Deep flow-net for EPI distortion estimation

Benjamin Zahneisen a,b,*, Kathrin Baeumler a,b, Greg Zaharchuk a, Dominik Fleischmann a,b, Michael Zeineh a,b

a Stanford University, Department of Radiology, Stanford, CA, USA
b Stanford 3D and Quantitative Imaging Laboratory, Stanford, CA, USA

A B S T R A C T

Introduction: Geometric distortions along the phase encoding direction caused by off-resonant spins are a major issue in EPI based functional and diffusion imaging. The widely used blip up/down approach estimates the underlying distortion field from a pair of images with inverted phase encoding direction. Typically, iterative methods are used to find a solution to the ill-posed problem of finding the displacement field that maps up/down acquisitions onto each other. Here, we explore the use of a deep convolutional network to estimate the displacement map from a pair of input images.

Methods: We trained a deep convolutional U-net architecture that was previously used to estimate optic flow between moving images to learn to predict the distortion map from an input pair of distorted EPI acquisitions. During the training step, the network minimizes a loss function (similarity metric) that is calculated from corrected input image pairs. This approach does not require the explicit knowledge of the ground truth distortion map, which is difficult to get for real life data.

Results: We used data from a total of $N_{train} = 22$ healthy subjects to train our network. A separate dataset of $N_{test} = 12$ patients including some with abnormal findings and unseen acquisition modes, e.g. LR-encoding, coronal orientation) was reserved for testing and evaluation purposes. We compared our results to FSL’s topup function with default parameters that served as the gold standard. We found that our approach results in a correction accuracy that is virtually identical to the optimum found by an iterative search, but with reduced computational time.

Conclusion: By using a deep convolutional network, we can reduce the processing time to a few seconds per volume, which is significantly faster than iterative approaches like FSL’s topup which takes around 10min on the same machine (but using only 1 CPU). This facilitates the use of a blip up/down scheme for all diffusion-weighted acquisitions and potential real-time EPI distortion correction without sacrificing accuracy.

1. Introduction

Geometric distortions limit the applications of echo planar imaging (EPI) (Mansfield, 1977) since they generally require some sort of correction (Jezzard, 2012). Geometric distortions can lead to errors when registering functional or diffusion-weighted imaging data to an undistorted anatomical scan. Another serious consequence is that they invalidate tractography tracts that pass through areas affected by distortions (Andersson et al., 2004). They are caused by a combination of local off-resonances, i.e., shim imperfections combined with the relatively low readout bandwidth of an EPI train along the phase encode direction which results in a spatial shift of the reconstructed voxel along that direction.

State-of-the-art post-processing software packages thus typically provide some sort of distortion correction functionality. The widely used topup method (Andersson et al., 2003) is based on the observation that by inverting the phase encoding direction, the direction of the distortions is also inverted. Regions stretched in one direction will be compressed in the other and vice versa. Employing a series of optimizations with descending spatial resolutions, the method iteratively finds the displacement map that best maps both images onto each other. More details on the blip up/down approach can be found in Bowtell et al. (1994); Chang and Fitzpatrick (1992a); Holland et al. (2010); Irfanoglu et al. (2015); Morgan et al. (2004) and Ruthotto et al. (2013). One of the problems in our application of topup is processing speed. The speed argument is especially critical if a blip up/down pattern were to be played out for every diffusion direction as required for the joint blip up/down reconstruction suggested in Chang and Fitzpatrick (1992b); Embleton et al. (2010); Irfanoglu et al. (2015); Zahneisen et al. (2017). Other traditional iterative methods exist as open source implementations that are highly optimized and therefore much faster than topup, e.g. (Holland et al., 2010; Irfanoglu et al., 2015; Morgan et al., 2004; Ruthotto et al., 2013).

As an alternative and in light of the recent interest in machine learning approaches, we propose a deep convolutional neural net to estimate the distortion map from an input pair of blip up and down images. The time required for inference is an order of magnitude faster than iteratively minimizing the up/down model. The network architecture is inspired from the deep flow net (Dosovitskiy et al., 2017; Fischer et al., 2015).
publicly available ground truth available (despite work by Graham et al. (2017), which provides some evidence of the existence of a ground truth) used to ensure consistency across various resolutions.

In contrast to the work in Dosovitskiy et al. (2017), we use an approach where we do not provide ground-truth displacement maps explicitly but train the network to minimize a similarity metric between corrected up and down images as employed in (de Vos et al., 2017) and (Shan et al., 2017). This approach allows us to train our network without the existence of a ground-truth fieldmap, which might not be readily available (despite work by Graham et al. (2017), which provides some publicly available ground-truth fieldmaps) or may not be guaranteed to represent a ‘true’ ground truth. The output of typical software packages that perform the field map estimation depends on a set of parameters like regularization, number of iterations, sub-sampling, etc. that all affect the resulting fieldmap. In contrast, our approach only requires minor total variation regularization to guide the network towards smoother displacement maps. Although our network architecture is fully convolutional which means that it learns a set of parameters for 2d convolutions, the actual unwarping of the images is implemented as 1d cubic interpolation with density weighting to account for signal pile-up.

Knowing the field map allows one to directly or indirectly account for the additional phase evolution during the reconstruction process. By training our network with different field maps, we are able to predict a displacement map from two input images with inverted phase encoding direction. Once the network is trained, the feed-forward estimation (inference) of the displacement maps is computationally efficient with processing times on the order of seconds for a typical EPI volume.

2. Methods

2.1. Deep flow net for distortion estimation

We implemented an unsupervised deep convolutional neural net, inspired by Flow-Net (Dosovitskiy et al., 2017), but we do not minimize the difference between the predicted displacement map and a ground truth map, and instead we minimize the mean squared difference of the un-warped images. This approach was previously described in Shan et al. (2017) in the context of affine registration of cardiac images.

