Deep-Learning Based Cortical Surface Reconstruction: A Review of Methods, Comparisons, and Recommendations

William Ashbee[a], Linlin Lu[a], Harshvardhan Gazula[b], Noah Lewis[a], Satrajit Ghosh[b], Vince D Calhoun[a], Sergey Plis[a]

[a] *TReNDS Center, Georgia State, Georgia Tech, Emory, Atlanta, GA, USA*
[b] *MIT, Cambridge, MA, USA*

Abstract

Accurate reconstruction of the brain's cortical surface is essential for neuroimaging analyses, enabling precise measurement of cortical structures and supporting studies of neurological conditions. Recent advances in deep learning have significantly improved the accuracy and efficiency of cortical surface reconstruction, offering promising alternatives to traditional methods. In this review, we analyze current deep learning approaches for cortical surface reconstruction, focusing on their design elements and evaluating their performance in terms of accuracy, surface quality, and computational efficiency. Our analysis reveals that some methods achieve high geometric accuracy at the expense of increased computational resources, while others maintain structural integrity with lower resource demands but may sacrifice some accuracy. By systematically evaluating these trade-offs, we identify methods that effectively balance accuracy, structural integrity, and resource efficiency. Our findings indicate that mesh-based methods with optimized loss functions offer the best compromise for most applications. This paper not only benchmarks recent deep learning methods for cortical surface reconstruction but also uncovers underlying design principles that influence performance. We provide actionable guidance on selecting appropriate approaches based on specific needs, highlighting the importance of considering both precision and computational practicality. By emphasizing the interplay between accuracy, resource consumption, and surface quality, we

*Corresponding author
Email addresses: wsashbee@gmail.com (William Ashbee), pku600gt@gmail.com (Linlin Lu), hgazula@mit.edu (Harshvardhan Gazula), lhd231@gmail.com (Noah Lewis), satra@mit.edu (Satrajit Ghosh), vcalhoun@gsu.edu (Vince D Calhoun), s.m.plis@gmail.com (Sergey Plis)

provide a valuable resource for researchers and clinicians in neuroimaging, neuroscience, and related fields.

Keywords: Deep Learning, Cortical Surface Reconstruction, Benchmarking, Topology, Neuroanatomy, sMRI

1. Introduction

Understanding the intricate structures of the human brain is fundamental to advancing neuroscience and developing treatments for various neurological disorders. Cortical Surface Reconstruction (CSR) from structural magnetic resonance imaging (sMRI) plays a pivotal role in this endeavor by enabling detailed analysis of cortical morphology, including measurements of cortical thickness, surface area, and folding patterns [28, 36]. Accurate reconstruction of cortical surfaces facilitates studies on brain development, neurodegenerative diseases, and cognitive functions, providing critical insights into the brain's architecture and its variations across individuals [14, 31].

FreeSurfer [13] has long been considered the gold standard for CSR, providing reliable reconstructions that serve as ground truth for training deep learning models [26, 20, 34, 23, 27, 6, 10]. However, *FreeSurfer* is burdened by extended processing times and topological errors, such as self-intersections in the reconstructed surfaces [20, 18]. These limitations have spurred the development of new deep learning-based methods that aim to overcome these challenges.

Recent advances in deep learning have ushered in a variety of models specifically designed to improve both the efficiency and accuracy of cortical surface generation [26, 20, 34, 23, 27, 6, 10]. The landscape of deep learning-based CSR models can be broadly categorized into three approaches [6]: voxel-based, implicit surface-based, and mesh-based methods. Voxel-based approaches typically involve segmenting the sMRI data followed by surface extraction [18, 16] to generate cortical surfaces. Implicit surface-based frameworks predict occupancy fields or signed distance function (SDF), which guide the surface extraction [10]. In contrast, mesh-based methods directly deform a template mesh to align with

the sMRI data, leveraging advanced techniques in geometric deep learning to achieve precise reconstructions [20, 23, 34, 6, 27, 26].

However, given the diversity of these approaches, there is no clear consensus on which model is best suited for specific tasks or datasets. Moreover, the architectural relationships and functional unit interactions within these models are underexplored, limiting the ability of researchers to make informed decisions when selecting models for their work. To address this gap, we provide a systematic analysis of recent deep learning models for CSR, examining their architectural differences, performance characteristics, and suitability for various applications.

This paper makes several key contributions. We evaluate the performance of recent deep learning models for CSR using various metrics such as distance to ground truth, self-intersections, inter-mesh collisions, and overall mesh quality, offering insights into their geometric and topological strengths and weaknesses. We also assess the computational performance of these models, including system memory requirements, processing time, and graphics processing unit (GPU) memory usage, providing practical information for selecting models based on available computational resources. Additionally, we conduct an in-depth examination of these models detailing their architectural choices and functional components including surface extraction methods, mesh deformation techniques, smoothing algorithms, loss functions, topology correction mechanisms, and neural network architectures. By elucidating the architectural relationships, performance characteristics, and key functional units of these models, this work serves as a valuable resource for researchers and clinicians aiming to leverage deep learning for cortical surface reconstruction.

The remainder of this paper is organized as follows. In Section 2, we define the core problem of cortical surface reconstruction and review representative deep learning-based approaches for CSR. Section 3 describes the methodology used for our comparative analysis. Section 4 presents the results of our evaluation, including mesh quality and computational performance measures. In Section 5, we discuss the functional units of CSR frameworks and provide guid-

ance on model selection. Finally, Section 6 concludes the paper with a summary of our findings and suggestions for future work.

2. CSR: Problem Statement and Representative Approaches

2.1. Problem Statement

Cortical surface reconstruction from brain imaging predominantly employs structural, T1-weighted sMRI images to generate surfaces, represented as triangular meshes, reflecting the boundaries between white and gray matter, and gray matter and cerebrospinal fluid (CSF), called the white and the pial surface, respectively [13, 28]. The goal is to ensure that these surfaces do not intersect, have no self-intersections, and form contiguous surfaces, while capturing the underlying structure of the brain.

Given a three-dimensional structural sMRI image denoted by $\mathbf{I} \in \mathbb{R}^{H \times W \times D}$, we aim to define a transformation ψ that yields a triangular mesh $\mathcal{M}$:

$$\psi : \mathbf{I} \rightarrow \mathcal{M}, \tag{1}$$

where H, W, and D correspond to the height, width, and depth of the image, respectively.

The triangular mesh $\mathcal{M}$ is described as a set of vertices $\mathbf{V}$, edges $\mathbf{E}$, and faces $\mathbf{F}$, expressed as $\mathcal{M} = \{\mathbf{V}, \mathbf{E}, \mathbf{F}\}$. Each edge $e \in \mathbf{E}$ is a line segment characterized by its endpoints $\mathbf{u}, \mathbf{v} \in \mathbf{V}$. Similarly, each face $f \in \mathbf{F}$ is defined by an ordered triplet $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbf{V}$, forming a triangular face that is one of the basic units of the mesh surface.

The transformation ψ is achieved by first preprocessing an image using a function ρ^* (where * indicates the function is optional or can be the identity transformation), then processing it through a neural network ω parameterized by $\boldsymbol{\theta}$. Finally, the neural network's output is post-processed with a function Δ^* to produce the final mesh. Typically, the function Δ^* involves Laplacian smoothing, although other post-processing functions can also be applied. Thus,

ψ is a composition of these three transformations, which can be expressed as:

$$\psi(\mathbf{I};\boldsymbol{\theta}) = \Delta^*(\omega(\rho^*(\mathbf{I});\boldsymbol{\theta})) \tag{2}$$

Within this framework, researchers train the neural network ω to ascertain the relationship between the preprocessed 3D image and the mesh. Alternatively, ψ could encapsulate the operation of the *FreeSurfer* pipeline [13], which the deep learning frameworks are attempting to learn and replace.

2.2. Representative Approaches

In this section, we review representative deep learning-based approaches for CSR providing an overview of their architectural foundations. We review several seminal models that execute modules similar to the *FreeSurfer* CSR pipeline in a fraction of the time.

2.2.1. DeepCSR

DeepCSR[1] [10] is an implicit surface framework that uses topology correction and marching cubes [25] for cortical surface generation. It was one of the first deep-learning-based methods for this task. The pipeline involves registering input sMRI and ground truth surfaces to an MNI152 T1 1mm template, using the sMRI as input to predict an implicit surface. After topology correction, marching cubes are applied to preserve topology.

The implicit surface $\mathcal{S}$ is defined as:

$$\mathcal{S} = \{\mathbf{p} \in \mathbb{R}^3 \mid d_{\mathcal{S}}(\mathbf{p}) = \ell\}. \tag{3}$$

where $d_{\mathcal{S}}(\mathbf{p})$ is the signed distance function (SDF), indicating whether a point is inside or outside $\mathcal{S}$.

The use of implicit functions for surface representation in DeepCSR aligns with the approach pioneered by DeepSDF [29], which demonstrated the effectiveness of modeling 3D surfaces as continuous signed distance fields to capture

[1] Cortical surface reconstruction via deep learning

fine geometric details and overcome the limitations of discrete voxel-based methods.

A neural network ω predicts $d_{\mathcal{S}}(\mathbf{p})$ using points $\mathcal{P}_{\text{sMRI}}$ sampled from the sMRI space during training. Inference involves uniform sampling within a fixed bounding box, a simpler approach compared to training.

For preprocessing, occupancies and distances are computed for $\mathcal{P}_{\text{sMRI}}$ using occupancy grids and the SDF. The grids help identify surface boundaries for marching cubes [10, 25], and distances are calculated with nearest-neighbor searches [4].

The architecture consists of an encoder that extracts features from the sMRI using convolutions and pooling, while a hypercolumn module interpolates activations to create dense feature representations. A decoder then uses these features to predict the implicit surface, employing conditional batch normalization (CBN) and fully connected layers.

