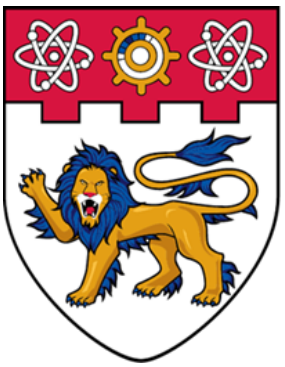

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC4000 Machine Learning

HuBMAP - Hacking the Kidney

Group 9

Name	Email (excluding @e.ntu.edu.sg)	Matriculation Number	Contributions
Durvasula Satya Sai Vasanth	durv0002	U2222819D	Ideated, Designed, and Implemented SAM Integration, ResNeXt-Mamba, and ResNeXt Models. Did Preprocessing, Training, Hyperparameter Tuning, Inference, Report Writing and Refinement, Slide Preparation and Presentation
Sunkara Bhargavi	bhargavi005	U2222578K	Literature Review, Pre-processing, Hyperparameter Tuning, Inference, Report Writing and Refinement, Slide Preparation and Presentation
Yu Boxuan	byu005	U2221435A	Manual Labels Preparation, ResNeXt-50/101 Implementation, Report Writing and Refinement, Slide Preparation and Presentation
Tejeswara Nehru	tejeswar001	U2220197K	Report Writing, Slide Preparation, Presentation and Video Editing
Priyadharshiny Rajasekaran	priy0022	U2221954B	Slide Preparation and Presentation

Table of Contents

1. Introduction

1.1 Significance of issue

Maintaining kidney health is crucial for overall well-being, yet millions worldwide suffer from chronic kidney disease and related disorders. Kidneys perform vital functions like filtering blood, regulating pressure, and balancing electrolytes, but a detailed understanding of their cellular structure and functional tissue units (FTUs) remains incomplete.

The Human BioMolecular Atlas Program (HuBMAP) seeks to bridge this gap by creating comprehensive cellular maps of the human body, beginning with the kidney's glomeruli. Glomeruli, the essential FTUs responsible for initial blood filtration, are central to this effort. Accurately detecting and segmenting these structures in complex histopathological images is a critical but challenging step towards analyzing kidney function at a cellular level, ultimately accelerating disease research, improving diagnostics, and advancing personalized medicine.

1.2 Motivation for project

We are drawn to projects where machine learning can make a real difference in healthcare. Working on Hubmap's open atlas feels like a great fit - it is technically challenging, directly relevant to clinical research, and gives us a chance to apply and test vision models on complex, real-world data. By focusing on glomerulus detection, we hope to demonstrate that thoughtful ML engineering can accelerate translational kidney research.

1.3 Problem statement

Given high‑resolution Periodic Acid-Schiff (PAS)‑stained images that vary widely by specimen preparation pipeline, develop a supervised semantic‑segmentation model that detects and segments glomeruli. Performance is assessed with the mean Dice coefficient, i.e. the average overlap between predicted and ground‑truth masks at the pixel level. The chief challenge is building a model that generalises across variations in staining, scanning equipment, and tissue‑processing.

1.4 Our Contributions

1.4.1 Dual SAM Integration Strategies

We propose and evaluate 2 distinct approaches for incorporating SAM-generated pseudo masks into our segmentation pipeline:

1. **Early Fusion** - The SAM mask is concatenated as a fourth channel alongside RGB input, directly guiding the encoder (used in our ResNeXt-50 and ResNeXt-101 architectures)
2. **Late Fusion** - The SAM mask is passed through a separate convolutional feature extractor and merged with decoder features via channel-wise concatenation (used in our ResNeXt-Mamba).

Our findings demonstrate that both methods significantly enhance Dice performance, with the late fusion strategy preserving encoder pre-training while still leveraging spatial priors provided by SAM.

1.4.2 Hybrid Encoder-Decoder Architecture

Our most effective architecture combines multiple advanced modules, including:

1. A lightweight SAM feature extractor
2. BiFPN for multi-scale feature fusion
3. A mamba inspired decoder leveraging Selective State Space Modules (SSMs) for efficient long-range context modeling

1.4.3 Empirical Validation

We conduct ablation studies and report comparative results for all architectures. The use of SAM—whether via early or late fusion—consistently improves segmentation accuracy. Notably, the SAM guidance contributed the majority of the Dice score gain, highlighting its effectiveness as a weak supervision signal.

2. Preprocessing

2.1 Generating Image Tiles

The training dataset consists of 8 high-resolution TIFF images, each ranging from approximately 500 MB to 5GB in size. Due to their large dimensions, loading entire images into memory can exceed system resources. Working with full images can lead to slow processing times as well, thereby hindering model training and evaluation.

Our preprocessing strategy focused on splitting the original large TIFF images into smaller, non-overlapping tiles that could be processed independently. This results in a reduction of memory consumption and is more computationally effective.