Our U-Net (Ronneberger et al., 2015) architecture (Fig. 1) consists of an encoder and a decoder section. The input to the encoder is the up/down pair as a NxxNyx2 tensor. The feature map dimension is then decreased, and the number of channels is increased between layers resulting in an 8x8 map with Nc = 256 channels. Each convolutional layer consists of a convolution operation (3x3 kernels for all layers), batch normalization of the output, and a leaky ReLu (Maas et al., 2013) activation function. The decrease in feature map size between the main layers is achieved by using a stride s = 2 for the convolution operation (Springenberg et al., 2014). The encoder section consists of Nenc = 4 layers and results in an 8 × 8 × 256 feature map that serves as the input to the decoder section (see Fig. 2).

The decoder part (Fig. 3) consists of Ndec = 5 main layers where the feature map size is gradually increased by means of an up-convolution operation, also known as transposed convolution. Up-convolution steps refer to the process of up-sampling followed by a convolution (5x5 kernels for all layers) and an activation function. The up-sampling is performed by mapping (replicating) each feature map entry to a 2x2 nearest neighborhood. The final output of our network is a voxel displacement map (VDM) that describes the spatial displacement (in pixel units) of each voxel of the input images along the y-direction, i.e. the phase encoding direction.

In order to facilitate information flow through the network from lower resolutions and earlier layers of the encoder section we concatenate the resulting feature map of each up-convolution with an up-sampled version of the previous VDM and one layer of the encoder section with corresponding feature map dimensions (dashed red line in Fig. 3). This ensures that at each resolution the estimation of the new VDM is informed by the information of a low-resolution version and thus is able to be self-consistent across multiple resolutions. From a pure neural network perspective these shortcuts between earlier and later parts of a deep network reduce the problem of vanishing gradients (Hochreiter and Frasconi, 2009) and lead to a better convergence.

We estimate voxel displacement maps (VDM) at three distinct spatial resolutions (32x32, 64x64, 128x128) and apply them to the input up/down pair with the corresponding resolution (see un-warp module). In this context, spatial resolution refers to the grid size of our image and not to the actual voxel size in physical units.

![Schematic network architecture](image)

Fig. 1. Schematic network architecture. The input image Iup and Idown are down-sampled to a 64x64 and to a 32x32 grid as a pre-processing step. Then the full resolution up/down pair is propagated through the encoder network with a series of convolution layers with decreasing feature map dimensions. The final feature map (8x8xNc = 128) of the encoder is fed into a decoder network where the full image resolution is gradually restored by means of up-convolutions. At three distinct spatial resolutions a voxel displacement map (VDM) is estimated. The VDM estimation receives information not only from the decoder layers but also from encoder layers with the corresponding resolution. Each estimated VDM is used to un-warp the input images (un-warp module) and calculate a loss function based on the mean squared difference (MSE) of the corrected up/down pair. Because the un-warp operation is implemented in TensorFlow, it is straightforward to backpropagate the loss into the encoder/decoder network and update its parameters. By minimizing the loss of the un-warp operation during the training phase, no ground truth VDM is required.
2.2. The unwarp module

To make full use of the internal gradient calculations and backpropagation of the loss, we implemented an unwarp module in TensorFlow. It receives the distorted up/down pair and the estimated VDM as input with arbitrary resolution and applies (un)-warps the input with the distortion map. The unwarping of the distorted input images is performed by 1D-cubic interpolation along the phase encode direction. We also account for density variations of the transformed voxel position grid. The need for density compensation (Studholme et al., 2000) is well known and virtually all published correction methods apply it in some form. The intensity of a voxel that experienced signal pile up from multiple voxels in the distorted case, is reduced accordingly prior to the actual unwarping. While the density compensation is not able to restore the information of the individual contributions to the signal pile up, it preserves the total image intensity for the unwarped case. For each resolution the un-warped images are returned as tensors which are then fed into the custom mean squared error loss function described in the next section.

The input to the unwarp module consists of an \( (N_x \times N_y) / C^2 \) vector \( \text{IM} \) of linearized image intensities and a vector \( \text{vdm} \) with equal dimensions that holds the voxel displacements (in pixel units). A grid vector \( y_{\text{grid}} \) of undistorted y-pixel coordinates is constructed as an array of increasing indices \([0, 1, 2, \ldots, N_x \times N_y - 1]\) since only the y-coordinate is of importance for the distortion correction. This kind of indexing facilitates vector operations but ignores circular permutations that can be present in an MRI scan, i.e. the image re-appearing on the left when shifted past the right edge.

For the transformation the inverse mapping \( u' \), i.e. the pixel shift values that maps a distorted pixel back to the undistorted grid, is calculated as \( u' = y / C \cdot \text{vdm} \). \( u' \) contains fractional pixel shift values and

![Fig. 2. Encoder section of the network:](image)

The up/down input pair \((N_x \times N_y \times 2)\) is fed into the first convolution layer with \( N_c = 16 \) filters and 5x5 kernel size followed by a leaky ReLu activation function and a batch normalization operation. The \( N_z \) direction of the feature map is omitted in the graph. A 2nd convolution layer with identical parameters is applied and stored and sent to the decoder section (blue arrows). The reduction in feature map size from Layer0 to Layer1 is achieved by using a stride \( s = 2 \) for the convolution operation. All following convolutions have a 3x3 filter kernel. These operations are repeated until the final feature map (dimensions 8x8 by \( N_c = 128 \)) is fed into the decoder network.

