Approaching the Vesuvius Challenge: Surface Detection in 3D CT Scans

1. Challenge Overview and Goals

The Vesuvius Challenge – Surface Detection is a Kaggle competition focused on segmenting the surfaces of rolled papyrus scrolls from high-resolution 3D X-ray CT scans. The scrolls, carbonized by the eruption of Mt. Vesuvius contains fragile layers of papyrus with ancient text. Physically unwrapping them is impossible, so the goal is to detect and segment the papyrus sheet surfaces in volumetric scans so they can be virtually "unwrapped" and readscrollprize.substack.comscrollprize.org. In practice, this means labeling the thin, crumpled papyrus layers within the 3D volume (as a binary mask) while avoiding errors like merging separate layers or leaving gaps. This surface segmentation is one of the most essential steps in the virtual unwrapping pipelinescrollprize.org – accurate surfaces allow researchers to flatten the scroll and then apply ink detection models on the flattened pagesscrollprize.orgscrollprize.org. The competition provides CT scan data in chunks along with binary masks marking smoothed papyrus sheet positions as ground truthscrollprize.substack.com. Participants must train models to reproduce these surface labels on unseen scans, with a $100K prize pool incentivizing creative and effective solutionsscrollprize.substack.com.

Key challenge considerations: The papyrus surfaces are very thin, convoluted structures in the volume, often only a voxel or two thick and closely packed. They can be difficult to distinguish from the background or each other in the CT images. The task thus requires segmenting complex 3D shapes with high precision. Importantly, the solution isn't just about voxel-wise accuracy, but also about preserving the topology of each sheet – no false connections between different sheets, and no holes or breaks in a continuous sheetscrollprize.substack.com. In other words, the model's output should follow each scroll layer cleanly, without "stitching" layers together or creating artificial tears. Achieving this demands careful modeling and validation, since even small misclassifications can lead to significant topological errors. Overall, the challenge's goal is to produce clean, continuous papyrus surface masks that will enable successful virtual unrolling of the scrolls and subsequent ink detectionscrollprize.substack.com.

2. Methodologies for 3D Surface Segmentation in Volumetric Data

Segmenting surfaces in 3D volumetric data is a well-studied problem, and various methodologies can be applied – from classical image processing to modern deep learning. In the context of the scroll CT scans, the surfaces correspond to thin sheet-like structures. Earlier efforts to segment the scrolls used semi-automatic approaches: for example, the project's research team employed a custom tool called Volume Cartographer which combined algorithmic detection with extensive manual tracing to map out papyrus layersscrollprize.org. While such tools can leverage domain knowledge (e.g. following high-contrast boundaries in the volume) and even incorporate Hessian-based filters or "sheetness" measures to enhance planar structures, they are labor-intensive. The Kaggle challenge instead encourages machine learning solutions that can generalize and automate this process.

3D segmentation with deep learning is the predominant approach for this task. Typically, a model is trained to classify each voxel in a 3D input patch as “surface” or “background.” A common strategy is to use a sliding window or patch-based inference: the full CT volume (or large chunk) is too big to process at once, so one divides it into smaller 3D patches (for example, 256×256×depth cubes) for training and predictiongithub.com. This tiling method was successfully used in the earlier Vesuvius ink detection competition to handle ultra-high-res imagesgithub.com, and similarly, for volumetric data it ensures memory feasibility. Care must be taken to use overlapping patches or stitching strategies so that surfaces crossing patch boundaries are consistently segmented (to avoid edge artifacts).

Two broad methodological paradigms exist for segmenting such 3D data:

- Full 3D Convolutional Approaches: These treat the data as a true volume. For instance, a network can take a 3D sub-volume (e.g. 128×128×64 voxels) and output a 3D mask of the same size. This leverages contextual information in all three dimensions, which is crucial for following papyrus sheets that may undulate in 3D space. A properly trained 3D model can learn the continuity of surfaces across slices (preventing broken or misaligned segments). However, 3D models are computationally heavy, and memory constraints limit the input size. Participants often find a balance by using moderately sized patches and pooling operations to get a larger receptive field. In practice, 3D CNNs have been effective: the winning ink detection solution in 2023, for example, used 3D models in the first stage to analyze volumetric datagithub.com. For surface detection, a pure 3D U-Net is a natural baseline approach (and indeed

Kaggle provided starter code with a simple 3D U-Net) to directly segment the volume.

- 2D or 2.5D Slice-Based Approaches: These simplify the problem by treating it as a series of 2D segmentations. A model can segment one CT slice at a time (possibly using neighboring slices as additional channels for context – a “2.5D” approach). This drastically reduces memory usage and allows use of high-capacity 2D CNNs or vision transformers on each slice. However, purely slice-by-slice methods risk losing the global 3D continuity of surfaces. A surface that is continuous in 3D might appear as disjoint contours in individual 2D slices, so a 2D model might inadvertently break it or connect it to the wrong structure. Indeed, research has shown that “slice-by-slice” segmentation struggles with complex 3D shapesnature.com. In the scrolls, surfaces are not axis-aligned – they twist through the volume – so a 2D model might confuse one layer for another without cross-plane awareness. That said, some participants experiment with 2.5D models (e.g. segmenting orthogonal projections or stacks of slices) because they are easier to train. If one goes this route, it’s important to add mechanisms to enforce consistency across slices (for example, merging 2D results into 3D and then applying connected-component filtering to correct any fragmented surfaces).

