RSNA 2025 Challenge: The Definitive Master Blueprint

An End-to-End Guide for a Physics-Informed, Oscillatory, and Topologically-Aware Network Strategy

1. The Core Philosophy: Why Our Approach Can Surpass the SOTA 🚀

Before detailing the technical architecture, it is crucial to understand the strategic philosophy that provides our competitive edge. Current State-of-the-Art (SOTA) models like **Swin UNETR** are exceptional **Pattern Recognition Experts**. They are akin to a brilliant radiologist who has memorized millions of scans; they know *what* an aneurysm looks like with unparalleled morphological accuracy.

Our proposed model, however, is designed to be a **Biomedical Physics Council**. It doesn't just learn *what* an aneurysm looks like; it is trained to understand *why* it is a pathological entity from first principles.

- It understands the disruption of **dynamical harmony** in anatomical features through **Artificial Kuramoto Oscillatory Neurons (AKOrN)**. It detects an aneurysm as a "dissonant note" in a biological symphony.
- It understands the fundamental **physical laws of fluid dynamics** through a **Physics-Informed Neural Network (PINN)**. It detects an aneurysm as a structure that creates a physically implausible blood flow pattern.

This multi-layered reasoning provides profound robustness. When faced with an ambiguous case where morphological patterns are unclear, our model can make a decision based on these deeper, fundamental principles. This is our path to victory.

2. The Definitive End-to-End Workflow 🗺️

This section details the complete, step-by-step workflow. Each phase is designed to build upon the last, creating a powerful and coherent system.

Phase 0: Foundational Asset Creation (Offline Preparation)

This strategic preparatory phase creates two critical, reusable assets, saving immense time and boosting performance.

- **Asset 1: The Pre-trained Backbone Model**
  - **Objective:** To create a powerful, general-purpose feature extractor with a deep understanding of neurovascular anatomy.
  - **Data Corpus:** An assembly of the **OpenMind Dataset** (for MRI diversity) and

the **NIH DeepLesion Subset** (for CT diversity).
    - **Methodology:** A **WaveFormer** backbone will be pre-trained using the **SparK** self-supervised algorithm, with a **Hierarchical Masking (MiM)** strategy to effectively learn vascular tree structures.
    - **Outcome:** A set of pre-trained WaveFormer weights—our "expert anatomist" model.
- **Asset 2: The Vessel Segmentation Masks**
    - **Objective:** To generate a high-quality vessel segmentation map for every scan in the official competition dataset.
    - **Data Corpus:** The segmentations/ directory provided in the competition data.
    - **Methodology:** A lightweight 3D U-Net will be trained on this subset using a **Tversky Loss** (e.g., α=0.7, β=0.3) to achieve a vessel recall of >98%. This model will then infer vessel masks for the entire dataset.
    - **Outcome:** A complete set of NIfTI vessel masks.

Phase 1: Candidate Generation ("The Smart Magnetic Net")

This stage intelligently filters the entire 3D scan to produce a short list of high-quality candidates. It is a two-step process that combines morphological and topological analysis.

- **Step 1.1: Morphological Feature Extraction**
    - **Architecture:** A lightweight and efficient **SegFormer3D** model.
    - **Input:** The full 3D scan volume.
    - **Process:** The model performs a fast scan to produce a 3D probability map, highlighting all voxels that *look like* they could be part of an aneurysm.
    - **Output:** A 3D probability map.
- **Step 1.2: Topological Anomaly Detection (The VGN Framework)**
    - **Architecture:** This is not a single model but a pipeline that processes the output of Step 1.1.
    - **Process:**
        1. **Probability Map → Binary Mask:** The probability map is thresholded to create a high-confidence binary mask of the vasculature.
        2. **Binary Mask → Skeleton:** A 3D morphological **skeletonization** algorithm is applied to extract the 1-voxel-thick centerline of the vessel tree, preserving its topology.
        3. **Skeleton → Graph Construction:** The skeleton is traversed to build a mathematical graph. **Nodes** are defined at junction points and endpoints. **Edges** are the vessel segments connecting these nodes. Features like local radius and segment length are assigned to nodes and edges.
        4. **Graph Analysis:** A **Graph Attention Network (GAT)** is trained to analyze

this graph. The GAT learns the rules of normal vascular topology and assigns a high anomaly score to nodes that violate these rules (e.g., a short, stubby edge leading to a high-radius node).
    - **Output:** A final, refined list of (x, y, z) coordinates for candidate regions that are both morphologically suspicious and topologically anomalous.

