Masterarbeit

Robust Registration to a template brain for the Drosophila larva

Harsha Yogeshappa

January 2, 2023

Contents

Chapter 1

Introduction

The purpose of this chapter is to introduce the main focus of this thesis and provide an overview of its key components. Specifically, we will present the research question that drives our study, the background and motivation for this work, the specific objectives and hypotheses we are aiming to test, and the overall structure of the thesis. Our goal is to provide a clear and concise introduction to the content of the following chapters, giving the reader a clear roadmap for the rest of the document.

1.1 Research question: Image Registration

Image registration is the process of aligning or registering two or more images so that they can be compared or combined. It is a fundamental technique in many fields, including medical imaging, computer vision, image processing, robotics, remote sensing, and computer graphics. Some examples of where image registration is used outside of medicine include:

1. **Computer vision**: Image registration is often used in computer vision to align images of the same scene or object, taken at different times or from different viewpoints, in order to facilitate image analysis and recognition tasks.
2. **Image processing**: Image registration is used in image processing to correct for distortions or align images from different sensors or modalities. This is often done to improve the accuracy of measurements or to combine information from multiple sources.
3. **Robotics**: Image registration is used in robotics to align images of the environment with a map or model of the environment in order to enable autonomous navigation.
4. **Remote sensing**: Image registration is used in remote sensing to align satellite or aerial images of the earth's surface, taken at different times or from different platforms, in order to monitor changes or extract information about the environment.
5. **Computer graphics**: Image registration is used in computer graphics to combine or align images or video frames in order to create seamless transitions or to generate special effects.

Image registration is widely used in the field of medicine, particularly in medical imaging. Some examples of its applications in medicine include:

1. **Aligning medical images from different modalities**: Medical images are often taken using different imaging technologies, such as CT, MRI, PET, and ultrasound, which can produce images with different contrasts, resolutions, and field of views. Image registration is used to align these images in order to facilitate diagnosis and treatment planning.
2. **Tracking the progression of a disease**: Image registration can be used to track the progression of a disease or the response to treatment by aligning medical images taken at different time points. This can be useful for monitoring the growth of a tumor, for example, or for evaluating the effectiveness of a treatment.
3. **Fusion of medical images**: Image registration can be used to fuse medical images from different modalities in order to provide a more complete picture of the anatomy or physiology of a patient. For example, PET and CT images can be fused to combine functional and structural information.

4. **Surgical planning and guidance**: Image registration can be used to align pre-operative images with real-time intra-operative images in order to guide surgical procedures or to assess the accuracy of a surgical approach.

5. **Image-guided therapy**: Image registration can be used to align images of a patient's anatomy with images of a therapeutic device, such as a catheter or a radiation beam, in order to guide the delivery of treatment.

There are many different techniques for image registration, including feature-based methods, intensity-based methods, and deformable registration methods. Feature-based methods are recommended if the images contain distinctive and easily detectable features. This is usually the case with remote sensing and computer vision where typically the image contains lot of details like roads, rivers, monuments etc. On the other hand, the medical images are not so rich in such details and thus intensity-based methods are usually employed. Sometimes, the lack of distinctive features in medical images are alleviated by introducing extrinsic features by an expert. Thus, the choice of technique depends on the specific needs of the application, such as the type and quality of the images being registered, the required accuracy of the registration, and the computational resources available.

1.2 Image Registration in the context of biological scans

Image registration is commonly used in the analysis of biological brain scans, such as those of drosophila larva, to align images taken at different times, or using different imaging modalities, or to compare images of different larvae. This allows researchers to compare and analyze the brain images in a common reference frame, which can provide valuable insights into the development and function of the brain.

For example, image registration can be used to track changes in brain structure and function over time, such as during development or in response to different stimuli. It can also be used to combine information from different imaging modalities, such as fluorescence microscopy and electron microscopy, to create a more comprehensive view of the brain.

In the context of drosophila larva, image registration can be particularly useful for studying the development and function of the nervous system, as these insects have a relatively simple and well-defined nervous. Image registration can help researchers to align and compare brain images from different experiments or conditions, and to accurately measure and analyze changes in brain structure and function.

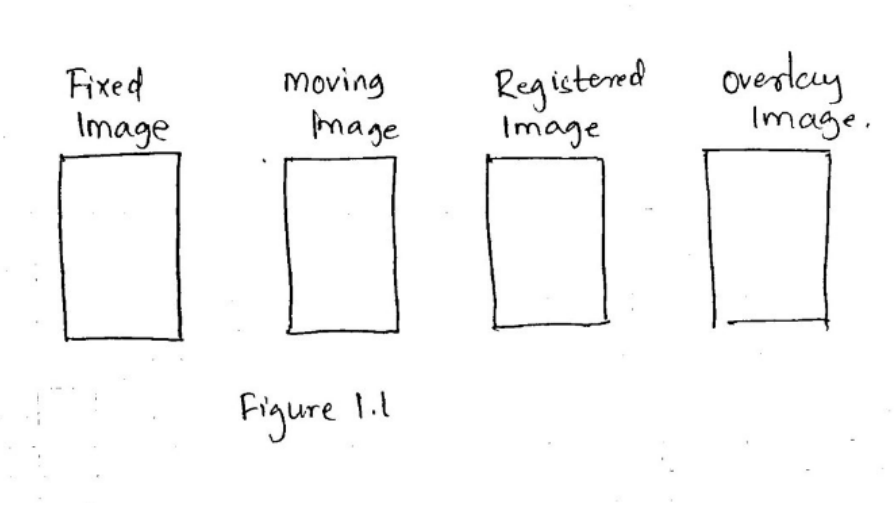

Figure 1.1: An image registration example

The above 1.1 shows an example of image registration of the drosophila larva brain obtained using the *larvalign* software [1]. In this example, image registration was used to align two images of a drosophila larva, which were taken as 3D volumes but are shown here as maximum intensity projections for visualization purposes. The input images are referred to as the fixed image and the moving image, and the output is a registered image in which the moving image is aligned with the fixed image. This alignment can make it easier to analyze the images.

1.3 Motivation

Image registration can be done in two methods - deep learning method and traditional method. Both methods aim to align multiple images of the same or different objects, taken at different times or from different viewpoints. However, they differ in the approaches they take to achieve this goal.

Traditional image registration methods typically rely on hand-crafted features or manually designed algorithms to align the images. On the other hand, image registration using deep learning methods relies on the use of neural networks to learn a mapping from the input images to the registered output. These methods can be more accurate and efficient than traditional methods, especially for complex images, as they are able to learn and adapt to the characteristics of the data. However, deep learning methods require a large amount of data for training.Traditional image registration can be prone to repeating errors and requires starting from scratch for each new image pair, whereas deep learning-based methods can learn from data and adapt to new image pairs more effectively.

In this study, we aim to investigate the effectiveness of deep learning-based image registration in aligning brain scans of drosophila larvae and compare it to the traditional method. We will also explore the possibility of using auxiliary information, such as landmark points, to guide the learning process and improve the accuracy and efficiency of image registration.

The focus of this thesis is to build upon the work presented in the *larvalign* paper [1], which is a 3D volume template and a registration method called *larvalign* for aligning brain scans of drosophila larvae. While *larvalign* showed promising results, it sometimes failed to accurately align images, particularly at the lower tip of the Ventral Nerve Cord (VNC) as shown in Figure 1.2.

Figure 1.2: An example of a registration failure of *larvalign* [1] at the lower tip of the Ventral Nerve Cord (VNC).

The figure 1.2 was plotted using MATLAB and displays four images as subplots in a row.

1. The **Fixed or Template image** is the first image in the figure and is the image against which the registration is performed using the larvalign software.
2. The **Moving or Subject image** is the second image in the figure and is the image that is being registered.
3. The **Moved or Registered image** is the third image in the figure and is the result of the registration.
4. The **Overlay image** is the fourth image in the figure and is an overlay between the fixed image and the moved image, which allows us to visually evaluate the quality of the registration.

In the overlay image, green structures represent the fixed image and magenta structures represent the moved image. When the two images overlap perfectly, the intensity values are displayed.

In this example case, the *larvalign* software was unable to accurately align the moving image with the fixed image, as indicated by the protruding magenta structure at the lower tip of the Ventral Nerve Cord (VNC) in the overlay. This is an example of a case where *larvalign* struggles to handle large deformations at the lower tip

of the Ventral Nerve Cord (VNC) and consistently produces registration errors in these situations. This issue occurs repeatedly whenever we attempt to register these image pairs again.

Larvalign is a non-learning method for image registration, which means it cannot improve with experience. We are exploring the use of a learning-based method of registration that can adapt and potentially outperform larvalign. Deep learning techniques have been effective at learning to compute the deformation field between image pairs [2–6], and we expect this approach to be more efficient in terms of registration time compared to larvalign.

1.4 Objectives and Hypothesis

The goal of this thesis is to investigate whether deep learning techniques like Voxelmorph [2] can be used to improve the robustness and efficiency of image registration for biological scans of drosophila larva that may exhibit large deformations. We hypothesize that deep neural network architectures have the ability to learn complex, inherent features and can therefore be as efficient as *larvalign* [1] in terms of accuracy in registration. Furthermore, we believe that by training these networks with more data, we can avoid the errors made by *larvalign* [1] and that the prediction time for image registration using deep learning will be significantly faster, due to the ability of these networks to learn a function for computing the deformation field between pairs of images.

1.5 Organization

Finally, in this section we provide an overview of the structure of the thesis, highlighting the main chapters and subtopics that we will cover. This will help the reader to understand the organization of the study and how each chapter contributes to the research question.

- **Literature review**: This chapter presents an overview of the existing research on the topic of the study, highlighting the main findings and debates in the field.
- **Drosophila larva biology and an overview of machine learning techniques**: This chapter provides an overview of Drosophila larva biology and a summary of the machine learning techniques that will be used in the study.
- **Methods and Implementations**: This chapter describes the research design, sample, and data collection and analysis procedures used in the study. It provides enough detail for readers to understand and replicate the study.
- **Results**: This chapter presents the findings of the study, including tables, figures, and other visualizations. In addition, we will conduct several ablation studies to explore the effect of different parameters on training.
- **Discussion**: This chapter interprets and contextualizes the results of the study, discussing their implications and limitations in relation to the literature review and research question. It also identifies areas for future research.
- **Conclusion**: This chapter summarizes the main findings and implications of the study, and will highlight the contribution of the research to the field.

Chapter 2

Literature review

Over the past few decades, given the numerous applications of image registration, a plethora of methods have been proposed. Until the success of AlexNet [7] in the 2012 ImageNet competition, many of the proposed methods formulated the registration problem as a physical problem to minimize the energy function [8] [9] [10] [11] [12] [13]. The success of AlexNet led to increased confidence in deep learning techniques and their widespread use in research. This can be seen in Figure **??**, which shows the increase in Deep Learning-based image registration work after 2012.

The adaptation to deep learning techniques led to exceptional performance results in many computer vision tasks such as object recognition, image segmentation, image classification, etc., which could not be achieved before, and the popularity of deep learning methods increased rapidly.

Not surprisingly, the current state-of-the-art methods for image registration are also based on deep learning methods, which are also used in this work. Traditional optimization methods minimize the dissimilarity function (or maximize the similarity function) between the fixed image and the moving image with respect to the registration parameters in an iterative process. In modern deep learning methods, the model is trained to learn the mapping between the fixed image and the moving image by optimizing an objective or cost function to perform registration in a single shot rather than iteratively. Although the training process can be resource intensive, the time required to perform registration after training a model is much less than the time required to perform mathematical optimization from scratch for each new image pair. Therefore, the methods can generally be divided into two types: traditional non-learning methods and modern deep learning methods. We'll talk about these in the next two sections, Traditional methods and Deep Learning Methods.

2.1 Traditional methods

Thirion's Demons algorithm [8] has been a popular choice for deformable registrations since the beginning of this century due to its linear complexity and simple implementation. The main idea of this work was to consider the object boundaries in one image as a semipermeable membrane and allow the other image, considered as a deformable grid model, to diffuse through these semipermeable boundaries by the action of effectors called demons located at these boundaries. The concept of demons was introduced by Maxwell in the 19th century to illustrate a paradox in thermodynamics, and this concept has been successfully adapted to image processing. If we want to match a moving image M with a fixed image F, it is assumed that the contour of an object O in F is a membrane and several demons are scattered on the contours of this object O. These demons act as local effectors, pushing the moving deformable model M to the inside of O if the corresponding point of M is labeled "inside" or to the outside of O if it is labeled "outside". Image matching is an iterative process to find the transformation form T that is the evolution of a family of transformations $\{T_0, T_1, ... T_i, ..\} \subset \tau$ (where τ is the set of admissible deformations).

Starting from an identity deformation T_0, for each iteration, the elementary force for each demon at the object boundary of the fixed image is calculated. From all elementary forces of the demons, the deformation T_{i+1} of T_i is calculated. The following figure 2.1 represents this iterative scheme.

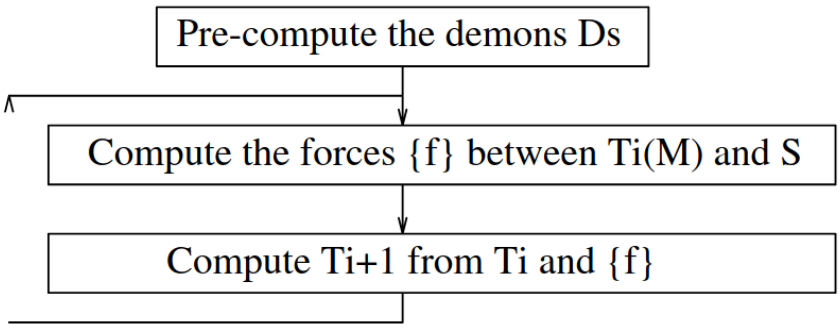

Figure 2.1: Iterative scheme in the case of diffusing models. [8]

One of the pioneering ideas in the field of large deformation registration was Christensen's modeling of the image as a fluid [9]. The image registration problem was formulated as a semilinear partial differential equation (PDE) system on the velocity field of the deformation, and it was shown how fixed point iteration generates a sequence of linear PDE systems. In [10], the demon algorithm was modified to perform curvature and fluid registration.

As mentioned earlier, there are a plethora of methods that have been proposed for image registration. Unfortunately, however, no universal method can be developed that is suitable for all registration problems. This may be due to the variety of images, assumed deformation types, required registration accuracy, application-dependent features, and other factors. To alleviate this problem, Klein et al. [11] in 2010 developed a software toolbox called *elastix* for intensity-based registration problems specifically used in medical image processing. The *elastix* software contains various optimization methods, multiresolution methods, interpolators, samplers, transformation models, and cost functions which is very helpful for the user to quickly compare different registration methods and select a satisfactory configuration for a particular application. *larvalign* [1] proposes such a method using *elastix*. *larvalign* [1] is an automatic and semi-automatic registration procedure, which is also the impetus for this thesis. In the following subsection, we shall talk more about what *larvalign* is and the results it achieved.

2.1.1 *larvalign*

As mentioned earlier, *larvalign* [1] is the stimulus for this work, i.e., this work is an extension of the work of *larvalign* [1]. *larvalign* [1] is both a standard template of the larval brain of the fruit fly *Drosophila melanogaster* and an automatic and semi-automatic registration method for registering different scans of the larval brain to this standard template. The *larvalign* [1] software package performs a specific sequence of image processing steps optimized specifically for the registration of microscopic images of the central nervous system (CNS) of third instar Drosophila larvae. The following is a brief overview of the registration algorithm, the evaluation of the results, and finally the results itself in *larvalign* [1].

Registration Algorithms

Image registration in *larvalign* is performed in two steps: global alignment of images (linear/affine/rigid registration) and local alignment of images (nonlinear/deformable/non-rigid registration).

The global alignment serves as initialization for the nonlinear registration. During the acquisition of brain samples, the samples can be randomly aligned and also mirrored in the z-direction. The goal of global alignment is to ensure that these acquired brain scans (moving images) have approximately the same orientation, size, and are in the same slice order in the z-direction as the standard template larval brain. Therefore, linear (affine) registration prior to nonlinear (deformable) registration would greatly facilitate the implementation of the non-rigid transformation.

Local alignment was performed using both non-parametric methods (variational methods) and parametric methods (B-spline models). Both methods were empirically evaluated to select the best of the two. Both categories have their own state-of-the-art toolkit for biomedical image registration, *ANTs* [12] and *elastix* [11], where *ANTs* image registration is best known for its symmetric diffeomorphic registration (SyN) algorithm [14]. An initial experiment was performed to *ANTs* SyN and *elastix* B-spline registration approach for pairwise nonlinear registration. Although comparable results were obtained with both methods, it was found that

registration with *ANTs* took more than 8 hours, whereas registration with *elastix* took about 4 minutes. Since the goal was accurate and *fast* image registration, the *elastix* approach was chosen.

Assessment

In addition to the visual inspection of the registered image, a quantitative value was defined to measure the quality of the registra¸tion. Although the correlation between the registered image and the fixed image captures the misalignment, such global measurements usually do not account for the local errors. To quantify local registration errors, statistical descriptors were developed for the regions at the entry points of the Ventral Nerve Cord (VNC) and thoracic nerve, similar to those used by Muenzing et al [15],. These descriptors were defined as VNC Terminal Error Indicator (**VI**) and Thoracic Nerve Error Indicator (**TI**), respectively. If these indicators had a value of less than 50%, this was an indication of a possible registration error. Apart from these two, another error called landmark registration error (**LRE**) was defined, which was nothing but the average of the Euclidean distance between all landmarks in the fixed image and the registered image.

Up to this point, registration is completely automatic and does not require human intervention. However, for those cases where **VI** and **TI** scored less than 50%, a framework for registration based on landmarks was developed as a fallback strategy. In the event that VNC registration fails (**VI** score less than 50 %), two landmarks were provided to guide this semi-automatic registration. In the event that thoracic registration failed (**TI** score less than 50 %), six landmarks were placed at all six entry points of the thoracic nerve. In contrast to automatic deformable registration, where only the correlation (NCC or MMI) between images controls the registration process, in semi-automatic deformable registration two similarity metrics control the registration process - the correlation metric and the Euclidean distance between the corresponding points [16].

In the tabular data provided in *larvalign* [1] it can be seen that for the partially failed registrations, landmark registration error (**LRE**) decreased after the semi-automatic registration was performed.

Results

The proposed *larvalign* registration method achieved an average landmark registration error (**LRE**) of **2.5 micrometers ± 1.5 micrometers** for the highest image quality data set, which is within the range of empirically measured human intra-rater variation measured on the same data set. However, the average landmark registration error (**LRE**) was **6.9 micrometers ± 6.4 micrometers** for the worst image quality within the representative data set, but it was noted that the intra-rater variation was measured on the high quality data and the baseline error could also be higher for the worst image quality data set.

2.2 Deep Learning Methods

Deep Learning (DL) belongs to a class of machine learning that uses neural networks with a large number of layers to learn the representation of the data and solve the task at hand. DL architectures had been limited in their ability to solve the then problems due to the vanishing gradient and the problem of overfitting. However, due to the availability of large datasets, large computational power in the form of GPUs and TPUs, and novel algorithms for training such deep networks, there has been a resurgence of interest in this concept. Recent works have also shown that they can even perform better than humans in some visual recognition tasks.

