Progressively Growing Convolutional Networks for End-to-End Deformable Image Registration

Koen A.J. Eppenhof, Maxime W. Lafarge, and Josien P.W. Pluim

Medical Image Analysis, Eindhoven University of Technology, The Netherlands

ABSTRACT

Deformable image registration is often a slow process when using conventional methods. To speed up deformable registration, there is growing interest in using convolutional neural networks. They are comparatively fast and can be trained to estimate full-resolution deformation fields directly from pairs of images. Because deep learning-based registration methods often require rigid or affine pre-registration of the images, they do not perform true end-to-end image registration. To address this, we propose a progressive training method for end-to-end image registration with convolutional networks. The network is first trained to find large deformations at a low resolution using a smaller part of the full architecture. The network is then gradually expanded during training by adding higher resolution layers that allow the network to learn more fine-grained deformations from higher resolution data. By starting at a lower resolution, the network is able to learn larger deformations more quickly at the start of training, making pre-registration redundant. We apply this method to pulmonary CT data, and use it to register inhalation to exhalation images. We train the network using the CREATIS pulmonary CT data set, and apply the trained network to register the DIRLAB pulmonary CT data set. By computing the target registration error at corresponding landmarks we show that the error for end-to-end registration is significantly reduced by using progressive training, while retaining sub-second registration times.

Keywords: Deformable image registration, multi-resolution methods, convolutional neural networks, deep learning, fast image registration

1. INTRODUCTION

In previous work we have shown that it is possible to train a neural network to estimate non-linear transformations on a thin-plate spline grid. The network was trained on pairs of synthetically deformed pulmonary CT images, but was shown to generalize to inhale-exhale lung image registration. This work has since been extended to a network that can learn full-resolution deformation fields directly from two input images. For this, we used the U-net architecture, which allows estimation of full-resolution deformation fields at high speeds because the registration consists of only one forward pass through the network. The accuracy of this method can compete with existing methods for pulmonary CT registration. One limitation of this previous work is that it requires affine pre-registration of the images, because the network can only estimate the smaller local deformations between affinely registered images and fails to accurately estimate the deformation field without pre-registration. This limitation has been reported by other registration methods based on convolutional neural networks. Examples include deep learning-based pulmonary CT registration, and brain MR registration, where the images are pre-registered to a brain atlas, or to each other.

1.1 Aim

Given the need for pre-registration in deep learning-based registration methods, we propose an extension to our previous method with the purpose to perform true end-to-end image registration, i.e. without requiring pre-registration, while still retaining the high registration speed. Instead of training the network to find the full resolution deformation field from the start, we reason that it is easier to learn large displacements at lower resolutions first, and then expand to higher resolutions. Similar strategies are used in image registration methods that iteratively optimize similarity metrics, by employing a multi-resolution image pyramid. A transformation model found for lower resolution version of the images is then used as initialization for the higher resolution. In optimization-based methods, this leads to a more robust optimization, that reduces local minima. In this paper, we adapt these ideas to training a multi-resolution neural network for deformable registration. Instead
of training the network directly to learn a full-resolution deformation field between two input images, we first let smaller versions of the network learn lower resolution versions of the deformation field. This way, we slowly build up the network during training, until the network can construct full resolution deformation fields. We implement this by parameterizing the network architecture such that it can be trained for a specific resolution of inputs and outputs. This parameterization also allows smooth transitions between the resolutions.

2. METHODS

2.1 Architecture

Growing neural networks have been previously applied in the context of generative adversarial networks (GANs) by Karras et al. Here, we apply a similar method to the U-net architecture. The U-net architecture is composed of several 'resolution levels' that consist of convolutional layers that operate on a specific resolution (image dimension), with low-resolution layers at the bottom and higher resolution layers at the top of the 'U'-shaped architecture. Each resolution level also has a distinct number of learned feature maps. This number doubles for every resolution level, e.g. if the convolutional layers in the top level learn $N$ feature maps, then the convolutional layers in the level below it learn $2N$ feature maps. When we apply this architecture to registration, we use two images as inputs: the fixed and the moving image. The output of the U-net consists of three maps corresponding to the deformation field’s three components, i.e. the displacement in $x$, $y$, and $z$ direction.