Phase 2: Candidate Classification ("The Expert Classifier")

This is the core of the system, where each candidate is scrutinized by our most advanced model.

- **Input (for each candidate):**
    1. A 3D patch (e.g., 96x96x96) from the **original image**.
    2. The corresponding 3D patch from the pre-computed **vessel mask**. These are stacked into a 2-channel input.
    3. Patient **metadata** (Age, Sex).
- **Internal Architecture & Layer-by-Layer Reasoning:**
    1. **Layer 1 (Input & Backbone):** The 2-channel 3D patch enters the pre-trained **WaveFormer** backbone. It efficiently extracts a rich, multi-scale, static feature map representing the spatial information.
    2. **Layer 2 (Dynamic Analysis):** The static feature map is reshaped via **Adaptive 3D Pooling** and fed into **AKOrN** blocks (placed via **Sequential Augmentation**). Here, the features act as stimuli for a system of coupled oscillators. The AKOrN block performs a dynamic simulation, answering the question: "Do these spatial features form a harmoniously synchronized system?" The final state of these oscillators becomes a new, dynamic feature representation.
    3. **Layer 3 (Contextual Fusion):** Patient metadata is embedded by an MLP and fused with the dynamic features from AKOrN using a **Gated Attention (FiLM-style)** mechanism. This allows the model to conditionally adjust its reasoning based on patient-specific risk factors.
    4. **Layer 4 (Multi-Task Prediction & Regularization):** The final, fused feature vector is passed to four parallel heads:
        - **Head 1 (Primary Task):** Predicts the 14 competition labels.
        - **Head 2 (Anatomical Regularizer):** Reconstructs the vessel mask within the patch.
        - **Head 3 (Localization Regularizer):** Predicts a heatmap of the aneurysm's center.
        - **Head 4 (Physics Regularizer):** A **PINN** head acts as a "physics auditor." It takes sampled coordinates from the vessel mask, predicts blood flow properties, and calculates a loss based on the **Navier-Stokes equations**.

The gradient from this physics loss flows back to the WaveFormer backbone, forcing it to learn physically plausible representations.

- **Training Loss & Optimization:**
  - A composite loss function combines the outputs of all four heads. The primary task head uses a hybrid L_ASL + L_AUC_Margin loss.
  - The weights for each task's loss are balanced automatically and efficiently using **Uncertainty Weighting**.
  - To mitigate PINN training failures, we will use **FP64 precision** for the physics calculations and an **R3 adaptive sampling** strategy.

Phase 3: Final Ensemble & Submission

- **Methodology: A Novel Stacking Ensemble**
  1. **Level-1 "Model Zoo":** We will train multiple variants of the full pipeline (e.g., from different cross-validation folds, with/without AKOrN/PINN) to create a diverse set of base models.
  2. **Level-2 Meta-Learner:** An **XGBoost** model will be trained on an enriched feature set. For each prediction from a level-1 model, we will provide the meta-learner with not just the **probability**, but also:
     - The **Kuramoto order parameter** from AKOrN (a measure of internal confidence).
     - The **physics residual loss** from the PINN (a measure of physical plausibility).
  - **Advantage:** This "secret weapon" allows our ensemble to learn sophisticated rules, such as "distrust a high-probability prediction if it is physically implausible."

3. Data Harmonization & Preprocessing Pipeline ⚙️

A rigorous, unified preprocessing pipeline is non-negotiable for success. All data, from both the pre-training corpus and the competition, will be processed through these steps.

1. **Format Conversion:** All source data (DICOM, PNG) will be converted to **NIfTI (.nii.gz)**. SimpleITK is the recommended tool for DICOM conversion. For the DeepLesion dataset, a custom script will be used to reconstruct 3D NIfTI volumes from the 2D PNG slices and metadata.
2. **Orientation Standardization:** All volumes will be reoriented to a canonical **RAS** (Right-Anterior-Superior) orientation using tools like monai.transforms.Orientationd.
3. **Isotropic Resampling:** All volumes will be resampled to a consistent **1.0mm x 1.0mm x 1.0mm isotropic voxel spacing**. This is the most critical step for

ensuring physical consistency. Use **B-spline or bilinear interpolation** for intensity images and **Nearest Neighbor interpolation** for segmentation masks.