When it comes to image registration, there are broadly three strategies that are prevalent in the current deep learning literatures: (1) using deep learning networks to estimate discriminate features for two images to drive an iterative optimization strategy, (2) using deep learning regression networks to directly predict transformation parameters, and (3) using deep learning networks to directly predict the deformation field. [17]

Wu et al. [18] [4] were the first ones to use deep learning to obtain application specific feature in contrast to the supervised feature extraction or handcrafted feature selection methods, and then the learnt features were deployed to the classical Demons [19], [20], [21], [22], [23] and HAMMER methods [24], [25] with defined interpolation strategy, transformation model, and optimization alogirthm. These features were learnt from the data and hence is optimal representation for specific dataset. Thus, the then state-of-the-art registration methods were imporved by integrating these feature representations via deep learning. However, the registration process is still slow because the later transformation estimation is still iterative. Thus, methods to directly predict the transformation parameters have been proposed such that the registered image can be obtained directly by warping the moving image with the predicted transformation parameters.

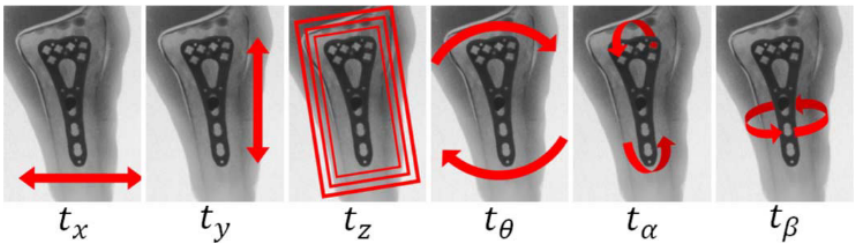

Figure 2.2: 6 transformation parameters in a 3D transformation. [26]

Miao et al. [26] were the first to use deep learning network to predict rigid transformation parameters. A rigid-body 3D transformation T can be parameterized by a vector $\boldsymbol{t}$ with 6 components of which 3 are in-plane (the translation parameters t_x and t_y, and rotation parameter t_θ) and 3 are out-of-plane (the translation parameter t_z and rotation parameters t_α and t_β) transformation parameters as shown in 2.2. A convolutional neural network (CNN) was used to predict the transformation parameters. Instead of regressing all parameters together, they were divided into three groups and regressed hierarchically.

Chee et al. [27] employed a CNN to predict all transformation parameters simultaneously, rather than predicting them in groups and then applying them in a hierarchical fashion. In this framework, their Affine Image Registration Network (AIRNet) took two images and outputs the rigid transformation parameters. The moving image is then warped by the resampler using transformation matrix. The training loss function was mean-squared error of the estimated transformation parameters.

Yang et al. [28], Rohe et al. [29], Jun et al. [30], Sokooti et al. [6] and many others used deep learning approaches to predict the deformation field in a supervised fashion either with ground truth data or synthetic data.

Interestingly, in one another work by Yang et al. [31], Generative Adversarial Network was used to estimate the rigid transformation. Generator network was trained to predict the rigid transformation parameters, while the discriminator was trained to discern between the images that were warped with predicted transformation parameters and the images that were warped with the ground truth transformation parameters. Both Eucledian distance to ground truth and an adversarial loss term were used to construct the loss function.

2.2.1 Unsupervised Transformation Estimation

Despite the success of the above methods, the difficulty of obtaining reliable ground truth in the form of a warped image or deformation field is a major obstacle. This has motivated a number of research groups to explore unsupervised approaches. In this thesis also, an unsupervised approach is pursued, and thus this separate subsection is devoted to it in order to express its importance.

A key innovation that has been useful for all work following the unsupervised method is the spatial transformer network (STN) [32]. The spatial transformer network is a differentiable module that can be inserted into existing convolutional architectures, giving neural networks the ability to spatially transform feature maps. Since it is a differentiable module, it can also be trained together in an end-to-end fashion through backpropagation. In the paper, "A Survey on Deep Learning in Medical Image Registration", [33] a nice visualization aptly describes the workflow of unsupervised deep learning methods for registration.

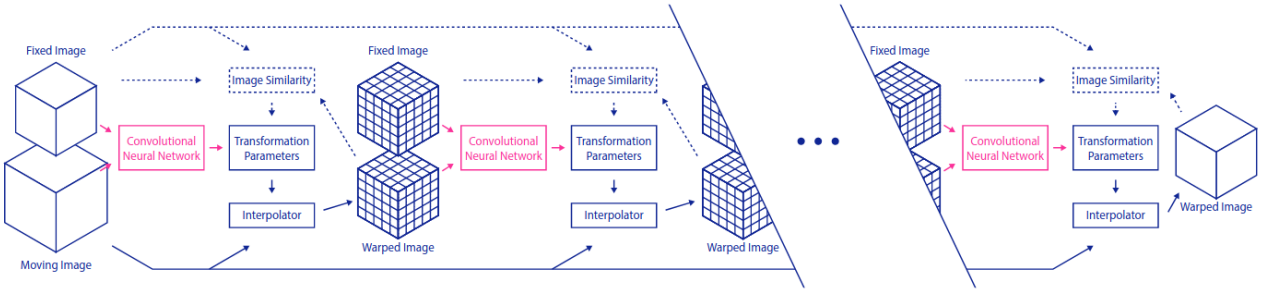

Figure 2.4: Schematic representation of the DLIR framework for hierarchical training of multi-stage ConvNet for multi-resolution and multi-level image registration. [3]

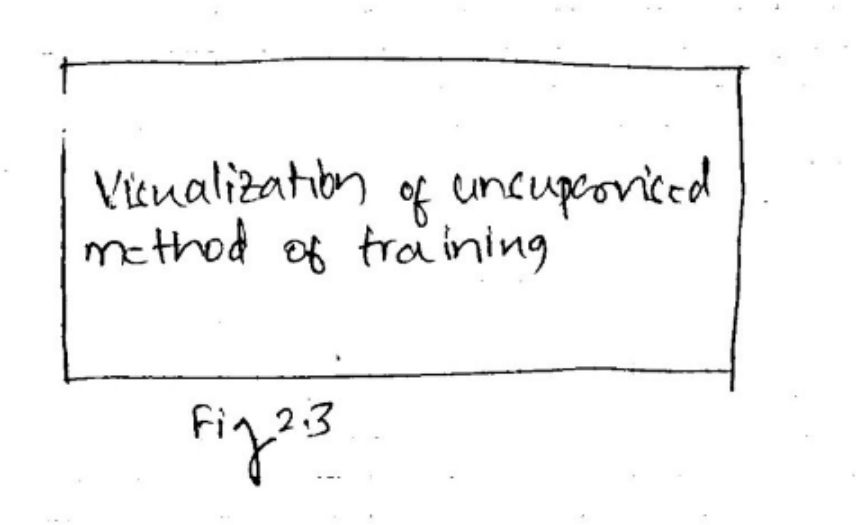

Figure 2.3: Visualization of unsupervised method of training deep neural networks for registration. [33]

One of the important works in this area is by de Vos et al. [3]. In their work they proposed a Deep Learning Image Registration (DLIR) framework for unsupervised affine and deformable registrations. The framework is trained for image registration by exploiting image similarity between fixed and moving image pairs. In the DLIR framework, the transformation parameters are learnt by analysing the fixed and the moving image pairs. The predicted transformation parameters are then used to make a dense displacement vector field (DVF). DVF is used to resample the moving image into a warped image. Unlike many works that expects the moving image to be already to affinely aligned with the fixed image, the DLIR is designed for affine as well as deformable registration. Additionally, the framework also supports multi-stage ConvNets for registration of coarse-to-fine complexity in multiple-levels and multiple image resolutions by stacking ConvNets in a serial fashion. It is also said that such hierarchical mutli-stage strategy makes conventional iterative image registration less sensitive to local optima and image folding [34]. Each stage is trained for its specific registration task, while keeping each of its weights fixed. After training, the multi-stage ConvNet is applied for one-shot image registration, similar to a single ConvNet.

In the next section, the current state-of-the-art methods which are also unsupervised is explained in detail.

2.3 State-of-the-art methods

There are many proposed methods in the literature for learning a function for medical image registration using convolutional neural networks in an unsupervised manner. In this work, we consider some of these methods, which are the current state of the art, as base methods and adapt them to achieve a successful registration result for our larval brain scans.

2.3.1 VoxelMorph: An Unsupervised Learning Model for Deformable Medical Image Registration

Like many other deep learning methods, the method proposed in VoxelMorph [35] [2] also formulate registration as a function that maps an input image pair to a deformation field that matches the moving and fixed images. However, unlike other methods, two different training strategies were proposed in this work. In the first (unsupervised) setting, the model is trained to maximize the similarity between the fixed image and the registered image based on the image intensities. In the second setting, auxiliary information in the form of anatomical segmentations were used to steer the network in the right direction of learning. It was found that the model trained with auxiliary information performed better than the model trained without at the test time. The performance of both settings was comparable to the then state-of-the-art methods.

3D MR scans of the human brain from healthy and sick people of different age groups were used for the study. In this work, more attention was paid to the slower deformable transformation than to the comparatively faster affine transformation. Therefore, it is assumed that the moving image and the fixed image are already affine aligned.

The registration problem was formulated to find the optimal displacement vector field such that the dissimilarity between the fixed and registered images is the smallest. In addition, a smoothing loss was included to act as a regularizer to enforce a spatially smooth deformation.

$$\begin{aligned} \hat{\phi} &= \arg\min_{\phi} \mathcal{L}(f, m, \phi) \\ &= \arg\min_{\phi} (\mathcal{L}_{sim}(f, m \circ \phi) + \lambda \mathcal{L}_{smooth}(\phi)) \end{aligned} \tag{2.1}$$

where m is moving image, f is fixed image, ϕ is the deformation field, and $m \circ \phi$ represents m warped by ϕ, function $\mathcal{L}_{sim}(\cdot, \cdot)$ measures the similarity between the two input images, $\mathcal{L}_{smooth}(\cdot)$ imposes regularization, and λ is the regularization trade-off parameter.

The common metrics used for $\mathcal{L}_{sim}$ include intensity mean squared error, normalized cross correlation, and mutual information. The latter two are useful when volumes have varying intensity distributions and contrasts.

VoxelMorph CNN Architecture

Figure 2.5 depicts the network used in VoxelMorph. The network accepts both moving image m and fixed image f which is further concatenated into a 2 channel 3D image before fed to encoder-decoder architecture with skip connections. 3D convolutions in both the encoder and decoder stages use a kernel size of 3, a stride of 1, and same padding. Each convolution is followed by a leakyReLU layer with parameter 0.2. The convolution layers capture hierarchical features of the input image pair, to estimate ϕ.

In the encoder, max-pooling is performed to reduce the spatial dimensons in half at each layer. Thus, successive layers of the encoder operate over coarser representations of the input. This multilayer encoder network is used to transfer the high-dimensional 3D image patches into the low-dimensional feature representations.

In decoder, alternate between upsampling, convolutions and skip connections are done. The decoder network is used to recover 3D image patches from the learned low-dimensional feature representations, acting as feedback to refine the inferences in the encoder network.

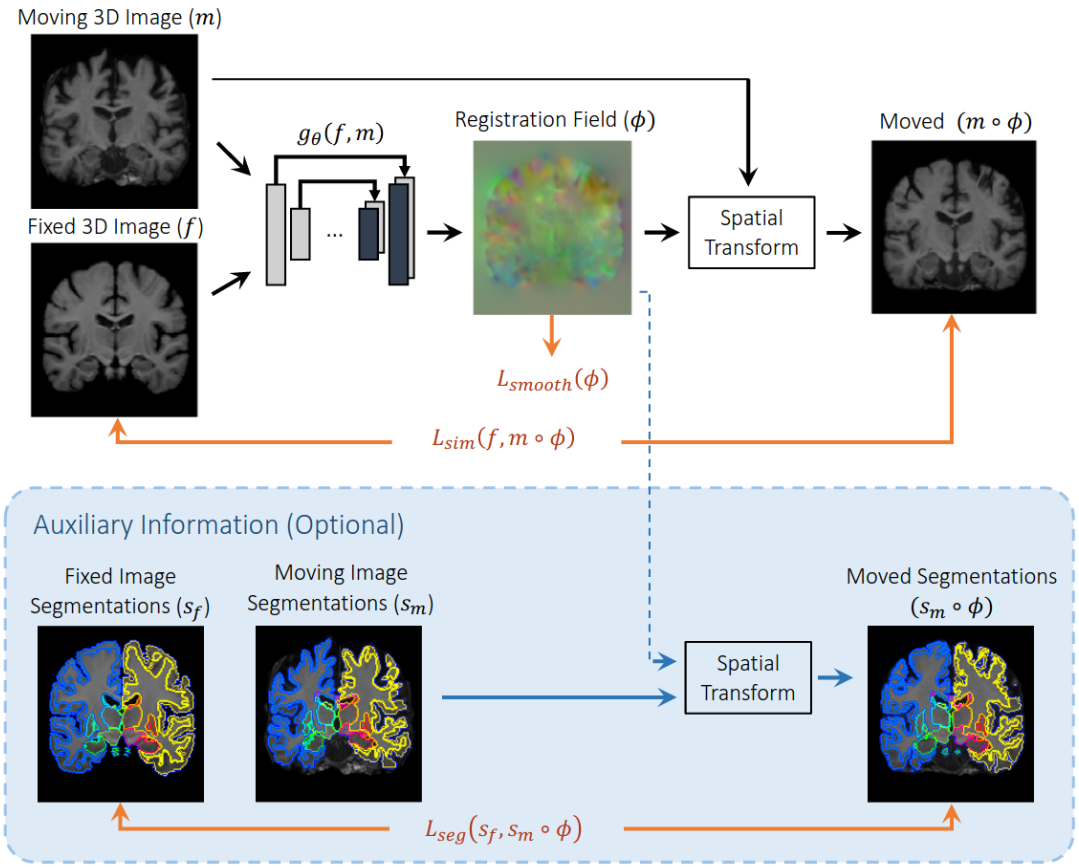

Figure 2.5: Overview of the VoxelMorph method. [2]

An important architectural constraint is that the receptive fields of the convolutional kernels of the smallest layer should be at least as large as the maximum expected displacement between corresponding voxels in f and m.

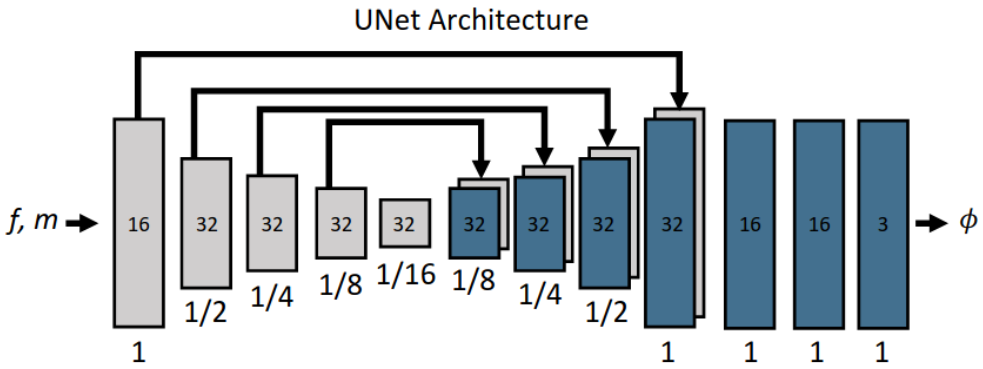

Figure 2.6: Implementation of the U-Net convolution architecture. Each rectangle represents a 3D volume created from the previous volume using a 3D convolutional network layer. The spatial resolution of each volume with respect to the input volume is given below. Multiple 32-filter convolutions are used in the decoder, each followed by an upsampling layer to bring the volume back to full resolution. The arrows represent skip connections that chain encoder and decoder functions. [2]

The figure 2.7 shows the average dice scores and runtime results for affine alignment, ANTs, NiftyReg and VoxelMorph. Standard deviations across structures and subjects are in parantheses. The average Dice score is computed over all the structures and subjects.

Method	Dice	GPU sec	CPU sec	$\lvert J_\phi \rvert \leq 0$	% of $\lvert J_\phi \rvert \leq 0$
Affine only	0.584 (0.157)	0	0	0	0
ANTs SyN (CC)	0.749 (0.136)	-	9059 (2023)	9662 (6258)	0.140 (0.091)
NiftyReg (CC)	0.755 (0.143)	-	2347 (202)	41251 (14336)	0.600 (0.208)
VoxelMorph (CC)	0.753 (0.145)	0.45 (0.01)	57 (1)	19077 (5928)	0.366 (0.114)
VoxelMorph (MSE)	0.752 (0.140)	0.45 (0.01)	57 (1)	9606 (4516)	0.184 (0.087)

Figure 2.7: Average Dice scores and runtime results for affine alignment, ANTs, NiftyReg and VoxelMorph. [2]

In another setup, the trained VoxelMorph was tested on the (unseen) Buckner40 dataset containing 39 scans. VoxelMorph with instance specific optimization was also performed. The dice scores in 2.8 show that VoxelMorph using cross correlation loss behaves comparably to ANTs and NiftyReg using the same cost function. Further, VoxelMorph with instance-specific optimization improved the results.

Method	Dice
Affine only	0.608 (0.175)
ANTs SyN (CC)	0.776 (0.130)
NiftyReg (CC)	0.776 (0.132)
VoxelMorph (MSE)	0.766 (0.133)
VoxelMorph (MSE) inst.	0.776 (0.132)
VoxelMorph (CC)	0.774 (0.133)
VoxelMorph (CC) inst.	0.786 (0.132)

Figure 2.8: Results for manual annotation experiment. [2]

Finally, the authors conclude that the VoxelMorph is a general learning model, and is not limited to any particular image type or anatomy - may be useful in medical image registration applications such as cardiac MR scans or lung CT images. Also, authors are of the opinion that with approporiate loss functions like mutual information, the model can also perform multimodal registration.

2.3.2 Cascade Networks for Unsupervised Medical Image Registration

Following the success of Voxelmorph [2], improvised frameworks like Volume Tweening Network (VTN) [36] and architectures like Recursive Cascaded Networks [37] were proposed. Both the methods aimed at addressing the issues where the registration meant resolving large displacement fields.

Although the authors note in the VoxelMorph paper [2] that the model may be useful for medical image registration applications such as MR scans of the heart or CT scans of the lung, Zhao et al. [36] note that VoxelMorph does not work well when large shifts are present between images. Therefore, both papers [36] and [2] offer a solution for dealing with such scenarios where large deformation fields are present by cascading the networks. In both the methods, final deformation field is a composition of field from the subnetworks. That is, the final flow prediction is composed of an initial affine transformation and ϕ_1, ϕ_2, $\cdots$, ϕ_n, each of which only performs a rather simple displacement.

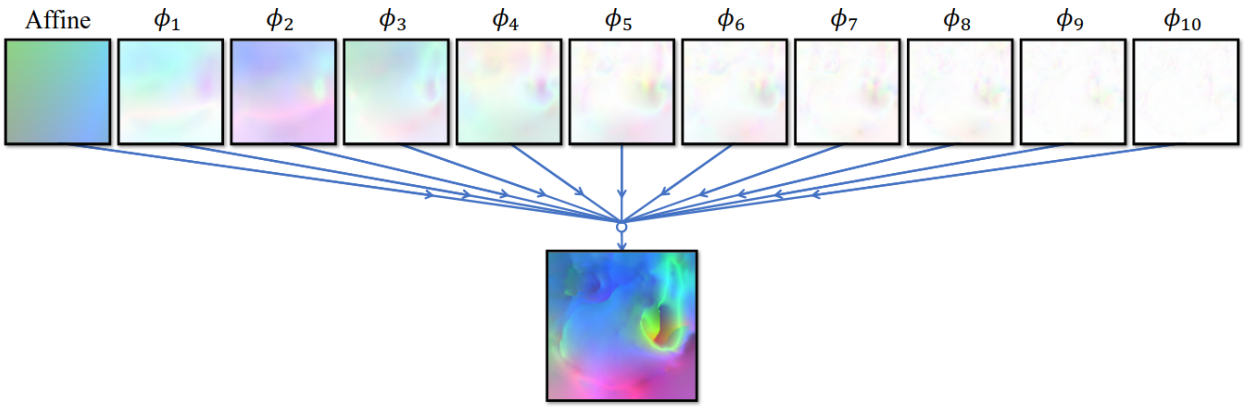

Figure 2.9: Composition of deformation fields. Flow fields are drawn by mapping the abosolute value of the three components (x, y, z) of flow displacements into color channels (R, G, B) respectively. White area indicates zero displacement. [37]

Both VTN [36] and DLIR [3] stack their networks. However, both are limited to a small number of cascades. DLIR trains each cascade one by one, i.e., after fixing the weights of previous cascades. VTN, in contrast, jointly trains the cascades measuring the similiarity between all successively warped images and the fixed image. The figure below 2.10 shows the Volume Tweening Network (VTN) [36]. Each registration subnet is responsible for determining the deformation field between the fixed image and the current moving image. The moving image is repeatedly distorted according to the deformation and fed into the next layer of cascaded subnetworks.