Topology correction ensures the implicit surface is genus 0 [2] before marching cubes extracts the surface mesh. The framework uses binary cross-entropy (BCE) loss for occupancy fields and L_1 loss for SDF predictions.

2.2.2. TopoFit

TopoFit [20] is designed to reduce processing time and computational demands by integrating image and graph convolutions to deform a genus 0 mesh template into a subject-specific white matter surface. This approach combines the power of Graph Neural Networks (GNN) [35] and the UNet architecture [33] for efficient mesh adaptation.

TopoFit uses an image-based UNet to process sMRI data, which is then aligned with mesh coordinates using a Talairach transformation. This alignment enables the creation of image embeddings that capture white matter structure, which are passed to the graph-based UNet. The graph UNet operates on mesh vertex features (coordinates and normals) and predicts deformation vectors to adapt the mesh to the subject's surface.

The deformation process begins with a low-resolution icosphere template,

which is refined through seven iterative modules to achieve the final deformed mesh thereby ensuring accurate mesh adaptation to the complex geometry of the white matter surface.

To maintain smoothness and geometric consistency, *TopoFit* uses a hinge spring regularization term $\mathcal{L}_{\text{reg}}$ on adjacent face normals:

$$\mathcal{L}_{\text{reg}}(\widehat{\mathcal{M}}) = \frac{1}{|\mathbf{E}|} \sum_{(f_a, f_b) \in \mathbf{F}_{adj}} \left(1 - \widehat{\mathbf{u}}_{f_a}^T \widehat{\mathbf{v}}_{f_b}\right)^2 \tag{4}$$

where $\widehat{\mathbf{u}}_{f_a}$ and $\widehat{\mathbf{v}}_{f_b}$ are the normals of adjacent faces. This regularization promotes smooth mesh deformation.

Additionally, *TopoFit* uses a localized Chamfer distance $\mathcal{L}_{\text{dist}}$ to measure mesh similarity:

$$\mathcal{L}_{\text{dist}}(\mathcal{M}, \widehat{\mathcal{M}}) = \frac{1}{2|\mathbf{V}|} \sum_{\mathbf{v} \in \mathbf{V}} \min_{n \in \mathcal{N}_{\mathbf{v}}} ||\mathbf{v} - \widehat{\mathbf{v}}_n|| + ||\widehat{\mathbf{v}} - \mathbf{v}_n||, \tag{5}$$

which quantifies the average pairwise distance between mesh vertices. By combining $\mathcal{L}_{\text{dist}}$ and $\mathcal{L}_{\text{reg}}$, *TopoFit* ensures smooth, accurate mesh reconstructions with minimal self-intersections.

2.2.3. CorticalFlow

CorticalFlow [23] is a neural ODE-based approach for cortical surface reconstruction. It uses a 3D UNet [8] to predict a diffeomorphic flow field, which is then integrated numerically to deform a template mesh. This process ensures a smooth, homeomorphic transformation, preserving topology during deformation.

CorticalFlow leverages the preprocessing pipeline of *DeepCSR*, training in the MNI152 T1 1mm template space. This choice likely facilitates a fair comparison between the two frameworks. At inference, only minimal processing is required for the sMRI, as the ground truth surfaces from *FreeSurfer* are not needed.

The Diffeomorphic Mesh Deformer (DMD) in *CorticalFlow* uses Euler's ODE solver to integrate through the flow field, ensuring that the mesh remains topologically valid. The deformation is governed by the evolution of a

time-dependent coordinate map ϕ, described by the first-order ODE:

$$\frac{d\phi(t;\mathbf{V})}{dt} = \psi(\phi(t;\mathbf{V}),t), \text{ with } \phi(0;\mathbf{V}) = \mathbf{V}_0, \tag{6}$$

where $\mathbf{V}_0$ is the initial template mesh configuration, and ψ is the predicted vector field.

Training involves three UNets, with each model learning progressively finer deformations. During training, the models are updated sequentially: first training the coarsest model, freezing its weights, then training the next, and so on. The UNets are applied in sequence during inference, repeating three times.

The training loss includes Chamfer distance and edge length terms to ensure both accurate vertex positions and smooth mesh edges. The edge length loss for a mesh with vertices $\mathbf{V}$ and edges $\mathbf{E}$ is computed as:

$$L_{\text{edge}} = \frac{1}{|\mathbf{E}|} \sum_{e \in \mathbf{E}} \left(\|\mathbf{v}_i - \mathbf{v}_j\|_2 - l_{\text{target}} \right)^2 \tag{7}$$

CorticalFlow guarantees that the deformation remains diffeomorphic and invertible, preserving topology, provided the space is Lipschitz continuous and the numerical integration step size is sufficiently small. This ensures no topological defects during deformation [34, 26], though self-intersections can still occur, as noted by Zhao et al. [40], highlighting an area for further exploration.

2.2.4. CorticalFlow++

CorticalFlow++ [34] builds upon the *CorticalFlow* framework with three key improvements that enhance its robustness:

1. **ODE Solver Update**: The Runge-Kutta method replaces Euler's method for numerical integration, improving the stability and accuracy of the deformation.
2. **Template Mesh Refinement**: The initial template mesh is smoothed and optimized to ensure it includes all vertices from the training set, with no topological defects.
3. **White-to-Pial Surface Deformation**: *CorticalFlow++* enables the deformation from the white matter surface to the pial surface, providing explicit correspondences between the two surfaces.

In *CorticalFlow++*, correspondence between meshes implies a bijection, where each vertex in one mesh corresponds to a unique vertex in the other. This allows for the efficient computation of loss functions such as Mean Squared Error (MSE) or L_1, both of which have linear complexity $O(N)$, as opposed to Chamfer Distance's quadratic complexity $O(N^2)$. This efficiency is particularly useful for large datasets and simplifies the calculation of cortical thickness between the white and pial surfaces.

The template creation process in *CorticalFlow++* involves the following steps:

1. Compute the Signed Distance Function (SDF) for each surface mesh in the training set.
2. Generate an initial mesh using the marching cubes algorithm on the binary union of these SDFs.
3. Smooth the mesh with the Laplacian smoothing algorithm and apply Delaunay triangulation to obtain a genus-0 template mesh.

2.2.5. Vox2Cortex

Vox2Cortex [6] integrates convolutional neural networks (CNNs) and graph neural networks (GNNs) to process sMRI data and generate detailed cortical meshes. Unlike traditional methods that rely on voxel-based reconstruction or implicit surfaces, it avoids techniques like marching cubes and topology correction. The preprocessing pipeline registers images, segmentations, and meshes to MNI152 T1 1mm space, ensuring consistency between the image and the mesh. The iterative mesh refinement strategy used in *Vox2Cortex* is conceptually similar to Pixel2Mesh [38], which deforms an initial 3D mesh into a target shape using a graph-based convolutional approach, leveraging perceptual and geometric loss functions to ensure detailed and smooth surfaces.

The framework starts with a UNet for segmenting the sMRI scans to identify cortical structures, and then it deforms a predefined mesh template using cortical features extracted from the scans. The features from each layer of the UNet (across both down- and up-convolution layers) are combined and mapped

to mesh vertices, which serve as input to a GNN that predicts the final cortical surface. The GNN updates vertex features by aggregating information from neighboring vertices, following the update rule:

$$\mathbf{h}_i' = \frac{1}{1+|\mathcal{N}(i)|}\left[\mathbf{W}_0\mathbf{h}_i + \mathbf{b}_0 + \sum_{j\in\mathcal{N}(i)}(\mathbf{W}_1\mathbf{h}_j + \mathbf{b}_1)\right] \tag{8}$$

where $\mathbf{h}_i'$ is the updated feature for vertex i, and $\mathcal{N}(i)$ represents the neighboring vertices.

The loss function in *Vox2Cortex* consists of several components:

1. **Segmentation loss** $\mathcal{L}_{\text{vox}}$: Measures the difference between predicted and ground truth segmentation maps, computed using binary cross-entropy (BCE):

$$\mathcal{L}_{\text{vox}}(\widehat{\mathbf{B}}_{seg}, \mathbf{B}_{seg}) = \sum_{l=1}^{L}\mathcal{L}_{\text{BCE}}(\widehat{\mathbf{B}}_l, \mathbf{B}_l), \tag{9}$$

2. **Mesh loss** $\mathcal{L}_{\text{mesh}}$: Measures the difference between the predicted and ground truth meshes using consistency terms, including mesh-to-mesh distance, surface normal alignment, and regularization:

$$\mathcal{L}_{\text{mesh}}(\widehat{\mathcal{M}}, \mathcal{M}) = \mathcal{L}_{\text{mesh,cons}}(\widehat{\mathcal{M}}, \mathcal{M}) + \mathcal{L}_{\text{mesh,reg}}(\widehat{\mathcal{M}}), \tag{10}$$

   where $\mathcal{L}_{\text{mesh,cons}}$ combines curvature-weighted Chamfer distance and inter-mesh normal consistency.

3. **Curvature-weighted Chamfer loss** $\mathcal{L}_C$: Focuses on penalizing discrepancies between predicted and true mesh points, with additional weighting based on local curvature to emphasize areas of high detail:

$$\begin{aligned}\mathcal{L}_C(\widehat{\mathcal{M}}_{s,c}, \mathcal{M}_c) = &\frac{1}{|\mathcal{P}_c|}\sum_{\mathbf{u}\in\mathcal{P}_c}\kappa(\mathbf{u})\min_{\widehat{\mathbf{p}}\in\widehat{\mathcal{P}}_{s,c}}||\mathbf{u}-\widehat{\mathbf{p}}||^2 \\ &+\frac{1}{|\widehat{\mathcal{P}}_{s,c}|}\sum_{\widehat{\mathbf{p}}\in\widehat{\mathcal{P}}_{s,c}}\kappa(\tilde{\mathbf{u}})\min_{\mathbf{u}\in\mathcal{P}_c}||\widehat{\mathbf{p}}-\mathbf{u}||^2\end{aligned} \tag{11}$$

   where $\kappa(\mathbf{u})$ is the curvature at a given point $\mathbf{u}$, helping to guide the model's focus toward regions with complex surface features.