2.1.1 Manual Labels

Several online discussions highlighted inconsistencies in the provided ground truth masks, with some glomeruli missing annotations. To address this and enhance model performance, we supplemented the dataset with manually labeled masks [1].

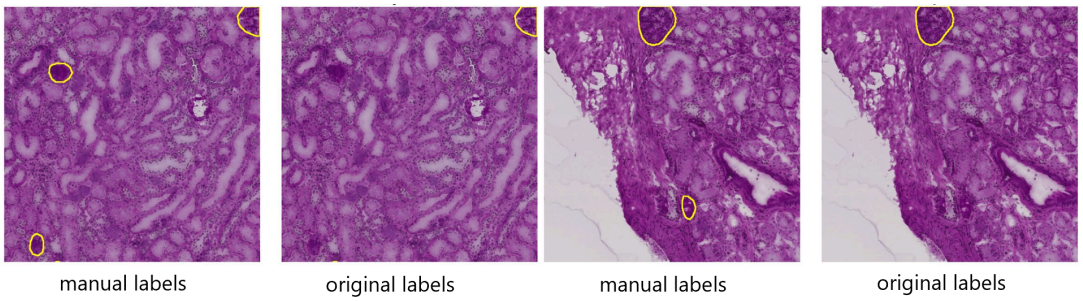

Fig 1: Comparison between manual and original labels

2.1.2 RLE Mask Conversion

The glomeruli annotations (ground truth) are provided in Run-Length Encoding (RLE) format. It is a compact way of representing binary masks where each RLE string specifies a sequence of start positions and lengths. These strings are parsed and mapped back into a 2D mask array of the original image shape.

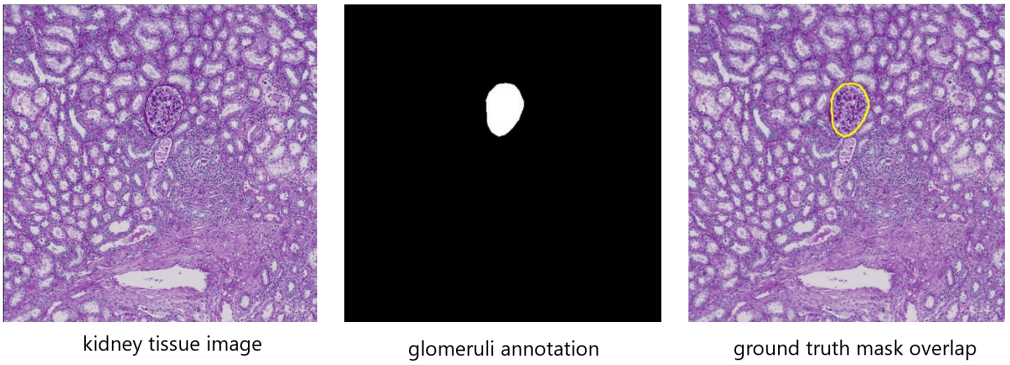

Fig 2: Kidney tissue image and ground truth mask

2.1.3 Tile Splitting and Padding

Each image is first divided into smaller tiles of size 2048*2048 pixels. Only the pixel data corresponding to the current tile is read into memory while processing, thereby optimizing resource usage.

Since the image dimensions are not always divisible by the tile size, padding was applied along the edges of the image. Black padding was added to ensure that the final tiles layout resulted in an integer number of complete tiles, therefore ensuring consistent input dimensions for model training.

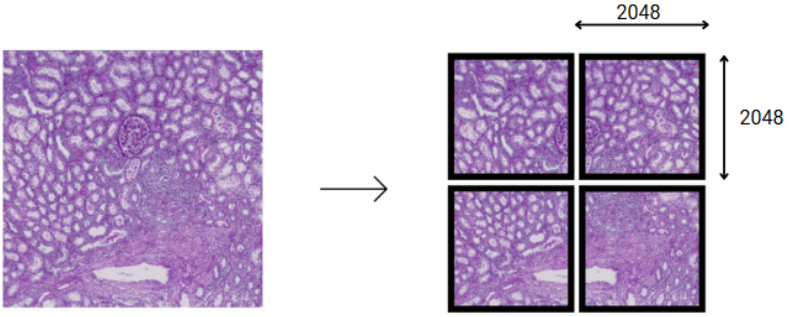

Fig 3: Tiling and Padding

2.1.4 Downsampling

To further reduce memory usage and computation cost, the tiles were downsized by a factor of 4, therefore resulting in 512*512 pixel tiles. This significantly decreased the resolution of each tile, reducing the computational burden during model training and evaluation, while still retaining sufficient structural detail for effective segmentation.

By focusing on smaller sections of the image, the segmentation model could more easily learn the features within those regions, leading to improved performance.