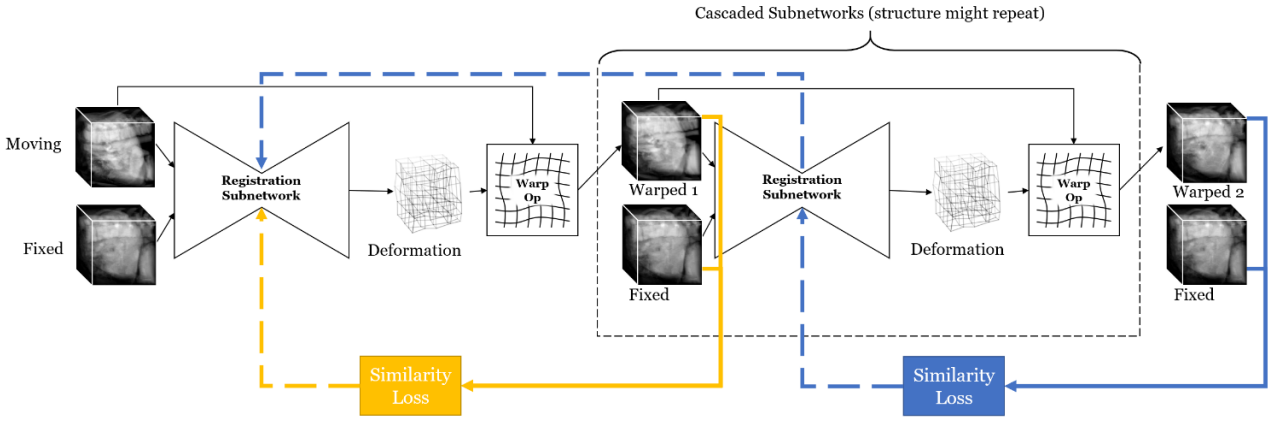

Figure 2.10: Illustration of the overall structure of the Volume Tweening Network (VTN) and the back-progression of gradients. Solid lines indicate how the loss is computed and the dashed lines indicate how gradients back-propagate [36]

Zhao et al. [37] note that such a training method does not allow progressive registration of images. They refer to these as non-cooperative cascades because they learn their own targets independently of each other's existence, making further improvement unlikely even if more cascades are performed. Therefore, Zhao et al. [37] propose the recursive cascade architecture, which allows unsupervised training of an unlimited number of cascades cascaded in a cooperative manner on any existing base network.

The architecture of the proposed recursive cascaded network is shown in the figure below 2.11. The difference between this architecture and the VTN is that the similarity is now measured only on the final warped image, so that all cascades can learn progressive alignments together.

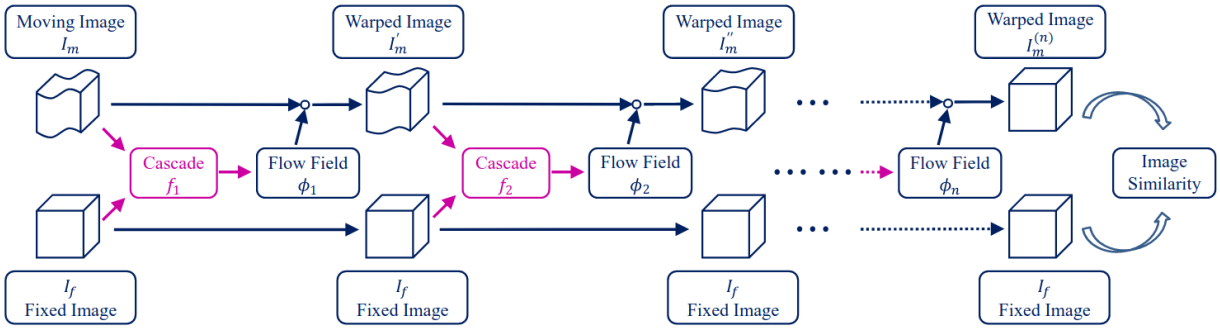

Figure 2.11: Illustration of the recursive cascade architecture. Circle denotes a composition. The unsupervised end-to-end learning is guided by the image similiarity between $I_m^{(n)}$ and I_f. [37]

The following figure 2.12 shows the performance of the recursive cascade network architectures for the SILVER, LiTS, and LSPG datasets. For testing purposes, both VoxelMorph and VTN were chosen as the base network here, and an empirical evaluation was performed. In all cases, the cascaded architecture performed better than the simple network alone.

Method	SLIVER		LiTS	LSPIG	LPBA	Time (sec)	
	Dice	Lm. Dist.	Dice	Dice	Avg. Dice	GPU	CPU
ANTs SyN [4, 5]	0.895 (0.037)	12.2 (5.7)	0.862 (0.055)	0.825 (0.063)	0.708 (0.015)	-	748
Elastix B-spline [35, 48]	0.910 (0.038)	12.6 (6.6)	0.863 (0.059)	0.825 (0.059)	0.675 (0.013)	-	115
VoxelMorph[1] [9]	0.883 (0.034)	14.0 (4.6)	0.831 (0.061)	0.715 (0.090)	0.685 (0.017)	0.20	17
VoxelMorph (reimplem.)[2]	0.913 (0.025)	13.1 (4.7)	0.870 (0.048)	0.833 (0.057)	0.688 (0.015)	0.15	14
5-cascade VoxelMorph	0.944 (0.017)	12.4 (4.9)	0.903 (0.055)	0.849 (0.062)	0.708 (0.015)	0.41	69
3×5-cascade VoxelMorph	0.950 (0.014)	11.9 (4.9)	0.905 (0.065)	0.842 (0.066)	0.715 (0.014)	1.09	201
VTN (ADDD)[3] [37]	0.942 (0.020)	12.0 (4.9)	0.897 (0.049)	0.846 (0.064)	0.701 (0.014)	0.13	26
10-cascade VTN	0.953 (0.014)	10.8 (4.9)	**0.909 (0.060)**	**0.855 (0.060)**	0.716 (0.013)	0.25	87
2×10-cascade VTN	**0.956 (0.012)**	**10.2 (4.7)**	0.908 (0.070)	0.849 (0.063)	**0.719 (0.012)**	0.42	179

Figure 2.12: Comparison among traditional methods (ANTs SyN and Elastix B-spline), baseline networks (VoxelMorph and VTN), and proposed recursive cascaded networks with and without shared-weight cascading. r x n -cascade means that every deformable cascade is repetitively applied for r times during testing, using the proposed shared-weight cascading method. [37]

After studying the literature, we decided to use VoxelMorph as the base model and incorporate concepts of VTN, DLIR, and recursive cascading networks for better performance. The cascading of the networks will be implemented in a coarse-to-fine fashion, similar to the image pyramid used in traditional image registration methods and proven for many years.

Chapter 3

Drosophila larva biology and an overview of machine learning techniques

The Drosophila larva is a widely studied model organism in the field of biology, due to its short lifespan and ease of genetic manipulation. In this chapter, we will provide an overview of the biological stages of Drosophila larva development and discuss their significance in the medical field. In addition, we will introduce the concepts of machine learning and deep learning, which are powerful techniques for analyzing and interpreting data. We will describe the key features and principles of these techniques, and how they are used in this thesis to address the research question.

3.1 Drosophila larva biology

Drosophila melanogaster, also known as the fruit fly, is a species of fly in the family Drosophilidae. Drosophila larvae, also known as maggots, are the immature stage of the fruit fly that precede the pupal stage and adult stage. They are commonly used as a model organism in genetics and developmental biology research due to their relatively simple nervous system, short generation time, and the availability of genetic tools for manipulating them.

Drosophila larvae are small, legless, and worm-like in appearance, with a length of about 5 mm. They are typically white or yellow in color and have a segmented body with three main regions: the head, thorax, and abdomen. The head of the larva contains the brain, eyes, and mouthparts, while the thorax and abdomen contain the respiratory, circulatory, and digestive systems.

During the larval stage, Drosophila undergoes significant changes in both form and function. For example, the larva grows in size, develops specialized organs and tissues, and undergoes metamorphosis to transform into the pupal stage. The larval stage is also a critical period for the development of the nervous system, with the formation and differentiation of neurons and glial cells, as well as the establishment of neural connections.

Drosophila melanogaster undergoes a complete metamorphosis during its life cycle, which consists of four stages: egg, larva, pupa, and adult.

1. **Embryo**: The embryo stage is the first stage of Drosophila development. Drosophila eggs are small and oval-shaped, with a length of about 0.5 mm. They are laid singly or in clusters on moist surfaces, such as rotting fruit or moist soil. The egg contains a single cell that divides rapidly to form a multicellular embryo.

2. **Larva**: The larva is the second stage of Drosophila development. Drosophila larvae, also known as maggots, are small, legless, and worm-like in appearance, with a length of about 5 mm. They are white or yellow in color and have a segmented body with three main regions: the head, thorax, and abdomen. The larva feeds on a variety of food sources, including rotting fruit, yeast, and other organic matter. During the larval stage, Drosophila undergoes significant changes in both form and function, including the growth of specialized organs and tissues and the development of the nervous system.

3. **Pupa**: The pupa is the third stage of Drosophila development. Drosophila pupae are typically about the same size as the larva, but are more solid and have a more defined shape. They are typically dark in color

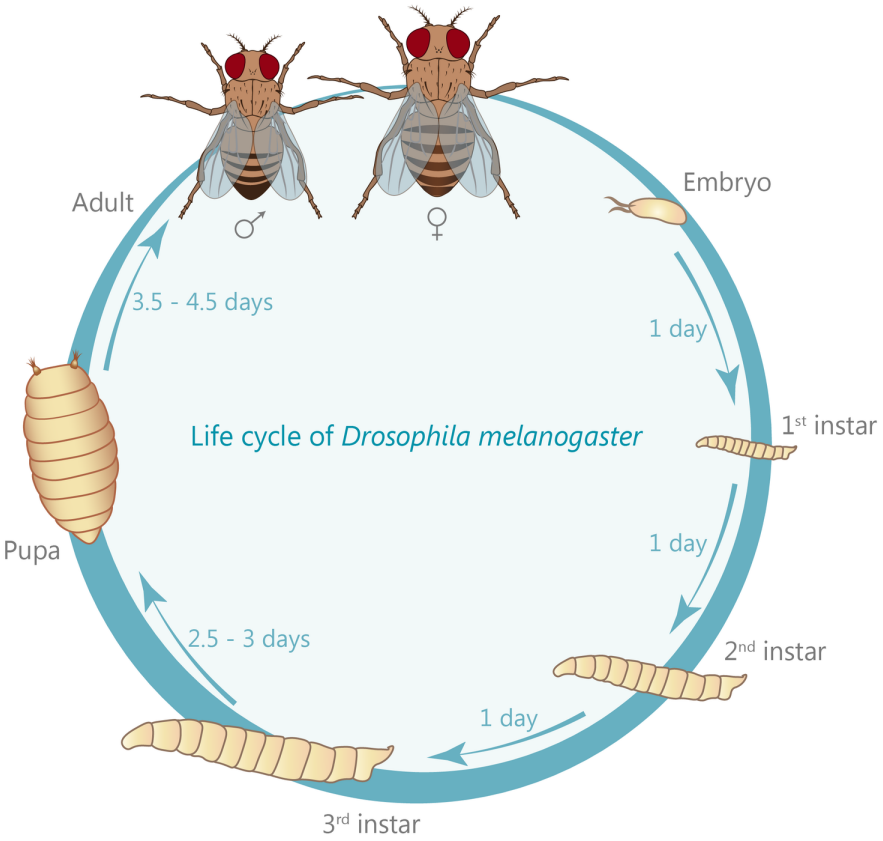

Figure 3.1: Life **cycle** of Drosophila melanogaster. [38]

and are enclosed in a hard, protective casing known as the puparium. During the pupal stage, Drosophila undergoes a process of metamorphosis, in which the larva transforms into the adult form. This process involves the reorganization and differentiation of tissues and organs, as well as the development of wings and other adult structures.

4. **Adult**: The adult is the fourth and final stage of Drosophila development. Drosophila adults are small flies with a length of about 4-7 mm. They have a distinctive appearance, with a yellow or tan thorax, a black abdomen, and red eyes. Drosophila adults are sexually mature and are capable of reproducing. They feed on a variety of food sources, including fruit, nectar, and other sweet substances.

3.2 Drosophila larva as model organism

Drosophila larva are an important model organism in biology research due to their relatively simple nervous system, short generation time, and the availability of genetic tools for manipulating them. These characteristics make Drosophila larvae a useful tool for studying development, genetics, and neurobiology.

1. Drosophila larvae has relatively simple nervous system, which consists of about 100,000 neurons, compared to the approximately 100 billion neurons in the human brain. This simplicity makes it easier to study the development and function of the nervous system, as well as the underlying genetic and molecular mechanisms that regulate these processes.

2. Drosophila larvae also have a short generation time, with a lifespan of about two weeks from egg to adult. This allows researchers to study development and evolution in a relatively short time frame, making Drosophila an important tool for studying the genetic basis of development and evolution.

3. In addition, Drosophila larvae are amenable to genetic manipulation, allowing researchers to study the function of specific genes and pathways in development and behavior. For example, researchers can use techniques such as knockdown, overexpression, and gene editing to study the role of specific genes in the development and function of the nervous system.

These characteristics make Drosophila larvae a valuable tool for studying development, genetics, and neurobiology, and for identifying potential therapeutic targets for treating human diseases. While the brain of Drosophila larvae is certainly different from the human brain in terms of size, complexity, and overall organization, it shares many fundamental features and molecular pathways with the human brain.

- Like the human brain, the Drosophila larval brain contains various types of neurons and glial cells that perform different functions, and it is organized into distinct regions that are involved in specific behaviors and functions.
- Furthermore, the development and function of the Drosophila larval brain are regulated by many of the same genes and molecular pathways that regulate the development and function of the human brain. For example, many of the signaling pathways and transcription factors that regulate the development and function of the human brain are also present in the Drosophila larval brain.

Overall, studying the brain of Drosophila larvae can provide valuable insights into the fundamental processes that underlie the development and function of the nervous system, and may help to identify potential therapeutic targets for treating neurological disorders in humans.

3.3 Machine Learning and Deep Learning Background

This section discusses the concept of machine learning, as well as its subfield, deep learning, and the techniques and tools used to train models in this field. Deep learning is a type of artificial intelligence and machine learning that allows a computer to learn from experiences and perform well on new tasks, just like a human brain. This approach can be used to discover patterns and correlations through observations without human intervention and has been applied in various fields, including computer vision, speech recognition, natural language processing, and medical image processing. The following subsections provide more information on key deep learning techniques that are important to understand in the context of this thesis.

3.3.1 Machine Learning and types

Machine learning is a branch of AI that deals with improving the computer algorithms from experience and data without being explicitly programmed [39]. There are four types of machine learning - supervised learning, unsupervised learning,semi-supervised learning, and reinforcement learning. In supervised learning, the algorithm or the model is trained using labeled data. During the training procedure, the model learns the parameters and uses these parameters to predict the outputs for unseen data. Supervised learning is of two types - regression and classification. Regression deals with continuous outputs whereas classification deals with discrete outputs. Supervised learning can be used in many applications like spam detection, object recognition, bioinformatics, etc. Though it has the above advantages, it may not generalize well to new data (overfitting), and pre-processing is a big challenge [40]. In unsupervised learning, the model is trained on unlabeled data to identify the hidden patterns and structures within the data. Unsupervised learning is also referred to as clustering. Some popular unsupervised learning algorithms are autoencoders, principal component analysis, etc. It is widely used for dimensionality reduction and exploratory analysis. The problems with unsupervised learning are that the results have high uncertainty and it is challenging to verify the outputs [41]. Semi-supervised learning is a combination of both supervised and unsupervised learning paradigms. The model is trained initially with the labeled data in a supervised fashion. Then unlabeled data is given to the model to predict the outputs.

Later the predicted outputs are considered as the true outputs for the unlabeled data. Then the model has trained again with the combined data along with the labels. This method is helpful when we have limited labeled data. They are widely used in image and document classification tasks [42]. Reinforcement learning works based on rewards and penalties. The goal of the model is to maximize the total reward. The model will not be given any labels, it has to figure out using trial and error approaches to come up with a solution to the problem. It gets feedback from the environment. Q-learning and Markov decision process are some of the most widely used reinforcement algorithms [43]. Even though machine learning algorithms have a lot of applications, they fail to perform well in image processing, natural language processing (NLP), etc. This is where deep learning is so powerful.

3.3.2 Deep Learning

Deep learning is a subset of machine learning that is inspired by the structure and function of the brain, specifically the neural networks that make up the brain. Deep learning algorithms are designed to learn and

extract features from data automatically, rather than being explicitly programmed with a set of rules or features to look for.

Deep learning algorithms are implemented using artificial neural networks, which are composed of multiple layers of interconnected artificial neurons, or "perceptrons." These neural networks are trained using large amounts of labeled data and a variant of the backpropagation algorithm, which adjusts the weights of the connections between the neurons based on the error between the predicted output and the actual output.

Deep learning algorithms are capable of learning complex, non-linear relationships in data and have been shown to achieve state-of-the-art results in a wide variety of tasks, including image and speech recognition, natural language processing, and predictive modeling. They have been applied to a wide range of applications, including self-driving cars, medical diagnosis, and language translation.

One of the key benefits of deep learning is that it can learn to extract meaningful features from raw data, without the need for manual feature engineering. This can save a lot of time and effort in the preprocessing stage of a machine learning project, and can often lead to better performance on the task at hand. However, deep learning algorithms can be computationally expensive to train and require a large amount of labeled data to be effective.

3.3.3 Neural Networks

Neural networks or artificial neural networks (ANN) are the core of deep learning algorithms, whose name and design are inspired by how neurons function in the human brain. It can learn and model non-linear and complex relationships between input and output. It also has the ability to generalize, which implies it can infer associations from unobserved data after training.

Neurons or perceptrons [44] are the basic blocks of a neural network. A perceptron takes a set of weighted inputs, exerts an activation function, and returns an output value. 3.2 illustrates the model of a perceptron. It takes any number of inputs (xi), which are weighted with corresponding weights (wi) and then summed up along with a bias (b). Then the sum is subjected to an activation function (σ) to produce the final output as shown in Equation 3.1.