![Fig. 3. Decoder section of the network:](image)

The purpose of the decoder section is to generate feature maps at various resolution stages that allow the calculation of displacement maps by means of 1x1 convolution operations (red arrows). Each node in the decoder receives input from an up-sampled version of the previous layer, the feature map from the encoder with corresponding resolution (blue rectangles) and, if available for Layer 7 and 8, an up-sampled version of the displacement map that was estimated at the previous stage. The VDMs, together with up/down input data, are sent to the unwarp module where they are used to correct the input. The mean squared error (MSE) of the corrected input is sent back to update the weights in the network (using TensorFlow’s backpropagation).
therefore mapping of the image intensities to the un-warped image grid has to be done by interpolation; 1d-cubic in our case. For each new coordinate $u$ we get the four nearest grid points $u^{(-1)}_g, u^{(0)}_g, u^{(+1)}_g, u^{(+2)}_g$ by $u^{(0)}_g = \text{floor}(u)$ $u^{(-1)}_g = u^{(0)}_g - 1$ $u^{(+1)}_g = u^{(0)}_g + 1$ $u^{(+2)}_g = u^{(0)}_g + 2$

where floor($x$) stands for the floor operation.

$\Delta y$ is the defined as the distance between $u'$ and the grid point on the left $u^{(0)}_g$ as $\Delta y = u - u^{(0)}_g$. The formula for cubic interpolation can be split into 4 terms where $IM(x)$ refers to image intensity at position $x$:

\[ cub = \left( \frac{1}{2} * IM\left( u^{(-1)}_g \right) + \frac{3}{2} * IM\left( u^{(0)}_g \right) - \frac{3}{2} * IM\left( u^{(+1)}_g \right) + \frac{1}{2} * IM\left( u^{(+2)}_g \right) \right) \Delta y^3 \]

\[ quad = \left( \frac{1}{2} * IM\left( u^{(-1)}_g \right) - \frac{5}{2} * IM\left( u^{(0)}_g \right) + 2 * IM\left( u^{(+1)}_g \right) - \frac{1}{2} * IM\left( u^{(+2)}_g \right) \right) \Delta y^2 \]

\[ lin = \left( -\frac{1}{2} * IM\left( u^{(-1)}_g \right) + \frac{1}{2} * IM\left( u^{(+1)}_g \right) \right) \Delta y \]

\[ z_0 = IM\left( u^{(0)}_g \right) \]

and finally combining all four terms for all grid points leads to the unwarped image:

\[ IM_{\text{unwarp}} = cub + quad + lin + z_0 \]

\[ w_{\text{right}} = w_{\text{left}} + w_{\text{middle}} \]

\[ w_{\text{left}} = \sum_{y=-\Delta y}^{1-\Delta y} (1 - \Delta y) \]

\[ w_{\text{right}} = \sum_{y=1}^{\Delta y_k} (1 - \Delta y) + \Delta y_k + \Delta y \]

\[ w_{\text{middle}} = \sum_{y=-\Delta y_k}^{\Delta y_k} (1 - \Delta y) \]

\[ w_{\text{final}} = w_{\text{left}} + w_{\text{right}} + w_{\text{middle}} \]

2.2.1. Density compensation

The sampling density is estimated by a smoothed linear interpolation of the grid point density. At each grid point $g$, the contributions from neighbors on the left $\sum_{y=-\Delta y}^{1-\Delta y} (1 - \Delta y)$ and on the right $w_{\text{right}} = \sum_{y=1}^{\Delta y_k} (1 - \Delta y) + \Delta y_k + \Delta y$ are calculated separately and summed up for the final value $w_g = w_{\text{left}} + w_{\text{middle}}$. An additional step performs smoothing of the density function with a gaussian kernel to remove high frequent oscillations. Prior to the unwarping, the uncorrected image is multiplied by the inverse sampling density which effectively reduces the intensity in regions with signal pile up and increases the intensity in regions with lower density. Fig. 4f illustrates the density estimation for a given grid point.

The effect of the density correction on the corrected images is shown in Fig. 4a-e. Uncorrected, the signal pile up in the input data leads to bright blobs in the frontal region of the brain for the unwarped images in c. The density compensated images are shown in d. The bright blobs are successfully corrected; however, a ripple artifact appears (arrows) due to high frequent changes of the density function. The unwarped images with smoothed density weighting function are shown in e.

We implemented the unwarp module as tensor operations in Tensorflow using the built-in tf.gather(), tf.pow() and tf.unsorted_segment_sum() functions. We refer to the official Tensorflow documentation for a complete description of the functionality and only give a brief overview here: tf.gather(input_array, indices) returns the values of input_array and positions given by indices and corresponds to the IM(x) operation. tf.unsorted_segment_sum(input_array, indices) works in a similar way but instead returns the sum of the input_array for all equal values in the indices array. This function is an efficient way to calculate $w_{\text{right}}$ and $w_{\text{left}}$ for the entire array of grid points.

The resulting VDMs are relative to the un-distorted image space, e.g. a positive version is applied to the up acquisition and sign inversion yields the map for the down case.