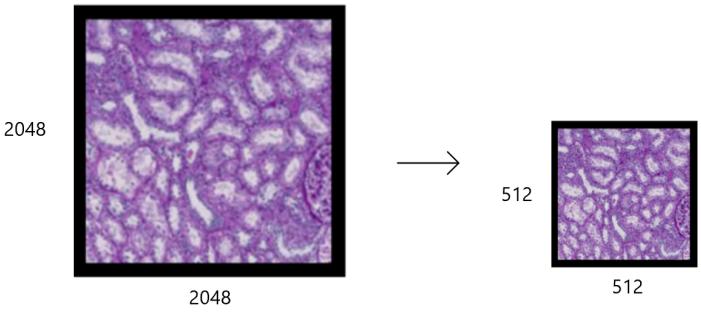

Fig 4: Downsampling by 4 times

2.1.5 Filtering Out Empty or Low Information Tiles

Few tiles lack relevant tissue structures or sufficient visual information for the segmentation task. To eliminate such tiles, we introduced a filtering step to our preprocessing pipeline.

Each tile was first converted to the HSV (Hue, Saturation, Value) color space. The saturation channel was then analyzed to assess the presence of meaningful color variation, which most often correlated with the presence of tissue. Tiles with a low number of pixels above a saturation threshold (set at 40) were excluded, as they likely contained only background or empty space with no meaningful tissue structure.

This filtering helped the model focus training on tiles that contain actual tissue structures and segmentation-relevant features, thereby improving data quality and model performance.

2.2 Using SAM Masks

2.2.1 Segment Anything Model (SAM)

The Segment Anything Model (SAM) is a high-performance, general-purpose segmentation model designed to produce high-quality masks with minimal user input. It has been trained on a dataset comprising more than 1 billion masks from 11 million diverse images. SAM's architecture is composed of three core components.

1. **Image Encoder** - A powerful image encoder computes rich, spatially aware embeddings of the entire input image, capturing both fine-grained local detail and broader contextual relationships.
2. **Prompt Encoder** - These embeddings are then conditioned using a flexible prompt encoder, which can handle various input prompts such as points, bounding boxes or existing masks.
3. **Mask Decoder** - A lightweight mask decoder combines the image and prompt embeddings to generate the corresponding segmentation masks.

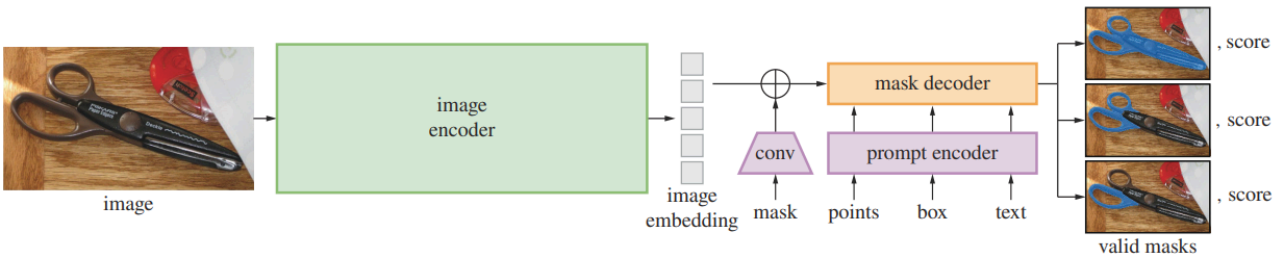

Fig 5: SAM Architecture [2]

The key strength of SAM lies in its zero shot segmentation ability, enabled by its strong generalisation capabilities and promptable architecture. This means that it can segment previously unseen objects or regions in new images without requiring additional fine-tuning or training.

2.2.2 Leveraging SAM as a Guidance Signal

Our method introduces a novel way of using SAM-generated masks as an additional guiding input channel during training. We modify the RGB (3-channel) kidney cells image input to a 4-channel input, where the fourth channel is the SAM-generated pseudo mask. This approach treats the SAM mask not as a source of hard supervision but as a contextual guide. It offers spatial hints and object-level cues that assist the model in focusing on relevant structures.

By incorporating the pseudo mask directly into the input, the model can learn to associate visual features with the rough location and shape of relevant anatomical regions. This helps especially in early training stages, where the model might otherwise struggle to localize glomeruli from labels alone. Additionally, this helps in improving data efficiency and guiding the model's attention even in noisy conditions.

We further tested a late fusion strategy where the SAM-generated pseudo mask is not concatenated to the input but instead is passed through a lightweight convolutional feature extractor. These SAM features are fused with deep decoder features after upsampling (e.g., using BiFPN or direct concatenation before the final convolutional prediction layer). This method allows the RGB encoder to remain pre trained and untouched, potentially preserving pretrained features while still benefiting from SAM context.

