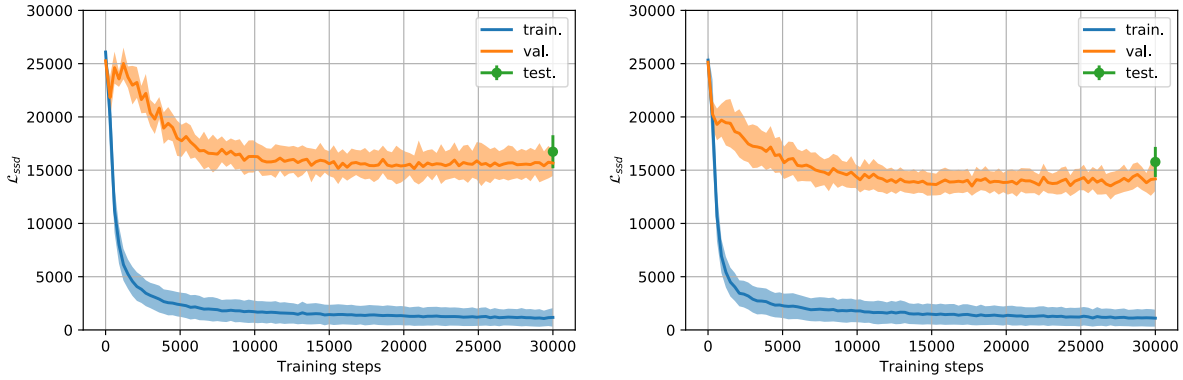


Figure 24: Per Landmark PE_{px} for $s_l \in [5, 10]$ compared to basline on iterative training task with Cephal

A.5.4. NTM Iterative Training Loss



(a) $s_l = 10$ (b) $s_l = 5$

Figure 25: Loss over training steps for NTM gate U-Net on iterative task with 6% Cephal