Fig. 4. Density compensation: a shows an uncorrected image with signal pile up (vertical arrows) in orbito-frontal part of the brain and the corresponding displacement map in b. Without density compensation (c) the artificially high signal intensity gets distributed to all voxels that contributed to the signal pile up resulting in two bright blobs. With density compensation, the image in d appears normal although the spatial resolution is lost in this particular region of the image. A residual ripple artifact can be observed (arrow) in d which is due to high frequent oscillations in the density function. Simple smoothing with a gaussian kernel greatly reduces this artifact.
2.3. Loss function and regularization

We aim to minimize a combined loss function which consists of the mean squared errors (MSE) between the corrected up/down images $I_{\text{down}}^{(n)}$ for each internal resolution stage $n$:

$$L_{\text{MSE}}^{(n)} = \frac{1}{N} \sum (I_{\text{down}}^{(n)} - I_{\text{up}}^{(n)})^2$$

This choice of the loss function minimizes the difference between corrected up/down images which, in the absence of noise and other MR imperfections, should approach zero for an ideal correction where the up-image is perfectly mapped onto the down-image. The squared error also puts more weight on greater differences and thereby puts more emphasis on the edges of an uncorrected image which are the regions where we expect the strongest pixel shifts.

Since the displacement map ultimately comes from a physical magnetic field variation with its boundary conditions governed by the Maxwell equations, it is reasonable to assume a certain smoothness of the displacement map. In addition, the problem of finding the underlying displacement map is generally ill-posed, in particular for compressed voxels, regularization is needed for convergence, see (Andersson et al., 2003) for more details. We therefore add the anisotropic total variation (TV) norm of each predicted VDM as a smoothness penalty to the total loss function:

$$L_{TV}^{(n)}(\text{VDM}) = \sum_{i,j} |\text{VDM}_{ij}^{(n)} - \text{VDM}_{ij}^{(n)}|^2 + |\text{VDM}_{ij}^{(n)} - \text{VDM}_{ij}^{(n-1)}|.$$ 

where $i,j$ are the pixel indices.

We add an additional penalty term to restrict the values that can occur in the displacement maps to reasonable values:

$$L_{\text{valley}}^{(n)} = \sum \min(|\text{VDM}_{ij}^{(n)}| - \text{vdmmax} , 0)$$

This valley shaped loss function only penalizes values above a certain threshold. We use a maximum displacement $\text{vdmmax} = 32$ pixels for the full resolution and scale it down for lower resolutions accordingly. The valley loss function only plays a role during the early training phase. Our resampling implementation does not account for a circular permutation of the image, i.e. the image re-appears on the other side if shifted outside the FOV. Instead we simply cut off the part that gets shifted too far out. The valley loss penalizes trivial solutions with extremely large shifts on the edges of an uncorrected image which are the regions where we expect the strongest pixel shifts.

The total loss $L_{\text{total}}$ then has the form:

$$L_{\text{total}} = \sum_{n=1}^{N_{\text{res}}} L_{\text{MSE}}^{(n)} + \lambda (L_{TV}^{(n)} + 10^4 L_{\text{valley}}^{(n)})$$

where $\lambda$ is the regularization parameter for the smoothness penalty.

2.4. Training and test data – image acquisition protocols

For a network to be able to generalize well it is important that the training data covers a wide range of potential image properties or characteristics. We therefore chose to train our network on wide variety of acquisition parameters and scanners. All images were acquired using a diffusion-weighted (DW) spin-echo EPI sequence at 3T. Images were acquired using a custom-built DW simultaneous multi-slice (SMS) spin-echo EPI sequence and with a vendor provided sequence without multi-band support. Various combinations of in-plane $R_p = \{1,2\}$ and multi-band acceleration $R_{\text{mb}} = \{1–4\}$ were included in the training data. The acquired in-plane voxel sizes ranged between 1.3 and 2.4 mm, slice thickness between 1.7 and 2.2 mm, and echo time (TE) between 65 and 105 ms and the acquisition bandwidth along the phase encode direction between 17.5 and 23.8 Hz/pixel. The phase encoding direction for all acquisitions used for training was Anterior-Posterior (AP). The network operates in the image domain, and therefore specific MR acquisition parameters are not relevant for the calculation of the voxel displacement maps. They become relevant if one wants to transform the VDM into an off-resonance map with actual values in Hz.

A total of $N_{\text{subjects}} = 37$ subjects (16 female) was split into $N_{\text{train}} = 22$ training cases, $N_{\text{test}} = 10$ test cases and $N_{\text{val}} = 4$ validation cases. A complete breakdown of the cases and imaging parameters can be found as supplementary material. Three datasets were acquired with blip up/down for all diffusion directions and included for data augmentation purposes. Because the distortions are expected to be very similar for all diffusion directions (ignoring the effects of motion and eddy currents), we only included 3 additional diffusion weighted volumes from each subject. Seven clinical cases were included in the test set based on retrospective IRB approval and after de-identification. The conditions (diagnosis) displayed in the results section included a cavernous malformation (in Fig. 6), an essential tremor patient (Fig. 7) and a glioblastoma multiforme (Fig. 8). Furthermore, we tested our network on publicly available datasets from the human connectome project (WU-Minn HCP Lifespan Pilot Data, www.humanconnectome.org) with left-right encoding and on cases with coronal orientation.

Each acquired volume was resampled from the original matrix size to a 128x128 grid and then each slice was treated as a separate input to the network resulting in $N_{\text{tot}} = 6354$ slices, i.e. the total number of training examples extracted from all available datasets. Although every up/down pair of images is treated as a separate input to the network, the split between training and test cases was performed on a subject level because of the expected strong correlations between neighboring slices within individual subjects that would alter the validity of the test measurements, as it would overestimate how well the network would generalize to unseen test cases.

Data augmentation is a widely used concept in machine learning applications to increase the available number of training examples. From every original subject, we generated two new synthetic datasets by a) flipping (mirroring) along the readout direction and b) interchanging which image is considered blip up and which down. Since the goal of the network is to generalize from the training dataset to whatever unseen test case it receives, the training dataset should span the space of potential datasets. Both augmentations achieve that without altering the underlying physical process of the distortions. In contrast, data augmentation by rotating or scaling the input images would alter the specific restriction that distortions only occur along the well-defined phase encoding direction and the transformed images would no longer be consistent with the change in the underlying susceptibility field due to these transformations.