Recent literature has shown the effectiveness of SAM in similar roles in the medical image segmentation domain. For instance, SimTxtSeg uses SAM with text-guided prompts to generate pseudo-labels [3] and MedCLIP-SAMv2 combines SAM with CLIP for accurate zero-shot segmentation [4]. These studies collectively validate the use of SAM-generated masks as a powerful tool for enhancing segmentation performance with minimal annotation cost, supporting our approach of using SAM as a guiding channel in histopathology.

2.2.3 SAM Pseudo Mask Generation Pipeline

To generate pseudo masks, we adopt a multi step strategy that combines blob detection, SAM predictions and morphological cleaning.
The pipeline is as follows -

1. **Blob Detection** - The RGB image is first converted into a grayscale image. We then use the Laplacian of Guassian(LoG) blob detection method to detect the candidate glomeruli regions with the image. LoG is well-suited for identifying roughly circular, blob-like structures in an image by detecting areas of rapid intensity change across multiple scales.[5] Hence, this method was used for locating glomeruli, which typically appear as dense, rounded regions with distinct intensity profiles.
2. **Prompting SAM** - These blob centers are then used as positive point prompts for SAM, which leverages them to generate segmentation masks around the blobs, which are predicted glomeruli locations. One mask per point is generated and all these masks are merged to form a binary pseudo mask.
3. **Morphological Cleaning** - The segmented regions are enhanced by removing small irrelevant objects and filling in minor holes. This helps reduce the noise and enhance the shape consistency of segmented regions.

Finally, these cleaned masks obtained are then saved in a designated directory for use during training.

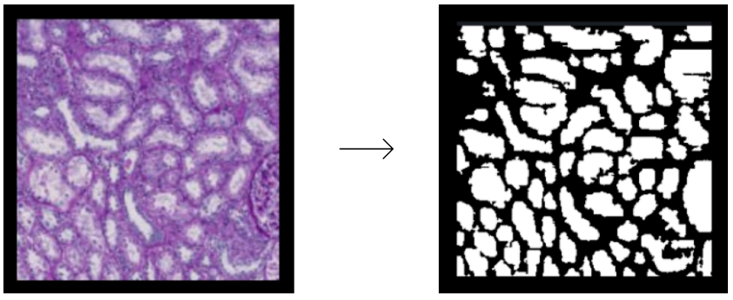

Fig 6: Tile and corresponding SAM Mask

2.3 Train Data Augmentation

To make the model more robust to variability in tissue appearance, orientation and staining, we use the Albumentations library to apply a range of augmentations to each training tile and their corresponding SAM generated masks. These include random flips, 90° rotations, and affine transformations (±15° rotation, ±20% scaling, ±6.25% shifting) with mirrored padding to avoid edge artifacts. With a 30% probability, we also apply a randomly chosen non-linear distortion—such as elastic, optical, or grid distortion, Gaussian blur, or noise. Color augmentations include random hue, saturation, and brightness shifts, contrast adjustments, and contrast-limited adaptive histogram equalization (CLAHE).

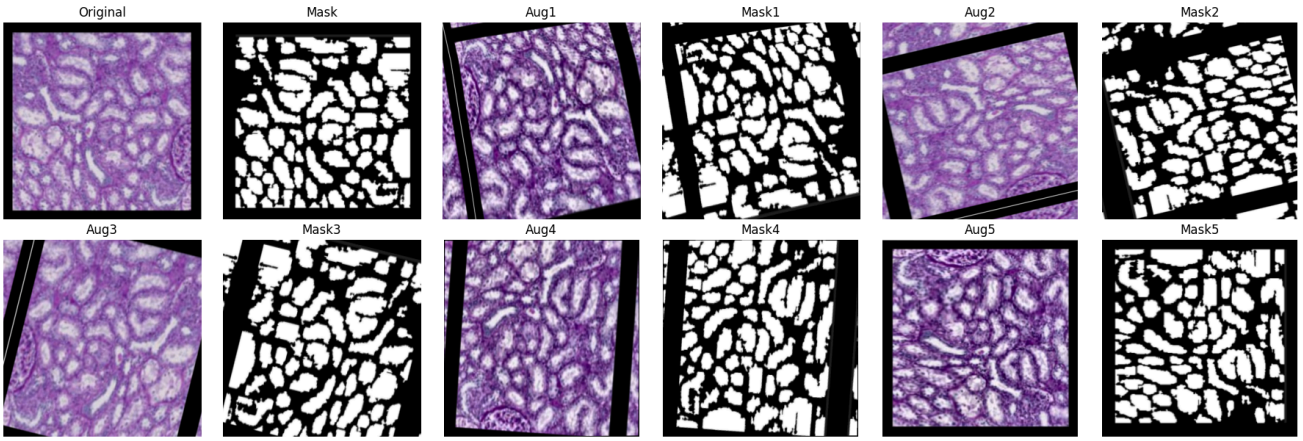

Fig 7: Augmentations to Tiles and Corresponding SAM Masks

3. Architecture

We began by evaluating several encoder–decoder backbones. An early experiment replaced the standard U-Net decoder in ResNeXt-50 with UNETR-Up blocks. Although this hybrid improved validation Dice, it also tripled the parameter count and pushed inference time beyond Kaggle's 9-hour submission cap, causing repeated time-outs.