2.2 Modifications to the architecture

The architecture remains similar to common implementations of a 3D U-net, except that there are inputs and outputs at every resolution level, instead of only the top level (Figure 1). These inputs and outputs have resolutions that match the resolution of their level. On the left side of the network, the input images are downsampled to the resolution of the level in which they enter using average pooling. The downsampled input passes through a single convolutional layer to match the number of feature maps $N$ for its resolution level $i$. The result and the output of the pooling layer of the resolution level above it, are summed, weighted by weights $\beta_i$ and $1 - \beta_i$. On the right side of each resolution level, an extra convolutional layer learns the three maps for the components of the vector field at the level’s resolution. The vector fields are upsampled to the original resolution, after which they are weighted with parameters $\alpha_i$, and summed to produce one final deformation field at the original resolution of the input images.

2.3 Progressive learning

The modifications to the architecture introduce a set of parameters $\alpha_i(t)$ that determine the influence of each resolution level to the output of the network. The $\beta_i(t)$ parameters determine the weighting between the output of pooling layers and incoming inputs, and are defined as

$$\beta_i = \sum_{j=0}^{i} \alpha_j.$$  \hfill (1)

The $\alpha_i(t)$ parameters are updated during training, and therefore depend on the training iteration $t$. At every point in training, the sum of these parameters is 1, i.e.

$$\sum_i \alpha_i(t) = 1 \text{ for all } t.$$  \hfill (2)

The individual parameters are defined as

$$\alpha_i(t) = \begin{cases} 0 & \text{for } t < \tau_i \text{ and } t > \tau_i + \Delta \\ \frac{t-\tau_i}{\delta} & \text{for } \tau_i \leq t \leq \tau_i + \delta \\ 1 & \text{for } \tau_i + \delta \leq t \leq \tau_i + \Delta - \delta \\ \frac{\tau_i + \Delta - t}{\delta} & \text{for } \tau_i + \Delta - \delta \leq t \leq \tau_i + \Delta. \end{cases}$$  \hfill (3)
Figure 1: The network architecture resembles the original U-net architecture in 3D, but with inputs and outputs at every resolution level. The parameters $\alpha_i$ determine which parts of the architecture are used.

Figure 2: Progressive growing scheme showing the parameters $\alpha_i(t)$ and $\beta_i(t)$ changing over time during training. The colors correspond to the colors of the levels in Figure 1.

The schedules for the five resolution levels in the U-net use $\delta = 200$, $\Delta = 600$, and $\tau_i = -300 + 400i$ and are shown in Figure 2. Whenever one of the weights $\alpha_i$ equals 1, we are essentially training a U-net with $i + 1$ resolution levels. Hence, once we reach $\alpha_0 = \alpha_1 = \alpha_2 = \alpha_3 = 0$ and $\alpha_4 = 1$ we have arrived at a normal five level U-net architecture. An example of the network architecture at three points in training is given in Figure 3, which shows the network at the start of training, at the transition between the first and second resolution level, and after that transition is complete and two resolution layers are being trained.

2.4 Data

We train on and apply the network to pulmonary CT images. We train the network on the public CREATIS and POPI data sets, and apply the trained network to the DIRLAB data set to show generalization to a different data set of pulmonary CT images. We crop the images such that the lungs are still completely visible, and resize the resulting image to a $128 \times 128 \times 128$ voxel resolution. We create lung masks by segmenting all voxels with Hounsfield units lower than -250, which corresponds to low density tissue inside the lungs. These masks are used to focus the training on the voxels inside the lungs (see Section 2.6).

2.5 Training set construction

During training, random synthetic transformations are applied to the fourteen (seven pairs) CREATIS and POPI images. Every iteration of training, one of the images in the training set is selected. Two random transformations are applied to this image: one to augment the data set ($T_{augm}$), and one that is actually learned by the network.
Figure 3: Illustration of the transition between a one level and a two level network. The remainder of the network (levels 3, 4, and 5) are not shown for clarity. The values of the parameters $\alpha$ and $\beta$ are shown in red for the state of the network at iterations 0, 150, and 300 respectively.

Figure 4: At training time, an image from the training set is transformed twice. The deformation field of the net transformation between the images is learned by the network. At test time, the two images are replaced by the moving and fixed image.