4. **Modality-Specific Intensity Normalization:**
    - **CT/CTA Data:** Apply a **multi-channel windowing** technique. Create a 3-channel image from the single CT volume, representing the Brain Window, Blood/Subdural Window, and Bone Window.
    - **MRI/MRA Data:** Apply **Nyúl-style histogram matching** to standardize the intensity distributions across the highly heterogeneous MRI scans. The intensity-normalization Python package is recommended.
5. **Brain Extraction (Skull-Stripping):** This step will be **avoided by default** due to the high risk of clipping peripheral vasculature. It will only be considered if VRAM limitations are insurmountable. If used, the SOTA tool **SynthStrip** is required, with a mandatory visual quality assurance check.
6. **VRAM-Aware Training Strategy:**
    - **Offline Harmonization:** The entire preprocessing pipeline will be run once offline to create a "preprocessed dataset".
    - **Online Augmentation:** During training, the data loader will read from this preprocessed data and perform lightweight, on-the-fly augmentations. **Patch-based training** (e.g., on 96x96x96 or 128x128x128 patches) is mandatory to fit within the 16-24GB VRAM constraint. monai.data.CacheDataset will be used to cache the entire dataset in RAM for maximum throughput.

4. Foundational Resources & Rationale 📚

This section provides the scientific and technical underpinnings for each key component of the blueprint, integrating findings from the latest research.

Phase 0: Foundational Assets

1.1 Pre-trained Backbone: SparK & Hierarchical Masking (MiM)

- **Foundational Papers:**
    - **SparK:** Tian, K., et al. (2023). "Designing BERT for Convolutional Networks: Sparse and Hierarchical Masked Modeling".
    - **MiM:** Zhuang, J., et al. (2024). "MiM: Mask in Mask Self-Supervised Pre-Training for 3D Medical Image Analysis".
- **Strategic Rationale:** The combination of SparK and MiM is a deliberate choice for robustness and anatomical awareness. SparK is proven to be superior to contrastive methods in data-scarce scenarios, a critical advantage for rare aneurysm subtypes. MiM's hierarchical masking provides a powerful inductive

bias for learning the tree-like structure of the vasculature, which is perfectly aligned with our downstream task.
- **High-Quality Implementation:**
  - **SparK:** Official PyTorch Repository
  - **MiM:** While an official repo is pending, a strong starting point is HybridMIM on GitHub, which implements a closely related concept.

1.2 Pre-trained Backbone: WaveFormer

- **Foundational Paper:** Perera, S., et al. (2024). "WaveFormer: A 3D Transformer with Wavelet-Driven Feature Representation for Efficient Medical Image Segmentation".
- **Strategic Rationale:** WaveFormer is chosen for its computational efficiency. By applying self-attention only to the low-frequency components of a wavelet-decomposed feature map, it drastically reduces VRAM and computational load while retaining the ability to model long-range dependencies. This efficiency is what creates the necessary "computational budget" for our more advanced AKOrN and PINN modules.
- **Implementation Note:** A public implementation for the 3D medical WaveFormer is a known gap. This component will require a custom implementation based on the detailed specifications in the foundational paper.

1.3 Vessel Segmentation: Tversky Loss

- **Foundational Paper:** Salehi, S. S. M., et al. (2017). "Tversky loss function for image segmentation using 3D fully convolutional deep networks".
- **Strategic Rationale:** Our two-stage pipeline is critically sensitive to false negatives (missed vessels) in the auxiliary segmentation mask. The Tversky Loss directly addresses this by allowing us to heavily penalize false negatives more than false positives (by setting $\alpha > \beta$). This explicitly optimizes the segmentation model to maximize recall, mitigating the most critical failure mode of the entire system.
- **High-Quality Implementation:** A well-regarded PyTorch implementation is available in this Kaggle Loss Function Library.