2.2.6. *CortexODE*

CortexODE [26] introduces a differential equation to model the evolution of cortical surfaces over time, similar to the *CorticalFlow* formulation of the initial value problem (Equation 6). It uses deformation techniques and provides proofs to minimize topological defects. Notably, *CortexODE* performs topology correction on the signed distance function (SDF) of the initial surface, ensuring the starting surface has genus 0.

The *CortexODE* pipeline begins by segmenting the surface into left and right hemisphere white matter regions using a 3D UNet, similar to *Vox2Cortex*. An SDF is computed for the segmentation map, and topology correction is applied to the discrete SDF after Gaussian blurring[2]. Marching cubes are then used to generate an initial surface, which is smoothed using Laplacian smoothing. This initial surface is converted into the white matter surface via the *CortexODE* neural network. The pial surfaces are generated by inflating and smoothing the white matter surface [26].

Like *CorticalFlow*, *CortexODE* employs neural ODEs to deform the initial surface during cortical surface reconstruction (CSR). However, unlike *CorticalFlow*, *CortexODE* uses a deformation network that is not a UNet, but instead has a localized receptive field focused on mesh vertices mapped into the sMRI. The deformation network relies on cube sampling, where features are extracted from the sMRI at the mesh vertex locations, reducing the computational cost compared to standard convolutional models like UNet. The sampled cubes are combined into a single matrix of size $|\mathbf{V}| \times C$, where C is the hidden unit size. This matrix is then passed through two fully connected layers to produce a deformation vector for each mesh vertex. These vectors are used to deform the mesh during the creation of the white and pial surfaces.

For training the white matter surface, *CortexODE* uses an initial surface generated by marching cubes and compares it to a ground truth white surface from *FreeSurfer*. For training the pial surface, *CortexODE* transforms the

[2] A custom Python multicore implementation using Numba [21] is provided [26]

FreeSurfer white surface by inflating and smoothing it, simulating real-world conditions where the white matter surface is available before the pial surface.

The inflation and smoothing process for generating the pial surface involves:

- **Inflation:** The white matter surface is inflated by adjusting vertex positions along their normal vectors, scaled by a predefined factor. This simulates the anatomical transition from the white matter to the outer cortical boundary (pial surface).

- **Smoothing:** Laplacian smoothing is applied to the inflated surface to remove local variations and noise, ensuring a smooth and continuous pial surface.

For white surface training, *CortexODE* uses the Chamfer distance loss $\mathcal{L}_{ch}$:

$$\mathcal{L}_{ch}(\widehat{\mathcal{M}}, \mathcal{M}) = \frac{1}{|\widehat{\mathbf{V}}|} \sum_{\widehat{\mathbf{v}} \in \widehat{\mathbf{V}}} \min_{\mathbf{v} \in \mathbf{V}} ||\widehat{\mathbf{v}} - \mathbf{v}||^2 + \frac{1}{|\mathbf{V}|} \sum_{\mathbf{v} \in \mathbf{V}} \min_{\widehat{\mathbf{v}} \in \widehat{\mathbf{V}}} ||\mathbf{v} - \widehat{\mathbf{v}}||^2, \qquad (12)$$

where $\widehat{\mathcal{M}}$ is the predicted white surface, $\mathcal{M}$ is the ground truth white surface, and $|\widehat{\mathbf{V}}|$ and $|\mathbf{V}|$ are the number of vertices in the predicted and ground truth surfaces, respectively.

For pial surface training, *CortexODE* uses mean squared error (MSE) $\mathcal{L}_{\text{mse}}$ between the predicted vertex locations $\widehat{\mathbf{V}}$ and the target vertices $\mathbf{V}$ of the ground truth pial surface $\mathcal{M}$. The use of MSE is feasible because the inflation and smoothing of the *FreeSurfer* white surface ensures a bijective mapping between the white and pial surface vertices.

2.2.7. PialNN

PialNN [27] is a partial cortical surface reconstruction framework that takes a white matter surface as input and produces the corresponding pial surface. It generates only one of the two cortical surfaces, requiring the missing surface (i.e., white matter) as input. This approach is useful for establishing mesh vertex correspondence between the two surfaces. However, *PialNN* is arguably the most incomplete framework reviewed here because it depends on both an

input white matter surface and an sMRI. The input white surface guides the algorithm's cube sampling, which significantly reduces the number of convolutions, making the number of operations proportional to the surface area rather than the sMRI volume. Inspired by approaches such as *Voxel2Mesh* [39], which generate 3D surface meshes directly from volumetric data using an end-to-end trainable architecture and iterative mesh refinement, *PialNN* adopts a similar methodology to accurately construct pial surfaces from MRI volumes.

The sMRI volume must be resampled to 256^3 voxels, with 1 mm isotropic resolution and 8-bit depth per voxel [3]. For training, *PialNN* requires both the white surface and ground truth pial surface, which are generated by *FreeSurfer*. For inference, *PialNN* requires only the white surface, also generated by *FreeSurfer*.

The key components of the *PialNN* architecture are a deformation block and cube sampling. The deformation block iteratively deforms the vertices of the input white mesh to better approximate the pial surface. This is achieved by computing the deformation vectors $\xi\mathbf{V}$ for each vertex and updating the vertex positions as follows:

$$\mathbf{V}_l = \mathbf{V}_{l-1} + \xi\mathbf{V}_{l-1} = \mathbf{V}_{l-1} + \omega(\mathbf{V}_{l-1}, \mathbf{N}_{l-1}, \mathbf{I}), \tag{13}$$

where $\mathbf{V}_l$ represents the updated vertex positions. The deformation $\xi\mathbf{V}_{l-1}$ is determined by the deformation network, which uses both localized convolutions and a multi-layer perceptron (MLP) $g_{\boldsymbol{\theta}_l}$. The network processes the previous vertex positions $\mathbf{V}_{l-1}$, vertex normals $\mathbf{N}_{l-1}$, and unprocessed sMRI data $\mathbf{I}$.

The network gathers features in two ways: (1) cube sampling to extract local sMRI features and (2) MLP-based point embeddings derived from the mesh vertex coordinates. These point embeddings are generated by processing the vertex coordinates and normals through fully connected layers with leaky ReLU activations. The point embeddings are then combined with image features obtained from cube sampling at multiple scales. These combined features are fed into fully connected layers, which predict the deformation for each vertex

[3] https://www.mail-archive.com/freesurfer@nmr.mgh.harvard.edu/msg64274.html

based on its local Cartesian coordinates and the 3D convolutional image features near each vertex.

The network uses three deformation blocks in total, followed by Laplacian smoothing to update each vertex, pulling it closer to the weighted average of its neighbors. Finally, the mean squared error (MSE) loss is computed. Since the connectivity of the input and target meshes is identical, MSE is preferred over the more computationally expensive Chamfer distance.

2.2.8. Other Recent Advances in Cortical Surface Reconstruction

During the preparation of this review, several new cortical surface reconstruction frameworks were introduced. While these frameworks are not included in our comparative analysis to maintain focus on the core aspects of CSR, it is important to acknowledge their noteworthy contributions.

The authors of *Vox2Cortex*, published two significant modifications to the original framework. In one modification, they developed subject-specific templates that achieve consistent surface correspondence over longitudinal studies involving a single subject [5]. In the other enhancement, a neural ODE to improve the performance of *Vox2Cortex*, analogous to approaches used in *CortexODE*, *CorticalFlow*, and *CorticalFlow++* was employed [7]. Additionally, *Vox2Cortex* was modified to incorporate an L_1 loss following surface registration for CSR and parcellation, thereby mitigating the limitations associated with Chamfer distance and adding class mappings for generated cortical surfaces [32].

To address the limitations of Chamfer distance, Wasserstein distance was employed to generate meshes with more uniformly shaped faces, resulting in improved mesh quality compared to those produced using Chamfer distance alone [22].

Ren et al. [30] introduced *FastCSR*, which employs the marching cubes algorithm with topology correction similar to that used in *DeepCSR*. Their implementation, which operates in approximately five seconds using a 3D U-Net, is an order of magnitude faster than *DeepCSR*, justifying the nomenclature *FastCSR*.

Zheng et al. [42] explore midpoint initial surface deformation between the

white and pial surfaces. Furthermore, they also proposed a cycle loss that minimizes the trajectories from the midpoint surfaces to both the white and pial surfaces, ensuring that the Chamfer distance between the midpoint surface and the white and pial surfaces remains equidistant [43].

In response to the growing popularity of transformer architectures, An *et al.* [1] integrated residual self-attention mechanisms for cortical surface reconstruction. Finally, the domain of cortical surface reconstruction has expanded to include subcortical structures with frameworks such as *Vox2Surf* [19] and *MeshDeform* [41]. These frameworks are capable of constructing surfaces for subcortical structures.

3. Methodology

By rigorously defining our evaluation criteria and experimental setup, we ensure that our comparison of cortical surface reconstruction methods is both valid and reproducible. This level of methodological transparency is essential for advancing the field, as it allows other researchers to replicate our study and build upon our findings. In this section, we present the methodology used to evaluate the mesh quality and computational performance of the frameworks. Detailed descriptions of the data and code, model training configuration, hardware setup, and containerization and code management are provided in Appendix A.

3.1. Mesh Quality Evaluation Methods

Evaluating the quality of the reconstructed cortical surfaces is crucial for determining the practical applicability of different methods. By employing a range of quantitative metrics including distance measures, self-intersections, and inter-mesh collisions, we provide a thorough assessment that highlights the strengths and limitations of each approach, thereby informing future developments in cortical surface reconstruction. These metrics provide an extensive assessment of the structural and topological accuracy of the generated surfaces.