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3.1}$$

In this equation 3.1., y is the output of the perceptron, x_1, x_2, ..., x_n are the input features, w_1, w_2, ..., w_n are the weights, and b is the bias term. The sigmoid function σ(x) is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

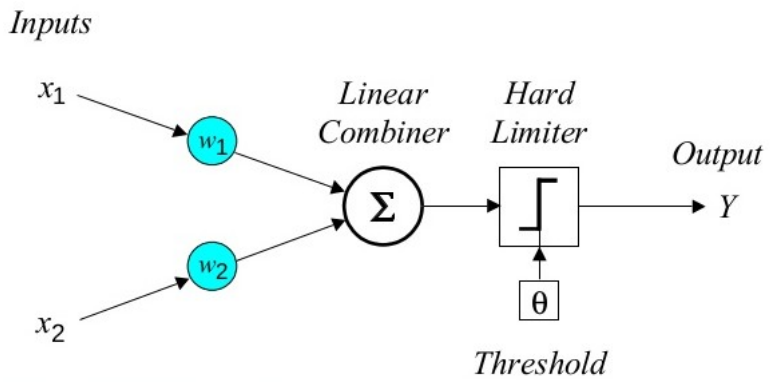

Figure 3.2: Illustration of model with a single perceptron. [45]

Several perceptrons can be combined to solve more complex problems. Such a combination of neurons, which are organized in layers called multi-layer perceptrons (MLPs) as shown in 3.3. A multi-layered perceptron (MLP) is a type of artificial neural network that is composed of multiple layers of artificial neurons, or "perceptrons." MLPs are used for supervised learning tasks, such as classification and regression.

$$
\begin{aligned}
a_1 &= x \\
a_2 &= f_2(w_{1,2}a_1 + b_2) \\
a_3 &= f_3(w_{2,3}a_2 + b_3) \\
&\vdots \\
y &= f_L(w_{L-1,L}a_{L-1} + b_L)
\end{aligned}
$$

In this equation, x is the input to the neural network, a_1, a_2, ..., a_L are the activations at each layer, w_l,l+1 are the weights between layer l and layer l+1, b_l is the bias at layer l, and f_l is the activation function at layer l. The output of the neural network is y.

The basic structure of an MLP consists of an input layer, one or more hidden layers, and an output layer like in figure 3.3. The input layer receives the input data and passes it through to the hidden layers, which process the data and pass it along to the output layer. The output layer produces the final output of the MLP, which can be a classification or a prediction based on the input data.

MLPs are trained using a variant of the backpropagation algorithm, which adjusts the weights of the connections between the neurons based on the error between the predicted output and the actual output. This process is repeated until the error is minimized and the MLP is able to accurately predict the output for a given input.

MLPs have been widely used in many applications, including image recognition, natural language processing, and predictive modeling. However, they can be computationally expensive to train and are not well-suited to tasks that require processing large amounts of sequential data, such as language translation or speech recognition.

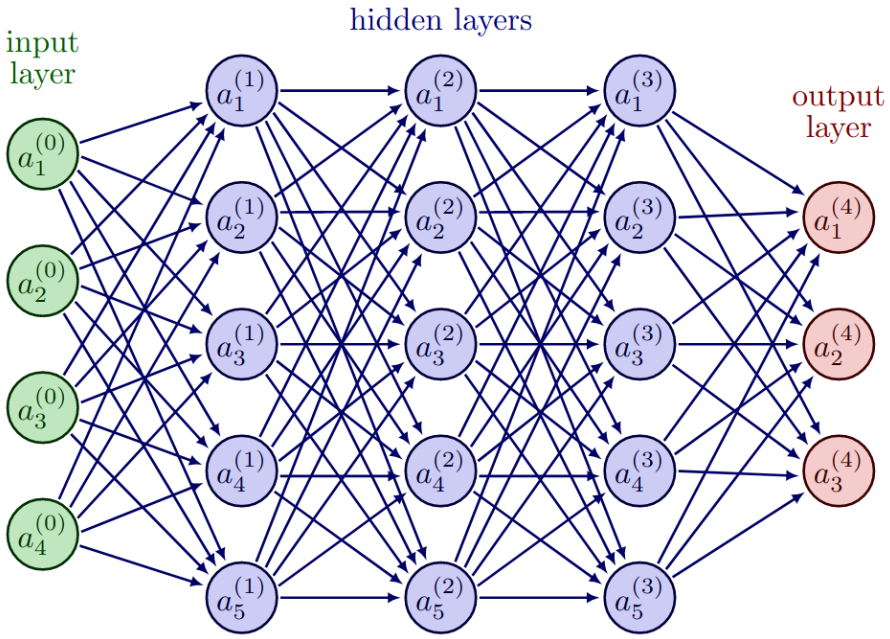

Figure 3.3: Illustration of model with a multi-layer perceptron. [46]

3.3.4 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are a type of artificial neural network specifically designed for image recognition and processing. They are inspired by the structure of the visual cortex in animals, which is arranged in such a way as to be sensitive to specific patterns or features in visual stimuli.

CNNs are composed of multiple layers of artificial neurons, or "perceptrons," arranged in a three-dimensional structure, with the input layer at the bottom, followed by one or more hidden layers, and an output layer at the top. The hidden layers are made up of multiple "convolutional" layers, which apply a set of learnable filters to the input data and produce a set of feature maps. These feature maps are then processed by one or more "pooling" layers, which downsample the data and reduce the spatial resolution of the feature maps. Finally, the output of the pooling layers can be passed through a fully connected layer, which produces the final output of the CNN.

CNNs are trained using a variant of the backpropagation algorithm, in which the weights of the filters and connections between the neurons are adjusted based on the error between the predicted output and the actual output. This process is repeated until the error is minimized and the CNN is able to accurately classify or predict the output for a given input.

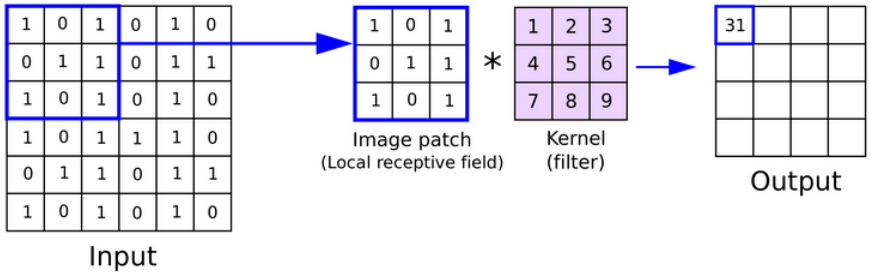

Figure 3.4: Illustration of working of Convolutional Neural Networks (CNNs). [47]

CNNs have achieved state-of-the-art results in many image recognition tasks and are widely used in applications such as object detection, image classification, and facial recognition.

3.3.5 Elements of Neural Network

Activation function

Activation functions are used to adjust the output of each layer in a neural network. There are two types: linear and non-linear. Linear activation functions are not commonly used. Instead, non-linear activation functions are used to introduce non-linearities to the network, allowing it to learn non-linear relationships between inputs and outputs. Some common non-linear activation functions include sigmoid, tanh, ReLU, and leaky ReLU. The graphs of these activation functions are shown in Figure 3.5 [48, 49].

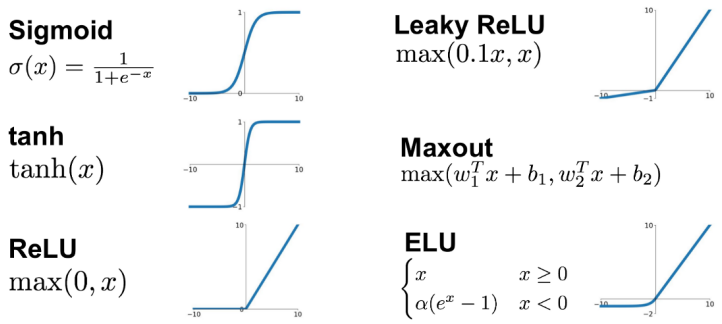

Figure 3.5: Activation functions

Loss function

A Loss function is used to check how our model compares to an ideal model. There are many types of loss functions, one of which can be selected based on the problem being solved. The most common ones for regression are mean absolute error (MAE) also referred to as L1 loss, mean squared error (MSE) also referred to as L2 loss. The most common ones for classification problems are binary cross-entropy (BCE) loss and hinge loss [50]. And others include normalized cross correlation and mutual information loss to measure the similarity between the two images. A typical loss function plotted against the number of epochs is given in the Figure 3.6 [51].

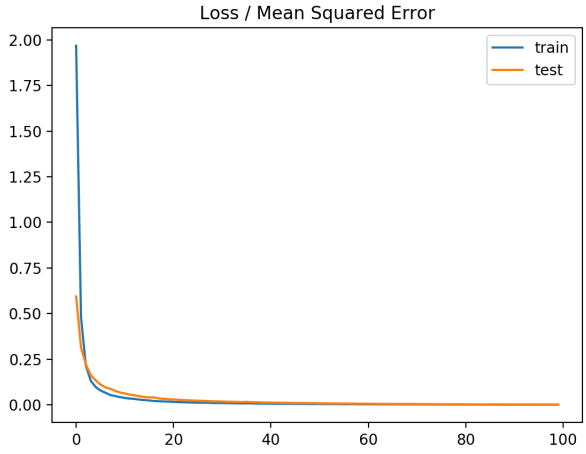

Figure 3.6: Loss function

Backpropagation

Backpropagation is an algorithm used to train artificial neural networks. It involves using gradient descent to propagate the error computed by the loss function backwards through the network. The gradients of the error with respect to the model's parameters are calculated using the chain rule of calculus, and these gradients are used to update the parameters of the model.

Learning rate

The learning rate is a hyperparameter that determines the size of the updates made to the model's weights during training. It is important to choose an optimal learning rate, as a value that is too small can cause the training process to be slow and potentially get stuck, while a value that is too large can cause the training process to be unstable and result in sub-optimal learned parameters. The learning rate can also be scheduled to change during the training process. [52].

Optimizer

Optimizers are important in the training process of a model, as they use the gradients computed through backpropagation to update the model's parameters and minimize the loss. Some common optimizers include Adam and stochastic gradient descent. Research has shown that the Adam algorithm tends to perform well in most cases [53].

3.3.6 Encoder-Decoder Architecture

The encoder-decoder architecture is a type of neural network architecture commonly used in natural language processing and machine translation tasks. It consists of two main components: an encoder and a decoder.

1. The encoder takes in a sequence of input data, such as a sentence in a source language, and converts it into a fixed-length internal representation, called a "latent representation" or "latent vector." This latent vector captures the meaning of the input sequence and is typically much smaller in size than the input sequence itself.

2. The decoder takes the latent vector as input and generates an output sequence, such as a translation of the input sentence into a target language. The decoder is typically trained to generate the output sequence by predicting the next word in the sequence based on the previous words and the latent vector.

The encoder-decoder architecture has been widely used in natural language processing tasks, such as machine translation, language modeling, and text summarization. It has also been applied to other domains, such as image generation and sequence-to-sequence modeling in time series data.

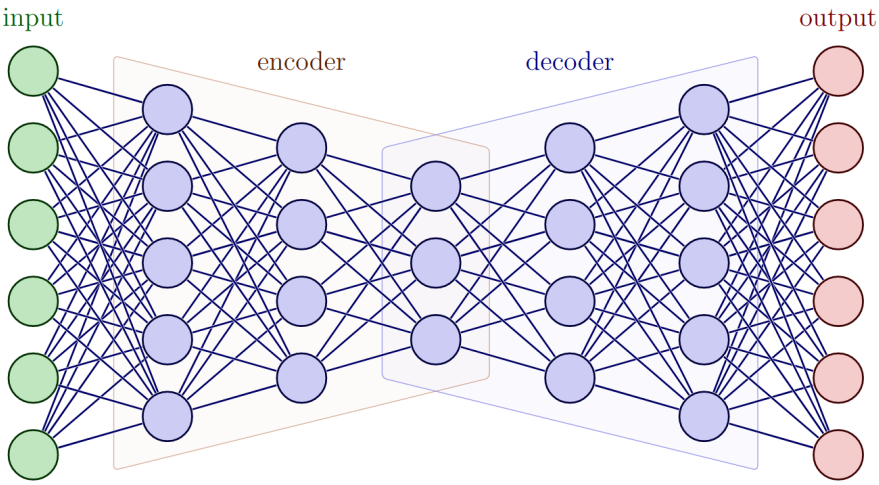

Figure 3.7: Illustration of encoder decoder architecture [46].

In image processing, the encoder-decoder architecture can be used to perform tasks such as image compression and image reconstruction. In image compression, the encoder takes in an input image and compresses it into a smaller representation, called a "latent representation" or "latent vector" that captures the important features of the image. This latent vector captures the important features of the input image and is typically much smaller in size than the input image itself. The decoder takes the latent vector as input and reconstructs an output image that is as close as possible to the original input image.

In image reconstruction tasks, the encoder-decoder architecture can be used to restore images that have been damaged or degraded in some way, such as by noise, blur, or missing pixels. It can also be used to perform tasks such as super-resolution, in which a low-resolution image is upscaled to a higher resolution.

3.3.7 U-Net

U-Net is a convolutional neural network (CNN) architecture designed for image segmentation tasks. It was developed by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in 2015 [54] and has since become a widely used model for image segmentation in medical imaging and other domains. U-Net is typically trained using supervised learning, with the goal of minimizing the error between the predicted output segmentation map and the ground truth segmentation map. It has been widely used in medical imaging tasks, such as tumor segmentation and organ segmentation, and has also been applied to other domains, such as satellite image analysis and surface defect detection.

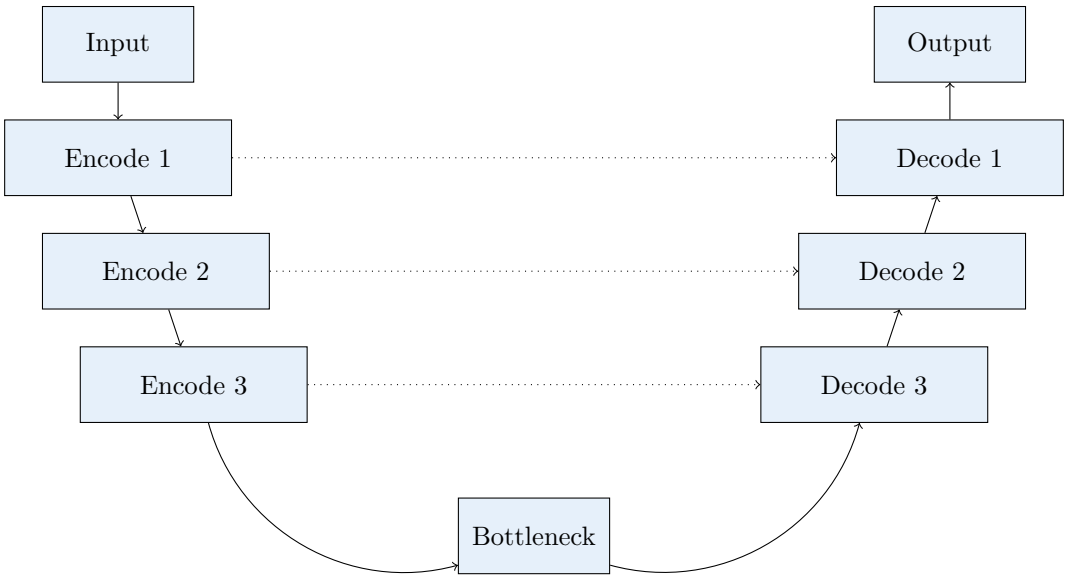

Figure 3.8: A graphical representation of the U-Net model.

Figure 3.8 shows the structure of the U-Net model. The model has two main parts: an encoder that reduces the spatial resolution of the input image and increases the number of feature maps, and a decoder that does the opposite. The encoder uses convolutional layers with a stride of 2 to downsample the input image, and the decoder uses transposed convolutional layers to upsample the feature maps. The bottleneck layer connects the encoder and decoder and acts as a bridge between them. It allows the decoder to use the high-level features extracted by the encoder to generate a detailed segmentation map. The model also has skip connections, shown as dotted lines in the figure, that allow the decoder to access the features extracted by the encoder at each level of the network. This helps the model to use both high-level and low-level features and preserves spatial information, leading to more accurate segmentation results.

The input to the model is an image and the output is a segmentation mask, with each pixel in the mask corresponding to a particular class in the image. The U-Net model is commonly used in medical imaging, where it can be trained to segment organs or other structures in images.

Chapter 4

Methods and Implementations

In Chapter 2, we reviewed various traditional and deep learning approaches for medical image registration. In this chapter, we will delve into the specific methods we selected for our study and explain why they were chosen and how they are effective in addressing the challenges of medical image registration.

In the following three sections, we will address the specifics of the Voxelmorph architecture [2] that served as the foundation for our study, explore the concept of landmark points and their importance in the field of image registration, and provide an overview of the dataset we used and the preprocessing steps we applied to it in order to prepare it for analysis.

4.1 The Voxelmorph Architecture: An Overview

In the paper, "VoxelMorph: a learning framework for deformable medical image registration," [2] the authors introduce a deep learning architecture called Voxelmorph for medical image registration. The architecture of Voxelmorph is designed specifically for medical image registration, which is the process of aligning two or more medical images of the same subject, taken at different times or using different modalities, in order to compare and analyze them.

The Voxelmorph architecture consists of two main components: an encoder-decoder CNN architecture (U-Net) [54] and a spatial transformer network (STN) [32]. The U-Net is responsible for learning the spatial transformations needed to align the images, while the spatial transformer network is responsible for applying those transformations to the images.

In the Voxelmorph architecture, the U-Net is modified to predict a dense displacement field, which describes how the pixels in the target image should be displaced in order to align with the source image. The displacement field is then passed through spatial transformer network (STN), which uses it to warp the moving image and bring it into alignment with the fixed image. The U-Net architecture is particularly well-suited for this task because it is able to capture both high-level semantic features and fine-grained spatial details in the input images.

The basic structure of Voxelmorph consists of an encoder-decoder architecture (U-Net) with skip connections between the encoder and decoder layers as illustrated in Subsection 3.3.7. The encoder takes in a pair of input images and extracts features from them, while the decoder generates a dense displacement field that maps pixels from one image to the other. The skip connections allow the decoder to access the features extracted by the encoder at each level of the network, which helps to preserve spatial information and improve the accuracy of the displacement field.

The figure 4.1 is an illustration of the Voxelmorph model. The model consists of two main components: a U-Net model and an STN model. The U-Net model takes as input the fixed image and the moving image, and produces a deformation field as output. The STN model, or spatial transformer network model, takes as input the deformation field and the moving image, and produces a moved image as output. The STN model is responsible for applying the deformation field to the moving image, effectively aligning it with the fixed image. The moved image is then compared to the fixed image using a similarity loss function, which is represented by the block labeled "Similarity Loss" in the illustration. The dotted arrows in the illustration indicate that the similarity loss is calculated between the fixed image and the moved image. The model is trained to minimize the similarity loss, which helps to align the moving image with the fixed image.

Figure 4.2 depicts a model that includes an auxiliary information block in addition to the fixed and moving images as input to the U-Net model. This auxiliary information can improve the accuracy of the deformation field produced by the U-Net model. Researchers have found that using segmentation masks as auxiliary input

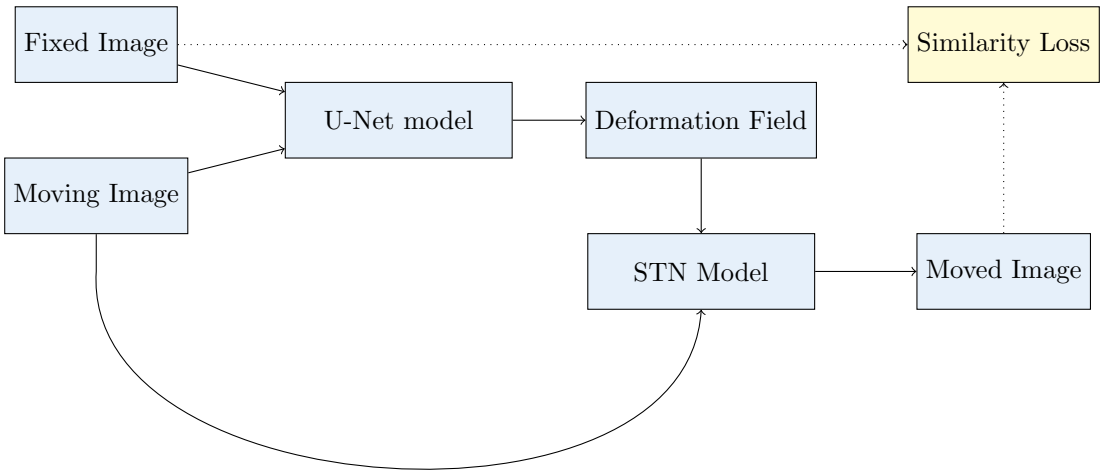

Figure 4.1: A graphical representation of the Voxelmorph model where the U-Net model takes as input the fixed image and the moving image, and produces a deformation field as output.

can enhance the model's performance on specific tasks. For example, segmentation masks can help the model focus on specific areas of the image, such as the brain or heart, and improve the accuracy of the displacement field in those areas.

