PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

AUTOMATIC PLANE REFORMATTING FOR 4D FLOW MRI USING CONTINUOUS REINFORCEMENT LEARNING

JAVIER E BISBAL

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
SERGIO URIBE

Santiago de Chile, April 2022

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

AUTOMATIC PLANE REFORMATTING FOR 4D FLOW MRI USING CONTINUOUS REINFORCEMENT LEARNING

JAVIER E BISBAL

Members of the Committee:
SERGIO URIBE
PABLO IRARRAZAVAL
JULIO SOTELO
CRISTIÁN ESCAURIAZA

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, April 2022

Gratefully to my family and loved ones

ACKNOWLEDGEMENTS

This work was funded by Agencia Nacional de Investigación y Desarrollo / Subdirección de Capital Humano / Beca Magíster Nacional/ 2021 – 22211103

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

4D flow MRI allows the calculation of hemodynamic parameters that provide valuable information to characterize cardiovascular diseases. One limitation is the time-consuming and user-dependent post-processing. We developed an automated reinforcement deep learning framework for plane reformatting of 4D flow data using the Asynchronous Advantage Actor Critic algorithm to train a 2D convolutional network that sequentially updates plane parameters towards a target plane based on a continuous policy. We processed 4D flow data from GE, Siemens and Philips MRI scans of 67 healthy volunteers and 20 patients with congenital heart defects (47 men, 34 ± 12.4 years of age). All datasets were split in 50% training, 25% validation and 25% testing and validated with 4 fold cross validation. Our method achieved excellent results in terms of angulation and distance errors (average 7.88 ± 4.33 degrees and 3.46 ± 3.25 mm) with a flow correlation of 0.82. We successfully adapted a continuous reinforcement learning method to plane reformatting in 4D Flow suited for data from different scanner vendors, with promising results in healthy volunteers and patients. Future work with public data and more planes is needed for clinical validation.

Keywords: 4D flow MRI, Deep Learning, **(Reinforcement Learning, Plane reformatting)**.

RESUMEN

La resonancia magnética de flujo 4D permite calcular parámetros hemodinámicos que proporcionan información valiosa para caracterizar las enfermedades cardiovasculares. Una de las limitaciones es el post procesamiento, el cual requiere mucho tiempo y depende del usuario. En esta investigación desarrollamos un método de aprendizaje profundo reforzado para el reformateo automático de planos en datos de flujo 4D utilizando el algoritmo *Asynchronous Advantage Actor Critic* para entrenar una red convolucional 2D que actualiza secuencialmente los parámetros del plano hacia un plano objetivo una política de acciones continua. Procesamos datos de flujo 4D de escáneres de resonancia magnética de GE, Siemens y Philips de 67 voluntarios sanos y 20 pacientes con defectos cardíacos congénitos (47 hombres, 34 ± 12,4 años de edad). Todos los conjuntos de datos se dividieron en 50% para entrenamiento, 25% para validación y 25% para prueba, y se confirmaron con validación cruzada de 4 carpetas. Nuestro método obtuvo excelentes resultados en términos de errores de angulación y distancia (media de 7,88 ± 4,33 grados y 3,46 ± 3,25 mm) con una correlación de flujo de 0,82. Adaptamos con éxito un método de aprendizaje de refuerzo continuo al reformateo de planos de flujo 4D, apto para datos de diferentes fabricantes de escáneres de resonancia magnética, con resultados prometedores en voluntarios sanos y pacientes. Trabajo futuro con más datos y planos es necesario para la validación clínica.

Palabras Claves: Flujo 4D, Aprendizaje profundo, (**Aprendizaje reforzado, Reformateo de planos**).

1. INTRODUCTION

4D flow MRI allows the quantification of advanced hemodynamic parameters that provide valuable information to characterize cardiovascular diseases (Markl, Frydrychowicz, Kozerke, Hope, & Wieben, 2012). The volumetric coverage of 4D Flow MRI offers retrospective positioning of planes for flow measurements at any location within the acquired data volume. However, this task is time consuming and user-dependent, especially when analyzing multiple planes. The 4D flow cardiovascular magnetic resonance consensus statement (Dyverfeldt et al., 2015) declares: "Efforts toward more standardized automated flow visualization approaches could be helpful in certain applications and would minimize operator-dependent variation, although users should understand the principles, strengths and limitations of different techniques".

Although several semi-automatic methods have been proposed to asses different quantitative parameter, only a few have been proposed to retrospectively reformat the 4D flow data and quantify flux through the vessels (Bustamante et al., 2015; Corrado, Seiter, & Wieben, 2021). Bustamante et al. (2015) proposed an automatic plane reformatting algorithm based on the registration of a vessel-specific atlas. In this method, an atlas is created from a Phase Contrast Magnetic Resonance Angiography (PC-MRA) (Markl, Kilner, & Ebbers, 2011) of a healthy volunteer , with 2D planes positioned at multiple vessel locations. In order to align vessels and planes, the 3D PC-MRA input dataset is registered to the atlas. Despite the good results in terms of flow correlation between the automatic and semi-automatic method (with user-defined planes), atlas registration requires substantial computation time and depends on the similarity between the image and the atlas.

Deep Learning has shown promising results with high reproducibility and low variability for biomedical image processing tasks with different frameworks and architectures (Marrone, Olivieri, Piantadosi, & Sansone, 2019). Corrado et al. (2021) recently proposed a deep learning-based method for plane reformatting in 4D flow. This algorithm samples 3D patches from the input 4D flow data and uses a 3D convolutional neural network (CNN) to predict the patch most likely to contain the reformatted planes, along with the plane reformatting parameters (plane

center and normal direction). Regardless of the promising results, the 3D CNN only computes the plane parameters of the plane reformatting from the 3D patches and not the plane reformattting itself. This implies that there is information from the plane reformatting in 2D that is not being used to solve the plane reformatting task. In addition, Corrado et al. only used 4D flow scans from one scanner vendor (GE Healthcare).

Blansit et al. (2019) proposed a deep learning method for plane reformatting in cardiac cine images that uses three short-axis localization models. The first model locates the mitral valve slice using a 2.5 dimensional VGG-19/LSTM (Simonyan & Zisserman, 2014; Hochreiter & Schmidhuber, 1997) ensemble network. The second model locates a bounding box around the heart with a 2.5 dimensional U-net (Ronneberger, Fischer, & Brox, 2015). The final model uses a U-net modified for heatmap regression (Payer, Štern, Bischof, & Urschler, 2016) to localize for 4 landmarks on the mitral valve slice for plane reformatting. This method could be used for plane reformatting in 4D flow, however, not all vessels have specific landmarks in one particular plane (mitral valve slice in the case of short-axis cardiac cine images) to compute the plane reformatting. In 4D flow, multiple planes are needed within the volume to locate reference points that allow the computation of the plane reformatting in each vessel. This adds extra difficulty to perform 4D flow plane reformatting using landmarks in specific planes.

Another approach to solve the plane reformatting task is using Reinforcement Learning (RL). In RL an agent learns how to make comprehensive decisions by mimicking navigation processes of an expert solving the plane reformattting task. RL allows agents to learn complex tasks that may need several steps to find a solution (Sutton & Barto, 2018). For plane reformatting we start with an initial plane in a specific position within the volume data and the agent sequentially updates the plane parameters using rotations and translations towards a target position. Alansary et al. (2018) developed a RL agent using a Deep Q-Network (DQN) (Mnih et al., 2015) to solve the plane reformatting task in brain and cardiac cine MRI images. This method achieved good performance and more comprehensive results because the RL agent performs actions similar to those an expert would use for the plane reformatting.

In this work, we adapted the RL framework (Alansary et al., 2018) to be used in 4D flow images from different scanner vendors. A limitation of using DQN as a RL agent is that only one action (rotation or translation) can be performed at the time of updating the parameters of each plane and must have a discrete size. In order to have more degrees of freedom over rotations and translations, our method trains a RL agent using the Asynchronous Advantage Actor Critic (A3C) (Mnih et al., 2016) algorithm to estimate a continuous action policy for rotations and translations. With this policy the agent rotates and moves three ortogonal planes within the 4D flow data towards a target vessel location.

2. METHODS

2.1. Data sets and image processing

We processed a total of 87 scans from different scanners including healthy volunteers and patients with congenital heart defects. The details of each dataset are in Table 2.1:

We preprocessed the data with an in-house developed MATLAB toolbox (MathWorks, Natick, MA, USA), and it involves 6 steps: isotropic interpolation, computation of angiography volume, segmentation, field of view (FOV) reduction, contrast enhancement and registration.

(i) Isotropic interpolation: linear interpolation to transform all datasets to a resolution of $2x2x2\ mm^3$.

(ii) Angiography volume: computation of Phase Contrast Magnetic Resonance Angiography (PC-MRA) (Markl et al., 2011) images for Philips and Siemens scanners (43 scans) and complex difference (Bernstein & Ikezaki, 1991) images from data acquired in GE scanner (44 scans). Complex difference and PC-MRA are two types of angiography images in MRI. From this point on we will refer to PC-MRA and complex difference images as angiography images.

(iii) Segmentation: segmentation on angiography images with a manually adjusted threshold for each image. From the segmentation we keep the largest connected element that corresponds to the whole heart and great vessels.

(iv) FOV reduction: reduction of field of view (FOV) using the smallest bounding box around vessels segmentation to confine the whole heart and great vessels. In some cases the segmentation was connected to areas that did not correspond to the heart and great vessels. This occurred in cases of data with higher noise (about 8 to 12 scans from GE scanner). To solve this issue we manually reduced the FOV to confine the heart and great vessels.

(v) Contrast enhancement: Contrast limited adaptive histogram equalization (CLAHE) (Reza, 2004) in angiography images. This method improves local contrast and enhance the definition on edges of the heart and great vessels.

(vi) Registration: Rigid registration to align all angiography data in the same orientation. This registration maximizes the mattes mutual information (Pluim, Maintz, & Viergever, 2003) between a fixed angiography volume and a moving angiography volume using a evolutionary optimization strategy (Styner, Brechbuhler, Szckely, & Gerig, 2000) using rotations and translations.

Table 2.1. 4D flow data description

MRI scanner	**Description**
GE 1.5T	32 scans from healthy volunteers
GE 1.5T	8 scans from patients with aortic coarctation
GE 1.5T	4 scans from patients with tetralogy of fallot
Siemens 3T	17 scans from healthy volunteers
Siemens 3T	8 scans from patients with bicuspid aortic valve (BAV)
Philips 3T	18 scans from healthy volunteers

2.2. Reinforcement learning (RL) framework

RL is a type of machine learning technique which an agent learns to behave in an environment by performing actions that maximize a reward signal. The agent is not told which actions to take, but must discover, by trying different actions, which ones produce the greatest reward. (Sutton & Barto, 2018). For the task of plane reformatting, an agent sequentially modifies the position and direction of a plane in state s. If the next state plane is closer to the target plane, a positive reward signal is granted, otherwise the agent gets a negative reward signal. Since the goal is to maximize the total reward, the agent will learn to navigate to the target plane within the 3D volume. At each time step t the agent gets a new state s and chooses an action following a policy

π. The agent attempts to learn a policy that maximizes both immediate and subsequent future rewards.

Deep reinforcement learning (DRL) combines Deep Learning with RL, where deep learning allows agents to make decisiones from unstructured data without knowing all posible states and transitions (Torres, 2021). For the task of plane reformatting an infinite number of planes can be computed from a single volume. Even if we discretize the problem, a large number of possible states and transitions makes it complex to compute a function that allows us to map all the states to the actions that will allow us to reach the target plane. With the Asynchronous Advantage Actor Critic (A3C) algorithm (Mnih et al., 2016) we can estimate this function using a Convolutional Neural Network (CNN) without knowing all possible states and transitions.