3.1.1. Distance to Ground Truth

Accurately quantifying the deviation of reconstructed cortical surfaces from the ground truth is vital for assessing the performance of different reconstruction methods. By understanding these differences between different methods, we can improve reconstruction algorithms to better capture the true cortical geometry, ultimately enhancing the accuracy of neuroscientific research and clinical diagnostics.

To assess the accuracy of the predicted meshes, the distances between the predicted meshes and the ground truth meshes were calculated. Since the preprocessing pipelines of the frameworks involved slightly divergent registration techniques, it was necessary to align the meshes to a common space for precise distance calculations. The iterative closest point (ICP) algorithm was employed without necessitating alterations to the frameworks under review, along with scaling and rotation as required for *PialNN*. These procedures yielded a unified linear transformation matrix $\mathbf{T}$, facilitating the reversion of meshes to their original coordinate system.

Conceptually, this process involves four distinct meshes: the *FreeSurfer* ground truth mesh $\mathcal{M}_{\text{fs}}$ with vertices $\mathbf{V}_{\text{fs}}$, the framework-specific representation of the ground truth mesh $\mathcal{M}_{\text{framework}}$ with vertices $\mathbf{V}_{\text{framework}}$, the model output mesh $\widehat{\mathcal{M}}_{\text{framework}}$ with vertices $\widehat{\mathbf{V}}_{\text{framework}}$, and the framework-specific mesh prediction registered to *FreeSurfer* space $\widehat{\mathcal{M}}_{\text{fsreg}}$ with vertices $\widehat{\mathbf{V}}_{\text{fsreg}}$.

The transformation matrix $\mathbf{T}$ satisfies the equation:

$$\mathbf{V}_{\text{fs}} = \mathbf{T}\mathbf{V}_{\text{framework}} \tag{14}$$

Distances between the transformed meshes are computed as:

$$\delta = d_{\text{dist}}(\mathbf{T}\widehat{\mathbf{V}}_{\text{framework}}, \mathbf{T}\mathbf{V}_{\text{framework}}) \tag{15}$$

Chamfer distance represents the average value of the closest point distances between two point clouds, while Hausdorff distance denotes the maximum distance of the closest points between two point clouds. Both distances provide a

comprehensive view of the mesh comparisons.

3.1.2. Self-Intersections

Self-intersections in the reconstructed meshes can lead to inaccuracies in surface-based analyses and impair the calculation of cortical thickness and other morphological features. By detecting and quantifying self-intersections, we assess the topological correctness of each method, which is vital for ensuring the reliability of neuroimaging studies that utilize these surfaces.

The assessment of self-intersections employs a KDTree [3] to represent the centers of the mesh triangles. The algorithm iterates through the triangles, and for each center, the k nearest triangles are retrieved using the KDTree to enhance computational efficiency. Intersections are checked exclusively with these k nearest triangles, and a counter tallies the number of intersections detected.

Since self-intersections are invariant with respect to the coordinate system, no coordinate transformations are applied to the predicted meshes prior to calculating self-intersections. The medial wall is excluded from topology correction by default in *FreeSurfer*. To ensure a fair comparison, medial wall predictions were excluded from all evaluated models when calculating self-intersections. The exclusion was achieved by utilizing the aparc.a2009s brain atlas and the parcellation scheme proposed by Destrieux et al. [11].

3.1.3. Inter-Mesh Collisions

Inter-mesh collisions between the white and pial surfaces can result in ambiguous or invalid cortical thickness measurements. Evaluating the occurrence of such collisions allows us to determine the suitability of each method for applications that depend on accurate cortical thickness estimation, thereby impacting studies on brain development and disease.

Inter-mesh collisions were evaluated to assess the interactions between the white and pial meshes for each hemisphere. Similar to the self-intersection assessment, a KDTree was constructed, and the algorithm identified collisions between the white and pial meshes by checking for intersections between triangles.

These collisions delineate boundaries where cortical thickness is ambiguously defined. The medial wall was excluded here as it was for self-intersections.

3.2. Computational Performance Evaluation Methods

Assessing the computational efficiency of each framework is critical for their practical deployment in research and clinical settings. By measuring system memory usage, execution time, and GPU memory consumption, we provide insights into the resource demands of each method, which can influence their adoption based on available infrastructure.

To assess the computational efficiency of each framework, we measured the system memory, execution time, and GPU memory usage. These metrics are essential for understanding the resource requirements and scalability of the frameworks under review.

3.2.1. Minimum Required System Memory

Determining the minimum system memory required for each framework helps in understanding their scalability and suitability for different computational environments. This information is particularly important for researchers with limited hardware resources, ensuring that the methods can be effectively utilized across various settings.

To establish the minimal system memory requirements for each framework, the lowest stable amount of memory necessary for effective execution was determined. This was achieved by incrementally increasing the allocated memory in steps of 250 MiB, ranging from 250 MiB to 12,750 MiB over 50 runs. Each of the 50 runs for a given configuration was tested three times to ensure consistency.

3.2.2. Time Measures

Measuring the execution time of each method provides practical insights into their efficiency and throughput. Faster methods can significantly accelerate neuroimaging pipelines, enabling larger studies and more timely analyses, which is crucial in both research and clinical applications.

During the incremental search for minimal memory requirements, the execution time for generating a single surface for one subject was recorded. This measurement reflects the duration from the start to the completion of the program's execution. The same subject was used for each run to maintain consistency. Each memory allocation configuration was executed three times, totaling 150 runs, to mitigate the impact of hardware-induced latency from cold starts.

3.2.3. Peak GPU Memory Usage

Evaluating the peak GPU memory usage of each framework is essential for understanding their compatibility with available hardware, especially for institutions or researchers with varying levels of access to high-end GPUs. This metric can influence the feasibility of adopting certain methods in different computational environments.

Peak GPU memory usage was monitored during the execution of each framework to assess the maximum memory consumption required. This metric is crucial for understanding the scalability and resource demands of each framework under evaluation conditions.

4. Results

The results of the mesh quality and computational performance evaluations are presented in this section. All frameworks were evaluated using an identical test set comprising 107 HCP subjects selected from a total of 897 subjects.

4.1. Mesh Quality Measures

The distance plot in Figure 1 provides significant insights into the performance of various frameworks. Notably, *CorticalFlow++* demonstrates superior performance for pial surfaces. A cluster of highly successful frameworks includes *CorticalFlow*, *CorticalFlow++*, *TopoFit*, and *CortexODE_rk4*. According to the *CortexODE* framework, utilizing the Euler method would require more steps to achieve comparable success to *CortexODE* with the Runge-Kutta 4 (RK4) solver, which is evident in the results [26]. While *DeepCSR* exhibits challenges in

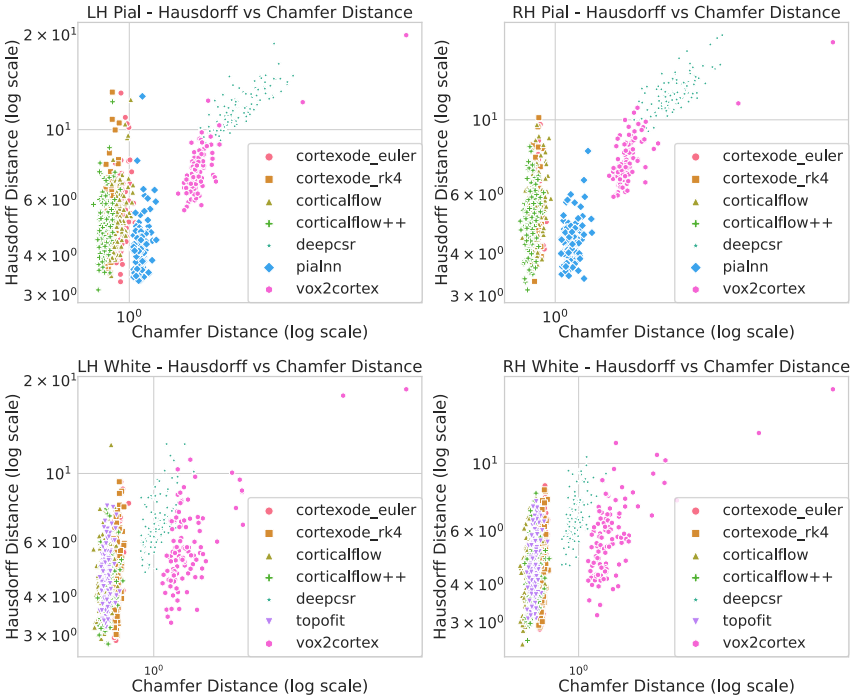

Figure 1: Distribution of distance measures for the test set predicted white and pial surfaces for both hemispheres. Hausdorff distance represents the maximum distance between corresponding points, while Chamfer distance represents the average distance between corresponding points.

generating competitive pial surfaces, it produces nearly competitive white surfaces. *Vox2Cortex* performs suboptimally compared to the state of the art and exhibits catastrophic failures in some instances. *PialNN* generates commendable but not exceptional pial surfaces, with significant improvements observed in the authors' implementation of *CortexODE* using the RK4 solver.