The Voxelmorph architecture is designed to be fast and accurate, and it has been widely used in the field of medical image analysis since its introduction. Considering its impressive performance and widespread adoption in the field of medical image analysis, we determined that the Voxelmorph architecture was the best choice for our research and therefore used it as the base architecture.

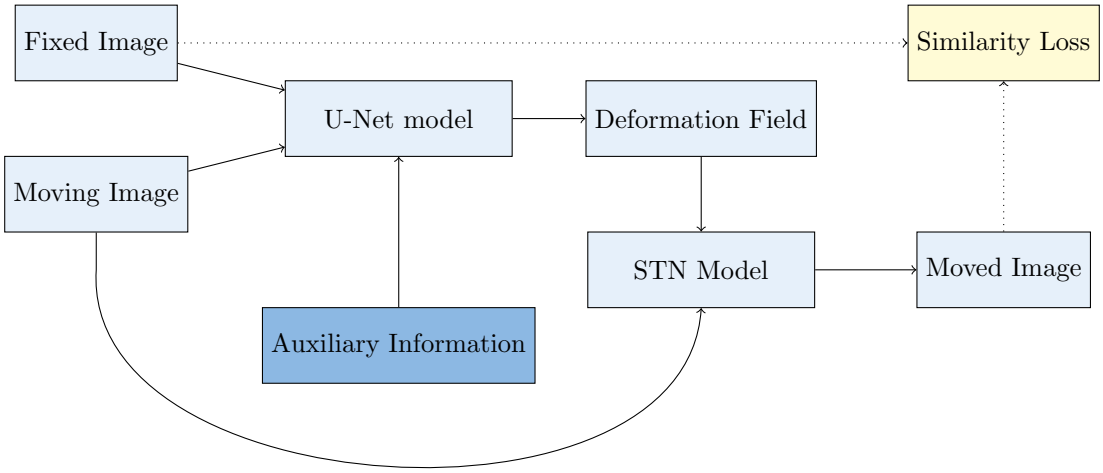

Figure 4.2: A graphical representation of the Voxelmorph model with auxiliary information where the auxiliary information in addition to the fixed and moving images are provided as input to the U-Net model.

4.2 Landmark Points: A Key Element in Image Registration

Landmark points in image registration, also known as fiducial points, are points of reference in an image that are used to align or register two or more images. These points are often selected based on their distinct and easily identifiable features, and are used to establish a common coordinate system for the images being registered.

The term "gold standard" refers to the fact that these points are considered to be the most accurate and reliable points of reference for aligning images. This term is used metaphorically, as gold is often considered a symbol of excellence and high quality. In the context of image registration, the use of landmark points as the

gold standard for alignment helps to ensure the accuracy and reliability of the registration process.

In a research paper on *larvalign* [1], biologists have identified 30 potential landmark locations in brain scans of drosophila larva. These landmarks are also used in this thesis.

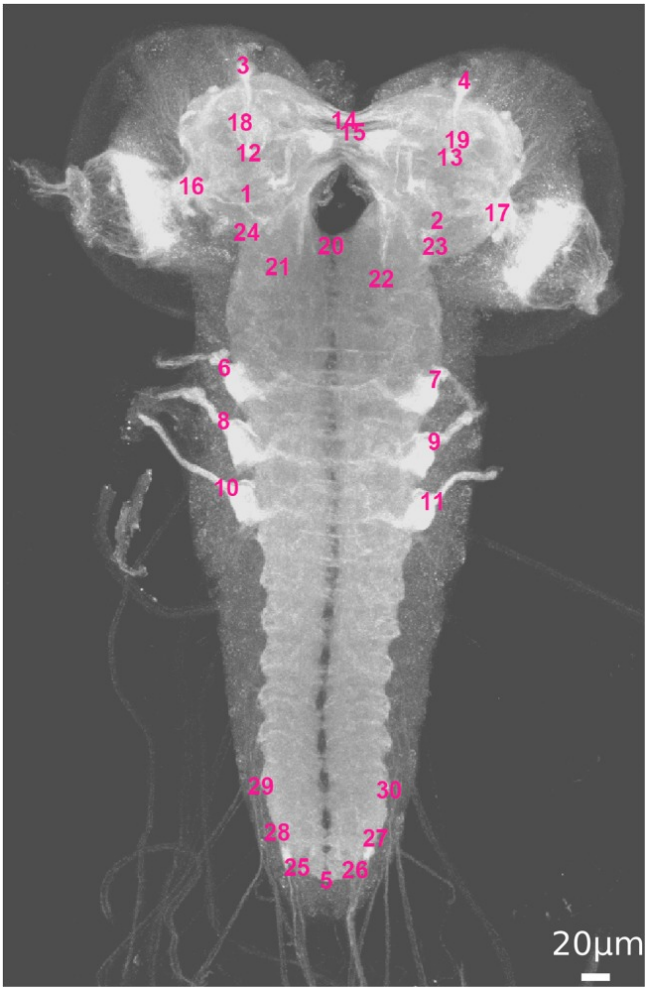

Figure 4.3: Maximum intensity projection (MIP) of an example scan with annotated landmarks [1].

To annotate or label these landmark points in the brain scan images, we can use ImageJ software. ImageJ is an open-source image processing software that offers "Point Tool" to label the landmark points. The "Point Tool" allows you to place a single point on an image by clicking on the desired location.

To use the "Point Tool" in ImageJ Fiji for landmark annotation, follow these steps:

1. Open the image you want to annotate in ImageJ Fiji.
2. Click on the "Point Tool" icon in the toolbar at the top of the window. This will activate the tool and allow you to place points on the image.
3. Click on the location in the image where you want to place a point. A small crosshair will appear at the location you clicked.
4. Use the "Set/Add Point" button to label the point with a name or identifier. You can enter the name in the text field that appears and then click "OK" to set the name for the point.
5. Repeat steps 3 and 4 for each additional point you want to place on the image.
6. When you are finished annotating the points, you can use the "Point Tool" icon again to deactivate the tool and return to the default cursor.

When you use the "Point Tool" to annotate an image in ImageJ, the result of the annotation will be saved in the same directory as the image as an XML file with a ".points" extension. This file will contain the coordinates of the points that you added to the image, as well as any other information about the points (such as labels and attributes).

4.3 Data Preparation: Selecting and Preprocessing Datasets

This section covers the selection and preparation of datasets for use with the Voxelmorph model. It will cover how to categorize the datasets for training and testing, as well as the necessary preprocessing steps to ensure that the data is properly formatted for the model.

4.3.1 Dataset

For this research, we have obtained three datasets from different sources: the `Leipzig dataset` from the University of Leipzig in Germany, the `Janelia dataset` from the Janelia Research Campus in Virginia, and the `Larvalign dataset` from the *Larvalign* research paper [1]. These datasets will be referred to by these names throughout the study.

As shown in Table 4.1, the datasets for this study were divided into three categories by the biologists: good quality, medium quality, and random quality. This separation was necessary to ensure that the training data was balanced and to allow us to evaluate the model's performance on different quality levels of the images.

	Quality	Number of Scans	Original Resolution	Scaled Resolution
`Leipzig dataset`		100	980x1440x81	256x512x64
		052	512x512x104	
		200	592x800x102	
`Janelia dataset`	Medium	200	977x1428x76	
	Good	200	981x1428x76	
	Random	200	973x1434x79	
`Larvalign dataset (Test Data)`	Medium	021	973x1434x79	
	Good	020	981x1430x79	
	Random	025	977x1432x77	

Table 4.1: Dataset distribution.

One issue that was encountered during the data preparation process was that the images were of different sizes and resolutions. To input the images into the network, they had to be resized to a consistent size. However, the Voxelmorph model was computationally intensive and required a significant amount of memory due to the various levels of encoding and the need to retain encoded feature maps for later use in the decoding stage through skip connections. Experiments showed that using the model with images that were `256x512x64` in size required at least a `16GB GPU`. If the image volumes were larger than this resolution, a GPU with a memory capacity proportional to the expansion of the input images was required. Empirically, it was found that the GPU memory requirement was directly proportional to the increase in image volume by more than 1. As a result, it was decided to use scaled-down resolution images of `256x512x64` for all experiments. Unless stated otherwise, all experiments from that point forward were based on images with this resolution. The scaling down operation was performed by resampling the image using the bicubic interpolation algorithm present in the ImageJ Fiji tool.

4.3.2 Affine registration

According to the Voxelmorph research paper [2], it was recommended to perform affine alignment on the input images before using them with the Voxelmorph model for deformable registration. This preprocessing step could improve the performance of the model compared to using the images without affine alignment.

Similar to Voxelmorph, the *larvalign* software [1] also performed affine alignment and then applied non-rigid registration to the affine-aligned image. Therefore, to compare the efficiency of non-rigid registration using *larvalign* versus a deep learning method, it was decided to use *larvalign* to perform affine registration on the input images so that both approaches would be working with the same pre-processed data. This allowed for a fair comparison of the two approaches.

Larvalign Software

Larvalign is based on the Elastix Registration Toolbox, which is a widely used, open-source medical image registration toolbox that allows for both rigid and non-rigid registration of medical images such as CT, MRI, and PET scans.

Elastix uses optimization algorithms to minimize a cost function that measures the difference between the images and finds the transformation that best aligns them. It offers a range of customization options, including the choice of optimization algorithm, cost function, and regularization terms, as well as pre-processing and post-processing tools to improve the accuracy and efficiency of the registration.

The figure below illustrates an example of an image before and after affine alignment.

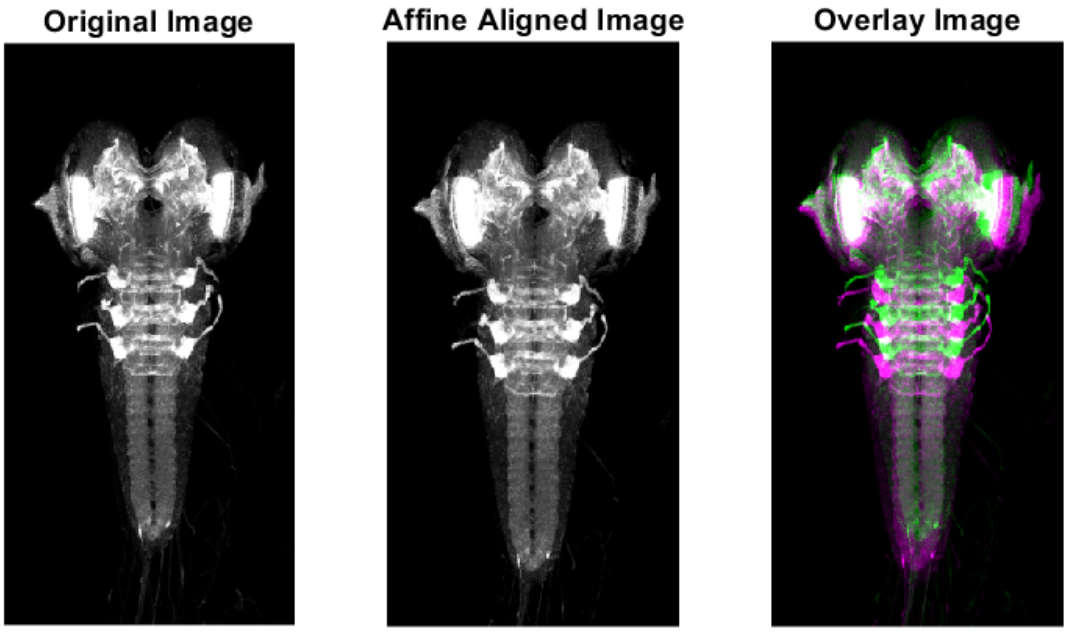

Figure 4.4: Affine registration example against the *larvalign* template [1].

Affine alignment is a method for aligning two images by applying an affine transformation to one of the images. An affine transformation can include rotation, scaling, translation, and shearing, and it can be used to match corresponding points in the two images. It is possible that the scale of the moving image could be changed during affine registration to match the scale of the fixed image so that moving image is aligned as much as possible with the fixed image. In the Figure 4.4, the affine registration is performed against the *larvalign* template (shown in Figure 1.1), and the resulting alignment is shown before and after registration in the left and center images, respectively. The right image shows an overlay of the two images, before and after registration.

4.3.3 Background Noise Removal and Normalizing

Background Noise Removal

To prepare the image for further processing, the noise in the background was removed by clipping the intensity values of the pixels between the mean intensity of the image and the maximum intensity (`255 for an 8-bit image`). Any values that are below the mean intensity are set to zero, while values above the mean intensity are retained. This helped to clean up the image and reduce the impact of noise on subsequent steps.

The below pseudo code defines a function called `cleanup_background` that takes an image as input and returns a modified version of the image with the background pixels set to zero.

```
1
2  def cleanup_background(image):
3    # Find the mean intensity of the image
```

```
    mean_intensity = find_mean_intensity(image)

    # Set the low and high values for clipping
    low = mean_intensity
    high = 255

    # Clip the pixel intensity values between the low and high values
    image = clip_intensity_values(image, low, high)

    # Set the low clipped values to zero
    image = set_low_values_to_zero(image, low)

    return image


```

The function shown in the Listing 4.3.3 performs the following steps:

1. It calls a function called `find_mean_intensity` to calculate the mean intensity of the image.
2. It sets the low and high values for clipping to the mean intensity and the maximum intensity (255), respectively.
3. It calls a function called `clip_intensity_values` to clip the pixel intensity values between the low and high values. Any values below the low value are set to zero, while values above the low value are kept the same.
4. It calls a function called `set_low_values_to_zero` to set the low clipped values to zero.

Normalizing the inputs

Normalizing the pixel values in the input images for use in a neural network could prevent issues such as vanishing or exploding gradients, which could further hinder the network's ability to learn effectively. One way to normalize the values was to scale them to be between 0 and 1, which could be done by dividing the pixel values by 255 (the maximum value for an 8-bit image). This approach was suggested by the authors of the Voxelmorph paper, as it was compatible with the Leaky ReLU activation function used in the network, and this was what was followed in this work. Other normalization techniques, such as subtracting the mean and dividing by the standard deviation of the pixel values, could also be used, but they may not have been as well-suited for use with the Leaky ReLU function because such normalization could result in negative values.

```
    def normalize(pixels):
      normalized_pixels = []
      for pixel in pixels:
        normalized_pixel = pixel / 255
        normalized_pixels.append(normalized_pixel)
      return normalized_pixels

    # Example usage
    pixels = [100, 150, 200, 50]
    normalized_pixels = normalize(pixels)
    print(normalized_pixels)  # Output: [0.39215686274509803, 0.5882352941176471,
    0.7843137254901961, 0.19607843137254902]

```

The code for normalization is shown in Listing 4.3.3. This code defines a function called "normalize" that takes a list of pixel values as input and returns a new list of normalized values. The pixel values are normalized by dividing each value by 255, the maximum value for a pixel in an 8-bit image.

4.4 Methods

In the following sections, we will describe the different methods that we used to evaluate the performance of deep learning neural networks, presented in order of increasing performance. All of the methods were implemented using the TensorFlow framework in Python. For each method, we will explain why we chose it and how it outperformed the previous method. In total, we will present four methods.

- **Method1**: Evaluating the Raw Voxelmorph Architecture.
- **Method2**: Enhancing Method 1 with Novel Landmark Information.

- **Method3**: Pyramid Gaussian Filtering and Cascaded Training.
- **Method4**: Enhancing Method 3 with Novel Landmark Information.

4.4.1 Method1: Evaluating the Raw Voxelmorph Architecture

As described in the Section 4.1, the architecture of the VoxelMorph network consists of a series of encoder-decoder blocks, which are inspired by the U-Net architecture.The encoder blocks down-sample the input images by applying convolutions and max pooling, while the decoder blocks up-sample the feature maps using transposed convolutions. The network also includes skip connections between the encoder and decoder blocks, which helps to preserve spatial resolution and fine details in the output image.

Unlike the vanilla voxelmorph model, this method employed a U-Net architecture with 6 encoding layers and 6 corresponding decoding layers, connected by skip connections. The number of channels in the encoding layers was [16, 32, 32, 32, 32, 32], and the number of channels in the decoding layers was [32, 32, 32, 32, 32, 32]. In addition, there were 3 additional convolutional layers with [32, 16, 16] channels that were used to extract additional features at the end stages. Between each pair of consecutive encoding layers, the size of the feature maps was reduced by a factor of 2 using max pooling. Similarly, between each pair of consecutive decoding layers, the size of the feature maps was increased by a factor of 2 using upsampling. Leaky ReLu activation functions were used to introduce non-linearity between each layer.

The U-Net produced a 3-channel feature map representing the 3D deformation field between the fixed and moving images. This deformation field, interpreted as displacement vectors, could be used to register the moving image to the fixed image using a spatial transformer network. This network applied the necessary transformations to the moving image based on the input deformation field, aligning it with the fixed image. After the moving image was registered to the fixed image using the U-Net and spatial transformer network, the resulting "warped" image was compared to the fixed image using a similarity score Normalized Cross Correlation. The network was trained for 100 epochs until the similarity score reached a maximum or saturated at a certain level, indicating satisfactory alignment of the moving and fixed images.

The NCC was the metric that measured the similarity between two images by comparing the intensity values of corresponding pixels. It was normalized to have a specific range, independent of the images' overall intensity values. A higher NCC score meant higher similarity between the images, while a lower score meant lower similarity.

$$L = -\frac{1}{N}\sum_{i=1}^{N} \mathrm{NCC}(Fixed, Moved_i) \tag{4.1}$$

The loss function for the registration process is shown in Equation 4.1.

In this Equation 4.1, N is the number of moving images, and $NCC(Fixed, Moved_i)$ is the normalized cross correlation between $Fixed$ and $Moved_i$. The loss function is the negative average of the normalized cross correlation between the fixed image and each of the moving images.

4.4.2 Results

The model with the above mentioned configuration was trained with the `Janelia dataset` and tested on the `Larvalign dataset`. It was found that this model performed poorly on images that contained large deformations. However, the model performed well on images that contained smaller deformations. This suggested that the voxelmorph model may have been less effective at registering images with large deformations, but effective for registering images with smaller deformations.

The moving image in Figure 4.6 has large deformation at the lower tip of the Ventral Nerve Cord (VNC), and the registration process is not successful as evident (magenta region at the lower tip of VNC) in the corresponding overlay image. In contrast, the Voxelmorph method produces good results in the Figure 4.7. The good results produced by the Voxelmorph method in this case can be attributed to the absence of significant deformation in the moving image and the model's ability to handle smaller deformations effectively.