Beyond direct segmentation, other methodologies can assist in detecting thin surfaces:

- Spatial Filtering and Feature Engineering: Classical techniques like edge detectors or Hessian-based filters can pre-process the CT data to highlight sheet-like structures. For example, analyzing the Hessian matrix of the intensity volume can reveal plate-like features (where two of the eigenvalues are small and one is large in magnitude). A "sheetness" filter could be applied to create an auxiliary volume emphasizing likely papyrus layers, which can then be fed as an input channel to a CNN. Similarly, a distance transform (DT) can be used: if one has an initial rough mask of papyrus, the DT's ridge (midline) might correspond to the center of the sheet. In fact, some discussions suggest using the midline of a distance transform to improve sheetness of predictions before skeletonization (thinning)kaggle.com. In practice, a model might output a thick mask and a post-processing step could thin it to a mid-surface using DT + skeletonization (ensuring one-voxel thickness). These classical methods can complement learning-based methods by enforcing the expected thin-sheet geometry.

- Multi-step or Hybrid Approaches: It's conceivable to break the task into stages – e.g., first identifying approximate sheet locations, then refining the exact surface boundaries. One might train a coarse model to label general areas of papyrus vs background, then a secondary model or algorithm to trace the precise surface within those areas. Another idea is using graph-based methods or watershed algorithms on the output probabilities: treating the volume of predicted papyrus probability as a landscape, one could use watershed (on the inverted probability) to separate distinct layers, using markers derived from predicted surface positions. While most Kaggle solutions likely stick to end-to-end segmentation, such hybrid post-processing can help fix mistakes like merged layers by leveraging the fact that real papyrus sheets should be separate surfaces.

In summary, the methodology centers on robust 3D segmentation. Best practice is to exploit 3D context as much as possible (since the task is inherently three-dimensional) and to incorporate any prior knowledge about the "sheet-like" nature of the target structures. Whether using a fully 3D CNN or a clever combination of 2D models, the goal remains the same: accurately trace the continuous surfaces of the scroll in the volumetric data.

3. Suitable Model Architectures for Surface Segmentation

Given the task's nature, several model architectures stand out as suitable:

- 3D U-Net and Variants: The 3D U-Net is a direct extension of the popular U-Net architecture to volumetric data (using 3D convolutions and 3D pooling/up-sampling) and is a strong baseline for medical and volumetric segmentation tasks. It consists of an encoder path that captures context and a decoder path that refines details, with skip connections to preserve spatial information. Many Kagglers start with a 3D U-Net for this challengekaggle.com. It's simple yet effective at learning the mapping from a raw CT volume patch to a binary mask. Efficient variants (with group norms, residual blocks, etc.) or lightweight 3D CNNs can be used to cope with memory limitations. For instance, the winning Kaggle ink detection team trained multiple 3D CNN-based models (including 3D U-Nets) as part of their two-stage pipelinegithub.com. They found that combining several architectures was beneficial, which underscores that a well-tuned 3D U-Net is a key component of a strong solution.

- Voxel Transformers and Hybrid Models: Transformer-based architectures have made inroads in segmentation, including volumetric cases. A notable example is UNETR (introduced by Nvidia), which uses a Vision Transformer encoder on 3D patches and a decoder like U-Net. The ink detection winners mention using UNETR alongside 3D CNNs in their first stagegithub.com. For surface detection, a voxel transformer could capture long-range dependencies in the volume – potentially useful if a papyrus sheet extends over a large area or if global context is needed to differentiate layers. Similarly, 3D Swin Transformer models (which apply self-attention in local windows through the volume) can be used to build a transformer-U-Net hybrid. These models are computationally heavy, but they can excel at learning shape continuity and global context. If resources permit, using a transformer encoder to capture the 3D structure, followed by a decoder that outputs the segmentation, can improve results – especially in avoiding fragmented surfaces (transformers naturally integrate information over distance). Some Kaggle notebooks (e.g. using the TransUNet or other medicAI modelskaggle.com) have explored this approach.

- “Surface-Aware” CNNs: This category isn’t a single architecture but rather design tweaks to make CNNs better at segmenting thin surfaces. One idea is a network that explicitly predicts surface orientation or distances. For example, the model could have auxiliary outputs like a signed distance field to the nearest papyrus surface, or a normal vector map. By learning these, the model inherently gains awareness of surface geometry. Although not standard in Kaggle, such approaches have precedent in research. Another concept is using multi-task learning: train the model to segment surfaces and detect sheet edges or thickness. A CNN could have one head that marks likely surface voxels and another that predicts where papyrus material exists (which might cover the thickness). The surface head can be guided by the material head but constrained to the outer boundaries. This way the CNN knows it should produce a single-voxel boundary rather than a thick region. Additionally, architectures that incorporate skeletonization or centerline extraction in the network (via differentiable operators) have been proposed in literature to enforce thin structure segmentationijcai.orgijcai.org. In practice, contestants might implement a simpler form: e.g., apply a thinning algorithm on the predicted mask as a post-process, but one could also try to bake that into the model’s training (for instance, using a differentiable clDice loss – see Section 6). Overall, “surface-aware” modeling means designing the network or loss to explicitly handle the thin, sheet-like nature of the target.

- Two-Stage Models (Coarse-to-Fine or 3D+2D): A notable architecture pattern from the ink detection competition was a two-stage model: first a 3D model outputs a volume or features, and then a 2D model refines itgithub.com. For ink, the winning team's 3D stage produced multi-channel 3D output that was flattened and fed to a 2D SegFormer modelgithub.com. A similar approach could be tried for surfaces – e.g., Stage 1: a 3D U-Net that outputs an initial surface probability map (or an enhanced volume where papyrus sheets are highlighted), Stage 2: a 2D segmentation model that processes each slice or a projection to clean up details. The rationale is that the 3D model captures the overall structure in 3D, while the 2D model (with potentially higher capacity per slice) can sharpen boundaries or remove noise. However, one must ensure the 2D stage does not break 3D continuity. If done slice-by-slice, some consistency enforcement (like using the 3D output as a guide or doing a final 3D connected-component check) is needed. The two-stage approach can also be coarse-to-fine: first identify regions where surfaces likely reside (lower resolution or downsampled volume), then zoom in and segment at full resolution in those regions with a specialized model.