In Figure 2, the component closest point distances are visualized for a single subject. Dark green indicates the best performance, light green the second best, and purple the worst. This figure mirrors the findings in the aggregate measures presented in Figure 1. Increased occurrences of purple and errors are observed in *Vox2Cortex*, *PialNN*, and *DeepCSR*, whereas dark green surfaces are prevalent

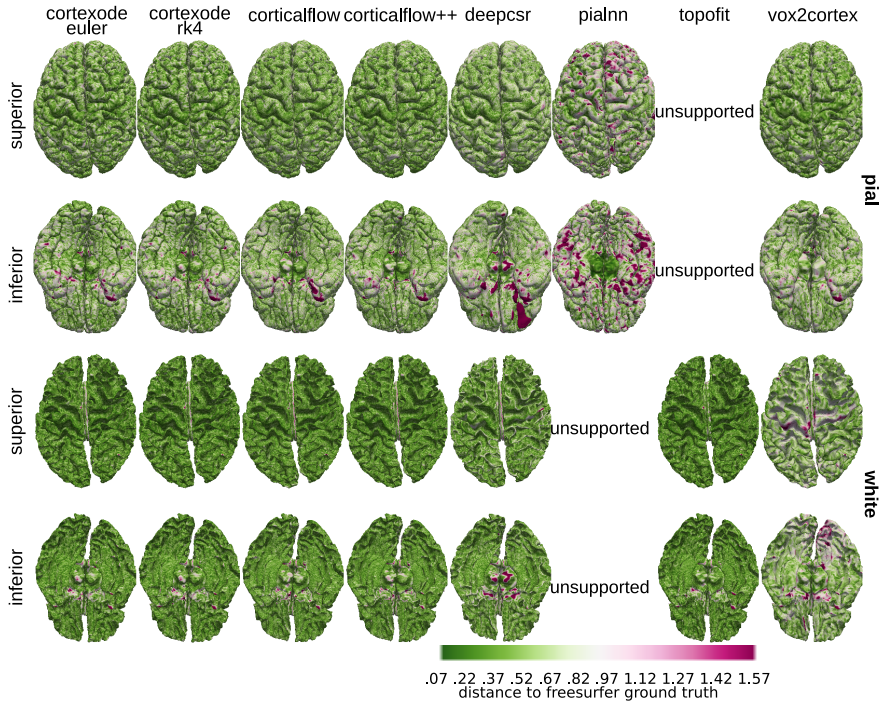

Figure 2: Closest point distances of framework-predicted surfaces to their ground truth for a single subject. Dark green indicates the best performance, light green indicates the next best, and purple indicates the worst performance. *PialNN* and *TopoFit* did not support both surface types at the time of their respective framework publications.

in the cluster of the most successful frameworks—*CortexODE* (Euler and RK4), *TopoFit*, *CorticalFlow*, and *CorticalFlow++*. Overall, the white surfaces exhibit better performance (darker green) than the pial surfaces, which is also reflected in the Chamfer versus Hausdorff plot.

The self-intersection distributions in Figure 3 depict the number of times triangles in the predicted meshes intersect with other triangles across the four surfaces: left and right hemispheres, white and pial. *PialNN* exhibits the highest topological error. While *CorticalFlow* does not perform optimally on pial surface topology given its performance in distance measures, *CorticalFlow++* significantly improves these metrics, potentially due to the use of better templates, the Runge-Kutta 4 solver, or deforming the white surface into a pial surface—likely

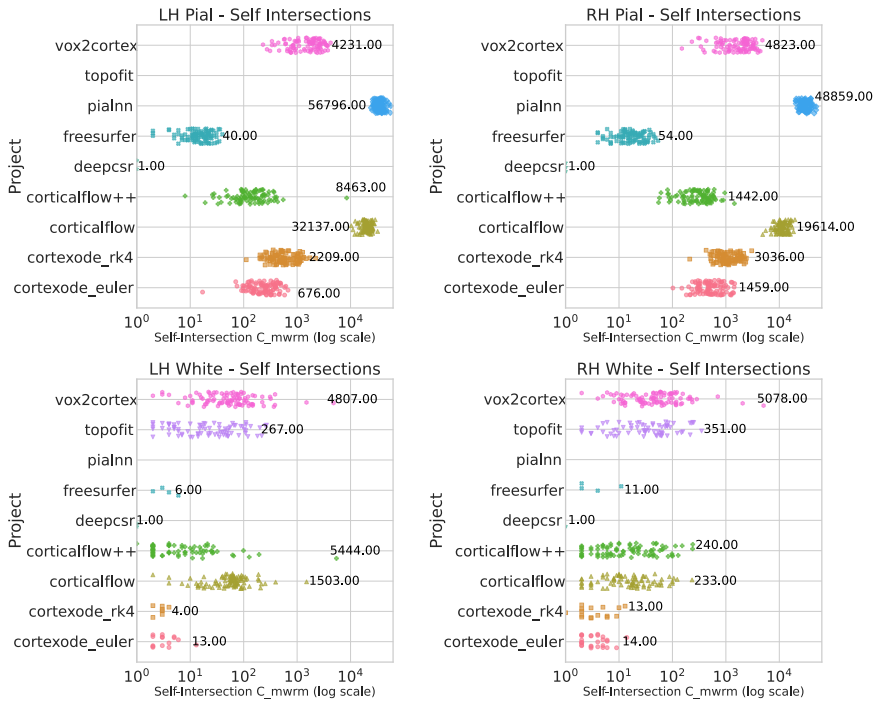

Figure 3: Distribution of collisions and self-intersections for predicted surfaces in the test set (higher values indicate greater severity). These defects can adversely affect geodesic distance measurements within the cortex due to their pervasive distribution. *FreeSurfer* serves as the ground truth, with its measurements provided for reference.

a combination of these factors. White surfaces appear to be generally easier to generate in terms of this measure. *DeepCSR* excels in avoiding self-intersections owing to its topology correction of the SDF and topology-preserving marching cubes algorithm. *CortexODE* produces excellent white surfaces but exhibits moderate performance on pial surfaces in terms of topology, indicating an area for potential improvement in future research. It is noteworthy that *FreeSurfer* also presents some self-intersections. Both *TopoFit* and *Vox2Cortex* perform reasonably well overall on their white surfaces, excluding some extreme outliers observed in *Vox2Cortex*.

It was observed that many defects occur in the sulci and gyri regions where

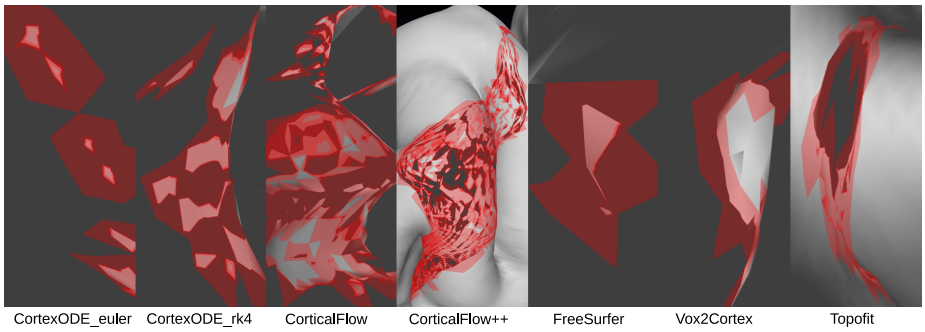

Figure 4: Visualizations and verification of self-intersections in the surfaces of the most affected subjects for selected frameworks. Red highlights indicate areas of self-intersection, light gray represents the outer face, and dark gray represents the inner face. The inversion of outer and inner surface colors within the red boundaries signifies self-intersection.

cortical folds arise, as illustrated in some of the failure cases in Figure 4, which shows self-intersections in red, outward faces in light gray, and inward faces in dark gray. Typically, defects result from either adjacent folds touching or a single fold crossing itself. The presence of collisions in a mesh can adversely affect certain types of analyses, such as geodesic distance calculations. Therefore, assessing the topological quality of these frameworks is crucial for researchers considering the incorporation of deep learning into their analysis pipelines. The measures provided in Figure 3 indicate that *DeepCSR* demonstrates exceptionally high quality in these aspects. This suggests that combining topology correction with deep learning yields superior results until faster solutions are developed.

For frameworks that produce both surfaces in each hemisphere, an investigation was conducted to determine whether the white and pial surfaces intersect, which ideally should not occur. Such white-pial intersections form boundaries where one surface protrudes into another, resulting in regions where cortical thickness is poorly defined. The measures in Figure 5 quantify the number of triangle intersections between the white and pial surfaces in each hemisphere. It was found that *CortexODE* outperforms other frameworks in these measures,

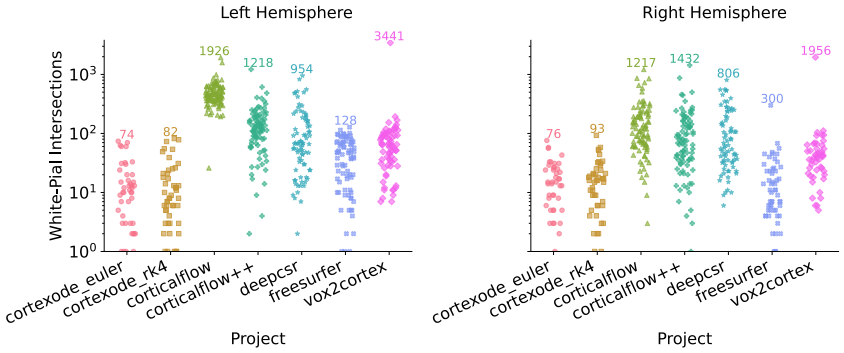

Figure 5: In cortical thickness measurements, white and pial surfaces should not intersect; however, this figure demonstrates that such intersections occasionally occur. The number of triangle-face intersections between these two surfaces is provided for both hemispheres. *FreeSurfer* serves as the ground truth, with its measurements included for reference.

with *FreeSurfer* being the next best. The worst-case outliers depicted in Figure 6 for *CorticalFlow*, *CorticalFlow++*, *DeepCSR*, and *Vox2Cortex* are significant enough to be considered macroscopic features. In the case of *Vox2Cortex*, a catastrophic failure was identified in a surface generated, as shown in Figure 6. Nonetheless, aside from this singular catastrophic failure in each hemisphere, *Vox2Cortex* performs reasonably well in these measures.