Phase 1: Candidate Generation

2.1 Morphological Extraction: SegFormer3D

- **Foundational Paper:** Perera, S., et al. (2024). "SegFormer3D: An Efficient Transformer for 3D Medical Image Segmentation".
- **Strategic Rationale:** For the candidate generation stage, speed and memory efficiency are paramount. SegFormer3D's lightweight, all-MLP decoder design

makes it significantly more efficient than other SOTA models, making it the ideal choice for a fast first-pass analysis of full 3D volumes.
- **High-Quality Implementation:** Official PyTorch Repository

2.2 Topological Analysis: VGN Framework

- **Foundational Papers:**
  - **VGN Concept:** Shin, S. Y., et al. (2019). "Deep vessel segmentation by learning graphical connectivity".
  - **GATs:** Veličković, P., et al. (2018). "Graph Attention Networks".
- **Strategic Rationale:** This framework is the "intelligence" of our Smart Net. A standard CNN might flag a simple vessel bend as a candidate. The VGN framework, however, builds a graph of the vasculature and uses a GAT to analyze its topology. The GAT can learn that a sudden, short edge leading to a high-radius node is a topologically anomalous pattern highly indicative of an aneurysm, allowing it to filter out morphologically simple but topologically normal false positives.
- **High-Quality Implementation:**
  - **Skeletonization:** skimage.morphology.skeletonize_3d
  - **GATs:** An excellent annotated PyTorch implementation can be found at LabML AI.

Phase 2: Candidate Classification

3.1 Dynamic Analysis: AKOrN

- **Foundational Paper:** Miyato, T., et al. (2025). "Artificial Kuramoto Oscillatory Neurons".
- **Strategic Rationale:** AKOrN replaces a static MLP block with a dynamic simulation. It allows the model to "bind" related features through synchronization. This provides a powerful mechanism to determine if the features extracted from a candidate patch form a coherent, harmonious system (healthy) or a dissonant, unsynchronized one (pathological).
- **High-Quality Implementation:** Official PyTorch Repository

3.2 Contextual Fusion: FiLM-style Gated Attention

- **Foundational Paper:** Perez, E., et al. (2018). "FiLM: Visual Reasoning with a General Conditioning Layer".
- **Strategic Rationale:** This is far superior to simple late-fusion. A FiLM-style layer allows patient metadata (age, sex) to act as a "controller," dynamically modulating the image feature channels. The model can learn to amplify or suppress certain visual features based on patient-specific risk factors.

- **High-Quality Implementation:** A robust implementation exists in PyTorch Geometric as torch_geometric.nn.conv.FiLMConv, the core logic of which is directly transferable.

3.3 Physics Regularizer: PINNs

- **Foundational Paper:** Raissi, M., et al. (2019). "Physics-informed neural networks...".
- **Strategic Rationale:** The PINN head acts as a "physics auditor." It regularizes the main backbone by penalizing it if it learns feature representations that correspond to physically implausible blood flow (i.e., violate the Navier-Stokes equations). This injects a powerful, fundamental prior into the model, improving generalization.
- **High-Quality Implementation:** While a full multi-task implementation is novel, the core logic for calculating Navier-Stokes residuals in PyTorch can be adapted from repositories like this PINN for Turbulence repo.

3.4 Advanced Loss Functions

- **Foundational Papers:**
  - **ASL:** Ridnik, T., et al. (2021). "Asymmetric Loss for Multi-Label Classification".
  - **AUC Margin Loss:** Yuan, Z., et al. (2021). "Large-Scale Robust Deep AUC Maximization...".
- **Strategic Rationale:** This hybrid loss directly tackles the two biggest challenges: ASL's asymmetric focusing handles the extreme class imbalance, while the AUC Margin Loss directly optimizes a surrogate for the competition's ranking-based metric.
- **High-Quality Implementation:**
  - **ASL:** Official PyTorch Repository
  - **AUC Margin Loss:** Official PyTorch Library: LibAUC

3.5 Loss Balancing: Uncertainty Weighting

- **Foundational Paper:** Kendall, A., et al. (2018). "Multi-Task Learning Using Uncertainty to Weigh Losses...".
- **Strategic Rationale:** With four different loss terms, manual weighting is intractable. Uncertainty Weighting is the ideal solution for our computationally constrained environment as it automatically balances the losses with negligible computational overhead compared to expensive gradient-based methods.
- **High-Quality Implementation:** A clean PyTorch implementation is available.

Phase 3: Final Ensemble

4.1 Stacking Ensemble

- **Foundational Paper:** Wolpert, D. H. (1992). "Stacked Generalization".
- **Strategic Rationale:** Our novel application of stacking uses interpretability metrics (AKOrN's order parameter, PINN's physics loss) as meta-features. This allows the level-2 XGBoost model to learn not just *what* the base models predict, but *how reliable and physically plausible* those predictions are, creating a more intelligent final arbiter.
- **High-Quality Implementation:** Standard libraries like scikit-learn (for StackingClassifier) and xgboost provide all the necessary tools.