4.4.3 Motivation for Method2

The model clearly struggled with handling large deformations, specifically in the lower ventricular nerve chord region of the brain. In order to improve the model's performance in these cases, we provided it with additional, manually annotated information for both the moving and fixed images. This helped guide the model towards

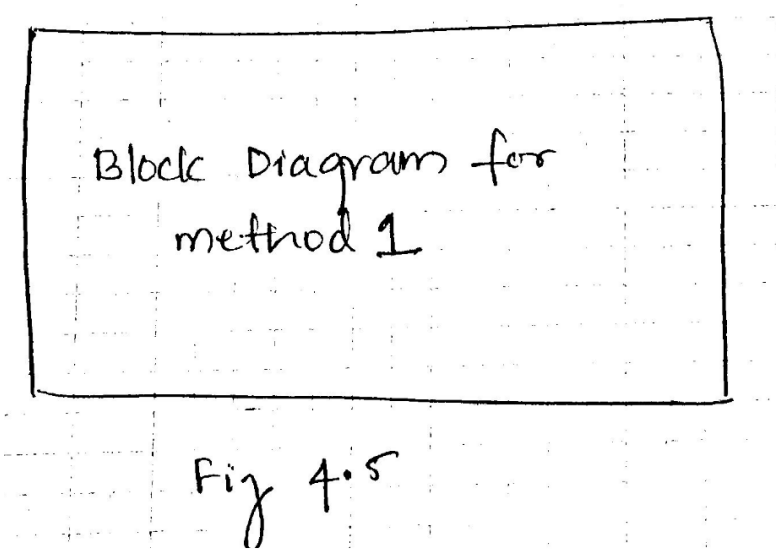

Figure 4.5: A visual guide to training Method 1 - the raw Voxelmorph model with the NCC optimization function.

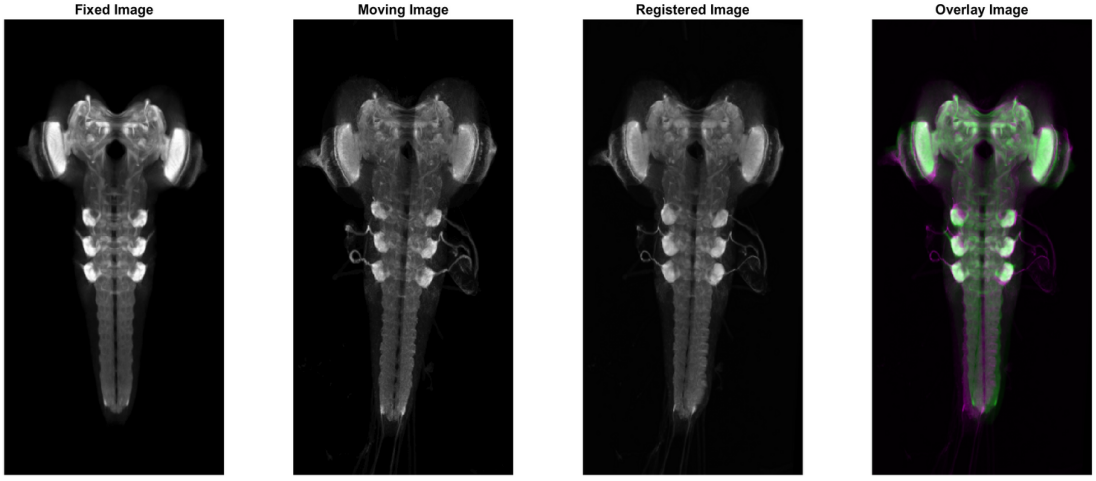

Figure 4.6: An example moving sample image with larger deformation where the Method 1 type of deformable registration did not produce a successful result. Four images are plotted in a row using Matlab. The first image is the fixed image against which the registration is being performed. The second image is the moving image that is being registered. The third image is the result of the registration process, and the fourth image is an overlay of the registered image and the fixed image.

achieving better alignment, particularly when dealing with large deformations. By including this extra information in the training process, we aimed to improve the model's ability to register brain images with significant deformations.

The practice of adding additional, manually annotated information for both the moving images and fixed image is referred to as Method 2. This method is explained in further detail in the following section.

4.5 Method2: Enhancing Method 1 with Novel Landmark Information.

As described in Section 4.2, in image registration, landmarks were considered to be the "Gold Standard" for determining the accuracy of the registration process. The goal was to align the fixed and moving images in such a way that the landmarks in both images matched perfectly, resulting in a landmark registration error (LRE) of zero. The LRE is the Euclidean distance between the landmarks in the registered and fixed images and to

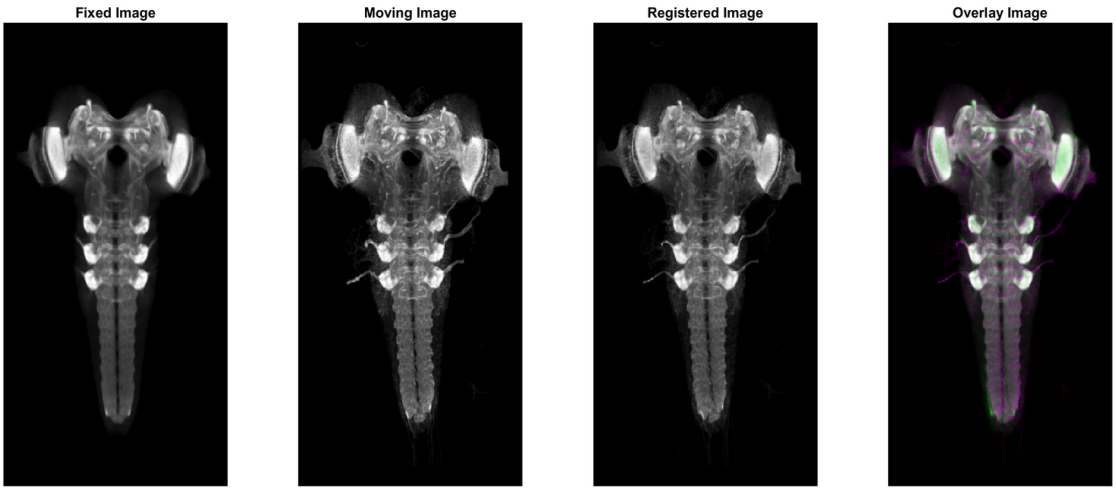

Figure 4.7: An example moving sample image with smaller deformation where the Method 1 type of deformable registration produced a successful result. Four images are plotted in a row using Matlab. The first image is the fixed image against which the registration is being performed. The second image is the moving image that is being registered. The third image is the result of the registration process, and the fourth image is an overlay of the registered image and the fixed image.

improve the accuracy of the registration, the LRE was used as an additional loss function for the network, and thus encouraging the model to learn the registration process to minimize the LRE.

Biologists identified 30 points on the drosophila larva that could be used as landmarks for image registration (Figure 4.3). However, manually annotating all 30 landmarks on every training sample was time-consuming. For samples with minimal deformations, the raw Voxelmorph model already performed well, so additional annotations were not needed. Even in cases where the Voxelmorph model did not perform well, the poor performance was only observed in specific regions, such as the tip of the Ventral Nerve Cord. Therefore, we only annotated landmarks on samples where the raw Voxelmorph model had difficulty achieving accurate registration. To be more specific, we chose to annotate only in such samples and regions where we were certain that Voxelmorph would require assistance. This means that each training sample may have had a different number of landmarks, including zero for samples with minimal deformations where Voxelmorph already performed well. To address this issue, we developed a method that could take into account of the presence of variable number of landmarks between the training samples and calculate the average LRE. This allowed us to minimize the annotation effort while still obtaining good performance from the model.

Incorporating landmark points as auxiliary information is a novel method that has not been used previously. The original Voxelmorph paper utilized segmentation maps to help improve registration results. The following block diagram illustrates the training process.

The optimization function we attempted to minimize was:

$$L_{\text{combined}} = L_{\text{main}} + L_{\text{aux}} \tag{4.2}$$

Let's call the main loss L_{main} and the landmark registration error L_{aux}.

Main loss:

$$L_{\text{main}} = -\frac{1}{N}\sum_{i=1}^{N} \text{NCC}(Fixed, Moved_i) \tag{4.3}$$

Landmark registration error:

$$L_{\text{aux}} = \frac{1}{NL}\sum_{n=1}^{N}\sum_{l=1}^{L}\sqrt{(p_l - q_{n,l})^2} \tag{4.4}$$

In this Equation 4.3, N is the number of moving images, and $NCC(Fixed, Moved_i)$ is the normalized cross correlation between $Fixed$ and $Moved_i$. The loss function is the negative average of the normalized cross correlation between the fixed image and each of the moving images.

In this Equation 4.4, N is the number of moving images, L is the number of landmarks, p_l is the coordinates of the l-th landmark in the fixed image, and q_n,l is the coordinates of the l-th landmark in the n-th moving

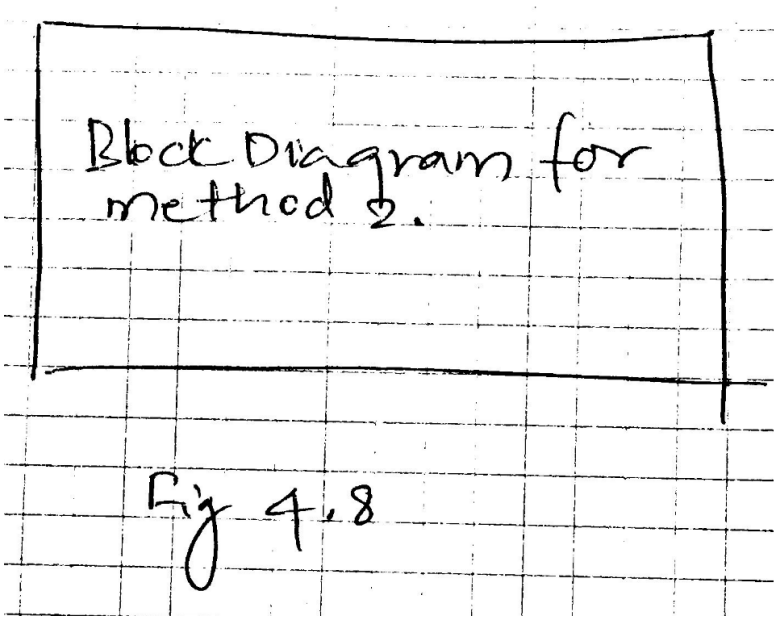

Figure 4.8: A visual guide to training Method 2 - the raw Voxelmorph model with landmark points as auxiliary information.

image. The auxiliary loss is the average of the euclidean distance between the landmarks in the fixed image and the landmarks in each of the moving images.

4.5.1 Results

Using the same configurations as in Method 1, landmark points as auxiliary information was incorporated into the training. The model was retrained on the `Janelia dataset` and tested on the `Larvalign dataset`. It was found that there is an improvement in the registration accuracy with landmark points as auxiliary information.

The following figures, Figure 4.9 and Figure 4.10, show the results of registration using Method 2 on the same example samples used in Method 1.

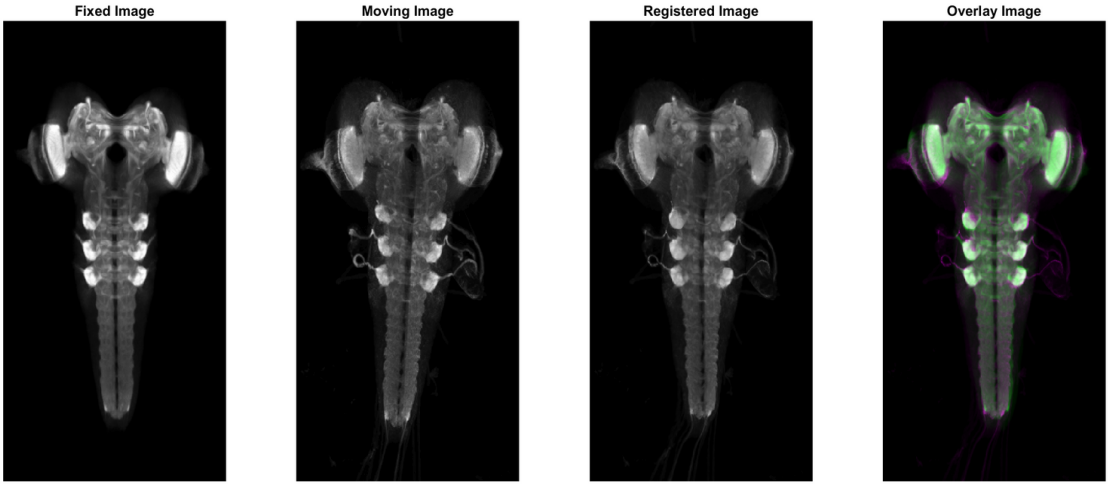

Figure 4.9: An example moving sample image with larger deformation where the Method 2 type of deformable registration could produce a successful result when large deformation is present in contrast to Method 1. Four images are plotted in a row using Matlab. The first image is the fixed image against which the registration is being performed. The second image is the moving image that is being registered. The third image is the result of the registration process, and the fourth image is an overlay of the registered image and the fixed image.

Method 2 produced significantly improved registered results compared to Method 1, as can be evidenced in the two figures Figure 4.6 and Figure 4.9. In Figure 4.6, Method 1 failed to produce satisfactory registration,

particularly at the lower tip of the VNC. However, in Figure 4.9, it can be witnessed that with the help of landmark information, Method 2 was able to successfully register the image. The results in both figures demonstrate the efficacy of using landmark information in improving registration accuracy.

For the moving image with little deformation like in Figure 4.7, the model continued to produce accurate registration results, as expected. This can be verified in the Figure 4.10.

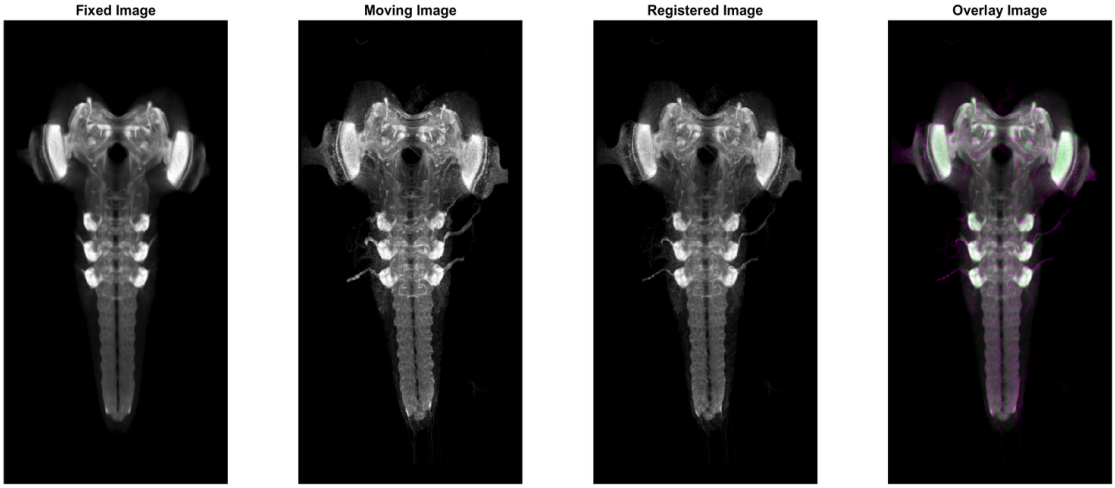

Figure 4.10: An example moving sample image with smaller deformation where the Method 2 type of deformable registration could contine to produce a successful result like Method 1. Four images are plotted in a row using Matlab. The first image is the fixed image against which the registration is being performed. The second image is the moving image that is being registered. The third image is the result of the registration process, and the fourth image is an overlay of the registered image and the fixed image.

While the incorporation of landmarks did improve registration accuracy, in some cases, on moving images with significant deformation, this model produced artifacts on the registered output. An example of this can be seen in the following Figure 4.11.

The two images in focus on the lower tip of the Ventral Nerve Cord (VNC) show the moving image before and after registration. As seen, it appears as though the structures in this area have been erased after the registration was performed.

4.5.2 Motivation for Method3

The artifact is commonly observed on moving images with large deformations and may be due to the model's inherent limitations in handling such large deformations. The addition of auxiliary information and loss is trying to enforce the model to achieve near perfect registration, while the model is not yet capable of doing it. This new force in play can cause the network to compromize on the qualitative features while gaining good scores in terms of error scores. To fix this problem, we need to make the model better at handling large deformations. We can also use the auxiliary information as a guide, but it should not be the main way to solve the problem.

4.6 Method3: Pyramid Gaussian Filtering and Cascaded Training.

Image registration can be challenging because the optimization surface can be complex and lead to suboptimal results if the model gets stuck in local optima. However, literature study has shown that performing registration in a coarse-to-fine fashion can help the model escape local optima and achieve more accurate results. This approach has been effective in both the pre-deep learning and deep learning eras.

This technique is called "registration in pyramid fashion," which involves applying pyramid-style Gaussian filtering and aligning images from the lowest resolution to the highest resolution. This method is effective because it ensures that both the large and small structures in the images are accurately aligned, whereas the raw Voxelmorph model only aligns fine details and not larger structures.

The method involved processing an image at five different resolutions, and aligning the images in a coarse-to-fine manner, starting at the lowest resolution and gradually increasing the resolution at each stage. At each stage, the transformation learned in the previous stage was incorporated into the current stage. To decrease the

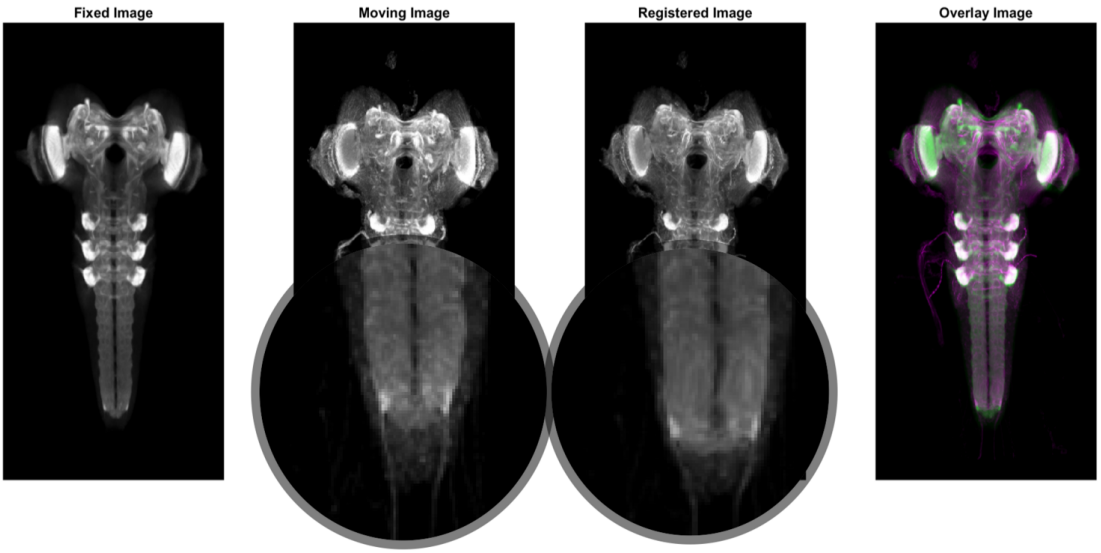

Figure 4.11: An example moving sample image with large deformations at the lower tip of Ventral Nerve Cord (VNC) where the Method 2 produced an artifact on the registered output. Four images are plotted in a row using Matlab. The first image is the fixed image against which the registration is being performed. The second image is the moving image that is being registered. The third image is the result of the registration process, and the fourth image is an overlay of the registered image and the fixed image.

resolution of the image, a Gaussian blur was applied using different values of σ, while keeping the width and height of the image constant. This process can be represented mathematically using the equation for Gaussian blurring.