4.2. Computational Performance Measures

The computational benchmark results provide insights into the system and GPU memory requirements and the processing time per subject. The results for minimum system memory were generally reasonable across the frameworks, with the exception of *DeepCSR*, as shown in Figure 7A. All frameworks except *DeepCSR* consumed between 2.5 and 5 gigabytes of system memory, which is acceptable for server environments. For client-side deployment, resource availability may vary, and it is advisable to consider both system and GPU memory requirements when determining resource allocation. *DeepCSR* consumed approximately 20 gigabytes of system memory to generate all four surfaces—left and right hemisphere white and pial surfaces. It was observed that providing

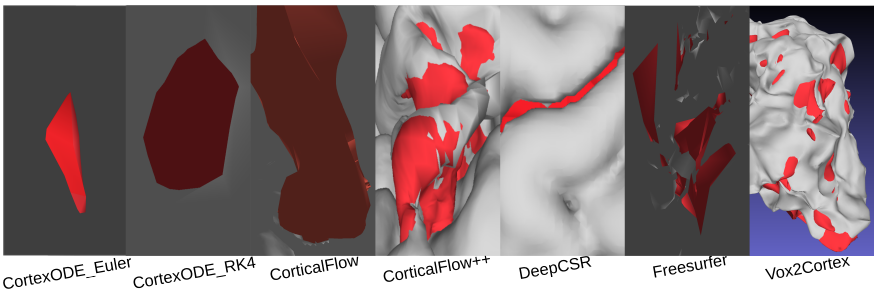

Figure 6: Visualization of interweaving between white and pial surfaces across all frameworks, which should be avoided. One of the two surfaces (white or pial) is colored red to highlight areas where a surface penetrates the other. Defects are located in the lowest quality surfaces for each framework, as indicated in Figure 5. Inter-mesh collisions in the cortex render cortical thickness measurements undefined in these regions (negative thickness values).

slightly less memory than required resulted in significantly slower performance without causing a crash. These findings can assist in deciding appropriate memory allocations for these frameworks. It is important to note that resolution changes can affect these numbers, particularly for *DeepCSR*.

The time measures presented in Figure 7B relate to the generation of cortical surfaces. Notably, *DeepCSR* emerges as the most time-consuming model at a resolution of 256, primarily due to its topology correction processes for implicit surfaces. Additionally, *DeepCSR* employs an iterative loop to sequentially generate sections of the surface during the construction of the implicit surface. Models incorporating ODE solvers, such as *CortexODE* and *CorticalFlow*, are slower at 30 steps compared to *CortexODE* with ten steps, although other architectural factors may also contribute to the performance differences. Their non-ODE counterparts, *PialNN* and *TopoFit*, which involve fewer sequential operations without ODE numerical integration, are slightly faster. Despite these variations, all frameworks achieve the objective of improving upon *FreeSurfer* in terms of speed, reducing processing time from hours to seconds or minutes.

As illustrated in Figure 7C, *DeepCSR* also has the largest GPU memory footprint, despite explicit calls to garbage collection during its iterative prediction of mesh subsections. This is partially attributable to *DeepCSR* generating

four surfaces simultaneously. In contrast, *TopoFit*, which generates only a single surface in this comparison (the left hemisphere white surface), consumes considerable memory (approximately 8 gigabytes). Other frameworks exhibit more modest memory requirements, with *Vox2Cortex* notably generating all four surfaces using approximately 4.5 gigabytes of GPU memory. *PialNN* demonstrates the lowest memory consumption among the measured frameworks for single surface generation.

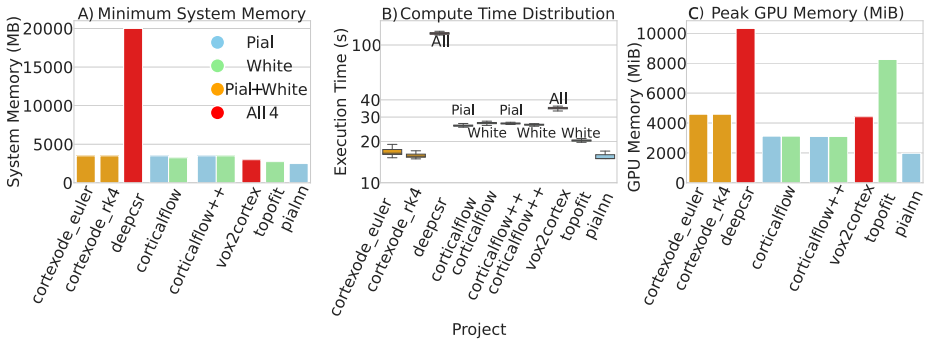

Figure 7: Resource requirements for selected frameworks are depicted in three measures: **A)** minimum system memory, **B)** execution time, and **C)** peak GPU memory usage. The color scheme denotes whether one (white or pial for the left hemisphere), two (white and pial for the left hemisphere), or four (all surfaces for both hemispheres) surfaces were generated. The number of surfaces generated varies based on design choices made by the original authors, which were not altered for this review.

The time measures presented in Figure 7B relate to the generation of cortical surfaces. Notably, *DeepCSR* emerges as the most time-consuming model at a resolution of 256, primarily due to its topology correction processes for implicit surfaces. Additionally, *DeepCSR* employs an iterative loop to sequentially generate sections of the surface during the construction of the implicit surface. Models incorporating ODE solvers, such as *CortexODE* and *CorticalFlow*, are slower at 30 steps compared to *CortexODE* with ten steps, although other architectural factors may also contribute to the performance differences. Their non-ODE counterparts, *PialNN* and *TopoFit*, which involve fewer sequential op-

erations without ODE numerical integration, are slightly faster. Despite these variations, all frameworks achieve the objective of improving upon *FreeSurfer* in terms of speed, reducing processing time from hours to seconds or minutes.

It was observed that *DeepCSR* also has the largest GPU memory footprint, despite explicit calls to garbage collection during its iterative prediction of mesh subsections—so much so that For *DeepCSR*, the memory allocation range was extended from 15,000 MiB to 28,500 MiB due to its higher resource demands. This is partially attributable to *DeepCSR* generating four surfaces simultaneously. In contrast, *TopoFit*, which generates only a single surface in this comparison (the left hemisphere white surface), consumes considerable memory (approximately 8 gigabytes). Other frameworks exhibit more modest memory requirements, with *Vox2Cortex* notably generating all four surfaces using approximately 4.5 gigabytes of GPU memory. *PialNN* demonstrates the lowest memory consumption among the measured frameworks for single surface generation.

5. CSR: A Guide on Functional Units and Their Use

CSR frameworks are composed of various functional units, each responsible for distinct aspects of the reconstruction process. These functional units include surface extraction methods, mesh deformation techniques, smoothing algorithms, loss functions, topology correction mechanisms, neural network architectures, and input modalities. Understanding and comparing these functional units across different frameworks is essential for evaluating their strengths, limitations, and suitability for specific applications. In this section, we present a comprehensive feature table that categorizes and compares the major CSR frameworks based on their implementation of these functional units, followed by guidance on major considerations for framework selection.

5.1. Functional Units

Our goal is to draw attention to the dominant techniques within the cortical surface reconstruction literature, as depicted in Table 1.

Table 1: Feature Table of Cortical Surface Reconstruction Frameworks

	CortexODE	CorticalFlow	CorticalFlow++	DeepCSR	PialNN	TopoFit	Vox2Cortex
Marching Cubes							
Primary Method				✔			
Initial surface	✔						
Mesh Deformation							
ODE Solver	✔	✔	✔				
Without Solver					✔	✔	✔
Smoothing							
Initial Surface	✔		✔				✔
Output Surface	✔				✔		
Loss term						✔	✔
Loss							
Chamfer distance	✔	✔	✔			✔Local	✔Curve
Other Distance	✔MSE			✔L_1	✔MSE		
Cross Entropy	✔			✔			✔
Inter/Intra Mesh Normal							✔
Edge Length		✔	✔				✔
Laplacian Loss							✔
Face Normals (Hinge Spring)						✔	
Topology Correction							
Implicit function (not mesh)	✔			✔			
Neural Net Architecture							
Graph Neural Networks						✔	✔
UNet	✔	✔	✔			✔	✔
Refinement Networks		✔	✔		✔		
Cube Sampling	✔				✔		
ODE Solver	✔	✔	✔				
Hypercolumn				✔			✔
Input							
sMRI Only	✔white	✔	✔white	✔		✔white	✔
sMRI + white surface	✔pial		✔pial		✔pial		

Marching Cubes: *DeepCSR* and *CortexODE* use marching cubes for initial surface extraction [10, 26]. *Vox2Cortex* moves away from marching cubes due to stair-step artifacts [6].

Mesh Deformation: Deformation frameworks like *CortexODE*, *CorticalFlow*, and *CorticalFlow++* integrate ODE solvers for continuous transformations [26, 23, 34]. *PialNN*, *TopoFit*, and *Vox2Cortex* employ mesh deformation without ODE solvers, relying on direct vertex updates [27, 20, 6].

Smoothing: Smoothing techniques are used to mitigate self-intersections and ensure smooth surfaces. Frameworks such as *PialNN*, *CortexODE*, and

Vox2Cortex incorporate smoothing at various stages [27, 26, 6].

These frameworks have the following smoothing strategies:

- *PialNN*: Applied to output surface.
- *CortexODE*: Used for the initial and output surfaces.
- *CorticalFlow++*: Employed during template creation.
- *TopoFit* and *Vox2Cortex*: Incorporated within the loss function.

The *TopoFit* and *FreeSurfer* related papers [20, 15, 14] highlight the inadequacy of some smoothing terms in optimization.

Loss Functions: Loss functions are crucial for training frameworks, with Chamfer distance being prominent in many [12]. *Vox2Cortex* uses a combination of Chamfer distance, edge length, and Laplacian loss [6]. Loss functions should ideally possess the following attributes [12]:

- Differentiability with respect to sampled points,
- Computational efficiency, and
- Robustness against outliers.

Chamfer distance is useful over short deformations, but requires additional smoothing terms over larger deformations [20, 26, 23].

