Published in final edited form as:

Image Anal Mov Organ Breast Thorac Images (2018). 2018 September ; 11040: 284–294.
doi:10.1007/978-3-030-00946-5_28.

Diffeomorphic Lung Registration Using Deep CNNs and Reinforced Learning

Jorge Onieva Onieva, Berta Marti-Fuster, María Pedrero de la Puente, Raúl San José Estépar

Applied Chest Imaging Laboratory, Department of Radiology, Brigham and Women's Hospital, Harvard Medical School, Boston, MA, USA

Abstract

Image registration is a well-known problem in the field of medical imaging. In this paper, we focus on the registration of chest inspiratory and expiratory computed tomography (CT) scans from the same patient. Our method recovers the diffeomorphic elastic displacement vector field (DVF) by jointly regressing the direct and the inverse transformation. Our architecture is based on the RegNet network but we implement a reinforced learning strategy that can accommodate a large training dataset. Our results show that our method performs with a lower estimation error for the same number of epochs than the RegNet approach.

Keywords

Deep learning; Reinforced learning; Lung registration Chest computed tomography; Diffeomorphism

1 Introduction

In this paper we address the problem of lung registration, with the aim of overlaying two chest CT scans from the same patient obtained during inspiration and expiration breath cycles. Numerous works have been published addressing lung registration in CT scans. Some of these methods competed in the EMPIRE10 Challenge [1], which evaluated registration methods on thoracic CT. Song *et al.* [2] proposed different configurations for their ANTS open source software package [3] to build diffeomorphic transformation models to perform a non-rigid image transformation that achieved good results in the challenge. Modat *et al.* [4] also achieved good results using a reformatted version of the Rueckert Free-Form Deformation algorithm [5] using the NiftyReg package. In 2013, Rühaak *et al.* [6] proposed a method based on minimizing the normalized gradient fields distance measure with curvature regularization.

Deep learning has emerged in the last years as a powerful tool to solve different medical image problems [7–9], including lung registration. The use of strategies based on deep learning allows to register the images without the need of a dissimilarity metric, as the

algorithm is optimized using just images saliency features. In this work, we used the RegNet architecture proposed by Sokooti *et al.* [10] to directly estimate a DVF using multiple resolution patches from a pair of input images. We focus on recovering the elastic diffeomorphic transformation. This is especially challenging in lung registration due to the large field displacement that takes places between the breathing cycle extremes.

The contributions of this work were twofold. First, we propose a new loss function that jointly estimates the direct and inverse diffeomorphic transformation. To train the algorithm, we obtained both fields using ANTs and visually validating the results. Second, we sequentially selected the data training patches more adequate for training using a reinforced learning strategy [11]. The reinforced learning strategy aims at selecting the most adequate training points allowing for a scalable approach in large training datasets.

2 Methods

2.1 Computation of Training Deformation Fields

We used 10 patients from the COPDGene cohort [12] to train our algorithm with inspiratory and expiratory high-resolution CT scanning. We tested the algorithm in another 5 different scans.

We registered the inspiration-expiration scans using ANTs [3] with the inspiration scan as the fixed image and the expiration scan as the moving one. We performed an initial affine registration using mutual information as the cost function. Then, a diffeomorphic B-spline registration was performed based on the Lagrangian diffeomorphic registration technique described in [13]. The parameters of the registration were optimized using Spearmint [14], a Bayesian optimization approach, using a publicly available reference dataset with corresponding landmarks [15].

We focused on the registration of the lung area and ignored the rest of the image. To that end, we generated a lung mask using the segmentation method described in [16] as implemented in the Chest Imaging Platform. All the images were preprocessed to rescale them to an isometric spacing of $1 \times 1 \times 1$ mm. The DVFs and the lung masks reformatted to match the new resolution. The scans were visually assessed to ensure that the original registration and the lung masks were correct. We input into the CNN the inspiratory and the affined transformed expiratory images, as we were only interested in estimating the elastic part. Figure 1 shows an example of the Jacobian of the DFV for one of the used scans after applying our lung segmentation mask.

For all our experiments, we used the RegNet architecture proposed by Sokooti *et al.* [10] (Fig. 2). The original network takes as an input four 3D patches centered in a voxel, and it outputs the DVF for that voxel. The first two patches have a $29 \times 29 \times 29$ voxel size, and they are obtained at the original scan resolution (one of the patches belong to the fixed image, while the other belongs to the moving one). Analogously, we select another two patches (one from each image) with a $27 \times 27 \times 27$ voxel size, but these patches are obtained at the half resolution to capture a bigger context in the image. The main difference in our approach is that we performed additional experiments to obtain not only the direct DVF, but

also the inverse diffeomorphic DVF (using the same architecture, just adding a second output vector).