Figure 2.1 shows the proposed DRL framework. Number (1) denotes the state s, which consists of three ortogonal planes where the desired plane is the first plane. At each time step a grid of size $70x70x3$ from each state is sampled using bilinear interpolation from the angiography volume. $70x70$ denotes the width and height of the sampled images and 3 the number of ortogonal planes. The parameters of the desired plane in cartesian coordinates are,

$$\cos(\alpha)x + \cos(\beta)y + \cos(\phi)z + d = 0 \tag{2.1}$$

where $\vec{n} = (\cos(\alpha), \cos(\beta), \cos(\phi))$ is the normal vector of the plane with α, β and ϕ the angles between the plane and the cartesian axes x, y and z, respectively. d is the distance from the plane to the volume center. The action space is defined as $\{a_\alpha, a_\beta, a_\phi, a_d\}$, where $\{a_\alpha, a_\beta, a_\phi\} \in (-\omega_{min}, \omega_{max})$ degrees and$\{a_d\} \in (-d_{min}, d_{max})$ mm. $(-\omega_{min}, \omega_{max})$ and $(-d_{min}, d_{max})$ are the lower and upper bounds for the action space. Each action updates the state plane parameters as $\alpha_t = \alpha_{t-1} + a_\alpha, \beta_t = \beta_{t-1} + a_\beta, \phi_t = \phi_{t-1} + a_\phi, d_t = d_{t-1} + a_d$. The reward of each transition is

$$\begin{aligned} r_t = & \left(D\left(P_{t-1}, P_T\right) - D\left(P_t, P_T\right)\right) + \\ & \lambda\left(NMI\left(P_t, P_T\right) - NMI\left(P_{t-1}, P_T\right)\right) \end{aligned} \tag{2.2}$$

where P_t represents the plane parameters in step t and P_T the target plane, D is the Euclidean distance between the parameters of the two planes, NMI is the normalized mutual information (Studholme, Hill, & Hawkes, 1999) between the plane in step t and the target plane and λ is a scalar weight to balance each term of the reward.

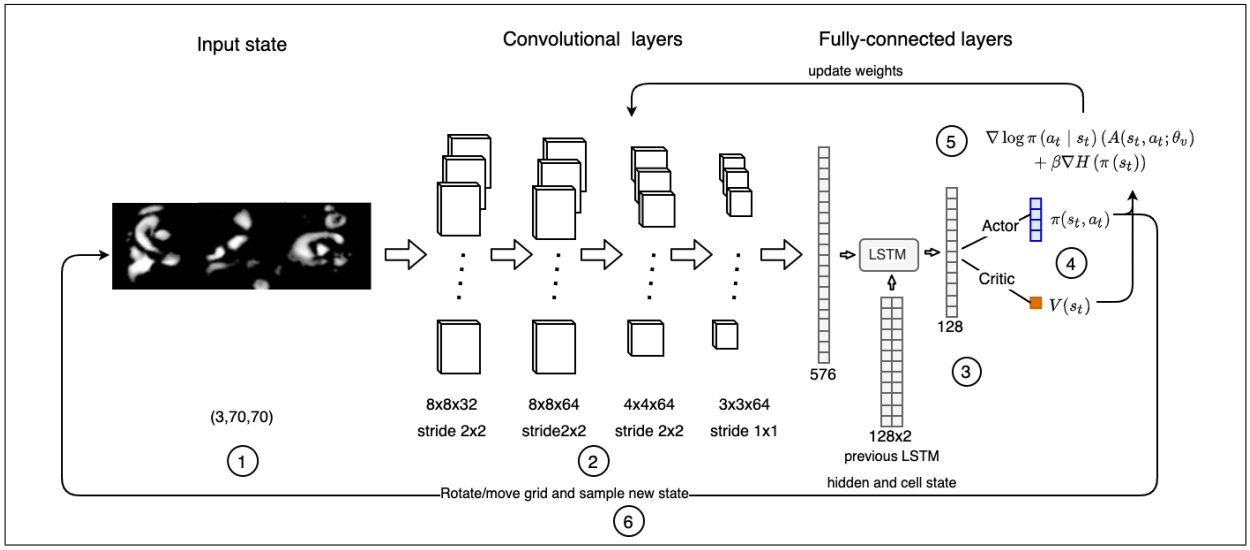

Figure 2.1. RL framework for 4D flow plane search. An initial stack of images (1) pass through 4 convolutional layers (2) and 1 fully-connected layer. The output of this layer enters a LSTM cell with the previous LSTM hidden and cell state (3). The LSTM calculates the policy ($\pi(a_t, s_t)$) and value function ($V(s_t)$) (4). (5) CNN parameters are updated with equation 2.4. Finally, the policy rotates and moves the grid to reach a new state (6).

The objetive is to maximize the expected return defined as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where $\gamma = 0.99$ is a discount factor that allow the summation to converge within a finite number of steps. We also defined a value function $V^{\pi}(s) = E[R_t \mid s_t = s]$ following an action policy $\pi(a_t \mid s_t)$. If we maximize $V^{\pi}(s)$, we also maximize the expected return. One way to estimate $V^{\pi}(s)$ is using a parametrization of the policy $\pi(a_t \mid s_t; \theta)$ and an estimation of the value function $V^{\pi}(s; \theta_v)$, with parameters θ and θ_v. The A3C algorithm (Mnih et al., 2016) estimates the parameters θ and θ_v with a neural network. We trained the network in order to find the optimal policy that maximizes the estimation of $V^{\pi}(s; \theta_v)$.

The network consists of four 2D convolutional layers (number 2 in Figure 2.1), a Long short-term memory (LSTM) cell, and two fully-connected layers (number 3 in Figure 2.1). The LSTM cell connects the previous and present LSTM hidden and cell states to exploit information between time steps. The last fully-connected layer estimates the policy $\pi(s_t, a_t)$ and state value $V(s_t)$ (number 4 on Figure 2.1). The policy is parameterized as a normal distribution

$$\pi(a \mid s, \theta) \doteq \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} \exp\left(-\frac{(a-\mu(s,\theta))^2}{2\sigma(s,\theta)^2}\right). \tag{2.3}$$

Here $\sigma(s, \theta)$ and $\mu(s, \theta)$ are the parameters estimated by the last layer of the neural network. Each parameter of the network is updated every t_{max} actions or in a terminal state. A terminal state is reached when the following two conditions are fulfilled in ten consecutive steps. First, if the agent reaches a state with a distance from the target plane lower than 3 mm and less than 4 degrees of angle error. In this case the agent a bonus of 1 to the reward. Second, if the agent continues to approach the target plane. The following equation updates the network parameters (number 5 in Figure 2.1):

$$\nabla_{\theta'} \log \pi\left(a_t \mid s_t; \theta'\right)\left(A(s_t, a_t; \theta_v) + \eta \nabla_{\theta'} H\left(\pi\left(s_t; \theta'\right)\right)\right. . \tag{2.4}$$

$\nabla_{\theta'} \log \pi\left(a_t \mid s_t; \theta'\right)$ denotes the direction of largest gradient of the probability of action a_t in the state s_t, which is weighted by $A(s_t, a_t; \theta_v)$, an estimate of the advantage function defined as

$$A(s_t, a_t; \theta_v) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V\left(s_{t+k}; \theta_v\right) - V\left(s_t; \theta_v\right), \tag{2.5}$$

where k is upper-bounded by t_{max}, in this implementation set to 8 steps. The gradient of an entropy estimate of the action policy denoted by $\eta \nabla_{\theta'} H\left(\pi\left(s_t; \theta'\right)\right)$ is added to improve exploration, preventing premature convergence to suboptimal policies. η adjusts the magnitude of the entropy regularization term. After the network parameters are updated, the policy distribution is

sampled to obtain the actions that rotate and move the grid and sample the next state from the 3D angiography data (number 6 n Figure 2.1).

2.3. Experimental setup

2.3.1. Data division and validation

To train our network we split each dataset in Table 2.1 in 50% training, 25% validation and 25% testing. To validate our results we performed a 4 folder cross validation. This process consists of 4 iterations, where in each iteration a different training, validation and test set is chosen, until each of the data is used for both training and testing. This allows us to evaluate the stability of the network by training with different scans within the datasets.

2.3.2. Manual annotations

We used the software Paraview (Kitware, Clifton Park, NY12065, USA) to generate 3D contours of the segmentation of the vessels and the velocity streamlines within the segmentation. Then, two observers with training in MR images visualization placed 1 perpendicular plane to the wall in 4 vessels: pulmonary artery (PA), right pulmonary artery (RPA), left pulmonary artery (LPA) and ascending aorta (AAsc). Each plane was located placing a point on the middle of the vessel and defining a normal vector. The 3D point coordinates were normalized between 0 and 1.

From these planes, we use inter-observer variability as a reference for later comparison with our algorithm.

2.3.3. Flow measurement

To measure hemodynamic parameters such as flow from the reformatted planes, we need a segmentation of the vessel to which the plane is perpendicular. To compute this segmentation we trained 2D U-net (Ronneberger et al., 2015) with the manually placed planes of each vessel. We

trained 4 networks (one for each vessel) for 40 epochs in Google Colab GPU (Nvidia K80 / T4) with Keras library (Chollet et al., 2015). We also applied affine transformations to the training dataset, including shear, rotation and scaling, to prevent overfitting. To measure the performance of the 2D U-Net segmentation we compute intersection over union (IoU) and Dice coefficient (equations 2.6 and 2.7) between U-net segmentation ($\hat{x}$) and manual segmentation (x) of vessels on the reformatted planes

$$Dice = \frac{2|x \cap \hat{x}|}{|x| + |\hat{x}|} \tag{2.6}$$

$$IoU = \frac{|x \cap \hat{x}|}{|x \cup \hat{x}|} \tag{2.7}$$

For statistical analyses of flow, we used Shapiro-Wilk (Shapiro & Wilk, 1965) test to determine the normality of flow data, and non-parametric Mann-Whitney test (Mann & Whitney, 1947) to measure the significance of the difference in flow between observers and between observer and algorithm.

2.3.4. Asynchronous training

The A3C algorithm has the advantage of training multiple agents asynchronuosly and updating the parameters of a shared model. For our model we trained 15 different agents, each one with a different training volume. This allows the network to learn from transitions across multiple volumes, decreasing training time and data overfitting. Data augmentation and dropout layers (Srivastava, Hinton, Krizhevsky, & Salakhutdinov, 2014) was used to prevent additional overfitting. Data augmentation consisted on random scaling, translations and rotations between (0.9,1.1), (-5.5) mm and (-5.5) degrees, respectively. These transformations allows the network to have invariance to small location and scaling variations, and they were implemented with TorchIO python library (Pérez-García, Sparks, & Ourselin, 2021). The dropout layers were used in the convolutional layers with probability 0.4.

Table 2.2 contains the chosen RL and networks hyperparameters for training and evaluation. We chose small action spaces for rotations and translations to make the transition between states slower, but more precise. For λ, a value was chosen so that the error range of the plane parameters and the error range of NMI in equation (2.2) were similar. The optimizer, t_{max} and the learning rate were chosen empirically to converge to a good solution in a short training time.

The network was trained using PyTorch library (Paszke et al., 2019) with A3C implementation code taken from (Kostrikov, 2018). Training was done on a Linux server with a NVIDIA Quadro RTX 8000 GPU for 150 epochs per plane. An epoch in this case consists of one agent training with all the volumes of the training set.

Table 2.2. RL and network hyperparameters

Hyperparameter	**value**
$(-d_{min}, d_{max})$	$(-3, 3)$ mm
$(-\omega_{min}, \omega_{max})$	$(-3, 3)$ degrees
λ	3
β	0.01
t_{max}	8
Optimizer	Adam
Learning rate	0.00001

2.3.5. Plane reformatting evaluation

For validation and test data, the network is evaluated with a sequence of 100 steps, being the final plane directly selected from the last step. We measured three metrics on the estimated planes.

(i) Distance error: defined as the Euclidean distance between a point in the ground truth plane in the middle of the vessel and the nearest point on the estimated plane.

(ii) Angle error: angle between estimated and ground truth normal vectors.

(iii) Normalized mutual information (NMI): Same NMI used in equation (2.2) to measure the correlation between ground truth plane and estimated plane. The range of NMI goes from 1 (no correlation between images) to 2 (perfect correlation between images).