- Other architectures: Some competitors might experiment with 3D residual networks or encoder-decoders with feature pyramids (3D FPNs) to better capture multi-scale features (small gaps vs large curving sheets). Others might try Graph Neural Networks if treating the surface as a graph problem (less common, but possible if one extracts an initial mesh). There is also research on level-set networks where the network predicts a level-set function whose zero-level corresponds to the surface; this implicitly ensures a smoother surface. Another advanced idea is leveraging Mesh segmentation: converting the volume to an initial mesh (isosurface) and then using a mesh-based CNN or transformer to label parts of the mesh that are papyrus. However, given the complexity, most Kaggle solutions will stick to volumetric pixel-wise segmentation.

In summary, a 3D U-Net or similar 3D CNN is a strong starting point, potentially enhanced by transformer components or multi-task heads to account for the nature of the problem. Ensembling different architectures often yields the best results – for example, combining a 3D CNN, a 3D Swin UNet, and a 2D model – as each may have different strengths. The first-place ink detection team explicitly cited an ensemble of 9 models (with 3D and 2D variants) as key to their successgithub.comgithub.com, and a similar strategy could improve surface detection robustness. The choice of architecture should consider memory vs. context trade-offs, and how easily the model can be guided to avoid topological mistakes (which we address next).

4. Data Preprocessing Techniques for CT Scan Data

Preprocessing the data correctly is crucial for training an effective model on this challenge. The raw input consists of 3D CT scan volumes (and possibly some derived 2D projections) with associated binary masks for surfaces. Notable preprocessing steps include:

- Intensity Normalization: CT scans are quantitative, but in this dataset the values might be arbitrary units scaled for the imaging system. It's common to normalize voxel intensities to a consistent range before feeding to a neural network. Many participants apply a simple normalization like min-max scaling to [0,1] or standardization (zero mean, unit variance) based on the training data. Normalizing per volume (i.e., subtract the mean and divide by std of each scan chunk) can account for slight differences in scanner calibration or reconstruction lighting. Consistent intensity scaling helps the model generalize across different scrolls. Additionally, since the task is essentially boundary detection, some might enhance contrast around the papyrus layers. For instance, one could apply a slight Gaussian blur or median filter to reduce high-frequency noise, or use a high-pass filter to emphasize edges. However, too much preprocessing could remove subtle signals (like slight density differences indicating papyrus). As a safe approach, many will use raw CT intensities with just linear normalization. If the CT contains outliers or a large dynamic range, clipping to a certain percentile range (to remove extreme values) and then scaling might be beneficial.

- Resampling and Alignment: If the dataset contains any anisotropy (different voxel spacing in z vs x/y), one could resample to isotropic voxels. The scroll CT scans likely have high resolution in all axes (perhaps on the order of tens of microns), but it's worth checking. Anisotropic data can harm a 3D model's performance (e.g., features look stretched in one dimension), so resampling to a uniform resolution and then resizing patches accordingly can improve results. Also, all volumes should be in the same orientation; if some scans are mirrored or rotated, one should reorient them to a common coordinate system if possible (the dataset likely already provides consistency, but competition forums sometimes flag any such issues).

- Patching (Tiling) Strategy: As mentioned, dividing the volume into patches is necessary for training. A good strategy is to generate patches that cover the entire training volume with overlap. Overlap ensures that surfaces that lie on patch borders are still learned correctly. One might use a sliding window of size (for example) 160×160×64 with a stride of 128×128×64 (just as an illustration), so neighboring patches overlap by 32 voxels. Overlap allows mixing predictions at inference (averaging or max-voting in the overlap region) to avoid seam artifacts. Another consideration is sampling more patches from areas that contain surfaces (since large portions of the volume might be empty background). If papyrus layers occupy only a small fraction of the voxels, class imbalance can be an issue – the model might output all zeros (no surface) and still get low error on most voxels. To combat this, one can sample patches with at least a certain percentage of surface voxels during training. For example, only patches that contain, say, >5% surface pixels could be used or at least oversampledgithub.com. The ink detection competition did something similar by only considering image tiles with enough inkgithub.com. In surface detection, this ensures the model sees positive examples frequently. That said, completely excluding empty patches could lead to edge effects (model never learning what pure background looks like), so a balanced approach (some patches mostly empty for specificity, and many patches with surfaces for sensitivity) is wise.

- Data Augmentation: Augmentations help the model generalize to variations in the data. For 3D CT volumes of scrolls, useful augmentations include:

  - Geometric transforms: Random rotations (90-degree or arbitrary angles) around the z-axis (in-plane) are typically easy (the scroll has no canonical orientation in XY plane). Rotations out-of-plane (tilting the volume) are trickier due to interpolation and the fact that the data is 3D – but small tilts could be done if handled carefully (or one could generate new slices by interpolation). Horizontal and vertical flips of the volume axes are also useful (since a flipped scroll is still a valid scroll). These augmentations address that the scroll might appear in different orientations or flipped arrangements in different data chunks.

  - Elastic deformations: Slight elastic distortion of the volume could simulate the variability in how the papyrus might be warped. However, this must be subtle to avoid creating unrealistic structures (one does not want to hallucinate a connection between layers). Elastic deformations are used in medical segmentation to augment organ shapes; for scrolls, a mild 3D elastic transform might improve robustness to natural shape differences.

    - Intensity augmentations: Varying brightness or contrast of the CT could help if there are differences between scans. One might randomly scale the intensity or apply slight gamma adjustments. Noise augmentation (adding Gaussian noise to simulate scanner noise) can also regularize the model.

    - Cutout or Masking: Randomly masking out small regions of the input (or setting to noise) might force the model to rely on broader context (this is like a dropout in input space). Given the importance of continuity, some might mask out parts of a surface in training to see if the model can still infer it (though this is speculative).