To stay within the competition's runtime budget while retaining diversity, we converged on three architectures:

Model	Encoder	Decoder / Head	Key traits
ResNeXt-50	50-layer ResNeXt	Lightweight U-Net	Fastest; strong baseline
ResNeXt-101	101-layer ResNeXt	Lightweight U-Net	Deeper context; modest added cost
ResNeXt-Mamba (**Our model**)	ResNeXt-50	BiFPN + *Mamba* decoder	Captures long-range dependencies with minimal params

3.1 ResNeXt-50 & ResNeXt-101

3.1.1 Introduction to ResNet

ResNet (short for Residual Network) is a deep convolutional neural network (CNN) architecture that was introduced by Kaiming He et al [6]. Before ResNet, deeper neural networks often performed worse than shallower ones due to difficulties in optimisation. This is known as the degradation problem.
ResNet introduces a shortcut (skip) connection, where the input of a layer is added directly to its output. As a result, it is easier to learn identity mappings, improves gradient flow during backpropagation and enables training of networks with 100+ or even 1000+ layers.

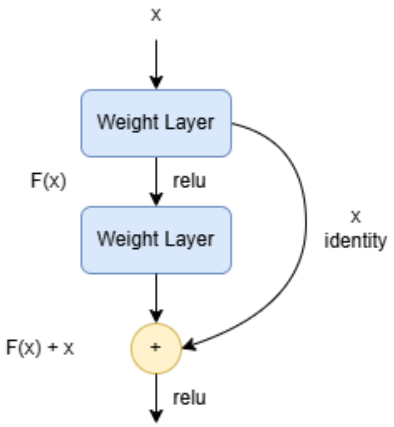

Fig 8: Residual Block of ResNet

The residual block formula is given by:

$$y = F(x) + x$$

- x = input
- F(x) = transformation (e.g., convolution → BN → ReLU)
- x+F(x) = residual connection

We chose to use the ResNeXt model since it builds upon ResNet, and has new concepts introduced in the following parts of the report. Additionally, ResNeXt is trained on the ImageNet 2012 classification dataset that consists of 1000 classes, 1.28 million training images, and evaluated on the 50k validation images [6].
Hence, we believe that it is suitable for transfer learning to our use case to detect diseases in the kidney glomeruli.

3.1.2 Architecture of ResNeXt-50

ResNeXt-50 is a convolutional neural network architecture that builds upon ResNet by incorporating the concept of cardinality, which is the number of independent paths in a block.

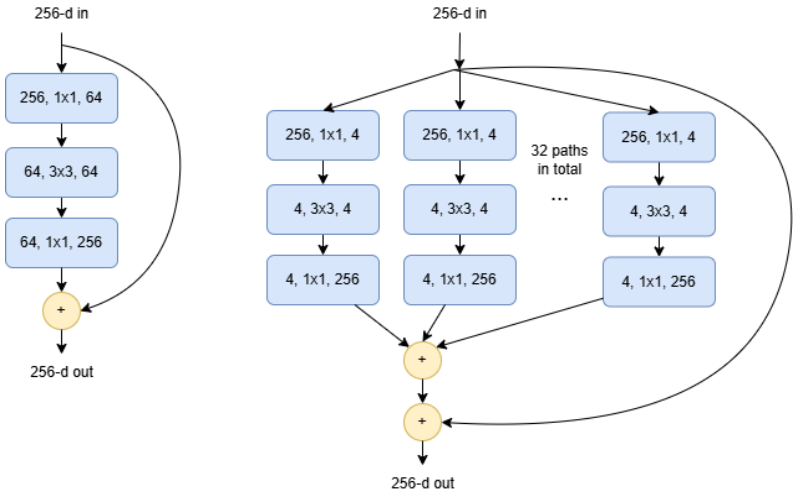

Fig 9: Diagram of ResNet (left) and ResNeXt-50 (right)

Each ResNeXt block is similar to a ResNet bottleneck block, but instead of just widening the layer (i.e., increasing channels), it splits the input into multiple paths (known as "cardinality"), applies transformations, and then aggregates them.
In ResNext-50 32x4d, the 50 stands for 50 layers deep, the cardinality is 32 (ie. 32 parallel paths per block), and the base width is 4 (ie. each path has a 4-channel bottleneck width). In each layer there are different numbers of bottleneck blocks.