MSE and L_1 losses provide a stricter structural constraint than chamfer and can be used without smoothing loss terms over short deformations especially if smoothing is applied to the mesh after it is generated [27, 26, 10].

Cross Entropy is used for preliminary segmentation tasks.

Additional loss terms ensure mesh quality:

- *Vox2Cortex* uses intra and inter-mesh vertex normals for mesh deformation consistency [6].
- *CorticalFlow* and *Vox2Cortex* use edge length loss to impose structural constraints [23, 6].

- *Vox2Cortex* employs Laplacian loss to encourage smooth deformations [6].
- *TopoFit* uses a face normal hinge spring term to maintain smooth surface normals [20].

Topology Correction: Topology correction ensures the precursor implicit surface representations are homeomorphic to a sphere. *DeepCSR* and *CortexODE* perform topology correction on implicit functions, not directly on meshes [10, 26].

Neural Net Architecture: Architectures include GNNs, UNets, refinement networks, cube sampling, and hypercolumns. Frameworks like *Vox2Cortex* and *TopoFit* leverage GNNs for mesh deformation [6, 20]. UNet architectures are employed for feature extraction in frameworks such as *CortexODE* and *TopoFit* [9, 33, 24, 6, 20, 23, 34, 26]. Refinement networks replicate weights to perform the same task multiple times at increasing levels of precision [27, 23, 34]. Cube sampling enables local receptive field in the sMRI [27, 26]. Hypercolumns behave like residual connections to preserve high frequency features [17, 10, 6, 7, 37].

Input: Some frameworks require only sMRI as input, while others require sMRI and existing white surface meshes. *CorticalFlow++* and *CortexODE* combine both approaches for different surface types [34, 26].

5.2. Recommendations

When choosing a CSR framework, understanding trade-offs between speed, topological accuracy, and loss function complexity is essential. Frameworks like *CorticalFlow++*, *CortexODE*, and *TopoFit* provide fast, high-accuracy reconstructions, suitable for applications needing quick results without sacrificing quality. For the highest topological accuracy, *DeepCSR* is recommended, though it requires longer processing times. *Vox2Cortex* explores advanced loss functions, such as curvature-weighted Chamfer distance, but the architecture itself did not perform well. Simpler frameworks like *PialNN* and *CortexODE* use straightforward loss functions with Laplacian smoothing, maintaining smooth

surfaces with less complexity, though only *CortexODE* can independently perform CSR for both surfaces without *FreeSurfer*.

- **Computational Resources**: Frameworks like *CorticalFlow++* are ideal for low-memory environments, especially with lower-resolution meshes. For projects with high computational resources, *CortexODE* and *TopoFit* provide finer mesh details and accuracy, while *DeepCSR* remains memory-intensive at high resolutions but is recommended where topological accuracy is a priority.

- **Desired Accuracy and Mesh Quality**: For studies demanding precise anatomical fidelity, frameworks like *CortexODE*, *CorticalFlow++*, and *TopoFit* with advanced loss functions and topology correction are ideal. For a balance between speed and accuracy, *CorticalFlow++* provides reliable reconstructions with efficient processing, making it suitable for general-purpose applications.

- **Model Complexity and Ease of Use**: *PialNN* offers the simplest implementation, though it relies on *FreeSurfer* for initial inputs. *CortexODE* extends this by enabling independent operation from *FreeSurfer*. Flexible and extensible options, such as *TopoFit*, *CortexODE*, and *CorticalFlow++*, allow customization for specific research needs. *DeepCSR* is well-documented and user-friendly, making deployment straightforward, with other frameworks also compatible with most workflows.

6. Conclusion

This review of deep learning-based CSR has revealed several key insights into the current state and future directions of the field. The emergence of diverse deep learning models has introduced various architectural approaches, yet determining which models are most effective in specific contexts remains an open question. Our analysis underscores the importance of leveraging high-dimensional features to enhance inference quality, as demonstrated by models

like *DeepCSR* with its hypercolumn features and the use of skip connections in UNet architectures. Additionally, the implementation of refinement networks across multiple stages, allowing for coarse-to-fine mesh generation, has proven effective in improving surface quality.

One of the significant findings from our results is that *CorticalFlow++* emerged as the leading framework for generating pial surfaces, achieving a superior balance between accuracy and computational efficiency. *CortexODE_rk4*, *TopoFit*, and *CorticalFlow* also exhibited strong performance, particularly in terms of surface accuracy and quality. However, *DeepCSR*, while showing commendable results for white surfaces and excelling in minimizing self-intersections due to its topology correction mechanisms, faced challenges with pial surfaces and required substantially more computational resources. Conversely, *Vox2Cortex* displayed several catastrophic failures, highlighting variability in model robustness and the need for further refinement.

The practical implications of these findings are significant for both research and clinical applications. By providing a detailed comparison of CSR frameworks, we equip researchers and clinicians with the knowledge to select the most appropriate tools based on specific needs such as computational resources, desired accuracy, and ease of integration into existing workflows. This guidance can streamline the decision-making process, optimize resource allocation, and potentially accelerate the adoption of deep learning methods in neuroimaging studies.

Our work contributes to the field by uncovering underlying design principles that influence CSR performance. By emphasizing the importance of architectural choices—such as the use of Graph Neural Networks (GNNs), loss function designs, and topology correction mechanisms—we offer actionable insights that can inform the development of next-generation CSR models. These insights can lead to improvements in cortical surface reconstruction accuracy, efficiency, and robustness, ultimately enhancing the quality of neuroimaging analyses.

In conclusion, deep learning-based CSR models have demonstrated great potential, often rivaling or surpassing traditional methods like *FreeSurfer* in terms

of speed and accuracy. The advances highlighted in this review not only facilitate a better understanding of the brain's architecture but also open avenues for more precise diagnosis and treatment of neurological conditions. By addressing the challenges of balancing precision with computational practicality, this work paves the way for broader adoption and continued innovation in the field of neuroimaging.

References

[1] An, M., Chen, J., Cao, Y., Tao, K., Wang, J., Wang, C., Zhao, K., Liu, Z., 2023. Resattn-recon: Residual self-attention based cortical surface reconstruction. Frontiers in Physics 11, 172.

[2] Bazin, P.L., Pham, D.L., 2007. Topology correction of segmented medical images using a fast marching algorithm. Computer methods and programs in biomedicine 88, 182–190.

[3] Bentley, J.L., 1975. Multidimensional binary search trees used for associative searching. Commun. ACM 18, 509–517. URL: https://doi.org/10.1145/361002.361007, doi:10.1145/361002.361007.

[4] Bentley, J.L., 1990. K-d trees for semidynamic point sets, in: Proceedings of the sixth annual symposium on Computational geometry, pp. 187–197.

[5] Bongratz, F., Fecht, J., Rickmann, A.M., Wachinger, C., 2024a. V2c-long: Longitudinal cortex reconstruction with spatiotemporal correspondence. arXiv preprint arXiv:2402.17438 .

[6] Bongratz, F., Rickmann, A.M., Pölsterl, S., Wachinger, C., 2022. Vox2cortex: fast explicit reconstruction of cortical surfaces from 3d mri scans with geometric deep neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 20773–20783.

[7] Bongratz, F., Rickmann, A.M., Wachinger, C., 2024b. Neural deformation fields for template-based reconstruction of cortical surfaces from mri. Medical Image Analysis 93, 103093.

[8] Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K., 2018. Neural ordinary differential equations. Advances in neural information processing systems 31.

[9] Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O., 2016. 3d u-net: learning dense volumetric segmentation from sparse annotation, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19, Springer. pp. 424–432.

[10] Cruz, R.S., Lebrat, L., Bourgeat, P., Fookes, C., Fripp, J., Salvado, O., 2021. Deepcsr: A 3d deep learning approach for cortical surface reconstruction, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 806–815.

[11] Destrieux, C., Fischl, B., Dale, A., Halgren, E., 2010. Automatic parcellation of human cortical gyri and sulci using standard anatomical nomenclature. Neuroimage 53, 1–15.

[12] Fan, H., Su, H., Guibas, L.J., 2017. A point set generation network for 3d object reconstruction from a single image, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 605–613.

[13] Fischl, B., 2012. Freesurfer. Neuroimage 62, 774–781.

[14] Fischl, B., Dale, A.M., 2000. Measuring the thickness of the human cerebral cortex from magnetic resonance images. Proceedings of the National Academy of Sciences 97, 11050–11055.

[15] Fischl, B., Sereno, M.I., Dale, A.M., 1999. Cortical surface-based analysis: Ii: inflation, flattening, and a surface-based coordinate system. Neuroimage 9, 195–207.

[16] Gopinath, K., Desrosiers, C., Lombaert, H., 2021. Segrecon: Learning joint brain surface reconstruction and segmentation from images, in: Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part VII 24, Springer. pp. 650–659.

[17] Hariharan, B., Arbeláez, P., Girshick, R., Malik, J., 2015. Hypercolumns for object segmentation and fine-grained localization, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 447–456.

[18] Henschel, L., Conjeti, S., Estrada, S., Diers, K., Fischl, B., Reuter, M., 2020. Fastsurfer – a fast and accurate deep learning based neuroimaging pipeline. NeuroImage 219, 117012.

[19] Hong, Y., Ahmad, S., Wu, Y., Liu, S., Yap, P.T., 2021. Vox2surf: Implicit surface reconstruction from volumetric data, in: Machine Learning in Medical Imaging: 12th International Workshop, MLMI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, September 27, 2021, Proceedings 12, Springer. pp. 644–653.

[20] Hoopes, A., Iglesias, J.E., Fischl, B., Greve, D., Dalca, A.V., 2022. Topofit: rapid reconstruction of topologically-correct cortical surfaces. Proceedings of machine learning research 172, 508.

[21] Lam, S.K., Pitrou, A., Seibert, S., 2015. Numba: A llvm-based python jit compiler, in: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, pp. 1–6.