- In the Kaggle ink competition, heavy augmentation was used (the winners mention rotations, flips, and ensuring invariances). Some even tried mixup/cutmix on images. For 3D, mixup of volumes might be less meaningful, but one could mix two volumes if they are of the same size by taking parts from each – however, since physical scroll structure wouldn't allow two different scrolls in one volume, it may be unrealistic. It's safer to augment within one volume's data. Tip: Always ensure that augmentation does not create broken or merged surfaces that wouldn't occur in reality, as that could confuse the model. For example, a large rotation might slice through a scroll oddly if not handled properly.

- Cropping and Padding: When training on patches, ensure that if a surface lies at the very border of a patch, the model isn't penalized unfairly if it predicts it slightly outside (which might actually be inside the neighboring patch in reality). Using valid convolutional crops or ignoring a few voxels near patch borders in the loss can help. At inference, one would typically pad the volume and then crop out the padding after segmentation to avoid boundary effects.

- Label Refinement: The ground truth masks provided are "smoothed sheet positions"scrollprize.substack.com – likely one-voxel thick representations of each papyrus layer. It might be beneficial to compute additional label representations for training. For instance, one can derive a distance map from the surface labels (distance to nearest surface voxel) and use that as auxiliary supervision (so the model doesn't just learn a binary, but also how close every voxel is to a surface, which might enrich its features). Another idea is to separate the surfaces by a small ID or coloring for analysis – though the task is binary, knowing which connected component is which might help in evaluating topology during development. Some competitors visualize connected components of predictions vs truth to identify errors (not a preprocessing per se, but a troubleshooting step).

In summary, preprocessing for this challenge involves making the volumetric data manageable for models (tiling, normalization), enriching the data with realistic variations (augmentation), and accounting for the class imbalance and structural peculiarities of the scrolls. Proper tiling and normalization were critical in the previous competitiongithub.com, and for surface segmentation an even more careful tiling (with overlaps and bias towards surface-containing patches) is recommended. Augmentations like flips and rotations ensure the model doesn't overfit to the particular orientation of training scans, which is important if the test data comes from different scrolls or different orientations.

5. Evaluation Metrics and Their Influence on Modeling

The competition uses a weighted blend of three segmentation metrics as the evaluation score: Surface Dice, VOI_score (Variation of Information), and TopoScore, each targeting different aspects of segmentation qualitykaggle.com. The combined score is calculated, for example, as: Score = 0.30 × TopoScore + 0.35 × SurfaceDice@τ + 0.35 × VOI_score (the exact weights and τ are defined in the competition)kaggle.com. Let's break down each metric:

- Surface Dice @ τ: This is a boundary-focused overlap metric. In essence, Surface Dice (also known as Normalized Surface Dice, NSD) measures how well the predicted surface aligns with the ground truth surface within a tolerance distance τ. Formally, it evaluates the proportion of surface points on one mask that lie within τ voxels of a surface point on the other masknature.com. A true positive is counted when a predicted surface voxel is sufficiently close to some true surface voxel (and vice versa), instead of requiring exact voxel-wise overlap. This accounts for the fact that a slight offset of a surface (within a small tolerance) is not a critical error in practice. For example, if τ=1 voxel, a prediction that is one voxel away from the actual surface can still be considered correct. The Surface Dice thus rewards accurate localization of the papyrus sheets' boundaries and penalizes large deviations. It is more forgiving than standard Dice on small shifts, but it focuses only on the boundary pixels, ignoring large correctly identified blank areas. A high Surface Dice means your predicted surfaces closely trace the true surfaces (geometrically), which is exactly what we want in virtual unwrapping. In modeling, this suggests that surface localization is key – losses that emphasize getting the boundary right (like boundary loss or level-set loss) might directly improve this metric.

- Variation of Information (VOI): VOI is an information-theoretic metric that measures the distance between two clusterings (here, the clustering of voxels into connected components of papyrus vs background)kaggle.com. In segmentation terms, it penalizes differences in how the segmentation partitions the space. VOI can be thought of as capturing over-segmentation and under-segmentation errors: if a single true surface is split into multiple predicted pieces (over-segmentation) or multiple true surfaces are merged into one predicted blob (under-segmentation), VOI increases (worsens). It computes the conditional entropy of the true labeling given the predicted labeling and vice versakaggle.com. Intuitively, VOI is very sensitive to topological changes like splits or mergesijcai.org. If your prediction has a different number of connected surface components than the ground truth, the VOI component of the score will suffer significantly. For example, merging two adjacent layers into one predicted region is a catastrophic error under VOI, as is producing a hole that divides one surface into two. The influence on modeling is that you must avoid merges and splits. A model that maybe slightly mis-aligns a surface (hurting Surface Dice a bit) could still be okay, but one that accidentally connects two layers will have a huge VOI penalty. This pushes us to design models or post-processes that preserve the count and identity of surfaces. It also means we should validate not just on Dice but also ensure the connected-component structure of predictions matches ground truth. In practice, one might regularly check the VOI or a proxy (like Rand index or counting components) on validation data to catch topology issues early. The VOI metric essentially tells us that instance-level segmentation quality matters – the model should "know" that each sheet is a separate entity and maintain those separations.