To choose the model for test data evaluation, we chose the network with better performance in validation data. The best performance was calculated with the lower average distance between the parameters of the ground truth (manually placed) planes and the estimated planes.

3. RESULTS

3.1. Plane reformatting performance

Figure 3.1 shows two representative test samples of a healthy volunteer and a patient with BAV. The network was trained with observer 1 manually placed planes. We observed that the difference between the estimated planes and their respective labels is similar to the difference between observers in terms of distance, but slightly higher in terms of angle error.

Figure 3.2 shows the 4 folder cross validation performance metrics on each plane. Cross validation iterations revealed similar performance in terms of mean and interquartil range. Also, the size of interquatil range of the different folders has similar values to inter-observer differences. This suggests stability of the proposed method among all data samples compared to human error.

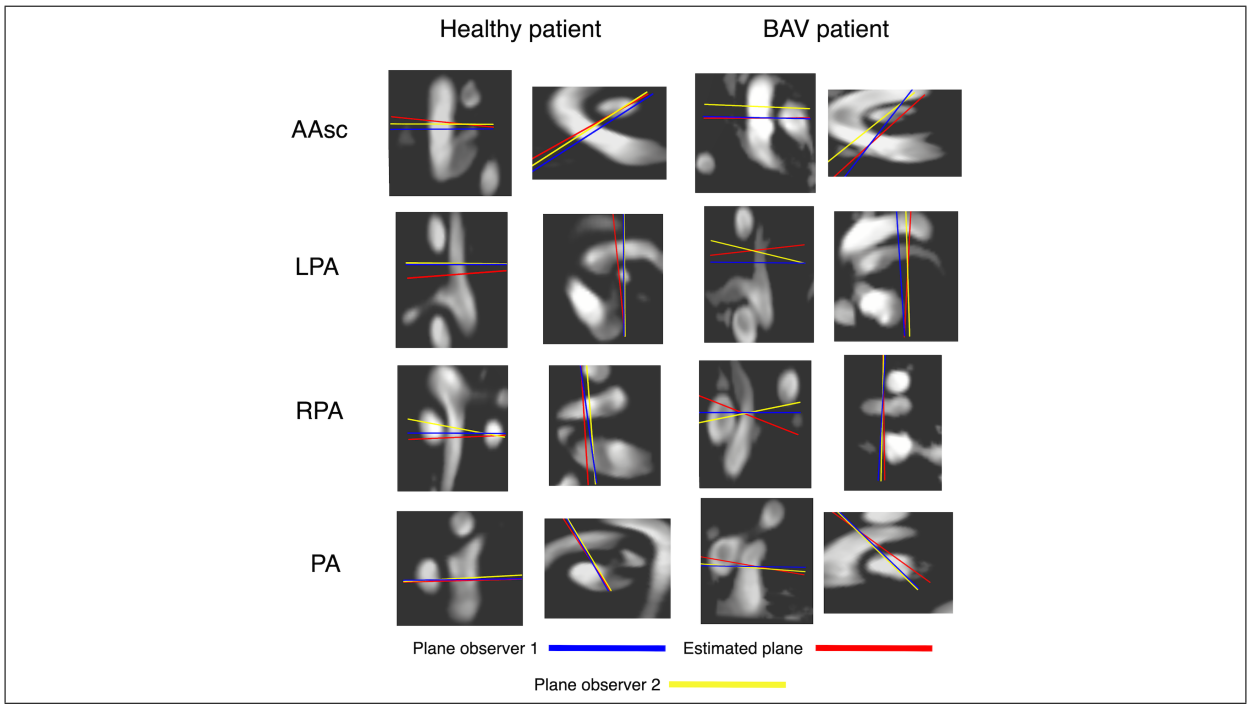

Figure 3.1. Visualization of plane localization shown as lines on two orthonormal images to the observer 1 plane. Estimated planes are shown in red and observers planes in blue and yellow.

Table 3.1 shows the average and deviation of angle error, distance error and NMI for all planes. The average and deviation of the performance metrics on all planes were 7.88 $\pm$ 4.33 degrees for angle error, 3.46 $\pm$ 3.25 mm for distance and 1.31 $\pm$ 0.08 for NMI. The best performance for angle error and distance error was on RPA plane and for NMI on PA plane. On the other hand, the worst performance for angle error was on LPA plane, and for distance error and NMI on AAsc plane. In AAsc plane we achieved low angle error, but with slightly higher distance error and lower NMI. This happens because AAsc is the bigger vessel and the plane planning has higher distance variation. This also impacts SSIM because the distance variations allow other tissues to show in the plane reformatting. Nevertheless, in 4D flow, a low angle error is sought in order to be able to recover well the flow values in the vessels, as long as the distance error is small enough not to leave the vessel. In the opposite case, in LPA plane we achieved slightly higher angle error with low distance error. LPA, being a small vessel, causes a small distance error, however, it is the one with the fewest tissues nearby, so the right angulation is more challenging to obtain.

Table 3.1. Performance metrics between observer 1 and the algorithm for all cross validation iterations. Average value in each plane and standard deviation in parenthesis.

Performance metric	**PA**	**RPA**	**LPA**	**AAsc**	**Average**
Angle between normals (degrees)	7.90 (4.44)	7.18 (3.86)	9.22 (5.21)	7.22 (3.35)	**7.88 (4.33)**
Distance error (mm)	3.47 (2.84)	2.19 (1.94)	2.97 (2.58)	5.18 (4.40)	**3.46 (3.25)**
Normalized mutual information	1.36 (0.06)	1.31 (0.06)	1.29 (0.10)	1.26 (0.06)	**1.31 (0.08)**

Table 3.2. Performance metrics between observers. Average value in each plane and standard deviation in parenthesis.

Performance metric	**PA**	**RPA**	**LPA**	**AAsc**	**Average**
Angle between normals (degrees)	4.23 (4.05)	2.57 (2.26)	6.08 (4.96)	4.82 (5.36)	**4.44 (4.4)**
Distance error (mm)	2.03 (2.03)	1.70 (1.50)	2.46 (2.27)	4.69 (4.47)	**2.72 (3.03)**
Normalized mutual information	1.40 (0.10)	1.39 (0.09)	1.37 (0.11)	1.32 (0.10)	**1.37 (0.10)**

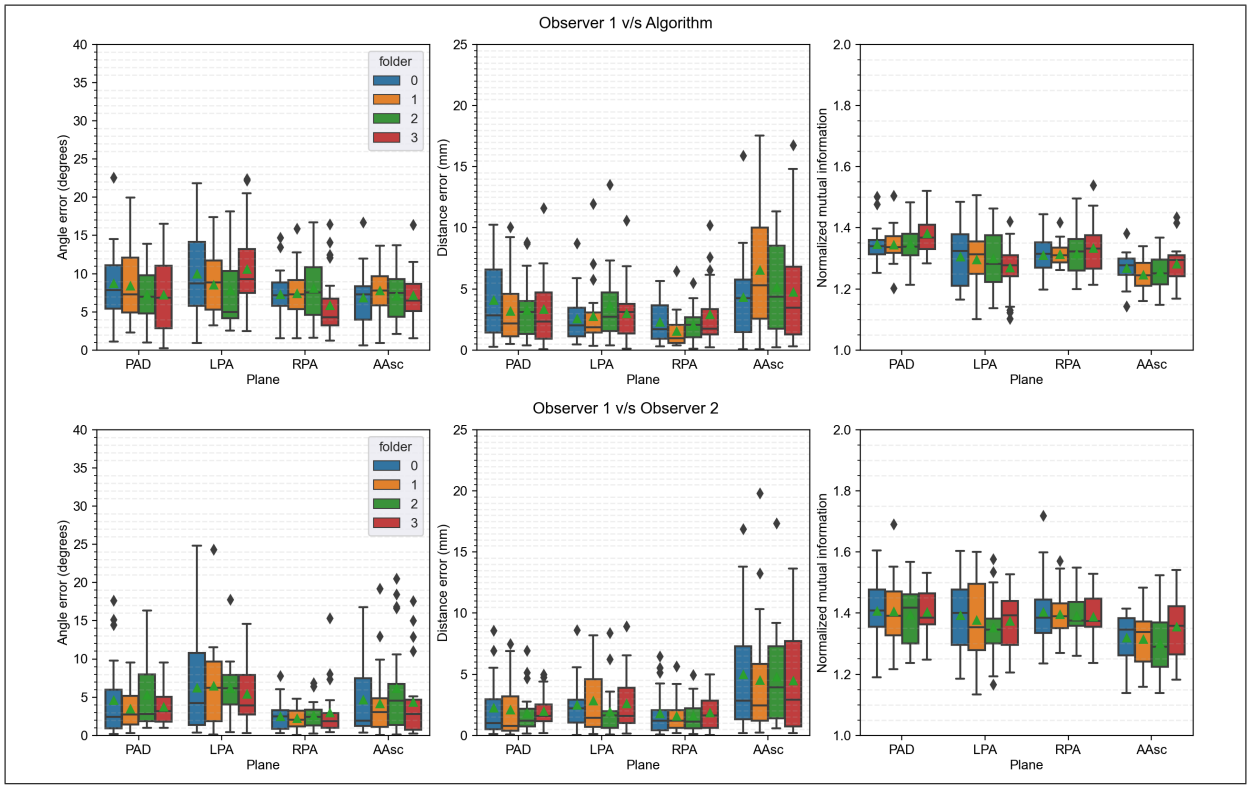

Figure 3.2. 4 folder cross validation performance metrics between observer 1 planes and algorithm estimated planes (first row), and between observer 1 and observer 2 planes (second row) . Green triangules show the mean of each boxplot and black diamonds the outliers. First column: angle error, second column: distance error and third column: Normalized mutual information

Comparing observer v/s algorithm errors with inter-observer errors (Table 3.2), we observed that our algorithm has great agreement in terms of distance, but there is still room to improve in terms of angulation error to get close to inter-observer variablity. Furthermore, the same pattern of errors as in the previous paragraph is observed in the inter-observer errors. With the best angle error performance in RPA and the worst in LPA, and the best distance error performance in RPA and the worst in AAsc. This suggests that both, the network and the observers, face similar visual problems when locating the planes.

3.2. Flow measurement

Figure 3.3 shows Bland-Altman plots for observer v/s algorithm and inter-observers differences. Inter-observer limits of agreement and average difference are lower than the limits of agreement and average difference between observer and algorithm. Nevertheless, difference between observer and algorithm is significant ($p < 0.05$) only on plane LPA (Table 3.3). We used Mann-Whitney non parametric test to measue difference beween flows because Shapiro-Wilk test revealed that flow data did not followed a normal distribution ($p < 0.05$). In addition, we found a flow agreement between observer and algorithm estimated planes with an average correlation of 0.82.

In terms of performance of the 2D U-Net segmentation (Table 3.4), the average and deviation of IoU on all planes was 0.89 ± 0.04 and for dice 0.91 ± 0.03. This values show high quality of segmentation using the 2D U-net network.

Table 3.3. Flow measurement. Flow values in L/min for both observers and the algorithm. Pearson correlation and Mann-Whitney test p-value between observers and observer v/s algorithm

Metric	**PA**	**RPA**	**LPA**	**AAsc**	**Average**
Flow Observer 1 (L/min)	5.14 (2.57)	2.05 (2.41)	2.99 (1.45)	4.79 (1.94)	3.74 (2.49)
Flow Observer 2 (L/min)	4.96 (2.57)	2.06 (2.35).	2.96 (1.45)	4.87 (1.90)	3.71 (2.45)
Flow Algorithm (L/min)	5.61 (2.48)	5.14 (2.57).	3.58 (1.9)	5.13 (2.31)	4.09 (2.63)
Correlation (Observer 1 v/s Algorithm)	0.80	0.82	0.86	0.80	0.82
Correlation (Observer 1 v/s Observer 2)	0.97	0.98	0.97	0.97	0.97
P-value (Observer 1 v/s Algorithm)	0.10	0.76	0.04	0.29	0.30
P-value (Observer 1 v/s Observer 2)	0.53	0.94	0.96	0.75	0.80

Table 3.4. U-net vessel segmentation performance. Average value in each plane and standard deviation in parenthesis.