Let's say we have an input image with pixel values $I(x, y)$, and we want to apply Gaussian blurring to it using a kernel with size $n \times n$. The resulting image, $G(x, y)$, can be computed as follows:

$$G(x,y) = \frac{1}{Z} \sum_{s=-\frac{n}{2}}^{\frac{n}{2}} \sum_{t=-\frac{n}{2}}^{\frac{n}{2}} I(x+s, y+t) \cdot K(s,t) \tag{4.5}$$

where $K(s, t)$ is the Gaussian kernel and Z is a normalization factor. The Gaussian kernel is defined as:

$$K(s,t) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{s^2+t^2}{2\sigma^2}\right) \tag{4.6}$$

Here, σ is a parameter that determines the amount of blur applied to the image. A larger value of σ results in more blur. The size of the kernel, n, is usually chosen to be an odd number such as 3, 5, or 7. We start by processing the image at the lowest resolution, where the value of σ is set to 4. In subsequent stages, we progressively increase the resolution of the image by decreasing the value of σ to 2, 1, 0.5, and finally to 0, which is the original resolution of the image.

The process is illustrated in the following block diagram.

This method for image registration involved using the raw Voxelmorph model without auxiliary information and with the same configurations at each stage of the cascaded process. Both the moving and fixed images were pre-processed with Gaussian blurring using different sigma values for each stage. The inputs for each stage are labeled input0 through input4, with input0 representing the input images for stage 0 (blurred with sigma = 4), input1 representing the input images for stage 1 (blurred with sigma = 2), and so on. The final input, input4, consists of the original, unmodified images. The models at each stage are labeled model0 through model4.

4.6.1 Stages

Stage 0

In stage 0, the model was trained using input images that had been pre-processed with Gaussian blurring using sigma = 4. The predicted deformation field from this model was used to transform the moving image, creating

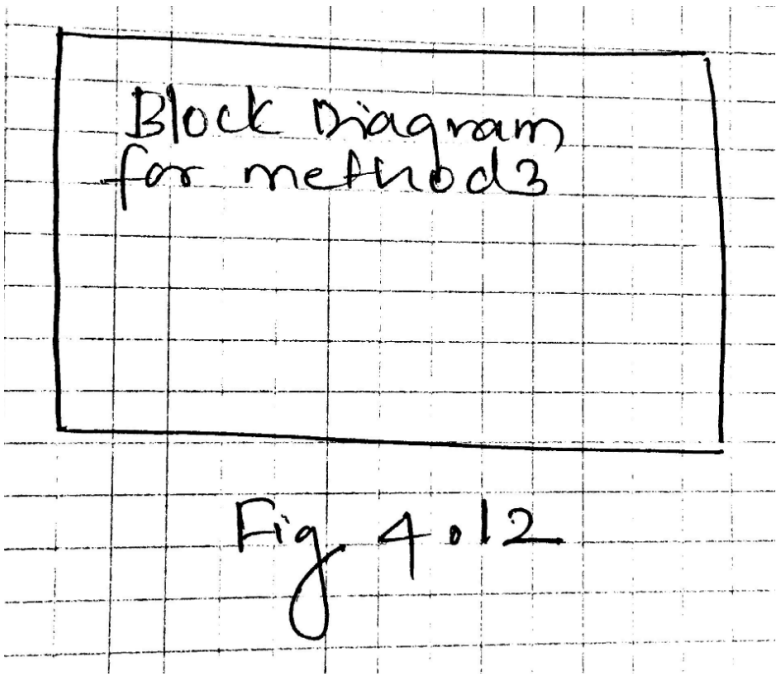

Figure 4.12: A visual guide to training Method 3 - cascade of Voxelmorph models without auxiliary information.

a version called the "moved image." This moved image was then compared to the fixed image, which had also been pre-processed with Gaussian blurring using sigma = 4. The optimization function (Normalized Cross Correlation) used in this stage aimed to maximize the similarity between the moved and fixed images.

$$L = -\frac{1}{N}\sum_{i=1}^{N} \text{NCC}(Fixed, Moved_i) \tag{4.7}$$

where:

- L is the loss function.
- N is the number of moving images.
- $\mathbf{Moved}_i$ is the ith moved image.
- **Fixed** is the fixed image.
- $NCC(\mathbf{Fixed}, \mathbf{Moved}_i)$ is the normalized cross correlation between the ith moved image and the fixed image.

Stage 1

After the training was completed for stage 0, the trained model parameters were used to roughly align the input1 images, which had been pre-processed with Gaussian blurring using sigma = 2. These roughly aligned input1 images were then used to train model1 to predict a deformation field, which was then applied to the moving image using a spatial transformer network. The resulting transformed moving image was compared to the fixed image, which had also been pre-processed with Gaussian blurring using sigma = 2, to calculate the similarity score. The optimization function used in this stage was the same as the one used in the previous stage.

Stage 2, 3 and 4

The aforementioned steps were repeated iteratively for all subsequent stages, incorporating the learned transformation from the previous stages into the current stage. This process followed a coarse-to-fine approach, meaning that the transformation was initially learned at a coarser resolution and then refined successively to the finer resolutions.

4.6.2 Results

There were no artifacts introduced when registering the moving sample using this model, as can be seen in the Figure 4.13. This was confirmed by comparing the results to the reference image in Figure 4.11

Method 3 performed better than the previous two methods in terms of registering the moving samples with large and smaller deformations, as shown in Figure 4.14 and Figure 4.15. However, there was room for improvement at the lower tip of the Ventral Nerve Cord (VNC) where the nerve entry points can be aligned more accurately.

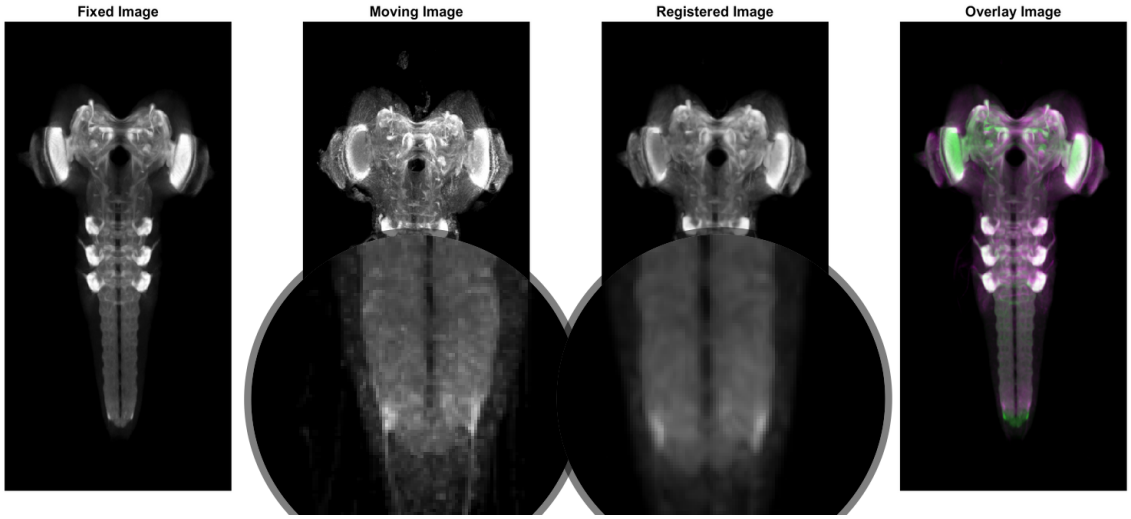

Figure 4.13: An example moving sample image with large deformations at the lower tip of Ventral Nerve Cord (VNC) where the Method 3 did not produce an artifact on the registered output. Four images are plotted in a row using Matlab. The first image is the fixed image against which the registration is being performed. The second image is the moving image that is being registered. The third image is the result of the registration process, and the fourth image is an overlay of the registered image and the fixed image.

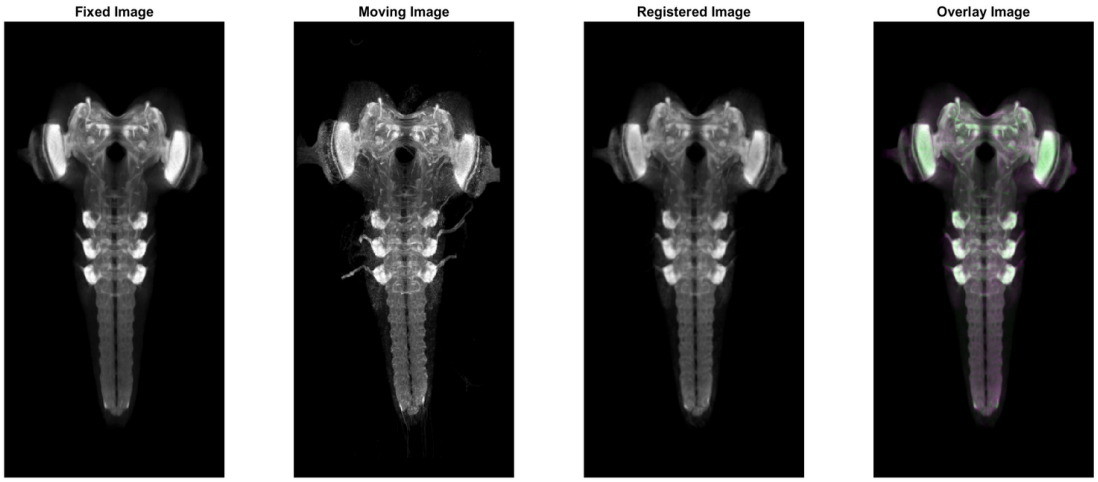

Figure 4.14: An example moving sample image with smaller deformation where the Method 3 type of deformable registration could continue to produce a successful result like Method 1 and Method 2. Four images are plotted in a row using Matlab. The first image is the fixed image against which the registration is being performed. The second image is the moving image that is being registered. The third image is the result of the registration process, and the fourth image is an overlay of the registered image and the fixed image.

The following Figure 4.16 demonstrate the output of the registered moving image as it is propogated through each stage of the cascade. The corrections are applied in a coarse to fine manner, with the transformation becoming more precise at progressively higher resolutions as the sample progresses through these stages.

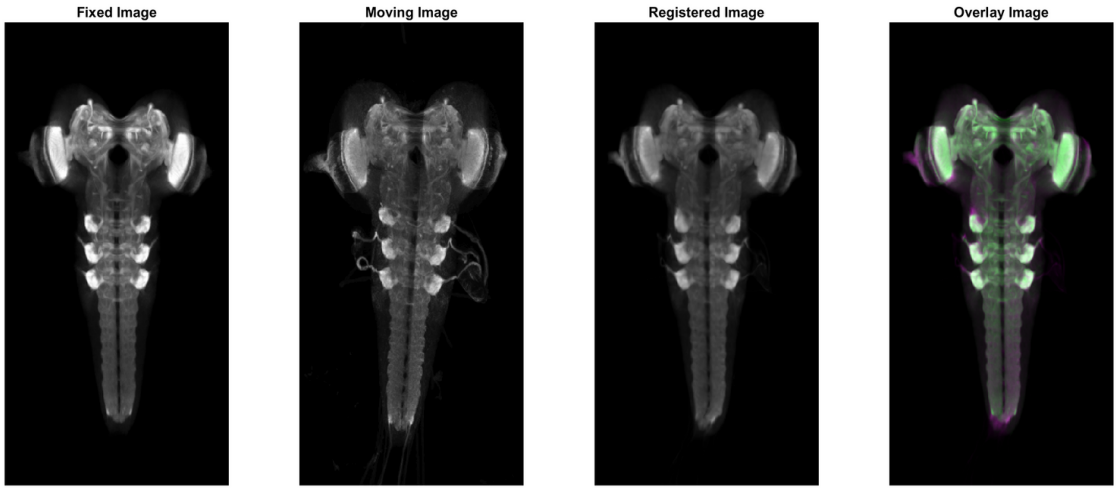

Figure 4.15: An example moving sample image with larger deformation where the Method 3 type of deformable registration could produce a successful result when large deformation is present in contrast to Method 1. Four images are plotted in a row using Matlab. The first image is the fixed image against which the registration is being performed. The second image is the moving image that is being registered. The third image is the result of the registration process, and the fourth image is an overlay of the registered image and the fixed image.

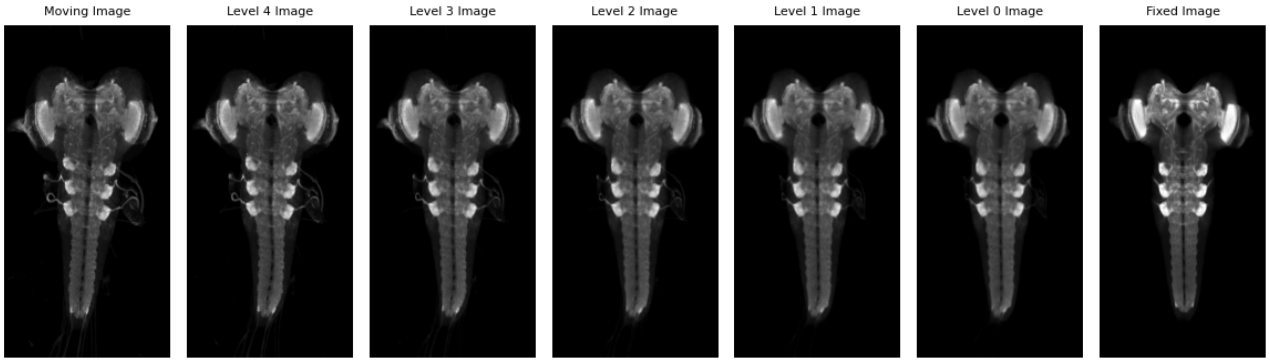

Figure 4.16: Registration output through a series of cascaded stages performed in a coarse-to-fine manner.

4.6.3 Motivation for Method4

This method was able to handle large deformations more effectively than the raw Voxelmorph model used in Method 1 and at the same time not producing any artifacts. The results demonstrated this. While the results were satisfactory, there was a clear potential for further improvement at the lower end of the Ventral Nerve Cord by incorporating landmark point information during training, as seen in the results of Method 2. Further experiments will be conducted to assess the benefits of adding this additional information in the next method.

4.7 Method4: Enhancing Method 3 with Novel Landmark Information.

In the Method 2, it was found through experimentation that using landmark information improved the performance of Method 2. However, it was difficult to incorporate this landmark information into Method 4 because the input for each stage consisted of different pre-aligned image samples. This meant that the landmarks, which had been annotated on one stage, could not be used in later stages because the pre-alignment process caused the landmarks to move to different positions. Re-annotating the landmarks for each stage would be impractical especially if there were many stages or a large number of images.

To address the issue of having to manually re-annotate landmarks for each stage in Method 4, we developed an approach that allowed us to incorporate landmark information at all stages without requiring manual re-

annotation. Our idea was based on the fact that we knew the input images for each stage were the same, except that they had different levels of Gaussian blurring applied and had been pre-aligned using the deformation correction from the previous stage. This meant that the initial annotations made on any of these images could still be used at later stages if we were able to propagate the landmarks to their new positions using the predicted deformation field.

This is again a novel method that has been introduced in this theis work. These moved landmarks, called "pseudo-annotated" landmarks, may not be as accurate as manually annotated landmarks, but they were precise enough to work with the blurred images in the earlier stages where the structures were not clear to require precise manual annotation.

$$l_n^{new} = l_n^{old} - d_n \tag{4.8}$$

where:

l_n^{new} is the new position of the landmark in image n

l_n^{old} is the old position of the landmark in image n

d_n is the deformation field for image n at the position of the landmark

This Equation 4.8 states that the new position of the landmark is obtained by subtracting the deformation field at that position from the old position of the landmark. The process of using the predicted deformation field to move both the image and the landmark points for the next stage is illustrated in the block diagram below.

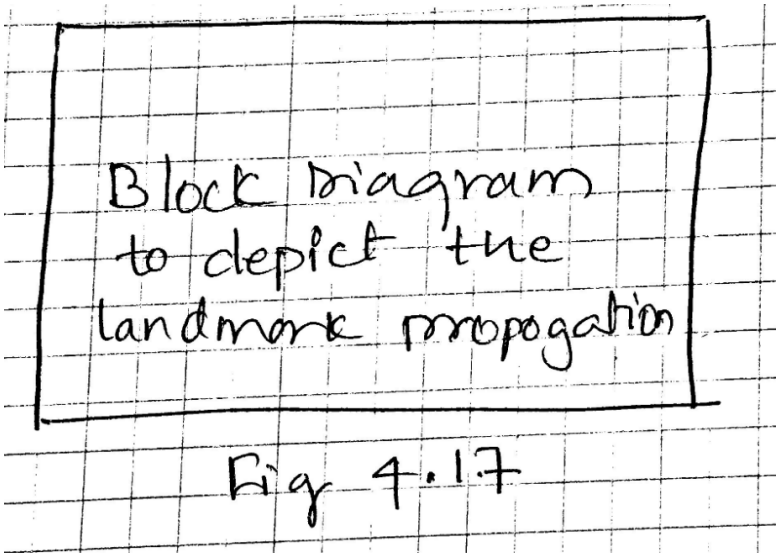

Figure 4.17: A visual guide to training Method 4 - cascade of Voxelmorph model with pseudo-landmark points as auxiliary information.

In each stage of this cascaded layer, the Voxelmorph model with auxiliary landmark information incorporated and configured in the same way as in Method 2 was used. Both the moving and fixed images were pre-processed with Gaussian blurring using different sigma values for each stage. The inputs for each stage are labeled input0 through input4, with input0 representing the input images for stage 0 (blurred with sigma = 4), input1 representing the input images for stage 1 (blurred with sigma = 2), and so on. The final input, input4, consists of the original, unmodified images. The models at each stage are labeled model0 through model4.

4.8 Stages

Stage 0

In this stage, the model was trained to predict a deformation field that warped the moving image to align it with the fixed image. The model used manually annotated landmarks and input images that had been blurred using Gaussian blur with a sigma of 4. The transformed version of the moving image, called the "moved image," was then compared to the fixed image, which had also been blurred using Gaussian blur with a sigma of 4, to

calculate a similarity score using normalized cross correlation. Additionally, the model was also evaluated based on the landmark registration error to guide the learning process.

Therefore, the optimization function attempted to minimize was:

$$L_{\text{combined}} = L_{\text{main}} + L_{\text{aux}} \tag{4.9}$$

Let us call the main loss L_{main} and the landmark registration error L_{aux}.

Main loss:

$$L_{\text{main}} = -\frac{1}{N}\sum_{i=1}^{N} \text{NCC}(Fixed, Moved_i) \tag{4.10}$$

Landmark registration error:

$$L_{\text{aux}} = \frac{1}{NL}\sum_{n=1}^{N}\sum_{l=1}^{L}\sqrt{(p_l - q_{n,l})^2} \tag{4.11}$$

In this Equation 4.10, N is the number of moving images, and $NCC(Fixed, Moved_i)$ is the normalized cross correlation between $Fixed$ and $Moved_i$. The loss function is the negative average of the normalized cross correlation between the fixed image and each of the moving images.

In this Equation 4.11, N is the number of moving images, L is the number of landmarks, p_l is the coordinates of the l-th landmark in the fixed image, and q_n,l is the coordinates of the l-th landmark in the n-th moving image. The auxiliary loss is the average of the euclidean distance between the landmarks in the fixed image and the landmarks in each of the moving images.