Stage	Output Size	Number of Bottleneck Blocks	Conv Layers per Block	Total Conv Layers
Conv1	112×112	-	1 (7×7)	1
Conv2	56×56	3	3	9
Conv3	28×28	4	3	12
Conv4	14×14	6	3	18
Conv5	7×7	3	3	9
FC	1×1	-	1	1
Total				50

Each bottleneck block helps to reduce the computation cost of training the neural network, since the number of channels is reduced before doing any heavy computation, and then expanded back afterwards. This reduction in channels makes the network smaller and easier to train, hence reducing the training cost.
Each ResNeXt bottleneck block consists of:

- 1x1 conv (reduce channels)
- 3x3 group conv (with cardinality = 32)
- 1x1 conv (expand channels)
- skip connection

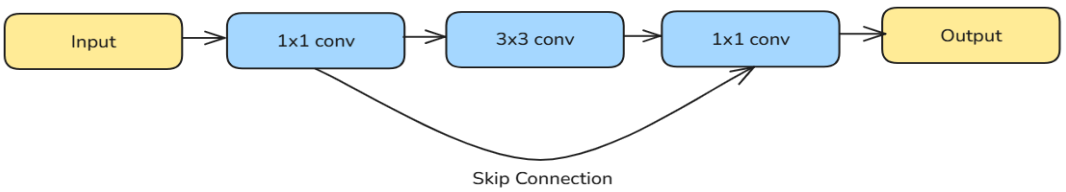

Fig 10: ResNeXt50 block specifications

ResNeXt uses a "split-transform-merge" strategy, which does the following:

1. **Split**: Input is divided into multiple groups.
2. **Transform**: Each group is processed through a small network
3. **Merge**: Outputs are summed (not concatenated)

3.1.2 Implementation of ResNeXt-50

In our implementation, we adjusted the first convolution (conv) layer to support a custom input channel, which enables us to input 4 channels instead of the common 3 channels denoted by the red, green, blue(RGB) colour input. The SAM pseudo mask is concatenated as the fourth channel to the 3 channel RGB input from the training image.

The overall segmentation model follows an encoder-decoder structure containing an encoder, Atrous Spatial Pyramid Pooling (ASPP) module, a decoder, and a final output layer.

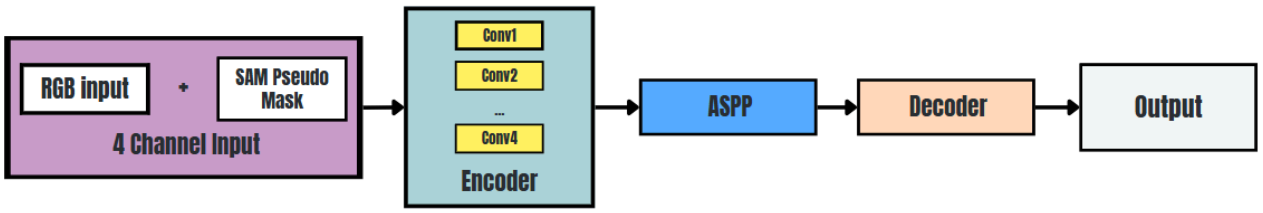

Fig 11: ResNeXt50 architecture

Section	Description
Encoder	• Consists of five stages (conv1, layer1 to layer4) derived from ResNeXt-50, producing progressively lower-resolution feature maps. • Modified initial convolution to accept 4-channel input.
ASPP Module	• Positioned at the deepest encoder layer to capture multi-scale contextual information using multiple dilated convolutions with dilation rates proportional to the stride. • Outputs a feature map with 512 channels.
Decoder	• Built using UnetBlock modules with PixelShuffle-based upsampling. • Each block fuses encoder features via skip connections, batch normalization, and optional self-attention layers. • Feature Pyramid Network (FPN) is employed to aggregate features from multiple decoder stages.

The final output layer is a 1x1 convolution layer that produces the final prediction.

3.1.3 Architecture of ResNext-101

The ResNeXt-101 architecture is an extension of the ResNeXt family that maintains the same modular design principles but with greater depth and capacity. Similar to ResNeXt-50, it utilizes split-transform-merge blocks with grouped convolutions, but expands the number of bottleneck blocks in each stage, enabling the model to learn richer and more hierarchical representations.
Similar to ResNeXt-50, the bottleneck block has the same specifications.
The main architectural changes from ResNeXt-50 are increased number of blocks in each stage (especially conv3_x and conv4_x), greater total depth (101 convolutional layers), and improved performance at a moderate increase in computational cost.