$\mathbf{T}_{\text{learned}}$. The latter is applied together with the augmentation transformation, such that we obtain a pair of images $I[\mathbf{T}_{\text{augm}}(\mathbf{x})]$ and $I[(\mathbf{T}_{\text{augm}} \circ \mathbf{T}_{\text{learned}})(\mathbf{x})]$. From these two images, the network will learn the underlying deformation vector field, i.e. of the transformation $\mathbf{T}_{\text{learned}}$ (Figure 4).

The augmentation transformation $\mathbf{T}_{\text{augm}}$ is created by sampling displacements on a $2 \times 2 \times 2$ grid from a uniform distribution in the $[-12.8, 12.8]$ voxel range. The learned transformation $\mathbf{T}_{\text{learned}}$ consists of a sequence of two transformations: one coarse B-spline transformation using a $4 \times 4 \times 4$ size B-spline grid, and one fine B-spline transformation using an $8 \times 8 \times 8$ size grid. The displacements on those grids are sampled from a uniform distribution in the ranges $[-25.6, 25.6]$ voxels and $[-6.4, 6.4]$ voxels respectively.

At test time, a pair of moving and fixed images replaces the pair of images: $I[\mathbf{T}_{\text{augm}}(\mathbf{x})]$ will be replaced by the moving image, and $I[(\mathbf{T}_{\text{augm}} \circ \mathbf{T}_{\text{learned}})(\mathbf{x})]$ will be replaced by the fixed image.

### 2.6 Training the network

The network is trained by minimizing the squared error between the true deformation field $\mathbf{u}_{\text{learned}}(\mathbf{x}) = \mathbf{T}_{\text{learned}}(\mathbf{x}) - \mathbf{x}$ and the network’s estimate $\hat{\mathbf{u}}_{\text{learned}}$, within the lung mask $M(\mathbf{x}) \in [0, 1]$. The loss function
is defined as

\[ L = \sum_{x \in \Omega_F} M(x) \left| u_{\text{learned}}(x) - \hat{u}_{\text{learned}}(x) \right|_2^2 \sum_{x \in \Omega_F} M(x) \]

This loss function is minimized using stochastic gradient descent with a momentum of 0.5 and an annealed learning rate

\[ \eta(t) = \frac{\eta_0}{1 + \lambda t} \]

with \( \eta_0 = 0.1 \) and \( \lambda = 10^{-4} \). We use a batch size of one, and use batch normalization for all convolutional layers using moving averages of the mean and variances of the parameters over all iterations, as proposed by Ioffe et al.\(^\text{10}\) To show the effect of progressive training, we train the architecture once using the schedule in Figure 2, and once using the setting \( \alpha_0 = \alpha_1 = \alpha_2 = \alpha_3 = 0 \) and \( \alpha_4 = 1 \) throughout training, i.e. the ‘conventional’ U-net. Because the architecture is the same in both cases, we use the exact same weight initialization, training data, and learning rate. At test time we use this setting for \( \alpha \) for both versions of the network.

### 3. RESULTS

We registered the ten pairs of DIRLAB images using both the progressive and conventional network, and evaluated the resulting deformation fields by computing target registration errors (TRE) on manually annotated corresponding landmarks in the DIRLAB set. The TRE metric captures misalignment by measuring the \( L_2 \)-norm between landmarks in the moving domain and the transformed landmarks from the fixed domain, i.e.

\[ \text{TRE} = \left| \left| x_M - T(x_F) \right| \right| \]

Results per image pair are shown in Table 1, together with the boxplot in Figure 6 that shows the distributions of TRE values for both architectures. We also show correlation plots for the estimated displacements of the 3000 landmarks in the DIRLAB set in \( x \)-, \( y \)-, and \( z \)-direction in Figure 5. Both show that

![Conventional training](image)

![Progressive training](image)

Figure 5: Correlation plots showing the correlation between the estimated and true displacements of the DIRLAB landmarks for both network variants, shown for the \( x \)-, \( y \)-, and \( z \)-components of the displacement.
Table 1: Target registration errors (TRE) in millimeters for no registration, registration using the conventional network, and using the progressive network.