Segmentation metric	**PA**	**RPA**	**LPA**	**AAsc**	**Average**
Dice coefficient	0.95 (0.02)	0.93 (0.04)	0.88 (0.01)	0.87 (0.04)	0.91 (0.03)
IoU	0.95 (0.01)	0.88 (0.07)	0.80 (0.02)	0.93 (0.03)	0.89 (0.04)

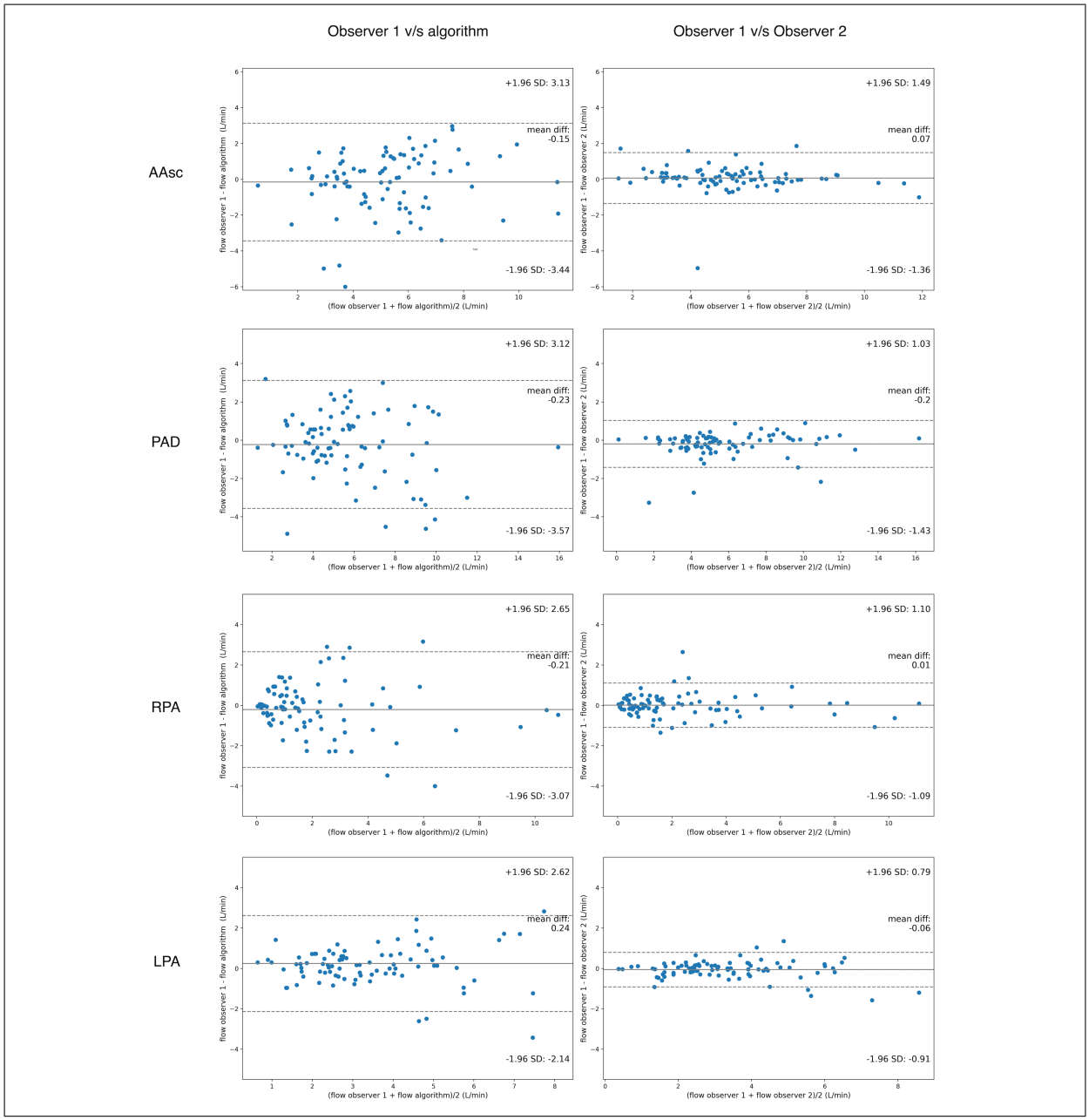

Figure 3.3. Bland–Altman plots comparing flow values in L/min. The first column shows the Bland-Altman plots for observer 1 v/s algorithm and the second column for observer 1 v/s observer 2. Each row shows the results for one plane. For each graph, the solid line represent the bias (average difference between the 2 measurements) while the dotted lines represent the limits of agreement (mean 1.96*SD of the differences between the 2 measurements).

3.3. Processing times

The longest step in our plane reformatting framework is the rigid registration of the angiography data that takes 20 seconds on average. With the trained CNN, plane search takes less than 5 seconds to find each plane. This task can be parallelized for all planes summing up to less than a 30 seconds for the complete plane reformatting. This is a significant reduction of time compared to manual plane reformatting which takes several minutes.

Training, regardless of the plane in which the networks were trained, took 4 GPU hours on average to converge to a stable performance in validation. The training of the 2D U-nets took 20 minutes per network.

4. DISCUSSION

Our results demonstrated a promising performance in terms of angulation and distance errors compared to other deep learning methods for 4D flow plane reformatting. According to the last reported method (Corrado et al., 2021), we improved the average accuracy of PA and AAsc plane reformatting by 10.93 and 5.18 mm for distance error, and 2.9 and 5.68 degrees for angle error, respectively. However, this comparison involves different training and test samples. Corrado et al. (2021) used 241 scans for training and 40 for testing without cross validation. Also, they reported higher inter-observer errors than ours (5.58 mm higher for distance error and 9.46 degrees for angle error) with a flow correlation of 0.81. Our correlation between the algorithm and one observer was similar to previous study inter-observer correlation and we improved the average correlation between flow and previous algorithm for plane reformatting. Our goal is to reach a higher correlation at the level between observers in our study (average correlation of 0.97) that shows almost perfect agreement in flow between the observers in the reformatting task. Our next step will be to apply our method on bigger public datasets for a fair benchmarking and on additional planes for a better clinical validation.

Whitehead et al. found a weak but significant correlation between angle and flow In (Whitehead, Doddasomayajula, Harris, Gillespie, & Fogel, 2009). For 15 degrees of angle error the flow is under or overestimated on $7\% \pm 5\%$ on average. For our plane localization, average angle and distance error are below the limits for appropiate calculation of the vessels hemodynamic parameters. Also, the value of correlation between flow measurements (average value of 0.82) indicates high agreement between observer and the algorithm for further clinical diagnostic. In addition, the value of the NMI index indicates high structure correlation between the manually defined and estimated planes, maintaining considerable part of the anatomy of the different planes of the heart.

Regarding the choice of the hyperparameters of the network, we experimented with different number of layers, filter sizes in the convolutional layers and output vector size of the convolutional layer. Our experiments showed that these changes converged to similar performances, but with different training times.

One limitation of our method is the dependency on the data pre-processing, in which segmentation and registration may produce errors that propagate and impact the final results. If the threshold-based segmentation fails, a manual FOV reduction is necessary, adding extra time to the plane reformatting processing time. Also, it is an indicator that the data has more signal noise than an average 4D flow acquisition. On the other hand, if the registration does not get the volume in the proper orientation, the transitions between different states will tend to fail. This occurs because the set of actions to move and rotate the planes are defined in a single set of coordinates. Therefore, when trying to change between states in a volume with a different orientation of the data which the network was trained with, the results are unpredictables. To face this problem we add random scaling, rotations and translations in the training to grant invariance of different locations to the network. However, this invariance only works for small variations. If the orientation and position are inconsistent from the one sought to obtain the registration, the network will not be able to converge to the desired plane. In the case of the data used in this work, part of them had greater signal loss that generated a worse registration and therefore greater plane location error. To overcome this problem we propose to use more advanced registration techniques like DeepReg (Fu et al., 2020) to converge to a good registration despite having different noise levels and data orientation.

Furthermore, we can improve our method with other plane reformatting methods. For example, the method described in (Corrado et al., 2021) finds 3D patches that isolates each vessel. If we use these patches instead of the full volume, the plane reformatting could be faster and more precise.

In this research we developed a fast and automatic deep learning framework for plane reformatting on 4D flow data that is suitable for data acquired from different MRI scanners vendors and for both healthy volunteers and patients.

REFERENCES

Alansary, A., Le Folgoc, L., Vaillant, G., Oktay, O., Li, Y., Bai, W., ... others (2018). Automatic view planning with multi-scale deep reinforcement learning agents. In *International conference on medical image computing and computer-assisted intervention* (pp. 277–285).

Bernstein, M. A., & Ikezaki, Y. (1991). Comparison of phase-difference and complex-difference processing in phase-contrast mr angiography. *Journal of Magnetic Resonance Imaging*, *1*(6), 725–729.

Bustamante, M., Petersson, S., Eriksson, J., Alehagen, U., Dyverfeldt, P., Carlhäll, C.-J., & Ebbers, T. (2015). Atlas-based analysis of 4d flow cmr: automated vessel segmentation and flow quantification. *Journal of Cardiovascular Magnetic Resonance*, *17*(1), 1–12.

Chollet, F., et al. (2015). *Keras.* `https://keras.io`.

Corrado, P. A., Seiter, D. P., & Wieben, O. (2021). Automatic measurement plane placement for 4d flow mri of the great vessels using deep learning. *International Journal of Computer Assisted Radiology and Surgery*, 1–12.

Dyverfeldt, P., Bissell, M., Barker, A. J., Bolger, A. F., Carlhäll, C.-J., Ebbers, T., ... others (2015). 4d flow cardiovascular magnetic resonance consensus statement. *Journal of Cardiovascular Magnetic Resonance*, *17*(1), 1–19.

Fu, Y., Brown, N. M., Saeed, S. U., Casamitjana, A., Baum, Z. M. C., Delaunay, R., ... Hu, Y. (2020). Deepreg: a deep learning toolkit for medical image registration. *Journal of Open Source Software*, *5*(55), 2705. Retrieved from `https://doi.org/10.21105/joss.02705` doi: 10.21105/joss.02705

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Kostrikov, I. (2018). *Pytorch implementations of asynchronous advantage actor critic.* https://github.com/ikostrikov/pytorch-a3c. GitHub.

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 50–60.

Markl, M., Frydrychowicz, A., Kozerke, S., Hope, M., & Wieben, O. (2012). 4d flow mri. *Journal of Magnetic Resonance Imaging*, *36*(5), 1015–1036.

Markl, M., Kilner, P. J., & Ebbers, T. (2011). Comprehensive 4d velocity mapping of the heart and great vessels by cardiovascular magnetic resonance. *Journal of Cardiovascular Magnetic Resonance*, *13*(1), 1–22.

Marrone, S., Olivieri, S., Piantadosi, G., & Sansone, C. (2019). Reproducibility of deep cnn for biomedical image processing across frameworks and architectures. In *2019 27th european signal processing conference (eusipco)* (pp. 1–5).

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... others (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529–533.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Payer, C., Štern, D., Bischof, H., & Urschler, M. (2016). Regressing heatmaps for multiple landmark localization using cnns. In *International conference on medical image computing and*

computer-assisted intervention (pp. 230–238).

Pérez-García, F., Sparks, R., & Ourselin, S. (2021). Torchio: a python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. *Computer Methods and Programs in Biomedicine*, 106236. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0169260721003102` doi: https://doi.org/10.1016/j.cmpb.2021.106236

Pluim, J. P., Maintz, J. A., & Viergever, M. A. (2003). Mutual-information-based registration of medical images: a survey. *IEEE transactions on medical imaging*, *22*(8), 986–1004.

Reza, A. M. (2004). Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement. *Journal of VLSI signal processing systems for signal, image and video technology*, *38*(1), 35–44.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International conference on medical image computing and computer-assisted intervention* (pp. 234–241).

Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, *52*(3/4), 591–611.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srivastava, N., Hinton, G., Krizhevsky, I., Alex and. Sutskever, & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Studholme, C., Hill, D. L., & Hawkes, D. J. (1999). An overlap invariant entropy measure of 3d medical image alignment. *Pattern recognition*, *32*(1), 71–86.

Styner, M., Brechbuhler, C., Szckely, G., & Gerig, G. (2000). Parametric estimate of intensity

inhomogeneities applied to mri. *IEEE transactions on medical imaging*, *19*(3), 153–165.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Torres, J. (2021). *Introducción al aprendizaje por refuerzo profundo. teoría y práctica en python*. WATCH THIS SPACE Book Series.

Whitehead, K. K., Doddasomayajula, R., Harris, M. A., Gillespie, M. J., & Fogel, M. A. (2009). Effect of flow angle and flow profile on phase contrast flow measurements: overestimation at extreme angles and skewed profiles. *Journal of Cardiovascular Magnetic Resonance*, *11*(1), 1–316.

APPENDIX

A. ARTICLE

From the next page to the end of the document, the article for the journal IEEE Transactions on Pattern Analysis and Machine Intelligence is included.

Automatic plane reformatting for 4D flow MRI using continuous reinforcement learning

Javier Bisbal, Julio Sotelo, Cristobal Arrieta, Pablo Irarrazaval, Cristian Tejos, Marcelo E Andia, Denis Parra, and Sergio Uribe

Abstract—4D flow MRI allows the calculation of hemodynamic parameters that provide valuable information to characterize cardiovascular diseases. One limitation is the time-consuming and user-dependent post-processing. We developed an automated reinforced deep learning framework for plane reformatting of 4D flow data using the Asynchronous Advantage Actor Critic algorithm to train a 2D convolutional network that sequentially updates plane parameters towards a target plane based on a continuous policy. We processed 4D flow data from GE, Siemens and Philips MRI scans of 67 healthy volunteers and 20 patients with congenital heart defects (47 men, 34 ± 12.4 years of age). All datasets were split in 50% training, 25% validation and 25% testing and validated with 4 fold cross validation. Our method achieved excellent results in terms of angulation and distance errors (average 7.88 ± 4.33 degrees and 3.46 ± 3.25 mm) with a flow correlation of 0.82. We successfully adapted a continuous reinforcement learning method to plane reformatting in 4D Flow suited for data from different scanner vendors, with promising results in healthy volunteers and patients. Future work with more data and planes is needed for clinical validation.

Index Terms—4D flow MRI, Reinforced learning, Deep learning, Localization, Plane reformatting.

1 Introduction

4D flow MRI allows the quantification of advanced hemodynamic parameters that provide valuable information to characterize cardiovascular diseases [1]. The volumetric coverage of 4D Flow MRI offers retrospective positioning of planes for flow measurements at any location within the acquired data volume. However, this task is time consuming and user-dependent, especially when analyzing multiple planes. The 4D flow cardiovascular magnetic resonance consensus statement [2] declares: "Efforts toward more standardized automated flow visualization approaches could be helpful in certain applications and would minimize operator-dependent variation, although users should understand the principles, strengths and limitations of different techniques".

- *J. Bisbal, J. Sotelo, C. Arrieta, P. Irarrazaval, C. Tejos, M. Andia and S. Uribe are with the Biomedical Imaging Center, Pontificia Universidad Catolica de Chile, Macul, Santiago, Chile.*
- *J. Bisbal, J. Sotelo, C. Arrieta, P. Irarrazaval, C. Tejos, M. Andia, D. Parra and S. Uribe are with Millennium Institute for Intelligent Healthcare Engineering, iHEALTH, Chile.*
- *J. Bisbal, J. Sotelo, C. Arrieta, P. Irarrazaval, C. Tejos, M. Andia and S. Uribe are with M Millennium Nucleus in Cardiovascular Magnetic Resonance, Cardio MR, Chile.*
- *P. Irarrazaval and C. Tejos are with the Department of Electrical Engineering, School of Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile.*
- *P. Irarrazaval and S. Uribe are with Institute for Biological and Medical Engineering, School of Engineering, Medicine and Biological sciences, Santiago, Chile.*
- *J. Sotelo is with School of Biomedical Engineering, Universidad de Valparaíso, Valparaíso, Chile.*
- *D. Parra is with Computer Science Department, School of Engineering, Pontificia Universidad Catolica de Chile, Santiago, Chile.*

Although several semi-automatic methods have been proposed to asses different quantitative parameter, only a few have been proposed to retrospectively reformat the 4D flow data and quantify flux through the vessels [3], [4]. Bustamante et al. proposed an automatic plane reformatting algorithm based on the registration of a vessel-specific atlas [3]. In this method, an atlas is created from a Phase Contrast Magnetic Resonance Angiography (PC-MRA) [5] of a healthy volunteer , with 2D planes positioned at multiple vessel locations. In order to align vessels and planes, the 3D PC-MRA input dataset is registered to the atlas. Despite the good results in terms of flow correlation between the automatic and semi-automatic method (with user-defined planes), atlas registration requires substantial computation time and depends on the similarity between the image and the atlas.

Deep Learning has shown promising results with high reproducibility and low variability for biomedical image processing tasks with different frameworks and architectures [6]. Corrado et al. recently proposed a deep learning-based method for plane reformatting in 4D flow [4]. This algorithm samples 3D patches from the input 4D flow data and uses a 3D convolutional neural network (CNN) to predict the patch most likely to contain the reformatted planes, along with the plane reformatting parameters (plane center and normal direction). Regardless of the promising results, the 3D CNN only computes the plane parameters of the plane reformatting from the 3D patches and not the plane reformattting itself. This implies that there is information from the plane reformatting in 2D that is not being used to solve the plane reformatting task. In addition, Corrado et al. only used 4D flow scans from one scanner vendor (GE Healthcare).

Blansit et al. proposed a deep learning method for plane reformatting in cardiac cine images that uses three short-axis

localization models [7]. The first model locates the mitral valve slice using a 2.5 dimensional VGG-19/LSTM [8], [9] ensemble network. The second model locates a bounding box around the heart with a 2.5 dimensional U-net [10]. The final model uses a U-net modified for heatmap regression [11] to localize for 4 landmarks on the mitral valve slice for plane reformatting. This method could be used for plane reformatting in 4D flow, however, not all vessels have specific landmarks in one particular plane (mitral valve slice in the case of short-axis cardiac cine images) to compute the plane reformatting. In 4D flow, multiple planes are needed within the volume to locate reference points that allow the computation of the plane reformatting in each vessel. This adds extra difficulty to perform 4D flow plane reformatting using landmarks in specific planes.

Another approach to solve the plane reformatting task is using Reinforced Learning (RL). In RL an agent learns how to make comprehensive decisions by mimicking navigation processes of an expert solving the plane reformattting task. RL allows agents to learn complex tasks that may need several steps to find a solution [12]. For plane reformatting we start with an initial plane in a specific position within the volume data and the agent sequentially updates the plane parameters using rotations and translations towards a target position. Alansary et al. developed a RL agent using a Deep Q-Network (DQN) [13] to solve the plane reformatting task in brain and cardiac cine MRI images [14]. This method achieved good performance and more comprehensive results because the RL agent performs actions similar to those an expert would use for the plane reformatting.

In this work, we adapted the RL framework [14] to be used in 4D flow images from different scanner vendors. A limitation of using DQN as a RL agent is that only one action (rotation or translation) can be performed at the time of updating the parameters of each plane and must have a discrete size. In order to have more degrees of freedom over rotations and translations, our method trains a RL agent using the Asynchronous Advantage Actor Critic (A3C) [15] algorithm to estimate a continuous action policy for rotations and translations. With this policy the agent rotates and moves three ortogonal planes within the 4D flow data towards a target vessel location.

TABLE 1
4D flow data description

MRI scanner	Description
GE 1.5T	32 scans from healthy volunteers
GE 1.5T	8 scans from patients with aortic coarctation
GE 1.5T	4 scans from patients with tetralogy of fallot
Siemens 3T	17 scans from healthy volunteers
Siemens 3T	8 scans from patients with bicuspid aortic valve (BAV)
Philips 3T	18 scans from healthy volunteers

2 METHODS

2.1 Data sets and image processing

We processed a total of 87 scans from different scanners including healthy volunteers and patients with congenital heart defects. The details of each dataset are in Table 1:

We preprocessed the data with an in-house developed MATLAB toolbox (MathWorks, Natick, MA, USA), and it involves 6 steps: isotropic interpolation, computation of angiography volume, segmentation, field of view (FOV) reduction, contrast enhancement and registration.

1) Isotropic interpolation: linear interpolation to transform all datasets to a resolution of $2x2x2\ mm^3$.
2) Angiography volume: computation of Phase Contrast Magnetic Resonance Angiography (PC-MRA) [5] images for Philips and Siemens scanners (43 scans) and complex difference [16] images from data acquired in GE scanner (44 scans). Complex difference and PC-MRA are two types of angiography images in MRI. From this point on we will refer to PC-MRA and complex difference images as angiography images.
3) Segmentation: segmentation on angiography images with a manually adjusted threshold for each image. From the segmentation we keep the largest connected element that corresponds to the whole heart and great vessels.
4) FOV reduction: reduction of field of view (FOV) using the smallest bounding box around vessels segmentation to confine the whole heart and great vessels. In some cases the segmentation was connected to areas that did not correspond to the heart and great vessels. This occurred in cases of data with higher noise (about 8 to 12 scans from GE scanner). To solve this issue we manually reduced the FOV to confine the heart and great vessels.
5) Contrast enhancement: Contrast limited adaptive histogram equalization (CLAHE) [17] in angiography images. This method improves local contrast and enhance the definition on edges of the heart and great vessels.
6) Registration: Rigid registration to align all angiography data in the same orientation. This registration maximizes the mattes mutual information [18] between a fixed angiography volume and a moving angiography volume using a evolutionary optimization strategy [19] using rotations and translations.

2.2 Reinforced learning (RL) framework

RL is a type of machine learning technique which an agent learns to behave in an environment by performing actions that maximize a reward signal. The agent is not told which actions to take, but must discover, by trying different actions, which ones produce the greatest reward. [12]. For the task of plane reformatting, an agent sequentially modifies the position and direction of a plane in state s. If the next state plane is closer to the target plane, a positive reward signal is granted, otherwise the agent gets a negative reward signal. Since the goal is to maximize the total reward, the agent will learn to navigate to the target plane within the 3D volume. At each time step t the agent gets a new state s and chooses an action following a policy π. The agent attempts to learn a policy that maximizes both immediate and subsequent future rewards.

Deep reinforcement learning (DRL) combines Deep Learning with RL, where deep learning allows agents to make decisiones from unstructured data without knowing all posible states and transitions [20]. For the task of plane reformatting an infinite number of planes can be computed from a single volume. Even if we discretize the problem, a large number of possible states and transitions makes it complex to compute a function that allows us to map all the states to the actions that will allow us to reach the target plane. With the Asynchronous Advantage Actor Critic (A3C) algorithm [15] we can estimate this function using a Convolutional Neural Network (CNN) without knowing all possible states and transitions.