Component	ResNeXt-50	ResNeXt-101
Encoder Backbone	50 layers	101 layers
Depth	Shallow (~4 stages)	Deep (~33 bottleneck blocks)
Feature Extractor	ResNeXt-50 (32×4d)	ResNeXt-101 (32×8d / 32×4d)
Benefit	Fast, lightweight	Better at capturing fine-grained features

3.1.4 Implementation of ResNeXt-101

The implementation of ResNeXt-101 is similar to the implementation of ResNeXt-50 (3.1.2).

3.2 ResNeXt-50-Mamba & ResNeXt-101-Mamba

3.2.1 Architecture & Implementation

ResNeXt-Mamba is our novel hybrid encoder–decoder architecture, built for the HuBMAP glomeruli segmentation task. It integrates the hierarchical feature extraction capability of a ResNeXt backbone with the efficient long-range spatial modeling of Selective State Space Modules (SSMs) in a Mamba-inspired style [7].

Instead of traditional convolutional decoders or resource-heavy transformer blocks, ResNeXt-Mamba leverages lightweight SSM blocks in the decoder. Each decoder block performs gated token-wise updates over flattened spatial features, inspired by the selective update mechanisms of Mamba. While we do not implement structured state recurrence as in full state-space models, this lightweight design captures global context effectively with minimal computational cost.

Formally, each decoder stage processes its feature map $x \in \mathbb{R}^{B \times c \times H \times W}$ as:

$$y = \sigma(W_g x) \cdot (W_h x) + [1 - \sigma(W_g x)] \cdot x$$

W_h learns spatially-aware projections, and σ(.) is a sigmoid gate that balances residual and new information.

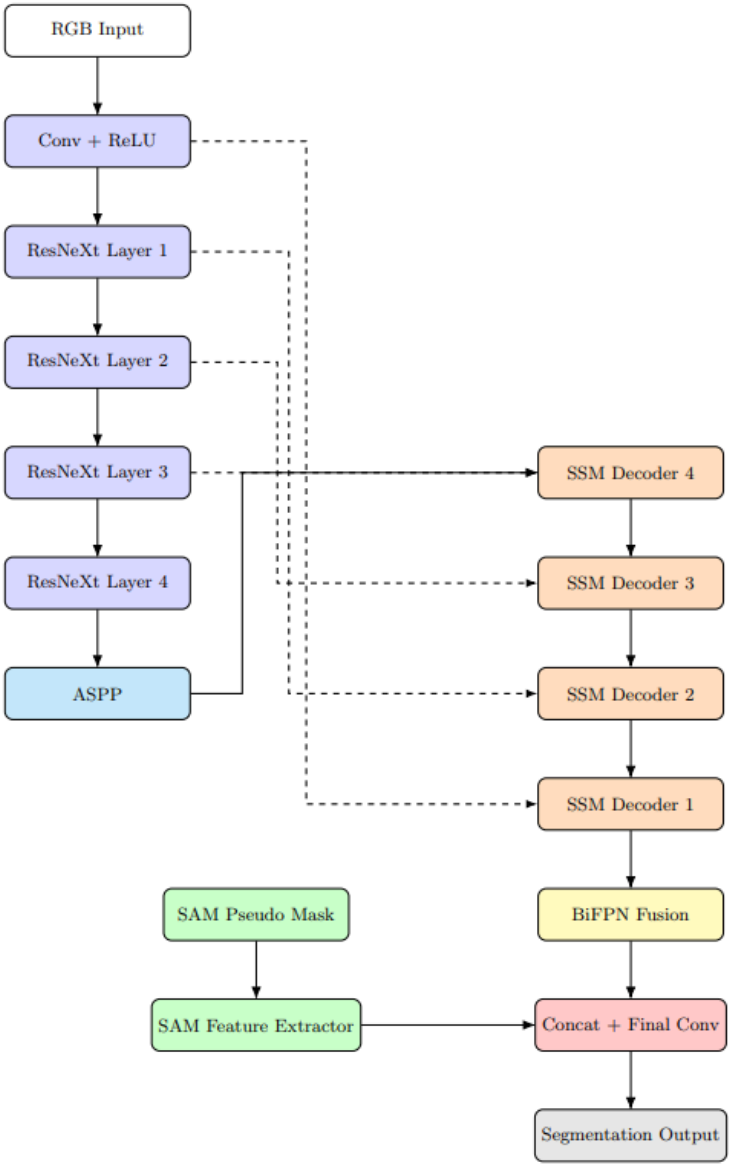

Fig 12: ResNeXt-Mamba architecture

Each decoder block also performs:

- PixelShuffle [8] upsampling with ICNR initialization to preserve edge sharpness and avoid checkerboard artifacts
- Linear projections in token space to inject global context
- Selective State Space modeling for long-range dependency capture
- Skip connection fusion with encoder outputs

To capture multi-scale semantics, we apply Atrous Spatial Pyramid Pooling (ASPP) [9] at the bottleneck stage using multiple dilation rates. The outputs of ASPP and all decoder stages are then aggregated via a lightweight BiFPN [10], ensuring effective multi-resolution feature fusion without heavy compute.