To create our training dataset, we randomly selected 20,000 points for each scan that were within the bounds of the lung mask. For each point (voxel), we obtained the four patches that compound the network input, and stored both the direct DVF and the inverse DVF from the elastic component of the original transformation (network output). We used a total of 200,000 points to train/validate the algorithm. We followed a similar approach to create a test dataset with the 5 test scans. We randomly selected 5,000 voxels for each scan in bounds with the corresponding lung mask, for a total of 25,000 test data points.

2.2 Reinforced-Sequential Training

We evaluated different sequential training strategies as an alternative to a traditional learning approach, based on the concept of reinforced learning in machine learning [11]. Instead of training using all the data points in the training dataset, we split them into batches of $n = 5,000$ that were trained independently. We also reserved a fixed number of 3,000 points for validation in each epoch, which were common to all the sequences.

We used an identical RegNet architecture, loss function and hyper parameters in all the sequences. The L2 loss error function is defined as:

$$L = \frac{1}{n} \sum_{i=1}^n (v_i - f(v_i))^2 \quad (1)$$

where v_i is a ground truth vector of 3 coordinates containing the DVF for the voxel i , and $f(v_i)$ is the output of the CNN that contains the corresponding DVF prediction.

Every sequence begins with an initial learning rate of 0.0001, which is reduced by a factor of 0.8 whenever two consecutive epochs do not improve the validation loss. The training for a sequence is interrupted after 3 consecutive epochs with no improvement in the loss function for the validation data. A batch size of 40 and the default Keras library (v2.1.5) implementation of RMSprop optimizer are used to minimize the loss function L .

After training each sequence, we use the current state of the model to evaluate all the data points that were used in it, and select the best b and the worst w ones based on the loss function value. p_b and p_w are two hyperparameters that control the percentage of points used with respect to the total number of points n used in the sequence training. After each sequence we select the $b = n * p_b$ best points and the $w = n * p_w$ worst points of the current training (where $p, b \in [0, 1]$), and we keep them to be reused in the next sequence. By doing so, the method can reuse the data points that are thought to contain the most useful information for the current model. Therefore, $p = n - b - w$ new data points are added to the training dataset at the end of each sequence. The full process is represented in Fig. 3.

Since each sequence is trained independently and sequentially, there is a need to define a strategy to initialize the model weights at the beginning of each one of them. We tested three different strategies: continue with the last model state (like a regular training), use the best

model that was found during the previous sequence, and use the best model that was found globally in all the previous sequences.

2.3 Use of Direct-Inverse DVFs for Training

We also tested the impact of training our algorithm using the diffeomorphic direct DVF (that contains the displacement for every voxel in the moving image to the closest one in the fixed image), the diffeomorphic inverse DVF (to go from the fixed image to the moving one) or both of them simultaneously. When using both DVFs, the value of the loss function L is the sum of the L2 error for the direct DVF and in the inverse DVF. Formally:

$$L = L_D + L_I \quad (2)$$

where L_D is the L2 loss function defined in Eq. 1 for the direct DVF and L_I is the L2 loss function for the inverse DVF. We compared the values of each one of the loss functions when using the different DVFs for the training.

3 Results

3.1 Evaluation of Reinforced-Sequential Training Strategies

We analyzed the performance of 5 different reinforced-sequential learning strategies, as described in the Sect. 2.2. Figure 4 shows the results of the validation loss value obtained during the first 3 *Epochs*. Note that in order to keep a consistent nomenclature and compare the results with a traditional learning approach, we define an *Epoch* like the moment where all the points in the training dataset have been used. It is important not to confuse it with each one of the regular epochs that happen during the training of each one of the sequences. Each reinforced learning *Epoch* is composed of around 60 sequential training steps, depending on the values of the hyperparameters p_b and p_w that are used to determine the number of data points to be reused in each sequence.