- TopoScore: This metric was formulated to explicitly quantify topological correctness for the scroll surfaces. While the exact implementation isn't given here, it is described as rewarding surface connectivity and penalizing gaps, holes, or mergersscrollprize.substack.com. We can infer that TopoScore checks for errors like: Did you introduce any holes in a surface (where a continuous sheet in truth is broken in prediction)? Did you merge two distinct sheets into one connected component? Did you create any "sheet-switches" (perhaps predicting a layer incorrectly such that it jumps from one sheet to another)? Essentially, it's a custom measure of whether the predicted surfaces have the same topological structure (in terms of connected components and surface genus) as the truth. It might be related to computing something like the Betti number errors (differences in count of connected components and holes), which is used in topological evaluations. In segmentation literature, similar metrics exist – e.g., the Betti error counts mismatches in number of holes or components. The competition discussion indicates that "VOI and TopoScore components are greatly affected by topology changes" – if your prediction has a different number of connected components, these metrics will drop sharply. TopoScore likely directly reflects if all sheets are intact and separate. For modeling, this metric underscores that even one small connection or break can ruin your score. It's not enough to get high voxel overlap; a slightly broken ring of segmentation might lose a lot of points. Therefore, contestants often incorporate topology-aware strategies like special loss terms or post-processing aimed at preserving connectivity. For example, one might add a loss penalty for false splits by using something like a clDice loss or connectivity regularizer (see Section 6), or do a morphological check on outputs (if two sheets got connected by a single voxel bridge, cut it).

The combined metric being a weighted average means you cannot ignore any single component. However, notably TopoScore (topology) and VOI are highly correlated – both punish connectivity errors. In fact, Kaggle organizers hinted that the topology component is the most important to focus on, since many participants would naturally optimize for Dice but might neglect topology. A quote from the discussion: "To win: (1) Correct loss or post-processing to improve topology score (most important), (2) surface dice, (3) VOI score. Many Kagglers will treat this like a normal segmentation challenge, but topology errors can be the biggest downfall." This implies that a model which naively maximizes Dice could still fail if it produces even a few mergers or holes, dragging down TopoScore/VOI. Successful approaches likely balance precision and recall carefully on the surfaces: you want to cover each true surface (no holes ↣ high recall), but also not hallucinate connections or extra bits (no merges or extra components ↣ need high precision in between layers).

Influence on modeling strategies: Participants have to adapt their training and post-processing to these metrics. Some concrete strategies influenced by the metrics are:

- Using a distance-based loss (for SurfaceDice): e.g., incorporate a boundary loss that penalizes distance of predicted surface to true surface, rather than just region overlap. This can directly optimize a surrogate to SurfaceDice. One could implement a differentiable surface distance by convolving the surface mask or using morphological operations to compute distances within the loss function.

- Topology-aware loss or regularization (for TopoScore/VOI): e.g., adding a term in the loss that penalizes changes in Euler characteristic between prediction and truth. There is research on topological losses (using persistent homology, etc.) that could be applied. Kaggle solutions might not go fully into homology calculations due to complexity, but a simpler approach is to use the clDice loss (which encourages maintaining connectivity for tubular structures) adapted to sheets. In fact, clDice (which stands for centerline Dice) has been used to preserve connectivity in segmentation by maximizing overlap of skeletons. While clDice is for thin lines, a 2D analog could be applied slice-wise, or one can take the skeleton of the 3D surface (essentially a graph representing each surface's "mid-line" network) and enforce overlap there.

- Prediction refinement via connectivity analysis: Because metrics reward getting connectivity exactly right, many will include a post-processing step to fix common errors. For example, after getting the raw prediction, one can find any small holes in a surface and fill them (since a small hole could cause a surface to be counted as two components). Likewise, if two surfaces are accidentally connected by a few voxels, one could detect that and sever the connection. Simple morphological operations can do this: small hole filling can be done by a binary closing or by analyzing connected components of the inverse (background inside a mostly-filled ring). Small connections between layers can sometimes be detected by checking the thickness of the predicted mask – if locally it exceeds 1 voxel in thickness or has a thin extrusion connecting to another layer, that's suspect. One could apply an opening operator or thickness filter to remove voxels that make a layer locally thicker than expected. Another approach is converting the predicted mask to a graph (skeleton) and seeing if any two separate ground truth skeletons became connected in the prediction; those connecting branches can be pruned. All these add complexity, but they directly target the TopoScore/VOI penalties.

- Metric-driven validation: Unlike simpler competitions where one might just use Dice or IoU on a validation set, here one should compute the competition metric (or a close proxy) on validation. The organizers provided a reference notebook with the metric implementation, which allows participants to test their approaches against all three components. For instance, one might notice that a certain model has great Surface Dice but poor TopoScore, indicating it often merges layers – that feedback would lead you to adjust thresholds or architecture to be more conservative in connecting predictions.

In summary, the evaluation metrics heavily emphasize structure over mere per-voxel accuracy. A winning approach must achieve a high Surface Dice (accurate placement of surfaces) and maintain the correct count and continuity of surfaces for high TopoScore and low VOI (VOI is typically reported such that lower is better, but Kaggle might invert it to a score). This has steered competitors to focus on topology-preserving segmentation. Strategies like stricter post-processing, topology-aware losses, and careful threshold tuning are all influenced by these metrics. It's a refreshing change from pure Dice optimization – here "no gaps, no mergers" is the mantrascrollprize.substack.com, and the modeling approach must reflect that.

6. Best Practices to Avoid Topological Errors

Avoiding topological errors is absolutely critical in this competition. A "topological error" in this context means either: (a) a merger, where two distinct papyrus layers that should be separate end up connected in the prediction; or (b) a split/hole, where a single continuous surface in truth is broken into pieces in the predictionkaggle.com. These errors correspond to massive penalties in

TopoScore and VOI, so it's worth investing effort to prevent them. Here are some best practices:

- Use Conservative Post-processing: One straightforward practice is to post-process the raw model outputs to enforce topology rules. For example, perform a connected component analysis on the predicted binary mask. Compare the number of connected components in prediction vs. expectation (if you know how many layers are in that volume from ground truth in training, or you can estimate from context in test). If there are predicted components that look unreasonably large or merge multiple layers, try to split them. A practical way is using morphological operations:

    - To fix mergers: If two layers are connected by only a thin bridge of voxels, applying a slight morphological opening (which erodes the mask and then dilates it back) can sometimes break that bridge. You have to be careful not to break real surfaces; thus, one might restrict this to areas where the predicted thickness exceeds a threshold. Another technique is to compute the skeleton of the predicted surfaces and see if any skeleton branch connects what should be two separate sheets. If yes, you can cut that branch. In image segmentation literature, a method was described that uses boundary dilation, skeletonization, then another dilation to eliminate small holes and spurious connectionsijcai.org. Essentially: first dilate boundaries slightly to fill tiny gaps, then skeletonize (which reduces structures to their 1-voxel-wide spine, potentially eliminating thin side connections), then dilate back to thicknessijcai.org. This can remove small extraneous connections and fill small internal holes.