Fig. 1 shows the proposed DRL framework. Number (1) denotes the state s, which consists of three ortogonal planes where the desired plane is the first plane. At each time step a grid of size $70x70x3$ from each state is sampled using bilinear interpolation from the angiography volume. $70x70$ denotes the width and height of the sampled images and 3 the number of ortogonal planes. The parameters of the desired plane in cartesian coordinates are,

$$\cos(\alpha)x + \cos(\beta)y + \cos(\phi)z + d = 0 \tag{1}$$

where $\vec{n} = (\cos(\alpha), \cos(\beta), \cos(\phi))$ is the normal vector of the plane with α, β and ϕ the angles between the plane and the cartesian axes x, y and z, respectively. d is the distance from the plane to the volume center. The action space is defined as $\{a_\alpha, a_\beta, a_\phi, a_d\}$, where $\{a_\alpha, a_\beta, a_\phi\} \in (-\omega_{min}, \omega_{max})$ degrees and$\{a_d\} \in (-d_{min}, d_{max})$ mm. $(-\omega_{min}, \omega_{max})$ and $(-d_{min}, d_{max})$ are the lower and upper bounds for the action space. Each action updates the state plane parameters as $\alpha_t = \alpha_{t-1} + a_\alpha, \beta_t = \beta_{t-1} + a_\beta, \phi_t = \phi_{t-1} + a_\phi, d_t = d_{t-1} + a_d$. The reward of each transition is

$$\begin{aligned} r_t = &\left(D\left(P_{t-1}, P_T\right) - D\left(P_t, P_T\right)\right) + \\ &\lambda\left(NMI\left(P_t, P_T\right) - NMI\left(P_{t-1}, P_T\right)\right) \end{aligned} \tag{2}$$

where P_t represents the plane parameters in step t and P_T the target plane, D is the Euclidean distance between the parameters of the two planes, NMI is the normalized mutual information [21] between the plane in step t and the target plane and λ is a scalar weight to balance each term of the reward.

The objetive is to maximize the expected return defined as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where $\gamma = 0.99$ is a discount factor that allow the summation to converge within a finite number of steps. We also defined a value function $V^\pi(s) = E\left[R_t \mid s_t = s\right]$ following an action policy $\pi\left(a_t \mid s_t\right)$. If we maximize $V^\pi(s)$, we also maximize the expected return. One way to estimate $V^\pi(s)$ is using a parametrization of the policy $\pi\left(a_t \mid s_t; \theta\right)$ and an estimation of the value function $V^\pi(s; \theta_v)$, with parameters θ and θ_v. The A3C algorithm [15] estimates the parameters θ and θ_v with a neural network. We trained the network in order to find the optimal policy that maximizes the estimation of $V^\pi(s; \theta_v)$.

The network consists of four 2D convolutional layers (number 2 in Fig. 1), a Long short-term memory (LSTM) cell, and two fully-connected layers (number 3 in Fig. 1). The LSTM cell connects the previous and present LSTM hidden and cell states to exploit information between time steps. The last fully-connected layer estimates the policy $\pi(s_t, a_t)$ and state value $V(s_t)$ (number 4 on Fig. 1). The policy is parameterized as a normal distribution

$$\pi(a \mid s, \theta) \doteq \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} \exp\left(-\frac{(a - \mu(s,\theta))^2}{2\sigma(s,\theta)^2}\right). \tag{3}$$

Here $\sigma(s, \theta)$ and $\mu(s, \theta)$ are the parameters estimated by the last layer of the neural network. Each parameter of the network is updated every t_{max} actions or in a terminal state. A terminal state is reached when the following two conditions are fulfilled in ten consecutive steps. First, if the agent reaches a state with a distance from the target plane lower than 3 mm and less than 4 degrees of angle error. In this case the agent a bonus of 1 to the reward. Second, if the agent continues to approach the target plane. The following equation updates the network parameters (number 5 in Fig. 1):

$$\nabla_{\theta'} \log \pi\left(a_t \mid s_t; \theta'\right)\left(A(s_t, a_t; \theta_v) + \eta \nabla_{\theta'} H\left(\pi\left(s_t; \theta'\right)\right)\right). \tag{4}$$

$\nabla_{\theta'} \log \pi\left(a_t \mid s_t; \theta'\right)$ denotes the direction of largest gradient of the probability of action a_t in the state s_t, which is weighted by $A(s_t, a_t; \theta_v)$, an estimate of the advantage function defined as

$$A(s_t, a_t; \theta_v) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V\left(s_{t+k}; \theta_v\right) - V\left(s_t; \theta_v\right), \tag{5}$$

where k is upper-bounded by t_{max}, in this implementation set to 8 steps. The gradient of an entropy estimate of the action policy denoted by $\eta \nabla_{\theta'} H\left(\pi\left(s_t; \theta'\right)\right)$ is added to improve exploration, preventing premature convergence to suboptimal policies. η adjusts the magnitude of the entropy regularization term. After the network parameters are updated, the policy distribution is sampled to obtain the actions that rotate and move the grid and sample the next state from the 3D angiography data (number 6 n Fig. 1).

TABLE 2
RL and network hyperparameters

Hyperparameter	value
$(-d_{min}, d_{max})$	$(-3, 3)$ mm
$(-\omega_{min}, \omega_{max})$	$(-3, 3)$ degrees
λ	3
β	0.01
t_{max}	8
Optimizer	Adam
Learning rate	0.00001

2.3 Experimental setup

2.3.1 Data division and validation

To train our network we split each dataset in Table 1 in 50% training, 25% validation and 25% testing. To validate our results we performed a 4 folder cross validation. This process consists of 4 iterations, where in each iteration a different training, validation and test set is chosen, until each of the data is used for both training and testing. This allows us to evaluate the stability of the network by training with different scans within the datasets.

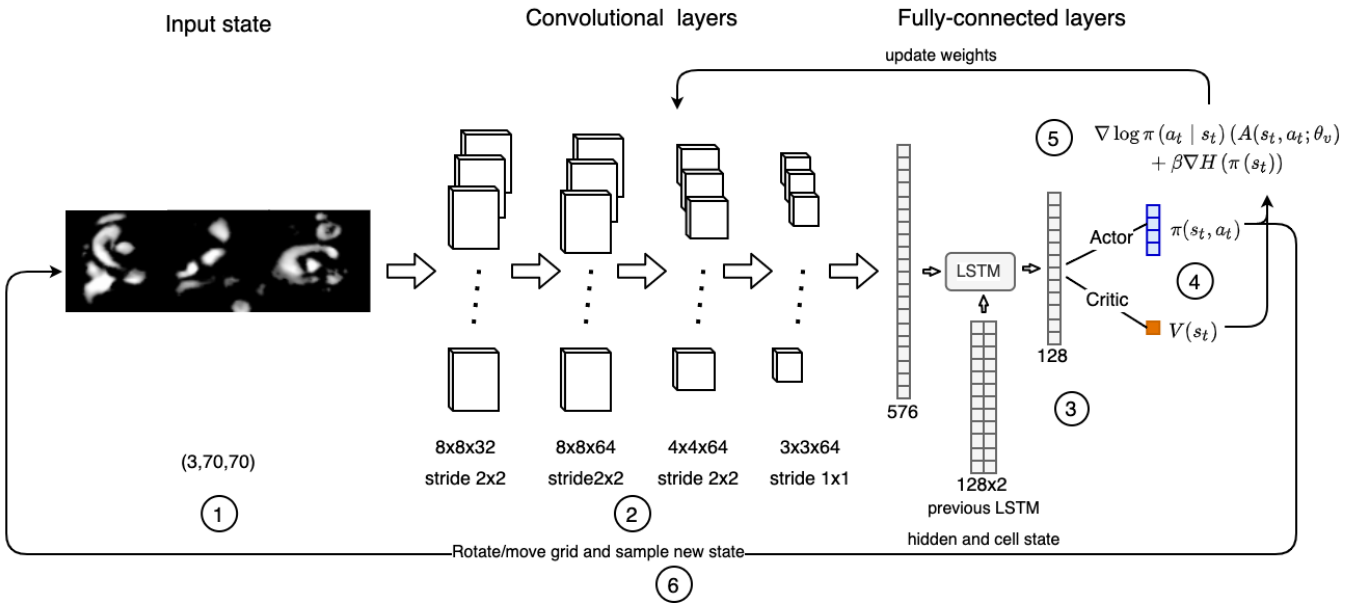

Fig. 1. RL framework for 4D flow plane search. An initial stack of images (1) pass through 4 convolutional layers (2) and 1 fully-connected layer. The output of this layer enters a LSTM cell with the previous LSTM hidden and cell state (3). The LSTM calculates the policy ($\pi(a_t, s_t)$) and value function ($V(s_t)$) (4). (5) CNN parameters are updated with equation 4. Finally, the policy rotates and moves the grid to reach a new state (6).

2.3.2 Manual annotations

We used the software Paraview (Kitware, Clifton Park, NY12065, USA) to generate 3D contours of the segmentation of the vessels and the velocity streamlines within the segmentation. Then, two observers with training in MR images visualization placed 1 perpendicular plane to the wall in 4 vessels: pulmonary artery (PA), right pulmonary artery (RPA), left pulmonary artery (LPA) and ascending aorta (AAsc). Each plane was located placing a point on the middle of the vessel and defining a normal vector. The 3D point coordinates were normalized between 0 and 1.

From these planes, we use inter-observer variability as a reference for later comparison with our algorithm.

2.3.3 Flow measurement

To measure hemodynamic parameters such as flow from the reformatted planes, we need a segmentation of the vessel to which the plane is perpendicular. To compute this segmentation we trained 2D U-net [10] with the manually placed planes of each vessel. We trained 4 networks (one for each vessel) for 40 epochs in Google Colab GPU (Nvidia K80 / T4) with Keras library [22]. We also applied affine transformations to the training dataset, including shear, rotation and scaling, to prevent overfitting. To measure the performance of the 2D U-Net segmentation we compute intersection over union (IoU) and Dice coefficient (equations 6 and 7) between U-net segmentation ($\hat{x}$) and manual segmentation (x) of vessels on the reformatted planes

$$Dice = \frac{2|x \cap \hat{x}|}{|x| + |\hat{x}|} \tag{6}$$

$$IoU = \frac{|x \cap \hat{x}|}{|x \cup \hat{x}|} \tag{7}$$

For statistical analyses of flow, we used Shapiro-Wilk [23] test to determine the normality of flow data, and non-parametric Mann-Whitney test [24] to measure the significance of the difference in flow between observers and between observer and algorithm.

2.3.4 Asynchronous training

The A3C algorithm has the advantage of training multiple agents asynchronuosly and updating the parameters of a shared model. For our model we trained 15 different agents, each one with a different training volume. This allows the network to learn from transitions across multiple volumes, decreasing training time and data overfitting. Data augmentation and dropout layers [25] was used to prevent additional overfitting. Data augmentation consisted on random scaling, translations and rotations between (0.9,1.1), (-5.5) mm and (-5.5) degrees, respectively. These transformations allows the network to have invariance to small location and scaling variations, and they were implemented with TorchIO python library [26]. The dropout layers were used in the convolutional layers with probability 0.4.

Table 2 contains the chosen RL and networks hyperparameters for training and evaluation. We chose small action spaces for rotations and translations to make the transition between states slower, but more precise. For λ, a value was chosen so that the error range of the plane parameters and the error range of NMI in equation (2) were similar. The optimizer, t_{max} and the learning rate were chosen empirically to converge to a good solution in a short training time.

The network was trained using PyTorch library [27] with A3C implementation code taken from [28]. Training was done on a Linux server with a NVIDIA Quadro RTX 8000 GPU for 150 epochs per plane. An epoch in this case consists of one agent training with all the volumes of the training set.

2.3.5 Plane reformatting evaluation

For validation and test data, the network is evaluated with a sequence of 100 steps, being the final plane directly selected from the last step. We measured three metrics on the estimated planes.

1) Distance error: defined as the Euclidean distance between a point in the ground truth plane in the middle of the vessel and the nearest point on the estimated plane.
2) Angle error: angle between estimated and ground truth normal vectors.
3) Normalized mutual information (NMI): Same NMI used in equation (2) to measure the correlation between ground truth plane and estimated plane. The range of NMI goes from 1 (no correlation between images) to 2 (perfect correlation between images).

To choose the model for test data evaluation, we chose the network with better performance in validation data. The best performance was calculated with the lower average distance between the parameters of the ground truth (manually placed) planes and the estimated planes.