The figure shows that the variation of the hyperparameters p_b and p_w do not seem to have a big impact in the overall result, at least in the ranges that we tested ([0.05–0.2]). However, note how the strategy used to initialize the weights in each sequence can impact dramatically in the performance of the algorithm. Given the same values for p_b and p_w ($p_b = 0.2$ and $p_w = 0.05$), using the best model found for all the past sequences performed much worse than the other two strategies. A possible explanation for this behavior is that using this strategy may break the continuity in the learning process, leading to poor performance of the optimizers used during the training. We can also see that indeed the continuous model strategy seems to perform slightly better than the best model found in the previous sequence, although the differences are smaller. This may happen because, since the size of the training data used in each sequence is small compared to the overall training dataset size, and we also used pretty aggressive early stopping conditions, the best model found in a sequence should be very similar to the one found when the sequence stopped. Therefore, the effect of the break in continuity should be limited. In any case, we can conclude that using a continuous training strategy is the best candidate for the tested reinforced-sequential training approach.

The best strategy found was a continuous learning with $p_b = 0.2$ and $p_w = 0.05$, which reached an L2 validation error of 0.77647 after 34 *Epochs*.

3.2 Comparison of Reinforced-Sequential and Traditional Learning Strategies

We compared the learning process of one of the reinforced-sequential training (Continuous learning with $p_b = 0.2$ and $p_w = 0.05$) to traditional learning, using the same train and validation datasets. The results are shown in Fig. 5. The figure shows the L2 error in the validation dataset in each *Epoch*. As it was described in the previous section, we are defining an *Epoch* as the moment when all the training data have been used for training purposes once.

We can see how after very few *Epochs* (around 4–5), the loss error in the reinforced-sequential learning is already quite close to the best-achieved result, especially when compared to a regular training. This indicates that the use of the proposed reinforced-sequential strategy allows having a pretty good estimation of the algorithm performance with very few iterations over the training dataset. This may be particularly useful in a context where the data generation process is difficult but virtually unlimited.

Besides, the overall error after 34 *Epochs* in the best reinforced-sequential strategy was 0.77647, which is lower than a traditional learning after 121 epochs (0.846). These results suggest that our reinforced-sequential learning strategy may be used in different problems to increase the efficiency of other deep CNN algorithms.

Both algorithms were trained in the same hardware (using a Nvidia GEFORCE GTX 1080 Ti GPU). The total training time for the traditional training over 121 epochs was 2 days, 18:47:05 s, while the compared reinforced learning over 34 *Epochs* took 5 days, 1:13:35 s. The higher training time in the reinforced learning is due to the bigger number of iterations for every sequence as well as the extra time needed to evaluate the best/worst training data points after each sequence. The training time in reinforced learning could be reduced by selecting a higher number of training data points for each sequence.

3.3 Direct DVF and Inverse DVF in the Loss Function

We finally compared the performance of three regular trainings using different DVFs in the training loss function, evaluating the trained models in the test dataset. For evaluation purposes, we report the error distribution using the norm-2 Euclidean distance (in a mm scale) for the different DVFs used.

In the first training, we used the direct DVF to train the model and to evaluate the distance D in the test dataset. We did a second analog training but using the inverse DVF for training/testing. Finally, we trained a third model using an L2 loss function whose total value is the sum of the individual L2 losses for the direct and the inverse DVF respectively (as described in the Methods section).

Figure 6 shows that the validation loss during the training of the direct DVF and the inverse DVF are very similar. However, when we look at the same metric in a training using a dual DVF loss, we can appreciate how the error is higher in the first epochs, but it ends up

converging to a similar validation error in advanced phases of the training process. This indicates that the complexity of the problem increased (as we are predicting 2 fields instead of one), but after some epochs the network is able to perform predictions at a similar level than the single DVF training.

Moreover, as we can see in Fig. 7, no significant differences in the test error were detected when using both DVFs in the loss function. Therefore, we can conclude that we are able to learn both the direct and the inverse DVFs simultaneously without the need for any adaptations in the network architecture or the training hyperparameters (learning rate, optimizer parameters, etc.).

4 Discussion

We proved the feasibility of using different training strategies to improve the accuracy of an algorithm based on deep CNNs, using the concepts of reinforced learning that have been applied to other tasks in the machine learning field. In the future, we will evaluate other strategies that can work in a more efficient way than the ones proposed since they could increase the total training time.

We evaluated the performance of our algorithm to study its convergence properties as well as the error when estimating just the direct and inverse DVF separately and jointly. Our results showed a lower error bound when the reinforced learning strategy was applied. We also showed that our diffeomorphic method can estimate both the direct and inverse DVF with an error that is similar to the one that is obtained when only estimating the direct or inverse DVF.

In the future, we will extend these strategies to bigger training datasets that can increase the generalization of the problem and reduce the total registration error.