  - To fix holes: Small holes in a surface can be filled with a morphological closing (dilate then erode) or simply by detecting them via connected components (a hole in a surface appears as a small connected component of background entirely surrounded by the surface – one can fill those if they're below a size threshold). Another approach is to enforce that each predicted surface component should be roughly the size/shape of a papyrus sheet. If you detect an anomalously small segment fully inside a larger layer, that's likely a hole artifact; you can fill it.

- It's worth noting that heavy post-processing might also remove correct features if overdone, so many solutions apply these fixes in a targeted way (for example, only remove connections shorter than X voxels, or fill holes smaller than Y voxels). The IJCAI 2024 paper on topology-aware segmentation suggests these simple steps can dramatically improve topological accuracy without degrading overall Diceijcai.org.

- Thresholding Strategy: The choice of threshold on model probability output can affect topology. A higher threshold makes the segmentation sparser (reducing merges but potentially causing holes by missing weakly predicted parts). A lower threshold yields more continuous surfaces (avoiding gaps, good for recall) but can introduce false connections. You may consider a multi-threshold approach: e.g., use a lower threshold to ensure surfaces are mostly complete, but then specifically remove regions below a higher threshold if they look like outliers. Some ensembles use one model at a low threshold (high recall) and another at high threshold (high precision for tricky regions) and combine them in a way that fills holes but avoids obvious false positives. Monitoring how threshold affects the number of connected components is a useful validation exercise.

- Topology-aware Training Losses: Incorporating losses that penalize topological mistakes can guide the model to avoid them in the first place. One such loss is clDice (differentiable connectivity Dice), originally developed for tubular structure segmentation, which ensures overlap of skeletons of prediction and ground truthijcai.org. For papyrus surfaces, one can apply a similar idea: compute a skeletal representation of the surfaces (which would be something like a 2D sheet "skeleton", possibly too advanced to compute on-the-fly in 3D). However, one simplification is to apply clDice on 2D projections or cross-sections. For example, in each axial slice, the papyrus surfaces appear as lines; one could enforce that the predicted lines have continuous skeletons matching ground truth. This is complex to implement, but even a simpler surrogate like adding a penalty for each predicted connected component that does not correspond to a ground truth component could help. There has been research using persistent homology to define a loss that explicitly penalizes differences in connected components or holespmc.ncbi.nlm.nih.gov. While implementing a full homology-based loss might be beyond the scope of most Kagglers, using ideas from it (like penalizing each extra component or missing component) can be approximated via standard operations. For instance, ensure that for each ground truth surface, the predicted mask intersects it in one piece (not broken into multiple pieces) – if not, add a loss term proportional to the number of fragments. Similarly, penalize if one predicted component overlaps multiple ground truth layers.

- Regularization to discourage overgrowth: One cause of merges is the model being overzealous in predicting positives in ambiguous areas (like between layers). Regularizing the model to be confident only when appropriate can help. Techniques include:

    - Dropout and spatial dropout in the model to avoid overfitting to small spurious features.

    - Ensembling multiple epochs or models: an ensemble tends to only keep voxels that multiple models agree on, which often eliminates odd stray connections. (Ensembling was noted as a key factor by winnersgithub.com, and it inherently provides a form of regularization.)

    - Training with a slightly higher penalty on false positives in the loss (e.g., using a weighted binary cross-entropy where false positives in low-probability regions are penalized more, or using Dice with a focus on precision). This must be balanced because we also hate false negatives (holes), but if the baseline model tends to over-predict connective tissue between layers, adjusting class weights could push it to back off those areas.

- Data augmentation for topology: Interestingly, one can augment training data in ways that teach the model to avoid certain mistakes. For example, one could simulate a merge error in training data and explicitly label it as wrong, so the model learns to avoid it. Concretely, take two ground truth surfaces that are close and artificially connect them in the training mask, then train the model that this connected region should be background (i.e., teach it that connection is not a valid surface). This is a form of adversarial augmentation. It might be tricky to implement, but the idea is to inform the model that "if you see a connection forming between layers, it's likely an error." Alternatively, one can augment by removing small pieces of a surface (simulating a hole) and still label those as surface in the target, forcing the model to infer through the gap. This might make the model robust to partial data and less likely to leave a real gap.

- Leveraging Domain Knowledge: Domain-specific knowledge can also help avoid topology errors. For instance, we know papyrus sheets should not branch or merge – they are essentially parallel layers. If the model prediction shows a branching structure (one layer splitting into two), that's definitely wrong physically. One could incorporate a check for branching: e.g., in a given cross-section, each sheet should appear as one continuous curve, not splitting. If a prediction shows a "Y" shape connecting two layers, that could be detected and removed. Volume Cartographer and human segmenters used their understanding that sheets are mostly continuous surfaces without bifurcationscrollprize.org. Encoding such priors explicitly into a model is hard, but one can design rules in post-processing to reflect them.