As a pre-processing step, the mean intensity of all input image data was normalized to 1 with unchanged standard deviation and resampled to the three different resolutions used in the network by bi-linear interpolation. Prior to the down-sampling step for the $n = 64$ and $n = 32$ matrix dimensions, the up/down images were filtered with a Gaussian smoothing kernel (kernel size 3x3, sd = 1.2 for n = 64 and sd = 2 for n = 32).

2.5. Machine learning framework

We used a combination of Python versions of Keras 2.1.5 (Chollet, 2015) and TensorFlow 1.6 to implement and train our network. Stochastic gradient descent was performed using an Adam optimizer (Kingma and Ba, 2015) with the default settings provided by the Keras package. We used batch normalization (Ioffe and Szegedy, 2015) for the layers in the encoder section only, since the decoder layers perform a regression task where the scaling of the output needs to be transmitted. Training was performed with $N_{\text{epochs}} = 100$ epochs and with a learning rate of $lr = 0.001$ using a batch size of 32. For all training settings, the loss function for both the training and the test loss reached a plateau after
around 80–100 epochs and further training did not improve the test performance. We trained our network on a server with an Intel Xeon CPU with 16 available cores for multi-threading without GPU support. We also trained our model using a cloud-based virtual machine (Google ML engine) with GPU support (1 Nvidia K80 card). In both cases, models typically converged after approximately 48 h.

2.6. Fieldmap/distortion field estimation using FSL

We chose to use FSL’s slip-up/down correction package (Andersson et al., 2003) as a gold standard for a fieldmap estimate because of its robustness and wide use within imaging community. We used FSL 5.0.9 (http://fsl.fmrib.ox.ac.uk/fsl) (Smith et al., 2004) and specifically topup, which uses a method similar to that described in Andersson et al. (2003), to calculate the distortion field and the field map. All corrections relied on settings from the standard configuration file that is part of the software distribution. Topup corrected images were pre-processed with the same pipeline that was used for the network input and the MSE image metric from the conv-net was used to calculate the topup loss for a set of input images.

3. Results

3.1. Learning phase and comparison with results from FSL’s topup

Fig. 5 displays the evolution of the loss metrics $L_{\text{MSE}}^{(128)}$ for the full resolution output images during the training phase. The training (blue line) and the test (orange) loss sharply decrease during the initial training phase and reach a plateau after around 100 epochs. The black dashed line represents the loss that was calculated from topup corrected up/down images of the test set. After around 20 epochs the network reaches the accuracy of the topup correction and with further training surpasses it by approximately a factor of 2. Fig. 5 compares two different VDM regularization factors $\lambda = 1$ in a and $\lambda = 0.1$ in b. A higher regularization factor leads to smoother VDMs, i.e. field maps in a, and results in a slightly higher final training loss. For the lower regularization factor in b, the network selects a solution with lower training error at the cost of pronounced regions with erroneous displacement values outside the brain. For the remainder of this paper we chose the smoother solution and therefore used a regularization factor of 1. For a fixed matrix size of 128x128 and 40 slices, topup processing is limited to single CPU usage and takes approximately 10 min per volume. A single network inference step on maximum available number (=12) of CPUs takes less than 10 s on the same computer. For a better comparison, we also limited the number of available CPUs to 1 by running Tensorflow processing within a virtual machine. The processing time then slightly increases to around 30s, which is still considerably faster than the iterative approach.

3.2. Evaluation of correction accuracy

The results from the correction is displayed in Fig. 6 with 3 axial slices from three different subjects randomly selected from the test dataset. The first and second column show uncorrected blip up and down images with typical distortions. The next two columns show the results of the correction (corrected up and down averaged together) and the residual difference between the corrected up and down images. The last two columns display the results from the topup corrected images and the respective residual difference. All difference maps were scaled with mean image intensity of the respective image pair and displayed with identical color mapping. The difference maps confirm what the loss function graphs in Fig. 5 implies: the neural network approach results in a slightly lower magnitude difference image compared to topup. We would like to emphasize here that the central message of this work is not a direct one-to-one comparison between our method and FSL; A comprehensive comparison might yield a slightly different outcome than these preliminary results.

The resulting correction for a single subject diagnosed with essential tremor is displayed in Fig. 7. The uncorrected up/down volumes are plotted in a and b. The pixel-wise difference between a and b is shown in d. The corrected and averaged output of the network is shown in c with the residual error in e dramatically reduced compared to d. Except for the edges of the brain and some CSF boundaries, residual error appears unstructured and has little resemblance to brain structures. To further validate the correction, the averaged output of the network in c was registered to a structural T1 scan using a global rigid-body plus scaling transformation, i.e. rotations, translations and scaling. Fig. 7f shows a contour plot of the T1 segmentation (red = gray matter, green = white matter) overlaid onto the corrected and registered image, showing that the network produces an undistorted image that can be registered to a structural scan with high accuracy using a single transformation matrix for the entire volume.