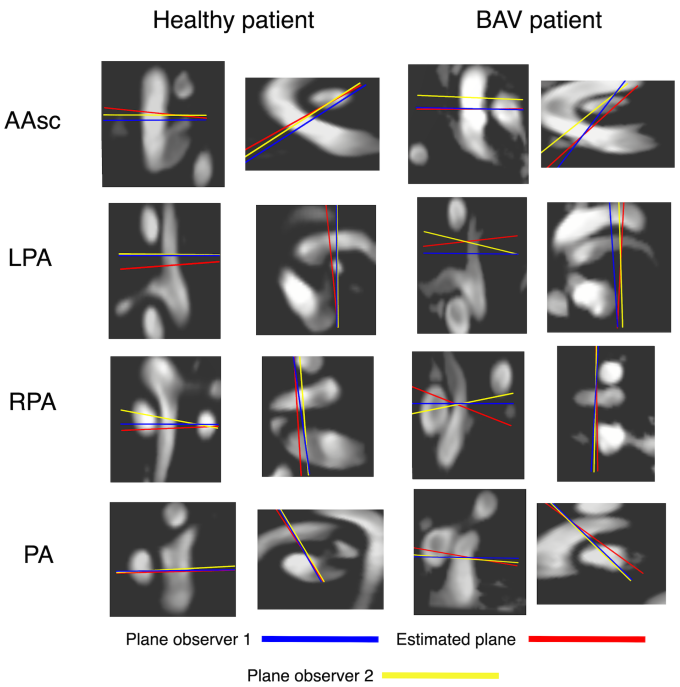

Fig. 2. Visualization of plane localization shown as lines on two orthonormal images to the observer 1 plane. Estimated planes are shown in red and observers planes in blue and yellow.

3 Results

3.1 Plane reformatting performance

Fig. 2 shows two representative test samples of a healthy volunteer and a patient with BAV. The network was trained with observer 1 manually placed planes. We observed that the difference between the estimated planes and their respective labels is similar to the difference between observers in terms of distance, but slightly higher in terms of angle error.

Fig. 3 shows the 4 folder cross validation performance metrics on each plane. Cross validation iterations revealed similar performance in terms of mean and interquartil range. Also, the size of interquatil range of the different folders has similar values to inter-observer differences. This suggests stability of the proposed method among all data samples compared to human error.

Table 3 shows the average and deviation of angle error, distance error and NMI for all planes. The average and deviation of the performance metrics on all planes were 7.88 $\pm$ 4.33 degrees for angle error, 3.46 $\pm$ 3.25 mm for distance and 1.31 $\pm$ 0.08 for NMI. The best performance for angle error and distance error was on RPA plane and for NMI on PA plane. On the other hand, the worst performance for angle error was on LPA plane, and for distance error and NMI on AAsc plane. In AAsc plane we achieved low angle error, but with slightly higher distance error and lower NMI. This happens because AAsc is the bigger vessel and the plane planning has higher distance variation. This also impacts NMI because the distance variations allow other tissues to show in the plane reformatting. Nevertheless, in 4D flow, a low angle error is sought in order to be able to recover well the flow values in the vessels, as long as the distance error is small enough not to leave the vessel. In the opposite case, in LPA plane we achieved slightly higher angle error with low distance error. LPA, being a small vessel, causes a small distance error, however, it is the one with the fewest tissues nearby, so the right angulation is more challenging to obtain.

Comparing observer v/s algorithm errors with inter-observer errors (Table 4), we observed that our algorithm has great agreement in terms of distance, but there is still room to improve in terms of angulation error to get close to inter-observer variablity. Furthermore, the same pattern of errors as in the previous paragraph is observed in the inter-observer errors. With the best angle error performance in RPA and the worst in LPA, and the best distance error performance in RPA and the worst in AAsc. This suggests that both, the network and the observers, face similar visual problems when locating the planes.

3.2 Flow measurement

Fig. 4 shows Bland-Altman plots for observer v/s algorithm and inter-observers differences. Inter-observer limits of agreement and average difference are lower than the limits of agreement and average difference between observer and algorithm. Nevertheless, difference between observer and algorithm is significant ($p < 0.05$) only on plane LPA (Table 5). We used Mann-Whitney non parametric test to measue difference beween flows because Shapiro-Wilk test revealed that flow data did not followed a normal distribution ($p < 0.05$). In addition, we found a flow agreement between observer and algorithm estimated planes with an average correlation of 0.82.

In terms of performance of the 2D U-Net segmentation (Table 6), the average and deviation of IoU on all planes was 0.89 $\pm$ 0.04 and for dice 0.91 $\pm$ 0.03. This values show high quality of segmentation using the 2D U-net network.

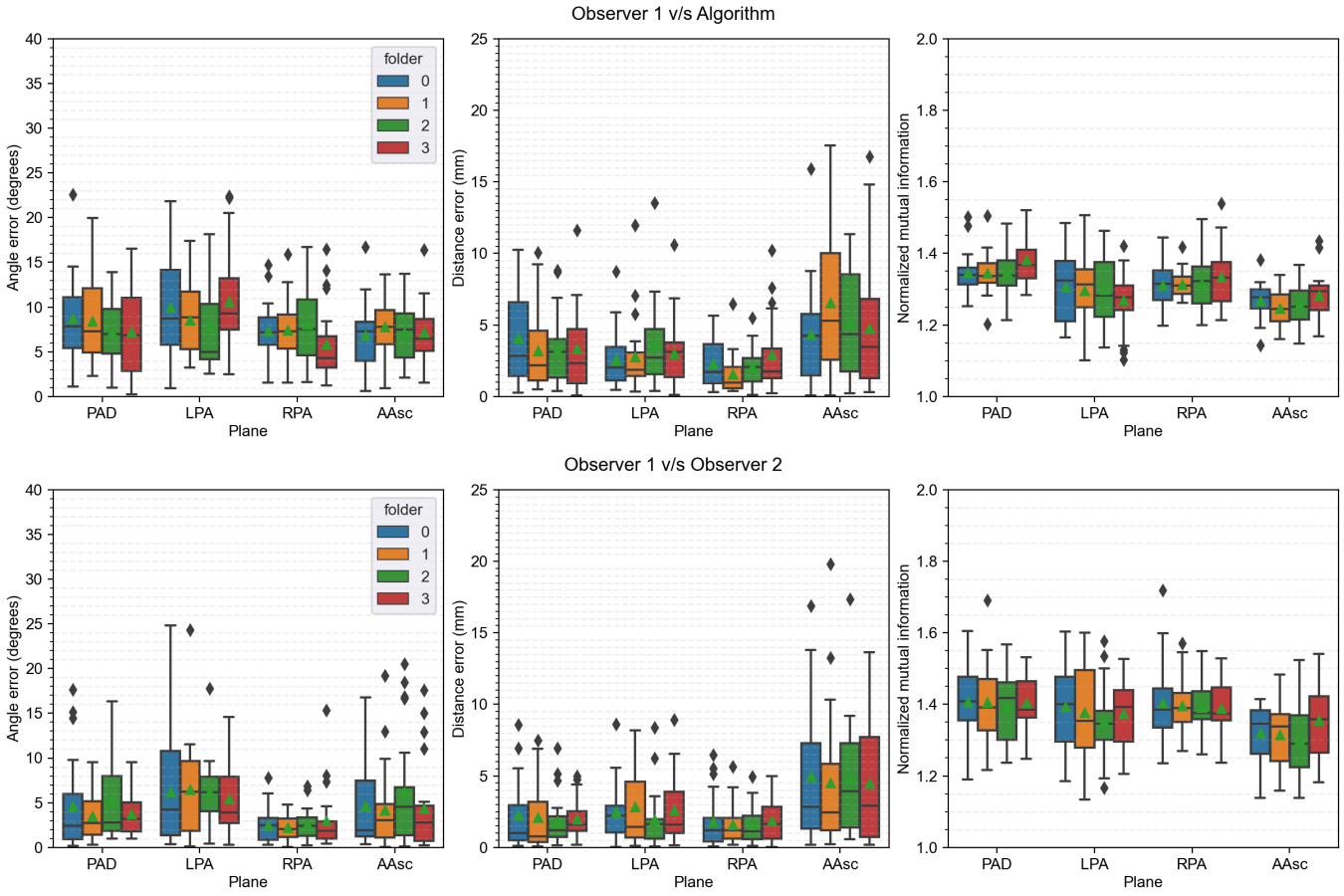

Fig. 3. 4 folder cross validation performance metrics between observer 1 planes and algorithm estimated planes (first row), and between observer 1 and observer 2 planes (second row) . Green triangules show the mean of each boxplot and black diamonds the outliers. First column: angle error, second column: distance error and third column: Normalized mutual information

TABLE 3
Performance metrics between observer 1 and the algorithm for all cross validation iterations. Average value in each plane and standard deviation in parenthesis.

Performance metric	PA	RPA	LPA	AAsc	Average
Angle between normals (degrees)	7.90 (4.44)	7.18 (3.86)	9.22 (5.21)	7.22 (3.35)	**7.88 (4.33)**
Distance error (mm)	3.47 (2.84)	2.19 (1.94)	2.97 (2.58)	5.18 (4.40)	**3.46 (3.25)**
Normalized mutual information	1.36 (0.06)	1.31 (0.06)	1.29 (0.10)	1.26 (0.06)	**1.31 (0.08)**

TABLE 4
Performance metrics between observers. Average value in each plane and standard deviation in parenthesis.

Performance metric	PA	RPA	LPA	AAsc	Average
Angle between normals (degrees)	4.23 (4.05)	2.57 (2.26)	6.08 (4.96)	4.82 (5.36)	**4.44 (4.4)**
Distance error (mm)	2.03 (2.03)	1.70 (1.50)	2.46 (2.27)	4.69 (4.47)	**2.72 (3.03)**
Normalized mutual information	1.40 (0.10)	1.39 (0.09)	1.37 (0.11)	1.32 (0.10)	**1.37 (0.10)**

3.3 Processing times

The longest step in our plane reformatting framework is the rigid registration of the angiography data that takes 20 seconds on average. With the trained CNN, plane search takes less than 5 seconds to find each plane. This task can be parallelized for all planes summing up to less than a 30 seconds for the complete plane reformatting. This is a significant reduction of time compared to manual plane reformatting which takes several minutes.

Training, regardless of the plane in which the networks were trained, took 4 GPU hours on average to converge to a stable performance in validation. The training of the 2D U-nets took 20 minutes per network.

4 Discussion

Our results demonstrated a promising performance in terms of angulation and distance errors compared to other deep learning methods for 4D flow plane reformatting. According