- Validation and manual inspection: It's a best practice to frequently inspect the 3D predictions overlayed on the CT (for a few validation samples) to visually catch topology issues. Sometimes the metrics will tell you there's a problem (e.g., VOI is high for a case), but looking at a 3D rendering of predicted vs true surfaces can reveal where the model is failing – often it might be in tricky regions like where two scroll layers are extremely close or touching. Focusing your model's attention on those hard cases (via either data augmentation or specialized sub-models) can reduce errors. For example, if you notice the model often merges layers when they are less than 2 voxels apart, you could try a distance-feature: provide as input a map of "distance to the nearest known surface" (from a first-pass prediction or from some heuristic) so that the model is aware "this area is a narrow gap, be careful." This is speculative but shows how analysis can lead to features that address the root cause of errors.

- Ensemble different error profiles: If you have one model that tends to never miss a surface (no holes, good recall) but sometimes connects layers (lower precision), and another model that is very clean but occasionally misses fine details, combining them cleverly can yield a result better than either. For instance, you could take the intersection of their predictions to ensure no extras, then add back any pieces from the high-recall model that are disconnected in the high-precision model's output but clearly should be there (perhaps identified by being a large portion of a known ground truth surface). This kind of ensemble post-processing was effectively used in some medical competitions to fix topology: one network ensures connectivity (fills gaps), another ensures no merges, and together you try to satisfy both. Since TopoScore demands both no merges and no gaps, achieving this might require multi-model cooperation.

In practice, the top teams will likely devote a lot of their submission paper to how they handled topological errors, because that's the differentiator in this competition. Simple segmentation networks can get a high Surface Dice, but making sure each papyrus layer is perfectly traced with continuity is harder. Summarizing best practices: treat topology errors as first-class citizens – actively search and destroy them. Use post-processing like hole-filling and bridge-cutting, consider topology-aware losses like clDice or persistence-based penalties, adjust your model's decision threshold to balance precision/recall, and test your pipeline on known tricky scenarios (like very close layers) to ensure it doesn't break. Kaggle organizers explicitly highlighted "no gaps, holes, or mergers"scrollprize.substack.com, so aligning your pipeline with that slogan is key.

7. Common Pitfalls and Practical Tips from Previous Challenges

This competition builds on experience from earlier Vesuvius Challenge iterations (notably the 2023 Kaggle Ink Detection competition) and shares similarities with other 3D segmentation contests (such as biomedical or connectomics challenges). Here are some common pitfalls and tips gleaned from those:

- Pitfall: Ignoring Topology Until the End. A mistake teams might make is to focus solely on improving Dice during development and only realize late that their model has severe topological flaws. Given the metric, this is fatal. In the Vesuvius Ink Detection competition, many solutions that performed well combined models and used domain-specific adaptationsscrollprize.orggithub.com. By analogy, for surface detection you should monitor not just the usual loss but also how many splits/merges your predictions have. Tip: Incorporate the TopoScore/VOI evaluation early in your validation pipeline. For instance, one can run a mini-evaluation that counts connected components of each predicted mask vs ground truth. If those counts don't match, figure out why. Being proactive here can save you from leaderboard surprises.

- Pitfall: Overfitting to Provided Data Distribution. The training data might come from specific scroll fragments or a specific scan, which could have particular characteristics (noise pattern, intensity range, etc.). If the test data is from different scrolls or scanned under slightly different conditions, a model that overfits to the training distribution may falter. For example, perhaps all training chunks have 5 layers visible and the model implicitly expects 5; if a test chunk has 6, will it erroneously merge two into 5? Or maybe the training scroll had slightly higher contrast between papyrus and background, and the model wasn't exposed to lower contrast scenarios. Tip: Use robust augmentations (as discussed) to expose the model to variation. Also, if unlabeled data or public data is available (the organizers provided some full scroll scans as unlabeled OME-Zarr formatkaggle.com), use them to sanity-check your model's behavior. Although you can't label them, you might run your model on a full scroll to see if it starts hallucinating or missing entire sections due to distribution shift. Domain adaptation techniques (used by one of the ink prize winners to adapt fragment models to whole scrollsscrollprize.org) might be relevant if, say, you train on small CT chunks and need to apply to a full scroll scan.

- Pitfall: Insufficient Patch Size / Context. If your input patch is too small (e.g., a network only sees a 64×64×16 voxel region), it might not have enough context to distinguish adjacent layers. A common error is that a model might confuse which surface is which if it cannot "see" the separation a bit further out. This can cause merges or misaligned continuation of a surface. Tip: Aim for as large a receptive field as memory allows. Use striding/pooling in the network to capture a global view. If memory is a bottleneck, consider a two-stage approach where a first network processes a downsampled larger patch to get contextual features, which are then used by a second network on a smaller crop. This mimics multi-scale processing. Ensure that at least the distance between layers (which might be just a few voxels in tight spots) is well within the network's view – ideally the network sees several layers up and down so it learns the pattern "there's a gap here, so don't connect these two." Some top solutions might use multi-scale inputs, feeding both a high-res crop and a low-res larger area to the network, concatenated as separate channels, to get the best of both worlds.

- Pitfall: Excessive False Positives in Gaps. Some models might light up any slight density difference as "surface", including noise or artifacts between layers. These false positives in the gap can create bridges. This often happens if the threshold is too low or the model is too sensitive. Tip: One practical trick is to use the fact that papyrus surfaces are relatively smooth and coherent. If you get tiny isolated blobs of prediction far from any other predicted surface, they are likely false. Removing very small connected components of prediction (below a certain voxel count) can eliminate speckle that could otherwise attach wrongly to a real surface. Similarly, if your model sometimes outputs two parallel surfaces very close (suggesting it marked both sides of a papyrus sheet instead of the middle), you may choose one side based on intensity or geometry. For example, maybe the inner side of the sheet has a slightly different intensity profile; if known, that could guide which surface is real. Lacking that, you could simply thin the double-surface to one via skeletonization.