A.5.5. Iterative Training Gate Output Maps

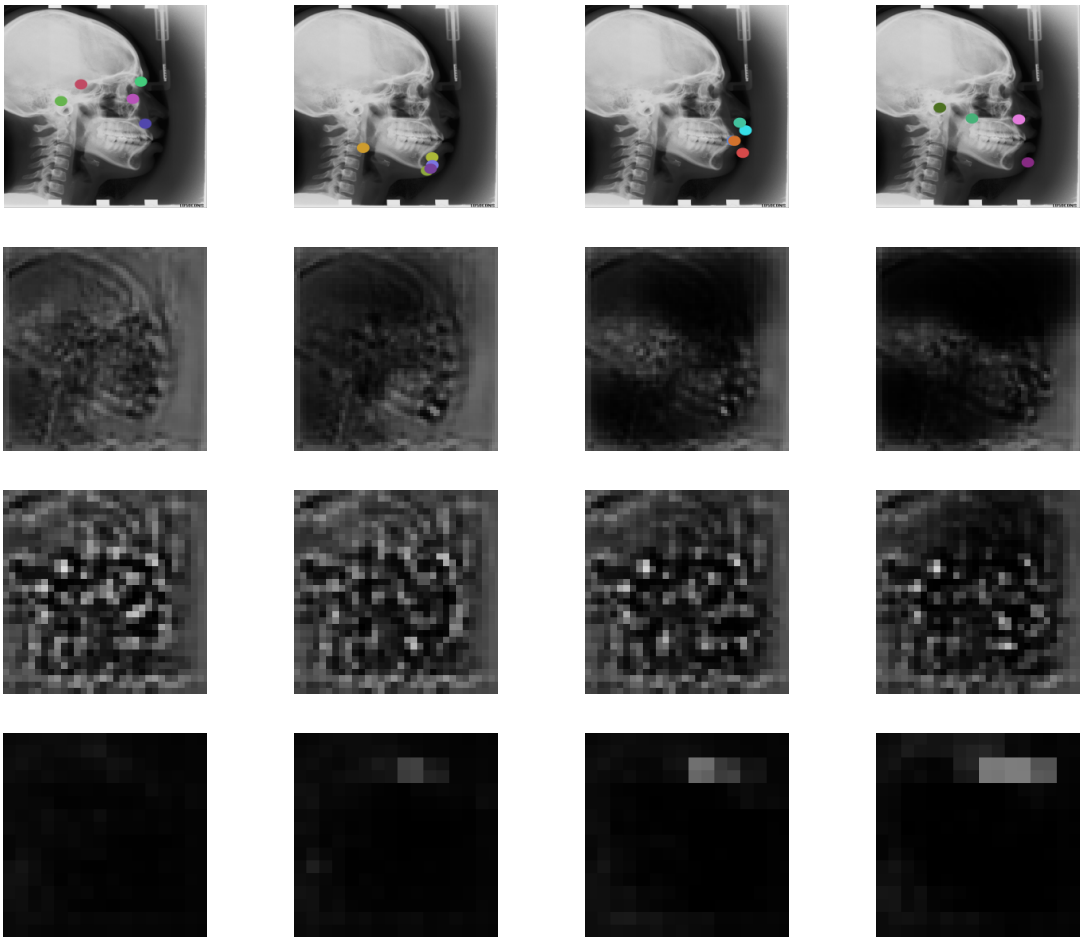


Figure 26: Attention gate output maps for individual levels at all sequence steps. Cephal, 66% training data.

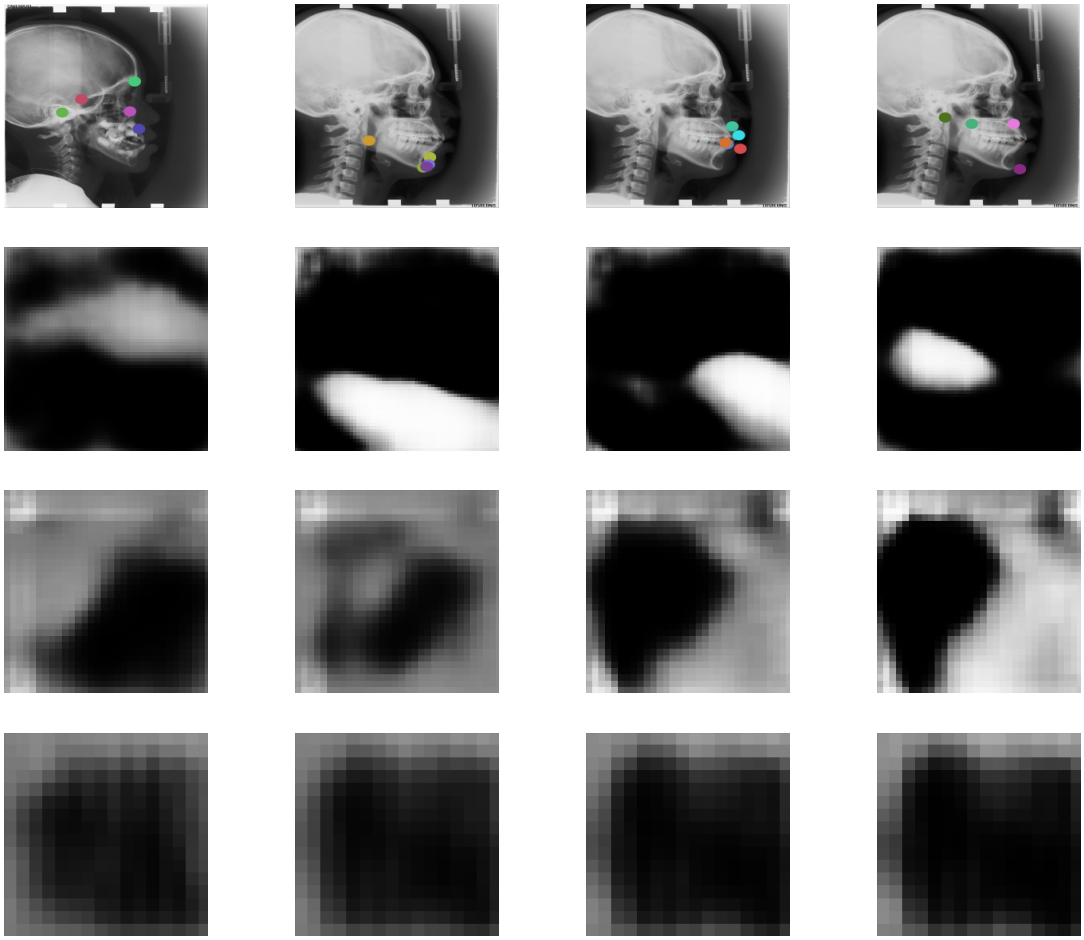


Figure 27: NTM gate output maps for individual levels at all sequence steps. Cephal, 66% training data. For NTM levels chosen are : 2,3,4 of 1-5

Declaration of Academic Honesty

I, Elias Baumann, hereby declare that I have not previously submitted the present work for other examinations. I wrote this work independently. All sources, including sources from the Internet, that I have reproduced in either an unaltered or modified form (particularly sources for texts, graphs, tables and images), have been acknowledged by me as such. I understand that violations of these principles will result in proceedings regarding deception or attempted deception.

Berlin, 16.11.2020 .....