[22] Le, T., Nguyen, K., Sun, S., Han, K., Ho, N., Xie, X., 2023. Diffeomorphic deformation via sliced wasserstein distance optimization for cortical surface reconstruction. arXiv preprint arXiv:2305.17555 .

[23] Lebrat, L., Santa Cruz, R., de Gournay, F., Fu, D., Bourgeat, P., Fripp, J., Fookes, C., Salvado, O., 2021. Corticalflow: a diffeomorphic mesh trans-

former network for cortical surface reconstruction. Advances in Neural Information Processing Systems 34, 29491–29505.

[24] Liu, Z., Tong, L., Chen, L., Jiang, Z., Zhou, F., Zhang, Q., Zhang, X., Jin, Y., Zhou, H., 2023. Deep learning based brain tumor segmentation: a survey. Complex & intelligent systems 9, 1001–1026.

[25] Lorensen, W.E., Cline, H.E., 1987. Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics 21, 163–169.

[26] Ma, Q., Li, L., Robinson, E.C., Kainz, B., Rueckert, D., Alansary, A., 2022. Cortexode: Learning cortical surface reconstruction by neural odes. IEEE Transactions on Medical Imaging .

[27] Ma, Q., Robinson, E.C., Kainz, B., Rueckert, D., Alansary, A., 2021. Pialnn: a fast deep learning framework for cortical pial surface reconstruction, in: Machine Learning in Clinical Neuroimaging: 4th International Workshop, MLCN 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, September 27, 2021, Proceedings 4, Springer International Publishing. pp. 73–81.

[28] Makropoulos, A., Robinson, E.C., Schuh, A., Wright, R., Fitzgibbon, S., Bozek, J., Counsell, S.J., Steinweg, J., Vecchiato, K., Passerat-Palmbach, J., et al., 2018. The developing human connectome project: A minimal processing pipeline for neonatal cortical surface reconstruction. Neuroimage 173, 88–112.

[29] Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S., 2019. Deepsdf: Learning continuous signed distance functions for shape representation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 165–174.

[30] Ren, J., Hu, Q., Wang, W., Zhang, W., Hubbard, C.S., Zhang, P., An,

N., Zhou, Y., Dahmani, L., Wang, D., et al., 2022. Fast cortical surface reconstruction from mri using deep learning. Brain Informatics 9, 6.

[31] Riccelli, R., Toschi, N., Nigro, S., Terracciano, A., Passamonti, L., 2017. Surface-based morphometry reveals the neuroanatomical basis of the five-factor model of personality. Social cognitive and affective neuroscience 12, 671–684.

[32] Rickmann, A.M., Bongratz, F., Wachinger, C., 2023. Vertex correspondence in cortical surface reconstruction, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer. pp. 318–327.

[33] Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, Springer. pp. 234–241.

[34] Santa Cruz, R., Lebrat, L., Fu, D., Bourgeat, P., Fripp, J., Fookes, C., Salvado, O., 2022. Corticalflow++: Boosting cortical surface reconstruction accuracy, regularity, and interoperability, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer. pp. 496–505.

[35] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G., 2008. The graph neural network model. IEEE transactions on neural networks 20, 61–80.

[36] Tamnes, C.K., Herting, M.M., Goddings, A.L., Meuwese, R., Blakemore, S.J., Dahl, R.E., Güroğlu, B., Raznahan, A., Sowell, E.R., Crone, E.A., et al., 2017. Development of the cerebral cortex across adolescence: a multisample study of inter-related longitudinal changes in cortical volume, surface area, and thickness. Journal of Neuroscience 37, 3402–3412.

[37] Ts'o, D.Y., Zarella, M., Burkitt, G., 2009. Whither the hypercolumn? The Journal of physiology 587, 2791–2805.

[38] Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G., 2018. Pixel2mesh: Generating 3d mesh models from single rgb images, in: Proceedings of the European conference on computer vision (ECCV), pp. 52–67.

[39] Wickramasinghe, U., Remelli, E., Knott, G., Fua, P., 2020. Voxel2mesh: 3d mesh model generation from volumetric data, in: Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part IV 23, Springer. pp. 299–308.

[40] Zhao, F., Wu, Z., Li, G., 2023a. Deep learning in cortical surface-based neuroimage analysis: a systematic review. Intelligent Medicine 3, 46–58. URL: https://www.sciencedirect.com/science/article/pii/S2667102622000493, doi:https://doi.org/10.1016/j.imed.2022.06.002.

[41] Zhao, J., Liu, S., Ahmad, S., Yap, P.T., 2023b. Meshdeform: Surface reconstruction of subcortical structures via human brain mri, in: International Conference on Information Processing in Medical Imaging, Springer. pp. 536–547.

[42] Zheng, H., Li, H., Fan, Y., 2023. Surfnn: Joint reconstruction of multiple cortical surfaces from magnetic resonance images. arXiv preprint arXiv:2303.02922 .

[43] Zheng, H., Li, H., Fan, Y., 2024. Coupled reconstruction of cortical surfaces by diffeomorphic mesh deformation. Advances in Neural Information Processing Systems 36.

Appendix A. Additional Methodology Details

Appendix A.1. Data and Code

The data utilized in this study were obtained from the Human Connectome Project (HCP). The specific datasets and subjects accessed from the HCP are available through ConnectomeDB at https://db.humanconnectome.org/.

To ensure reproducibility and facilitate further research, the code developed for the analyses presented in this study is available on GitHub[4]. This repository includes links to additional repositories where the original frameworks were forked and modified for analysis, benchmarking, and deployment with Singularity.

For each framework, HCP data processed by *FreeSurfer* were used as the starting point, followed by the application of framework-specific preprocessing scripts, as documented in the respective framework overviews.

Appendix A.2. Model Training and Configuration

Appendix A.2.1. Model Training

Framework-specific instructions and default configurations, as recommended in their respective README files and configuration files, were adhered to during model training. Given that architectural choices, schedulers, learning rates, and other design parameters are interdependent, the responsibility for reproducibility lies with the original authors to provide detailed instructions that yield effective models.

Appendix A.2.2. Template and Resolution Settings

The highest resolution mesh settings available for each framework requiring templates were employed. Specifically, for *DeepCSR*, which is not template-based, a resolution setting of 256 was utilized for experiments and benchmarking due to its stability. Although lower resolutions are available for each framework and could significantly impact memory and GPU benchmarks, the focus

[4] https://github.com/neuroneural/SurfaceReconstructionReview

was on evaluating the performance of the highest-resolution meshes. Although *DeepCSR* supports a resolution of 512, this setting was not explored further due to excessive memory and GPU requirements that rendered it impractical within the available computational environment.

PialNN utilized the white surface from *FreeSurfer* as the input template and was constrained to *FreeSurfer*'s resolution. Frameworks such as *TopoFit*, *CorticalFlow*, *CorticalFlow++*, *Vox2Cortex*, and *CortexODE* each required unique templates corresponding to their respective resolutions, with the highest available resolution selected for each.

Appendix A.2.3. ODE Solver Configurations

For frameworks employing ODE solvers, the default number of steps and solver types were utilized. Specifically, *CorticalFlow* and *CortexODE_euler* employed Euler solvers, while *CorticalFlow++* and *CortexODE_rk4* utilized Runge-Kutta 4 (RK4) solvers. *CortexODE* was configured with a step size of 0.1 for both solver variants, effectively demonstrating the strengths of the RK4 method within their framework.

Both *CorticalFlow* and *CorticalFlow++* defined their step parameters in terms of the number of steps, which were set to the default of 30 steps per mesh. The Singularity training scripts were configured to reflect these settings.

Appendix A.3. Hardware Configuration

Appendix A.3.1. Computational Benchmarks and Mesh Generation

During computational benchmarks and mesh generation in evaluation mode, the server's hardware configuration comprised an AMD EPYC 7551 32-Core Processor, built on the x86_64 architecture. Using the Slurm job scheduler, four cores were allocated for each framework's inference tasks per surface. The processor operates at a clock speed of 2.532 GHz and features an extensive cache system, including 2 MiB of L1d cache, 4 MiB of L1i cache, 32 MiB of L2 cache, and 128 MiB of L3 cache.

The server is equipped with an NVIDIA GeForce RTX 2080 Ti GPU possessing 11,264 MiB of memory, operating with NVIDIA driver version 535.104.12 and CUDA version 12.2.

Appendix A.3.2. Training Environment

The training system is built on an x86_64 architecture running Ubuntu 20.04 OS. For each surface type and framework, four Intel Xeon Platinum 8468 CPU cores with a base clock speed of 3.1 GHz were allocated during training sessions, accommodating up to 16 cores for four surfaces. The CPU is equipped with a 4.5 MiB Level 1 data cache, a 3 MiB Level 1 instruction cache, a 192 MiB Level 2 cache, and a 210 MiB Level 3 cache.

One NVIDIA A100-SXM4-80GB GPU was allocated per framework, utilizing NVIDIA driver version 535.104.12. The A100 GPUs require libraries compatible with CUDA 11 or higher. The Slurm scheduler facilitated parallel training of frameworks, dispatching each framework using Singularity containers and scripts.

Appendix A.4. Containerization and Code Management

For security reasons, many high-performance computing centers prefer Singularity over Docker. Docker offers the capability to sandbox development and production environments, providing unique sets of dependencies. Additionally, Docker supports the deployment of microservices and facilitates the distribution of framework dependencies. Singularity provides similar functionalities to Docker but operates with fewer privileges at runtime, offering enhanced security.

Each reviewed framework was forked, and corresponding Dockerfiles and Singularity runtime scripts were created. Each framework repository includes an updated README detailing the steps required to reproduce the results. The companion repository for this study[5] contains extensive documentation linking to each fork, analysis scripts, and other pertinent information.

[5] https://github.com/neuroneural/SurfaceReconstructionReview