- Pitfall: Neglecting Edge of Volume Effects. If a papyrus layer goes out of the provided chunk (for instance, a layer might start at the edge and continue beyond the chunk), the ground truth mask might show it ending at the border. A model could mistakenly think the layer truly ends there and maybe merge it with something at the border. Tip: Whenever possible, pad the volume during inference so that the model doesn't get confused by truncated surfaces. Also, if the competition data is split into chunks arbitrarily, realize that a surface touching the boundary might actually continue in an adjacent chunk that's not seen. You can't recover what you don't see, but you can at least avoid trying to "cap off" the surface in a weird way. This is more about careful interpretation – in evaluation, they probably only measure within each provided volume, so it's fine, but be cautious in assumptions about surfaces at volume boundaries.

- Learn from Connectomics: A very similar problem exists in connectomics, where segmenting neuronal membranes in EM images requires no mergers (don't merge two neurons) and no splits (don't break a neuron's membrane). They use metrics like VOI and Rand Index, just like hereijcai.org. In those challenges, top methods often used over-segmentation then merge strategies or special losses. One proven approach is to first intentionally over-segment (produce more fragments than necessary, but no merges), then merge fragments that belong together using a secondary algorithm. For scrolls, an analogous approach could be: ensure your initial segmentation might err on the side of splitting layers (if uncertain, leave a gap rather than connect), then use a second step to join pieces that have strong evidence of being the same layer (for example, if two broken segments align well and there's no gap in intensity between them, one could connect them). This two-step strategy (first avoid merges at all cost, then fix splits) aligns with the idea that VOI is more forgiving of splits than merges – a split can be fixed by a later merge, but a merge (two true objects as one) is hard to disentangle without ground truth. So if unsure, prefer a hole over an incorrect connection; you might fill the hole later with a heuristic.

- Read Competition Discussions and Notebooks: Often, Kaggle competitions have rich discussions. Since this competition is ongoing (and perhaps by 2025 some preliminary discussions or notebooks are public), it's wise to read those. For example, a discussion might point out that the test volumes come in certain sizes (256, 320, 384) and you should code your model to handle various sizeskaggle.com. Or someone might share a starter notebook with 3D U-Net implementation and some findings (like "increasing depth of the U-Net helped" or "using GroupNorm was necessary for stable training"). Utilizing these community insights can save time. One Kaggle notebook example was an advanced 3D U-Net with a topology-aware designkaggle.com – if provided, studying its architecture and augmentations is valuable. Kaggle is as much a community effort as a competition, and past experience (like how the ink detection winners approached ensemble, data processing, etc.) is instructive. The ink detection winning solution's README highlights the importance of ensemble, data augmentations (rotations, flips), and the use of a robust segmentation model (they used SegFormer in stage 2)[github.com](github.com). By analogy, don't shy away from ensembling different approaches for surfaces, and ensure your augmentations cover the possible range of scroll presentations.

- Time Management Pitfall: 3D models are slow to train and infer. A common pitfall is running out of time to iterate because each experiment is costly. Tip: Use smaller validation experiments to test ideas. For instance, try training on a downsampled volume or a smaller region to quickly see if a loss function or architectural change yields fewer topological errors, before committing to full resolution. Also, use mixed precision and efficient libraries to speed up training. Many Kagglers utilized GPUs like A100s or multiple 3090s; if resources are limited, consider using Kaggle's notebooks or Google Colab for extra compute, or optimize your code (e.g., use MinkowskiEngine for sparse 3D conv if applicable, since surfaces are sparse structures).

- Documentation and Visualization: Keep track of what methods you tried and how they affected the metrics. It's easy to get lost in numerous tweaks. Since the output is 3D, using visualization tools (perhaps exporting a few predictions to MeshLab or ParaView) can give insights. One practical tip: convert your predicted surfaces to a mesh (via marching cubes) – then you can visually inspect if the mesh has holes or extra connections. This is how you ultimately would "unwrap" it, so it's a good sanity check. If the mesh looks messy, the segmentation likely needs improvement.

In conclusion, previous competitions (Vesuvius Ink and similar segmentation tasks) teach us to balance ambition with caution: use powerful models and ensembling (for accuracy), but always guard against the edge cases that ruin topology. Many teams will likely achieve ~80-90% Surface Dice, but the winner will be the one who also nails the topology (perhaps by a combination of careful loss design and clever post-processing). By learning from past pitfalls – overfitting, ignoring topology until too late, lack of context – and implementing the tips above, one can formulate a strong approach to the Vesuvius Challenge Surface Detection task. The end result, with cleanly segmented papyrus surfaces, will bring us one step closer to virtually unrolling and reading these ancient textsscrollprize.substack.com – a prime example of AI and cultural heritage coming together.

References and Resources:

Vesuvius Challenge (Surface Detection) – Competition Overviewscrollprize.substack.comscrollprize.substack.com

Scrollprize Blog – “Back to the Challenge: $100K Kaggle Surface Detection” (Nov 2025)scrollprize.substack.comscrollprize.substack.com

Scrollprize Grand Prize Report (2023) – Context on scanning, segmentation, and virtual unwrapping pipelinescrollprize.orgscrollprize.org

Kaggle Vesuvius Ink Detection Winning Solution (GitHub README) – Two-stage model with 3D UNet/UNETR and 2D SegFormer, ensemble, etc.github.comgithub.com

IJCAI 2024 Paper on Topology-Preserving Segmentation – discusses metrics (VOI, Betti) and skeleton-based fixesijcai.orgijcai.org

npj Digital Medicine Paper on 3D Segmentation (PAM) – defines Normalized Surface Dice (NSD)nature.com

Kaggle Forum Discussions – various snippets on metric definition and strategykaggle.comkaggle.comkaggle.com (access via Kaggle).

Kaggle Notebooks – e.g., 3D U-Net baseline, Zur AI topology-aware U-Netkaggle.com (for code inspiration; ensure to cite any external code if used).

Connectomics Challenges (like SNEMI3D) – for understanding VOI and topological errors in segmentationijcai.org.