Fig. 8 shows the results for a different subject. Again, the first row (a-c) depicts sagittal slices of the uncorrected and corrected up/down acquisition. In d, an axial slice of the T1 scan is displayed with manual

![Image](https://example.com/image.png)

**Fig. 5.** Displays the evolution of the training and test loss as a function of training iterations. Only the mean squared error (MSE) for the full resolution is displayed. The losses for lower resolution maps behave in a similar way. For $N_{\text{train}} = 6354$ and $N_{\text{test}} = 815$ we see a sharp decrease in the training and test for the first 20 epochs with slower progress for the following epochs. Naturally, the test error is slightly higher than the training error with both metrics reaching a plateau after around 100 epochs. The black dashed line represents the error that was calculated from the topup corrected test dataset. Our network surpasses this loss after around $N = 20$ epochs. The learning curves for two different VDM regularization parameters $\lambda$ are plotted in a and b with $\lambda = 1$ and 0.1, respectively.
segmentation of the lateral ventricle and the affected region due to a diagnosed Glioblastoma multiforme. In e, the outline of the manual segmentation is copied onto the corresponding slice of the corrected and registered DWI volume, showing excellent co-registration.

3.3. Generalization of the learned model

The network was trained exclusively on axial images with up/down encoding and only 3 vol with b-values > 0. Fig. 9 displays the results for axial cases with left-right encoding and for coronal slice orientations.

None of these acquisition settings were used during the training phase. Row a and b display the results for left-right encoding for b-values of 0 and 1000, respectively. The results for an EPI acquisition with coronal orientation are shown in row c and d (different b-values and slices, same subject).

4. Discussion

We have shown that we can train a deep convolutional network to estimate a distortion map from a pair of blip up/down input images. We...
are using a learning approach where we update the network parameters to minimize a similarity metric between the corrected up/down pair. This approach does not require actual ground truth distortion maps and allows training with a relatively small number of images assuming that they span the expected space of possible distortions. We are using one network architecture with a fixed input size of 128x128 which

Fig. 8. Single subject with ventricular/periventricular brain tumor (glioblastoma multiforme, or GBM). Sagittal slices that depict the uncorrected blip up and down scans in a and b. The corrected and averaged volume is shown in c. The corrected volume was registered to an anatomical T1 scan where the left lateral ventricle and a Glioblastoma multiforme were outlined (d). In e, the corresponding slice of the corrected and registered DWI volume is shown with the mask copied from d.

Fig. 9. Application to images with left-right (L–R) encoding and coronal orientation with and without diffusion weighting. The top two rows are axial images from human connectome data (HCP) from a Siemens scanner, while the bottom two were locally collected coronal images on a GE scanner. The first two columns show the distorted images (same subject, different slices). The third column displays the sum-of-squares combination of the output of the neural network. The difference maps in percent of the maximum signal intensity are shown in columns four (difference of the corrected output) and five (raw difference of the input images).
corresponds to 1.7 mm² voxel size for a typical FOV of 220 mm. DWI acquisitions with a different matrix size are re-sampled to a 128x128 grid in a pre-processing step. The estimated displacement maps are then re-sampled to the original input resolution and or transformed to an off-resonance map in [Hz] that is used in a separate correction step or directly during off-resonance corrected reconstruction from k-space. Our training dataset is composed of up/down acquisitions with a wide range of imaging parameters, and 5% of the data were acquired with diffusion weighting (b > 0) from healthy subjects. Our test datasets also include pathologies (see Table 1) that are unique for each subject and were therefore not seen by the network during the training phase. Figs. 6 and 7 demonstrate that the learned mapping of the network is able to generalize to unknown pathologies. From lesion-free brains, the network learns how to handle air-tissue interfaces, regions with varying contrast and signal intensities of arbitrary shape, etc. As long as the encoding time along the multi-band direction is sufficiently short (e.g. 1–2 ms), which is the case for state-of-art blipped-CAIPI-EPI sequences, it is safe to ignore if neighboring kz samples. Limiting this time to a few lines of the EPI train, correction therefore only happens during the time it takes to acquire blipped-CAIPI-EPI can be fully explained in terms of a 3d k-space with along the multi-band direction is sufficiently short (e.g. 1–2 ms), which is the case for state-of-art blipped-CAIPI-EPI sequences, it is safe to ignore if neighboring kz samples. Limiting this time to a few lines of the EPI train, correction therefore only happens during the time it takes to acquire blipped-CAIPI-EPI can be fully explained in terms of a 3d k-space with along the multi-band direction is sufficiently short (e.g. 1–2 ms), which is the case for state-of-art blipped-CAIPI-EPI sequences, it is safe to ignore if neighboring kz samples. Limiting this time to a few lines of the EPI train, correction therefore only happens during the time it takes to acquire

### Table 1

| ID  | matrix size | resolution [mm] | vendor/ |
|-----|-------------|-----------------|---------|
| 0001 | 180x180x36  | 1.2x1.2x2       | GE/3T  |
| 0002 | 140x140x85  | 1.7x1.7         | GE/3T  |
| 0003 | 128x128x60  | 1.6x1.6         | GE/3T  |
| 0004 | 128x128x72  | 1.8x1.8         | GE/3T  |
| 0005 | 180x180x36  | 1.2x1.2         | GE/3T  |
| 0006 | 128x128x72  | 1.8x1.8         | GE/3T  |
| 0007 | 96x96x48    | 2.2x2.2         | GE/3T  |
| 0008 | 128x128x72  | 2 x 2           | GE/3T  |
| 0009 | 128x128x74  | 2 x 2           | GE/3T  |
| 0101 | 128x128x74  | 1.95x1.95       | GE/3T  |
| 0102 | 100x100x74  | 2 x 2           | GE/3T  |
| 0103 | 128x128x74  | 1.95x1.95       | GE/3T  |
| 0104 | 256x256x80  | 1 x 1           | GE/3T  |
| 0105 | 128x128x72  | 2 x 2           | GE/3T  |
| 0106 | 128x128x72  | 2 x 2           | GE/3T  |
| 0107 | 128x128x72  | 2 x 2           | GE/3T  |
| 0108 | 128x128x72  | 2 x 2           | GE/3T  |
| 0109 | 128x128x72  | 2 x 2           | GE/3T  |
| 0110 | 128x128x72  | 2 x 2           | GE/3T  |
| 0111 | 128x128x72  | 2 x 2           | GE/3T  |
| 0112 | 256x256x40  | 0.78x0.78       | GE/3T  |