Acknowledgments

This work has been funded by NIH NHLBI grants R01-HL116931 and R21HL140422. The Titan Xp used for this research was donated by the NVIDIA Corporation.

References

1. Murphy K, et al.: Evaluation of registration methods on thoracic CT: the EMPIRE10 challenge. *IEEE Trans. Med. Imaging* 30(11), 1901–1920 (2011) [PubMed: 21632295]
2. Song G, Tustison NJ, Avants BB, Gee JC: Lung CT image registration using diffeomorphic transformation models. In: *Medical Image Analysis for the Clinic: A Grand Challenge*, pp. 23–32 (2010)
3. Avants BB, Tustison N, Song G: Advanced Normalization Tools (ANTs). *Insight J* 2, 1–35 (2009)
4. Modat M, McClelland J, Ourselin S: Lung registration using the NiftyReg package. In: *MICCAI2010 Workshop: Medical Image Analysis for the Clinic - A Grand Challenge*, pp. 33–42 (2010)
5. Rueckert D, Sonoda LI, Hayes C, Hill DLG, Leach MO, Hawkes DJ: Nonrigid registration using free-form deformations: application to breast MR images. *IEEE Trans. Med. Imaging* 18(8), 712–721 (1999) [PubMed: 10534053]
6. Rühaak J, Heldmann S, Kipshagen T, Fischer B: Highly accurate fast lung CT registration. In: *SPIE Medical Imaging 2013: Image Processing*, vol. 8669, pp. 86690Y-1–86690Y-9 (2013)

7. Litjens G, et al.: A survey on deep learning in medical image analysis. *Med. Image Anal* 42, 60–88 (2017) [PubMed: 28778026]
8. Miao S, Wang ZJ, Liao R: A CNN regression approach for real-time 2D/3D registration. *IEEE Trans. Med. Imaging* 35(5), 1352–1363 (2016) [PubMed: 26829785]
9. Eppenhof KAJ, Pluim JPW: Supervised local error estimation for nonlinear image registration using convolutional neural networks. In: *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, vol. 10133, February 2017
10. Sokooti H, de Vos B, Berendsen F, Lelieveldt BPF, Išgum I, Staring M: Nonrigid image registration using multi-scale 3D convolutional neural networks In: Descoteaux M, Maier-Hein L, Franz A, Jannin P, Collins DL, Duchesne S (eds.) *MICCAI 2017. LNCS*, vol. 10433, pp. 232–239. Springer, Cham (2017). 10.1007/978-3-319-66182-7_27
11. Kaelbling LP, Littman ML, Moore AW: Reinforcement learning: a survey. *J. Artif. Intell. Res* 4, 237–285 (1996)
12. Regan E.a., et al.: Genetic epidemiology of COPD (COPDgene) study design. *COPD* 7(1), 32–43 (2010) [PubMed: 20214461]
13. Avants BB, Epstein CL, Grossman M, Gee JC: Symmetric diffeomorphic image registration with cross-correlation: evaluating automated labeling of elderly and neurodegenerative brain. *Med. Image Anal* 12(1), 26–41 (2008) [PubMed: 17659998]
14. Snoek J, Larochelle H, Adams RP: Practical Bayesian optimization of machine learning algorithms. *Adv. Neural Inf. Process. Syst* 25, 2951–2959 (2012)
15. Castillo R, et al.: A reference dataset for deformable image registration spatial accuracy evaluation using the COPDgene study archive. *Phys. Med. Biol* 58(9), 2861–2877 (2013) [PubMed: 23571679]
16. Ross JC, et al.: Lung extraction, lobe segmentation and hierarchical region assessment for quantitative analysis on high resolution computed tomography images In: Yang G-Z, Hawkes D, Rueckert D, Noble A, Taylor C (eds.) *MICCAI 2009. LNCS*, vol. 5762, pp. 690–698. Springer, Heidelberg (2009). 10.1007/978-3-642-04271-3_84

Fig. 1.
Example of the DVF Jacobian for a test patient

Fig. 2.

Inspiratory-Expiratory registration regression network based on the RegNet architecture [10]

Fig. 3.
Schema for the proposed reinforced learning workflow for one *Epoch*

Fig. 4.
Evaluation of reinforced-sequential training strategies

Fig. 5.
Comparison of reinforced-sequential and traditional learning strategies

Fig. 6.
Validation loss in full training

Fig. 7.
Test error comparison when using different DVFs in the training loss function