<table>
<thead>
<tr>
<th>Pair</th>
<th>Before registration</th>
<th>Conventional training</th>
<th>Progressive training</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.89 ± 2.78</td>
<td>2.05 ± 0.97</td>
<td>2.18 ± 1.05</td>
</tr>
<tr>
<td>2</td>
<td>4.34 ± 3.90</td>
<td>4.10 ± 2.01</td>
<td>2.06 ± 0.96</td>
</tr>
<tr>
<td>3</td>
<td>6.94 ± 4.05</td>
<td>2.96 ± 1.21</td>
<td>2.11 ± 1.04</td>
</tr>
<tr>
<td>4</td>
<td>9.83 ± 4.85</td>
<td>3.75 ± 2.28</td>
<td>3.13 ± 1.60</td>
</tr>
<tr>
<td>5</td>
<td>7.48 ± 5.50</td>
<td>4.00 ± 2.46</td>
<td>2.92 ± 1.70</td>
</tr>
<tr>
<td>6</td>
<td>10.89 ± 6.96</td>
<td>6.38 ± 6.05</td>
<td>4.20 ± 2.00</td>
</tr>
<tr>
<td>7</td>
<td>11.03 ± 7.42</td>
<td>5.00 ± 3.65</td>
<td>4.12 ± 2.97</td>
</tr>
<tr>
<td>8</td>
<td>14.99 ± 9.00</td>
<td>11.33 ± 5.01</td>
<td>9.43 ± 6.28</td>
</tr>
<tr>
<td>9</td>
<td>7.92 ± 3.97</td>
<td>4.50 ± 2.02</td>
<td>3.82 ± 1.69</td>
</tr>
<tr>
<td>10</td>
<td>7.30 ± 6.34</td>
<td>4.29 ± 2.61</td>
<td>2.87 ± 1.96</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>8.46 ± 6.58</strong></td>
<td><strong>4.84 ± 4.03</strong></td>
<td><strong>3.68 ± 3.32</strong></td>
</tr>
</tbody>
</table>

in general, the progressively trained network performs better. We used the Wilcoxon signed-rank test to test for pairwise equality between the TRE values for all landmarks in both architectures. This indicated that the TRE-values of the progressive architecture (median: 2.82 mm) and the TRE-values of the static architecture (median: 3.72 mm) differed significantly \((N = 3000, T = 16.2, r = 0.55, p < 0.001)\). The median TRE values per image for each architecture also significantly differ (3.68 mm versus 4.79 mm, \(N = 10, T = 0, r = 1.0, p = 0.005\)), as can also be concluded from Figure 6. On average, the estimation of the deformation vector fields took 0.90 ± 0.05 seconds on an Nvidia Titan XP GPU.

4. DISCUSSION

In this paper we propose a method for training a fully convolutional neural network that lets the network grow progressively during training. We show on publicly available pulmonary CT data sets that this approach results in significantly lower registration errors, and better correlation with the ground truth compared to the conventional training approach. We hypothesize that because the network can focus on lower resolution data at the start of training, it can quickly optimize a limited number of weights to model a large range of displacements on a small vector field. The smooth transitions between resolution levels then allow the network to estimate the same range of displacements in higher resolution vector fields. Once the final transition is complete, the architecture is no different from an ordinary U-net.

The resulting network can perform end-to-end estimation of full size deformation fields directly from two images. It can generalize from the training set to a separate set of data acquired on different hardware, at a different institute. By making elaborate use of data augmentation, we only require a few images to train on, without need for manually annotated ground truths. Furthermore, because the method requires only one forward pass through the network, it takes less than one second to estimate a full-resolution deformation field. Although the results are promising, they are not yet up to the standard of the state-of-the-art for lung registration, which result in average TREs between 1.36 ± 1.01 and 2.13 ± 1.82 mm on the same DIRLAB data set.\(^{11,12}\) To address this, future work will focus on improving the training set’s deformations to be more realistic for this particular application.

5. CONCLUSION

In this paper we propose a method for training a convolutional neural network that lets the network grow progressively during training. We show on publicly available pulmonary CT data sets that this approach results in lower registration errors, and better correlation with the ground truth compared to conventional training approaches. The resulting network can perform end-to-end estimation of full size deformation fields in less than a second, directly from two images.
REFERENCES