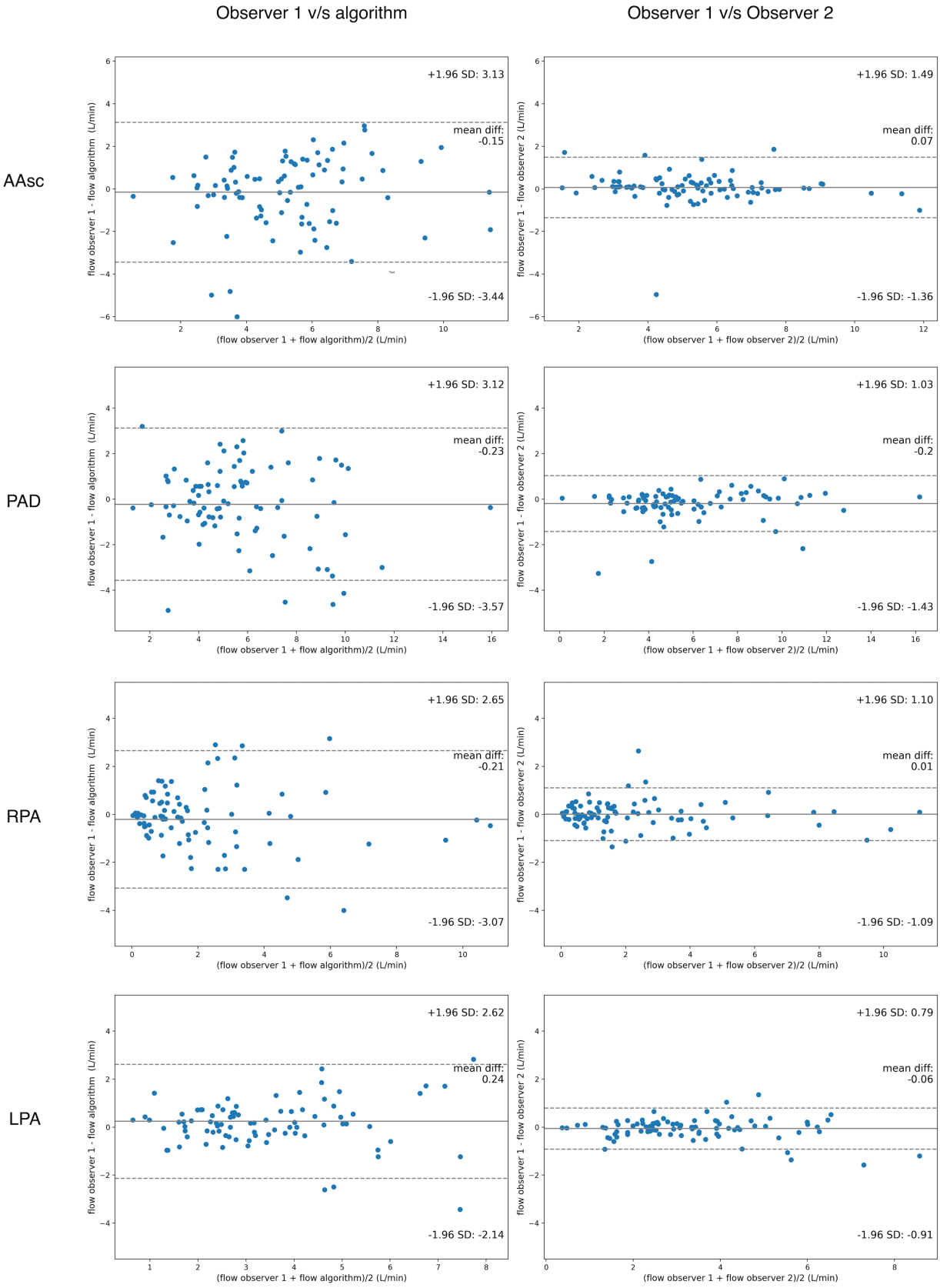

Fig. 4. Bland–Altman plots comparing flow values in L/min. The first column shows the Bland-Altman plots for observer 1 v/s algorithm and the second column for observer 1 v/s observer 2. Each row shows the results for one plane. For each graph, the solid line represent the bias (average difference between the 2 measurements) while the dotted lines represent the limits of agreement (mean 1.96*SD of the differences between the 2 measurements).

TABLE 5
Flow measurement. Flow values in L/min for both observers and the algorithm. Pearson correlation and Mann-Whitney test p-value between observers and observer v/s algorithm

Metric	PA	RPA	LPA	AAsc	Average
Flow Observer 1 (L/min)	5.14 (2.57)	2.05 (2.41)	2.99 (1.45)	4.79 (1.94)	3.74 (2.49)
Flow Observer 2 (L/min)	4.96 (2.57)	2.06 (2.35).	2.96 (1.45)	4.87 (1.90)	3.71 (2.45)
Flow Algorithm (L/min)	5.61 (2.48)	5.14 (2.57).	3.58 (1.9)	5.13 (2.31)	4.09 (2.63)
Correlation (Observer 1 v/s Algorithm)	0.80	0.82	0.86	0.80	0.82
Correlation (Observer 1 v/s Observer 2)	0.97	0.98	0.97	0.97	0.97
P-value (Observer 1 v/s Algorithm)	0.10	0.76	0.04	0.29	0.30
P-value (Observer 1 v/s Observer 2)	0.53	0.94	0.96	0.75	0.80

TABLE 6
U-net vessel segmentation performance. Average value in each plane and standard deviation in parenthesis.

Segmentation metric	PA	RPA	LPA	AAsc	Average
Dice coefficient	0.95 (0.02)	0.93 (0.04)	0.88 (0.01)	0.87 (0.04)	0.91 (0.03)
IoU	0.95 (0.01)	0.88 (0.07)	0.80 (0.02)	0.93 (0.03)	0.89 (0.04)

to the last reported method [4], we improved the average accuracy of PA and AAsc plane reformatting by 10.93 and 5.18 mm for distance error, and 2.9 and 5.68 degrees for angle error, respectively. However, this comparison involves different training and test samples. Corrado et al. used 241 scans for training and 40 for testing without cross validation. Also, they reported higher inter-observer errors than ours (5.58 mm higher for distance error and 9.46 degrees for angle error) with a flow correlation of 0.81. Our correlation between the algorithm and one observer was similar to previous study inter-observer correlation and we improved the average correlation between flow and previous algorithm for plane reformatting. Our goal is to reach a higher correlation at the level between observers in our study (average correlation of 0.97) that shows almost perfect agreement in flow between the observers in the reformatting task. Our next step will be to apply our method on bigger public datasets for a fair benchmarking and on additional planes for a better clinical validation.

Whitehead et al. found a weak but significant correlation between angle and flow In [29]. For 15 degrees of angle error the flow is under or overestimated on 7% $\pm$ 5% on average. For our plane localization, average angle and distance error are below the limits for appropiate calculation of the vessels hemodynamic parameters. Also, the value of correlation between flow measurements (average value of 0.82) indicates high agreement between observer and the algorithm for further clinical diagnostic. In addition, the value of the NMI index indicates high structure correlation between the manually defined and estimated planes, maintaining considerable part of the anatomy of the different planes of the heart.

Regarding the choice of the hyperparameters of the network, we experimented with different number of layers, filter sizes in the convolutional layers and output vector size of the convolutional layer. Our experiments showed that these changes converged to similar performances, but with different training times.

One limitation of our method is the dependency on the data pre-processing, in which segmentation and registration may produce errors that propagate and impact the final results. If the threshold-based segmentation fails, a manual FOV reduction is necessary, adding extra time to the plane reformatting processing time. Also, it is an indicator that the data has more signal noise than an average 4D flow acquisition. On the other hand, if the registration does not get the volume in the proper orientation, the transitions between different states will tend to fail. This occurs because the set of actions to move and rotate the planes are defined in a single set of coordinates. Therefore, when trying to change between states in a volume with a different orientation of the data which the network was trained with, the results are unpredictables. To face this problem we add random scaling, rotations and translations in the training to grant invariance of different locations to the network. However, this invariance only works for small variations. If the orientation and position are inconsistent from the one sought to obtain the registration, the network will not be able to converge to the desired plane. In the case of the data used in this work, part of them had greater signal loss that generated a worse registration and therefore greater plane location error. To overcome this problem we propose to use more advanced registration techniques like DeepReg [30] to converge to a good registration despite having different noise levels and data orientation.

Furthermore, we can improve our method with other plane reformatting methods. For example, the method described in [4] finds 3D patches that isolates each vessel. If we use these patches instead of the full volume, the plane reformatting could be faster and more precise.

In this research we developed a fast and automatic deep learning framework for plane reformatting on 4D flow data that is suitable for data acquired from different MRI scanners vendors and for both healthy volunteers and patients.

ACKNOWLEDGMENTS

This work has been funded by ANID-PIA-ACT192064 and ANID – Millennium Science Initiative Program – NCN17_129 and ICN2021_004. FONDECYT 1181057,

1191710, 1180525. Sotelo J. thanks to FONDECYT de iniciación en investigación 11200481. Arrieta C. was partially funded by CONICYT FONDECYT Postdoctorado 2019 3190763 and CONICYT PCI REDES 180090.

REFERENCES

[1] M. Markl, A. Frydrychowicz, S. Kozerke, M. Hope, and O. Wieben, "4d flow mri," *Journal of Magnetic Resonance Imaging*, vol. 36, no. 5, pp. 1015–1036, 2012.

[2] P. Dyverfeldt, M. Bissell, A. J. Barker, A. F. Bolger, C.-J. Carlhäll, T. Ebbers, C. J. Francios, A. Frydrychowicz, J. Geiger, D. Giese *et al.*, "4d flow cardiovascular magnetic resonance consensus statement," *Journal of Cardiovascular Magnetic Resonance*, vol. 17, no. 1, pp. 1–19, 2015.

[3] M. Bustamante, S. Petersson, J. Eriksson, U. Alehagen, P. Dyverfeldt, C.-J. Carlhäll, and T. Ebbers, "Atlas-based analysis of 4d flow cmr: automated vessel segmentation and flow quantification," *Journal of Cardiovascular Magnetic Resonance*, vol. 17, no. 1, pp. 1–12, 2015.

[4] P. A. Corrado, D. P. Seiter, and O. Wieben, "Automatic measurement plane placement for 4d flow mri of the great vessels using deep learning," *International Journal of Computer Assisted Radiology and Surgery*, pp. 1–12, 2021.

[5] M. Markl, P. J. Kilner, and T. Ebbers, "Comprehensive 4d velocity mapping of the heart and great vessels by cardiovascular magnetic resonance," *Journal of Cardiovascular Magnetic Resonance*, vol. 13, no. 1, pp. 1–22, 2011.

[6] S. Marrone, S. Olivieri, G. Piantadosi, and C. Sansone, "Reproducibility of deep cnn for biomedical image processing across frameworks and architectures," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[7] K. Blansit, T. Retson, E. Masutani, N. Bahrami, and A. Hsiao, "Deep learning–based prescription of cardiac mri planes," *Radiology: Artificial Intelligence*, vol. 1, no. 6, p. e180069, 2019.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[11] C. Payer, D. Štern, H. Bischof, and M. Urschler, "Regressing heatmaps for multiple landmark localization using cnns," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2016, pp. 230–238.

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

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[14] A. Alansary, L. Le Folgoc, G. Vaillant, O. Oktay, Y. Li, W. Bai, J. Passerat-Palmbach, R. Guerrero, K. Kamnitsas, B. Hou *et al.*, "Automatic view planning with multi-scale deep reinforcement learning agents," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 277–285.

[15] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[16] M. A. Bernstein and Y. Ikezaki, "Comparison of phase-difference and complex-difference processing in phase-contrast mr angiography," *Journal of Magnetic Resonance Imaging*, vol. 1, no. 6, pp. 725–729, 1991.

[17] A. M. Reza, "Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 38, no. 1, pp. 35–44, 2004.

[18] J. P. Pluim, J. A. Maintz, and M. A. Viergever, "Mutual-information-based registration of medical images: a survey," *IEEE transactions on medical imaging*, vol. 22, no. 8, pp. 986–1004, 2003.

[19] M. Styner, C. Brechbuhler, G. Szckely, and G. Gerig, "Parametric estimate of intensity inhomogeneities applied to mri," *IEEE transactions on medical imaging*, vol. 19, no. 3, pp. 153–165, 2000.

[20] J. Torres, *Introducción al aprendizaje por refuerzo profundo. Teoría y práctica en Python*. WATCH THIS SPACE Book Series, 2021.

[21] C. Studholme, D. L. Hill, and D. J. Hawkes, "An overlap invariant entropy measure of 3d medical image alignment," *Pattern recognition*, vol. 32, no. 1, pp. 71–86, 1999.

[22] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[23] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[24] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[26] F. Pérez-García, R. Sparks, and S. Ourselin, "Torchio: a python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning," *Computer Methods and Programs in Biomedicine*, p. 106236, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260721003102

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[28] I. Kostrikov, "Pytorch implementations of asynchronous advantage actor critic," https://github.com/ikostrikov/pytorch-a3c, 2018.

[29] K. K. Whitehead, R. Doddasomayajula, M. A. Harris, M. J. Gillespie, and M. A. Fogel, "Effect of flow angle and flow profile on phase contrast flow measurements: overestimation at extreme angles and skewed profiles," *Journal of Cardiovascular Magnetic Resonance*, vol. 11, no. 1, pp. 1–316, 2009.

[30] Y. Fu, N. M. Brown, S. U. Saeed, A. Casamitjana, Z. M. C. Baum, R. Delaunay, Q. Yang, A. Grimwood, Z. Min, S. B. Blumberg, J. E. Iglesias, D. C. Barratt, E. Bonmati, D. C. Alexander, M. J. Clarkson, T. Vercauteren, and Y. Hu, "Deepreg: a deep learning toolkit for medical image registration," *Journal of Open Source Software*, vol. 5, no. 55, p. 2705, 2020. [Online]. Available: https://doi.org/10.21105/joss.02705