Stage 1

After training was completed for stage 0, the trained model parameters were used to roughly align the input1 images, which had been pre-processed with Gaussian blurring using a sigma value of 2. These roughly aligned input1 images were then used to train model1 to predict a deformation field, which was then applied to the moving image using a spatial transformer network. The resulting transformed moving image was compared to the fixed image, which had also been pre-processed with Gaussian blurring using a sigma value of 2, to calculate the similarity score. The optimization function used in this stage was the same as the one used in the previous stage. In this stage, "pseudo-annotated" landmark points were used. These landmark points had been obtained by propagating manually annotated landmarks from the previous stage using the deformation field predicted by model0.

$$landmark_{stage1,m,n} = landmark_{stage0,m,n} - deformation\_field_{stage0,m,n} \tag{4.12}$$

Here,

- "$landmark_{stage1,m,n}$" is the landmark point for stage 1 at pixel m for image n.
- "$landmark_{stage0,m,n}$" is the landmark point for stage 0 at pixel m for image n.
- "$deformation\_field_{stage0,m,n}$" is the deformation field from stage 0 at pixel m for image n.

This Equation 4.12 is valid for all values of m such that m is a landmark point for an image in stage 0, and for all values of n such that n belongs to N, where N is the number of images in stage 1.

Stage 2 and 3

The aforementioned steps were repeated iteratively for all subsequent stages, gradually incorporating the learned transformation from the previous stage into the current stage. This process followed a coarse-to-fine approach of registration using landmark. The landmark points were propagated from stage 0 to stage s by using the deformation field predicted by each model.

```
# N: The number of landmark lists.
# M: The number of landmark points in each landmark list.
# S: The number of stages.
for n in range(N):
# n: The index of the current landmark list.
  for m in range(M):
```

```
    # m: The index of the current landmark point.
      for s in range(S):
      # s: The index of the current stage.
        # deformation_field: The deformation field for the current stage.
        deformation_field = find_deformation_field(s)
        # old_landmark_point: The old landmark point for the current landmark point
        # and stage.
        # new_landmark_point: The new landmark point for the current landmark point
        # and stage, which is calculated by subtracting the deformation field from
        # the old landmark point.
        new_landmark_point = old_landmark_point - deformation_field
        # Update the landmark point with the new value.
        update_landmark_point(new_landmark_point)

```

The pseudo code loops through each landmark list, landmark point, and stage, and calculates the deformation field for the current stage. The new landmark point is then calculated by subtracting the deformation field from the old landmark point, and the *update_landmark_point* function is used to update the landmark point with the new value. This process is repeated for all landmark points in all landmark lists for all stages.

Stage 4: The last stage

The above software approach could, in principle, be used to propagate the landmarks from stage 3 to stage 4 as well. However, it was decided to perform one more manual annotation of the landmark points for the final stage in order to obtain more accurate results. This was because the input images for stage 4 are the original, unmodified images, which were not blurred, and using "pseudo-annotated" landmark points for this stage may not be accurate. The input images for stage 4 were already nicely pre-aligned due to the coarse-to-fine registration approach, and using accurate landmark points would allow the network to achieve better final registration.

4.9 Results

The Figure 4.18 consists of 3 moving image samples in 3 rows. Each column represents the output of the corresponding movie image through different stages through the cascaded network. This figure is representative of the output from Method 4.

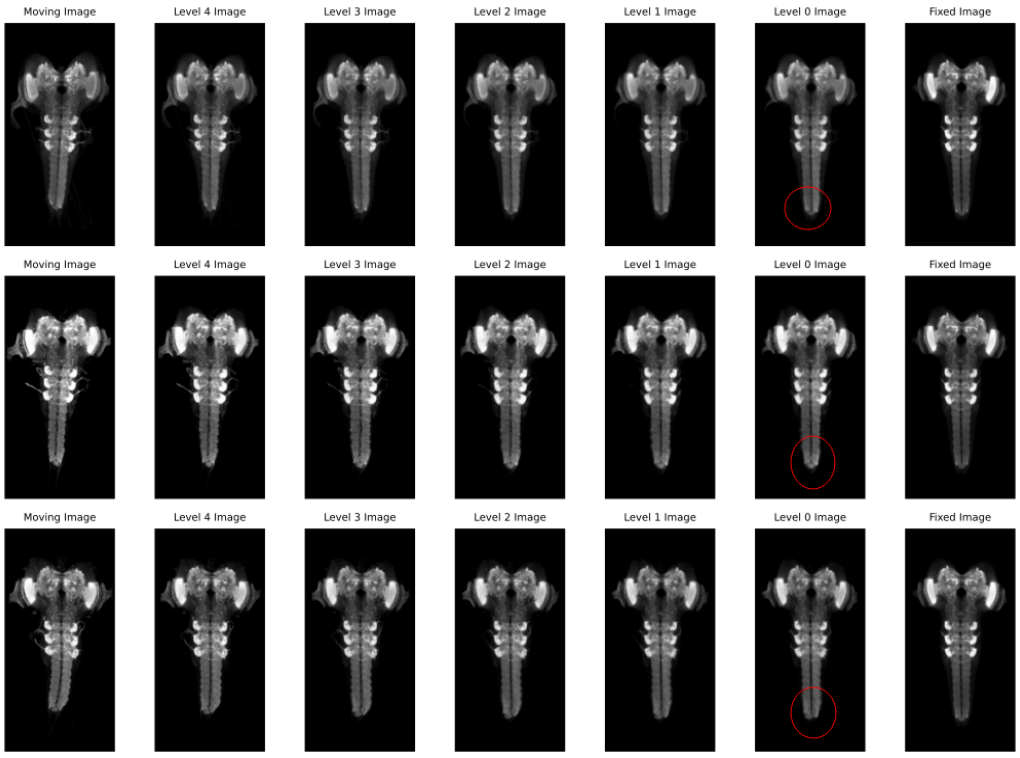

Figure 4.18: Coarse-to-fine registration using landmark information.

Similarly, the Figure 4.19 consists of same 3 moving image samples in 3 respective rows. Each column, again, represents the output of the corresponding movie image through different stages through the cascaded network. This figure is representative of the output from Method 3.

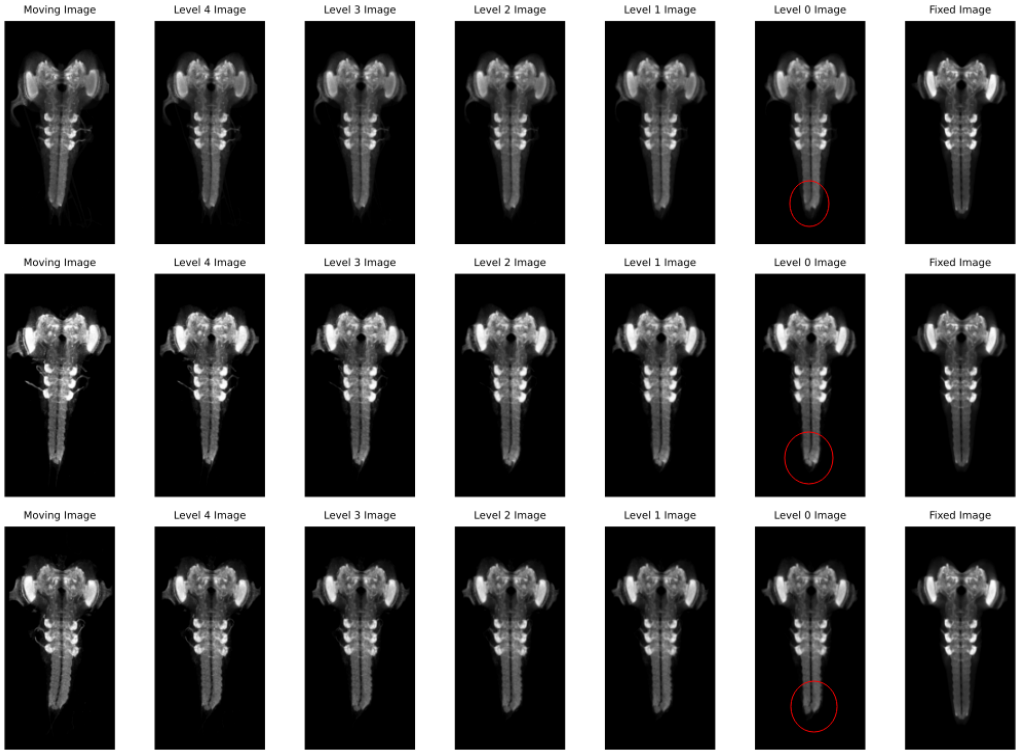

Figure 4.19: Coarse-to-fine registration without using landmark information.

Comparing the figures Figure 4.18 and Figure 4.19, it is evident that adding landmark points at the lower tip of the Ventral Nerve Cord has resulted in better registration accuracy for Method 4 than for Method 3.

Chapter 5

Results

5.1 Metrics

5.2 Quality Assessment

5.2.1 Qualitative Assessment

5.2.2 Quantitative Assessment

Chapter 6

Discussion

Chapter 7

Conclusion

Bibliography

[1] S. Muenzing, M. Strauch, J. Truman, K. Bühler, A. Thum, and D. Merhof, "larvalign: Aligning gene expression patterns from the larval brain of drosophila melanogaster," *Neuroinformatics*, vol. 16, 01 2018.

[2] G. Balakrishnan, A. Zhao, M. R. Sabuncu, J. Guttag, and A. V. Dalca, "VoxelMorph: A learning framework for deformable medical image registration," *IEEE Transactions on Medical Imaging*, vol. 38, pp. 1788–1800, aug 2019.

[3] B. D. de Vos, F. F. Berendsen, M. A. Viergever, H. Sokooti, M. Staring, and I. Išgum, "A deep learning framework for unsupervised affine and deformable image registration," *Medical Image Analysis*, vol. 52, pp. 128–143, feb 2019.

[4] G. Wu, M. Kim, Q. Wang, Y. Gao, S. Liao, and D. Shen, "Unsupervised deep feature learning for deformable registration of mr brain images," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013* (K. Mori, I. Sakuma, Y. Sato, C. Barillot, and N. Navab, eds.), (Berlin, Heidelberg), pp. 649–656, Springer Berlin Heidelberg, 2013.

[5] Y. Fu, Y. Lei, T. Wang, W. J. Curran, T. Liu, and X. Yang, "Deep learning in medical image registration: a review," *Physics in Medicine and Biology*, vol. 65, p. 20TR01, oct 2020.

[6] H. Sokooti, B. de Vos, F. Berendsen, B. P. F. Lelieveldt, I. Išgum, and M. Staring, "Nonrigid image registration using multi-scale 3d convolutional neural networks," in *Medical Image Computing and Computer Assisted Intervention MICCAI 2017* (M. Descoteaux, L. Maier-Hein, A. Franz, P. Jannin, D. L. Collins, and S. Duchesne, eds.), (Cham), pp. 232–239, Springer International Publishing, 2017.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[8] J.-P. Thirion, "Image matching as a diffusion process: an analogy with maxwell's demons," *Medical Image Analysis*, vol. 2, no. 3, pp. 243–260, 1998.

[9] G. Christensen, R. Rabbitt, and M. Miller, "Deformable templates using large deformation kinematics," *IEEE Transactions on Image Processing*, vol. 5, no. 10, pp. 1435–1447, 1996.

[10] N. D. Cahill, J. A. Noble, and D. J. Hawkes, "Demons algorithms for fluid and curvature registration," in *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pp. 730–733, 2009.

[11] S. Klein, M. Staring, K. Murphy, M. A. Viergever, and J. P. W. Pluim, "elastix: A toolbox for intensity-based medical image registration," *IEEE Transactions on Medical Imaging*, vol. 29, no. 1, pp. 196–205, 2010.

[12] B. B. Avants, N. Tustison, G. Song, P. A. Cook, A. Klein, and J. C. Gee, "A reproducible evaluation of ants similarity metric performance in brain image registration," *NeuroImage*, vol. 54, pp. 2033–2044, 2011.

[13] H. Johnson and G. Christensen, "Consistent landmark and intensity-based image registration," *IEEE Transactions on Medical Imaging*, vol. 21, no. 5, pp. 450–461, 2002.

[14] B. B. Avants, C. L. Epstein, M. Grossman, and J. C. Gee, "Symmetric diffeomorphic image registration with cross-correlation: Evaluating automated labeling of elderly and neurodegenerative brain," *Medical image analysis*, vol. 12 1, pp. 26–41, 2008.

[15] S. Muenzing, B. Ginneken, K. Murphy, and J. Pluim, "Supervised quality assessment of medical image registration: Application to intra-patient ct lung registration," *Medical image analysis*, vol. 16, 07 2012.

[16] M. Baiker-Sørensen, M. Staring, C. Löwik, J. Reiber, and B. Lelieveldt, "Automated registration of whole-body follow-up microct data of mice," vol. 14, pp. 516–23, 09 2011.

[17] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, dec 2017.

[18] G. Wu, M. Kim, Q. Wang, B. C. Munsell, and D. Shen, "Scalable high-performance image registration framework by unsupervised deep feature representations learning," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 7, pp. 1505–1516, 2016.

[19] T. Vercauteren, X. Pennec, A. Perchant, and N. Ayache, "Diffeomorphic demons: Efficient non-parametric image registration," *NeuroImage*, vol. 45, no. 1, Supplement 1, pp. S61–S72, 2009. Mathematics in Brain Imaging.

[20] X. Pennec, P. Cachier, and N. Ayache, "Understanding the "demon's algorithm": 3d non-rigid registration by gradient descent," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI'99* (C. Taylor and A. Colchester, eds.), (Berlin, Heidelberg), pp. 597–605, Springer Berlin Heidelberg, 1999.

[21] J.-M. Peyrat, H. Delingette, M. Sermesant, X. Pennec, C. Xu, and N. Ayache, "Registration of 4d time-series of cardiac images with multichannel diffeomorphic demons," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2008* (D. Metaxas, L. Axel, G. Fichtinger, and G. Székely, eds.), (Berlin, Heidelberg), pp. 972–979, Springer Berlin Heidelberg, 2008.

[22] D. Forsberg, Y. Rathi, S. Bouix, D. Wassermann, H. Knutsson, and C.-F. Westin, "Improving registration using multi-channel diffeomorphic demons combined with certainty maps," in *Multimodal Brain Image Analysis* (T. Liu, D. Shen, L. Ibanez, and X. Tao, eds.), (Berlin, Heidelberg), pp. 19–26, Springer Berlin Heidelberg, 2011.

[23] J.-M. Peyrat, H. Delingette, M. Sermesant, C. Xu, and N. Ayache, "Registration of 4d cardiac ct sequences under trajectory constraints with multichannel diffeomorphic demons," *IEEE transactions on medical imaging*, vol. 29, pp. 1351–68, 03 2010.

[24] G. Wu, P.-T. Yap, M. Kim, and D. Shen, "Tps-hammer: Improving hammer registration algorithm by soft correspondence matching and thin-plate splines based deformation interpolation," *NeuroImage*, vol. 49, pp. 2225–2233, 2010.

[25] D. Shen, "Image registration by local histogram matching," *Pattern Recognition*, vol. 40, pp. 1161–1172, 04 2007.

[26] S. Miao, Z. J. Wang, and R. Liao, "A cnn regression approach for real-time 2d/3d registration," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1352–1363, 2016.

[27] E. Chee and Z. Wu, "Airnet: Self-supervised affine registration for 3d medical images using neural networks," 2018.

[28] X. Yang, R. Kwitt, and M. Niethammer, "Fast predictive image registration," in *Deep Learning and Data Labeling for Medical Applications* (G. Carneiro, D. Mateus, L. Peter, A. Bradley, J. M. R. S. Tavares, V. Belagiannis, J. P. Papa, J. C. Nascimento, M. Loog, Z. Lu, J. S. Cardoso, and J. Cornebise, eds.), (Cham), pp. 48–57, Springer International Publishing, 2016.

[29] M.-M. Rohé, M. Datar, T. Heimann, M. Sermesant, and X. Pennec, "Svf-net: Learning deformable image registration using shape matching," in *MICCAI*, 2017.

[30] J. Lyu, M. Yang, J. Zhang, and X. Wang, "Respiratory motion correction for free-breathing 3d abdominal mri using cnn-based image registration: A feasibility study," *The British Journal of Radiology*, vol. 91, p. 20170788, 12 2017.

[31] P. Yan, S. Xu, A. R. Rastinehad, and B. J. Wood, "Adversarial image registration with application for mr and trus image fusion," 2018.

[32] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[33] G. Haskins, U. Kruger, and P. Yan, "Deep learning in medical image registration: a survey," *Machine Vision and Applications*, vol. 31, jan 2020.

[34] J. A. Schnabel, D. Rueckert, M. Quist, J. M. Blackall, A. D. Castellano-Smith, T. Hartkens, G. P. Penney, W. A. Hall, H. Liu, C. L. Truwit, F. A. Gerritsen, D. L. G. Hill, and D. J. Hawkes, "A generic framework for non-rigid registration based on non-uniform multi-level free-form deformations," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2001* (W. J. Niessen and M. A. Viergever, eds.), (Berlin, Heidelberg), pp. 573–581, Springer Berlin Heidelberg, 2001.

[35] G. Balakrishnan, A. Zhao, M. R. Sabuncu, A. V. Dalca, and J. Guttag, "An unsupervised learning model for deformable medical image registration," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9252–9260, 2018.

[36] S. Zhao, T. Lau, J. Luo, E. I.-C. Chang, and Y. Xu, "Unsupervised 3d end-to-end medical image registration with volume tweening network," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 5, pp. 1394–1404, 2020.

[37] S. Zhao, Y. Dong, E. Chang, and Y. Xu, "Recursive cascaded networks for unsupervised medical image registration," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, oct 2019.

[38] A. Walter, "The methods we're using. . . ." https://www.walter-lab.com/methods, 2020. Accessed on 2023-01-01.

[39] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, "Automated design of both the topology and sizing of analog electrical circuits using genetic programming," in *Artificial Intelligence in Design'96*, pp. 151–170, Springer, 1996.

[40] "The a – z of supervised learning, use cases, and disadvantages."

[41] "What is unsupervised learning."

[42] A. S. Yoon, T. Lee, Y. Lim, D. Jung, P. Kang, D. Kim, K. Park, and Y. Choi, "Semi-supervised learning with deep generative models for asset failure prediction," *arXiv preprint arXiv:1709.00845*, 2017.

[43] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[44] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[45] K. P. D. Kamdem, "2 inputs perceptron training." https://www.mathworks.com/matlabcentral/fileexchange/62842-2-inputs-perceptron-training, 2023. MATLAB Central File Exchange. Retrieved January 1, 2023.

[46] I. Neutelings, "Neural networks with TikZ." https://tikz.net/neural_networks/, n.d.

[47] A. Reynolds, "Convolutional neural networks (cnns)." https://anhreynolds.com/blogs/cnn.html, 2023. Retrieved January 1, 2023.

[48] Unknown, "Image." https://miro.medium.com/max/1200/1*ZafDv3VUm60Eh10OeJu1vw.png, 2023. Retrieved January 1, 2023.

[49] T. Szandaa, "Review and comparison of commonly used activation functions for deep neural networks," pp. 203–224, 2021.

[50] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A comprehensive survey of loss functions in machine learning," *Annals of Data Science*, pp. 1–26, 2020.

[51] Unknown, "Line plot of mean squared error loss over training epochs when optimizing the mean squared error loss function." https://machinelearningmastery.com/wp-content/uploads/2018/11/Line-plot-of-Mean-Squared-Error-Loss-over-Training-Epochs-When-Optimizing-the-Mean-Squared-Error-L png, 2023. Retrieved January 1, 2023.

[52] A. Senior, G. Heigold, M. Ranzato, and K. Yang, "An empirical study of learning rates in deep neural networks for speech recognition," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6724–6728, IEEE, 2013.

[53] R. Zaheer and H. Shaziya, "A study of the optimization algorithms in deep learning," in *2019 Third International Conference on Inventive Systems and Control (ICISC)*, pp. 536–539, IEEE, 2019.

[54] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.