| TR  | b-value [ms] | acq | bw/pix | multi- | #     |
|-----|--------------|-----|--------|--------|-------|
| 0001 | 3000         | 57  | 17.54  | 1      | normal|
| 0002 | 4000         | 49  | 20.41  | 1      | normal|
| 0003 | 4300         | 50  | 20.00  | 1      | normal|
| 0004 | 9750         | 45  | 22.22  | 1      | normal|
| 0005 | 3000         | 57  | 17.54  | 2      | normal|
| 0006 | 2500         | 42  | 23.81  | 3      | normal|
| 0007 | 2500         | 42  | 23.81  | 3      | normal|
| 0008 | 8000         | 44  | 22.73  | 1      | normal|
| 0009 | 8000         | 44  | 22.73  | 1      | normal|
| 0101 | 7500         | 44  | 22.73  | 1      | normal|
| 0102 | 3500         | 53  | 18.87  | 4      | normal|
| 0103 | 9300         | 44  | 22.73  | 2      | normal|
| 0104 | 8000         | 44  | 22.73  | 1      | normal|
| 0105 | 8000         | 57  | 17.54  | 1      | normal|
| 0106 | 9500         | 44  | 22.73  | 1      | normal|
| 0107 | 8000         | 44  | 22.73  | 1      | normal|
| 0108 | 8000         | 45  | 22.22  | 1      | normal|
| 0109 | 8000         | 45  | 22.22  | 1      | normal|
| 0110 | 8000         | 45  | 22.22  | 1      | normal|
| 0111 | 8000         | 45  | 22.22  | 1      | normal|
| 0112 | 8000         | 45  | 22.22  | 1      | normal|

| Diagnosis                        | fig. |
|----------------------------------|------|
| Normal                           |      |
| Cavernous malformation           | 6    |
| Essential tremor                 | 7    |
| Glioblastoma multiforme          | 8    |
| Intracran. pressure sup.         | 9    |
| Coronial                         | 9    |
from axial images with AP encoding to images with left-right encoding and coronal orientations. A next iteration of the network could be trained on a dataset that includes this acquisition modes and thereby making it even more robust.

4.2. Limitations

The de facto standard to calculate a distortion map from an up/down pair consists of an iterative multi-resolution method to minimize a difference metric between input pairs (similar to our metric). We chose to compare our approach to the widely used topup method (Andersson et al., 2003) with default parameters. These parameters are also used in our clinical setting on patients and work reliably. By manually optimizing the correction parameters, either on a subject or a group basis, one can very likely further reduce the topup loss (dashed line in Fig. 5) to levels that are closer to our network loss, and for a single subject, topup could even outperform our approach. For a larger set of subjects, however, our network is able to learn from a wide variety of different images and therefore able to generalize to unseen cases. The main advantage of our approach is processing speed since the displacement map is calculated by a single forward-pass through the network with shared information between resolution stages instead of an iterative series of resampling and update operations. A different correction software like DR-BUDDI (Irfanoglu et al., 2015) might yield slightly different results when compared to our approach. An overview on existing methods, e.g., (Holland et al., 2010; Irfanoglu et al., 2015; Morgan et al., 2004; Ruthotto et al., 2013), for susceptibility corrections in EPI can also be found in Gu and Eklund (2019). These methods are typically highly optimized and much faster than topup which reduces the processing speed advantage of our method to some degree. However, we would like to point out again that our network learned to correct images from an internal feedback loop and does not rely on ground truth data provided by another method.

4.3. Future work

Our current implementation is strictly 2d with each slice of a volume treated as an independent training example. If the slice direction of a DWI acquisition is sampled sufficiently dense, one could think of using 3d convolutions that would allow the network to exploit correlations and minimize TV along the slice direction due to the smoothness of the distortion field. It would also be possible to include a spatial transformer module to account for head motion between the up and down acquisition like shifts and other affine transformations. The output of the network would then not be a displacement map with one scalar per voxel but three orthogonal displacement components for each voxel, i.e. x, y, z. The more complex output might require extra layers to increase the network capacity and therefore require additional training data to prevent overfitting. It would also be necessary to replace the 1d cubic interpolation with a bi-linear/cubic interpolation to allow for shifts and displacement in arbitrary directions. Since the number of trainable parameters in a neural network with 3d convolutions is not necessarily much higher than in the 2d case, the feasibility of this approach is most likely not limited by computation time considerations. Future work will also examine the ability of the network to successfully correct eddy current induced distortions if a bilp up/down pattern is played out for every diffusion direction. The expectation is that the distortion corrected dataset is inherently eddy current corrected since each VDM contains dynamically changing contributions to the effective off-resonance map like eddy currents or changes due to breathing. It also remains to be further studied how well the network will perform in the low SNR regime of high b-values without additional training; more problem specific training cases or a modified network architecture might be needed to perform well under these circumstances.

5. Conclusion

By using a deep convolutional network, we can bring down the processing time to a few seconds per volume, which is significantly faster than iterative approaches and can enable clinical implementation. This facilitates the use of a bilp up/down scheme for all diffusion weighted acquisitions and potential real-time EPI distortion correction without sacrificing accuracy.

Author contribution

Benjamin Zahneisen is the main contributor to this manuscript. All co-authors equally contributed to this manuscript.

Acknowledgements

This project was supported by Stanford in-House funding.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.neuroimage.2020.116886.

References

Andersson, J.L., Richter, M., Richter, W., Skare, S., Nunes, R., Behrens, T., 2004. Effects of susceptibility distortions on tractography. In: ISMRM Twelfth Scientific Meeting, p. 87.
Andersson, J.L.R., Skare, S., Ashburner, J., 2003. How to correct susceptibility distortions in spin-echo echo-planar images: application to diffusion tensor imaging. Neuroimage 20, 870–888. https://doi.org/10.1016/S1053-8119(03)00336-7.
Bowtell, R.W., McIntyre, D.J., Commandre, M.-J., Glover, P.M., 1994. Correction of geometric distortion in echo planar images. In: Proc. 2nd Meet. Soc. Magn. Reson, 311.
Chang, H., Fitzpatrick, J., 1992a. A technique for accurate magnetic resonance imaging in the presence of field inhomogeneities. IEEE Trans. Med. Imag. 11, 319–329.
Chang, H., Fitzpatrick, J.M., 1992b. A technique for accurate magnetic resonance imaging in the presence of field inhomogeneities. IEEE Trans. Med. Imag. 11, 319–329.
Chetelat, F., 2015. Keras. GitHub Repos.
de Vos, B.D., Berendse, F.F., Viergever, M.A., Staring, M., Isgum, I., 2017. End-to-End Unsupervised Deformable Image Registration with a Convolutional Neural Network. https://doi.org/10.1007/978-3-319-67558-9_24.
Dosovitskiy, A., Springenberg, J.T., Tatschendorf, M., Brox, T., 2017. Learning to generate chairs, tables and cars with convolutional networks. IEEE Trans. Pattern Anal. Mach. Intell. 39, 692–705. https://doi.org/10.1109/TPAMI.2016.2567384.
Embleton, K.V., Haron, H.A., Morris, D.M., Ralph, M.A.L., Parker, G.J.M., 2010. Distortion correction for diffusion-weighted MRI tractography and fMRI in the
temporal lobes. Hum. Brain Mapp. 31, 1570–1587. https://doi.org/10.1002/hbm.20959.

Fischer, P., Dosovitskiy, A., Ilg, E., et al., 2015. FlowNet: learning optical flow with convolutional networks. Iccv 8. https://doi.org/10.1109/ICCV.2015.316.

Graham, M.S., Drobnjak, I., Jenkinson, M., Zhang, H., 2017. Quantitative assessment of the susceptibility artefact and its interaction with motion in diffusion MRI. PloS One 12, 1–25. https://doi.org/10.1371/journal.pone.0185647.

Gu, X., Eklund, A., 2019. Evaluation of six phase encoding based susceptibility distortion correction methods for diffusion MRI. bioRxiv. https://doi.org/10.1101/766139.

Hochreiter, S., Frasconi, P., 2009. Gradient flow in recurrent nets: the difficulty of learning LongTerm dependencies. In: A F. Guid. to Dyn. Recurr. Networks. https://doi.org/10.1109/9780470544037.ch14.

Holland, D., Kuperman, J.M., Dale, A.M., 2010. Efficient correction of inhomogeneous static magnetic field-induced distortion in Echo Planar Imaging. Neuroimage 50, 175–183. https://doi.org/10.1016/j.neuroimage.2009.11.044.

Ioffe, S., Szegedy, C., 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. https://doi.org/10.1109/42.896788.

Jezzard, P., 2012. Correction of geometric distortion in fMRI data. Neuroimage 62, 648–651. https://doi.org/10.1016/j.neuroimage.2011.09.010.

Kingma, D.P., Ba, J., 2015. Adam: a method for stochastic optimization. In: 3rd International Conference for Learning Representations, pp. 1–15. https://arxiv.org/abs/1412.6980.

Maas, A.L., Hannun, A.Y., Ng, A.Y., 2013. Rectifier nonlinearities improve neural network acoustic models. In: Proc. 30 th Int. Conf. Mach. Learn, 28, p. 6.

Mansfield, P., 1977. Multi-planar image formation using NMR spin echoes. J. Phys. C Solid State Phys. 10, 55–58.

Morgan, P.S., Bowtell, R.W., McIntyre, D.J.O., Worthington, B.S., 2004. Correction of spatial distortion in EPI due to inhomogeneous static magnetic fields using the reversed gradient. Method 507, 499–507. https://doi.org/10.1002/jmri.20032.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional Networks for Biomedical Image Segmentation, pp. 1–8. https://doi.org/10.1007/978-3-319-24574-4_28.

Ruthotto, L., Mohammadi, S., Heck, C., Modersitzki, J., Weiskopf, N., 2013. Hyperelastic Susceptibility Artifact Correction of DTI in SPM. Springer, Berlin, Heidelberg, pp. 344–349. https://doi.org/10.1007/978-3-642-36480-8_60.

Shan, S., Guo, X., Yan, W., Zhang, E.I.-C., Fan, Y., Xu, V., 2017. Unsupervised end-to-end learning for deformable medical image registration, 14, 1–12.

Smith, S.M., Jenkinson, M., Woolrich, M.W., Beckmann, C.F., Behrens, T.E., Johansen-Berg, H., Bannister, P.R., De Luca, M., Drobnjak, I., Flitney, D.E., Niazy, R.K., Saadenn, J., Vickers, J., Zhang, V., De Stefano, N., Brady, J.M., Matthews, P.M., 2004. Advances in functional and structural MR image analysis and implementation as FSL. Neuroimage 23 (Suppl. 1), S208–S219. https://doi.org/10.1016/j.neuroimage.2004.07.051.