A distinguishing design choice in ResNeXt-Mamba is our late fusion strategy for SAM-generated pseudo masks. Unlike early fusion (e.g., 4-channel input stacking), we process the SAM mask using a lightweight convolutional extractor and fuse the resulting features only at the final decoder stage. This

preserves the pre-trained RGB encoder's generalization while letting the SAM features act as spatial priors.

The complete pipeline consists of:

- A ResNeXt-101 or ResNeXt-50 encoder pre trained on ImageNet [11],
- A Mamba-style decoder built from SSM-enhanced PixelShuffle blocks,
- ASPP for bottleneck context aggregation,
- BiFPN for multi-scale feature fusion,
- A SAM Feature Extractor and late fusion head for enhanced spatial grounding.

The final output is produced via a 1×1 convolution followed by bilinear upsampling, yielding a segmentation mask at the original resolution.

Our architecture delivers a compelling balance of generalization, precision, and efficiency, making it well-suited for complex biomedical segmentation tasks where resolution, speed, and accuracy must co-exist.

4. Training

Model training was performed locally on a single NVIDIA RTX 4070 SUPER (12 GB VRAM). We also employed 5-fold cross-validation to rigorously evaluate model generalization. The training set was divided into five folds, with each model trained on four folds and validated on the remaining one. For each fold $i \in \{0,1,2,3,4\}$, a separate model was trained. Table 1 lists the key hyperparameters and implementation details for each experiment.

Component	**Value / Configuration**	
	ResNeXt	**ResNeXt-Mamba**
Optimiser	Adam	Adam
Learning Rate	1e-4	1e-4
Weight Decay	1e-5	1e-5
Loss Function	DiceBCELoss (0.6 × DiceLoss + 0.4 × BCE)	DiceBCELoss (0.6 × DiceLoss + 0.4 × BCE)
Mixed Precision	Enabled using NVIDIA APEX (opt_level='O1')	Enabled using NVIDIA APEX (opt_level='O1')
Scheduler	ReduceLROnPlateau	ReduceLROnPlateau
Epochs	50	50
Batch Size	8	8

Learning Rate Schedule	ReduceLROnPlateau with factor=0.5, patience=3, min_lr=1e-6	ReduceLROnPlateau with factor=0.5, patience=3, min_lr=1e-6
Metric	Soft Dice Coefficient	Soft Dice Coefficient
Checkpointing	Best-performing model saved based on Dice	Best-performing model saved based on Dice

5. Comparing Local CV Scores of Different Architectures

We first evaluate the impact of SAM integration by comparing models trained with and without SAM-generated pseudo masks. Using ResNeXt-50 as the base encoder, we observe that:

- ResNeXt-50 (RGB only): achieves a highest local CV Dice score of **0.9112**
- ResNeXt-50 + SAM (4th channel Early Fusion): improves to **0.9431**, demonstrating a consistent ~0.01–0.025 uplift in Dice when guided by the SAM pseudo mask

This validates our hypothesis that SAM guidance acts as a valuable spatial prior, even in early fusion form.

Next, we compare the effect of model depth and fusion strategy, keeping SAM guidance consistent across models:

Model	SAM Integration Strategy	Highest Soft Dice (Local CV)	Epochs to Reach Best Dice	Average Training Time per Epoch	Total Time to Converge (Training)
ResNeXt-50	4th Channel (Early Fusion)	0.9431	44	1min 1s	44min 44s
ResNeXt-101	4th Channel (Early Fusion)	0.9422	49	1min 12s	58min 48s
ResNeXt-Mamba (ResNeXt-50 encoder)	Late Fusion (SAM Feature Merging)	0.9429	27	1min 14s	33min 18s
ResNeXt-Mamba (ResNeXt-101 encoder)	Late Fusion (SAM Feature Merging)	0.9432	30	1min 22s	41min

Our proposed ResNeXt-Mamba with a ResNeXt-101 encoder achieved the best local CV Dice score of **0.9432**, converging significantly faster in just **30 epochs**, compared to **44 epochs** for ResNeXt-50 (early fusion), which peaked at **0.9431**. Notably, it also required less total training time—**41 minutes**,

versus **44 minutes** for ResNeXt-50 and **almost 59 minutes** for ResNeXt-101. Furthermore, it demonstrated strong early performance, achieving a Dice score of **0.9032 from the very first epoch**, while the ResNeXt baselines started near **0.10** and only crossed **0.90** after about five epochs.

6. Inference

Our end-to-end inference workflow couples fast, memory-aware prediction with several accuracy boosters. The main steps are detailed below.

Component	What it does	Why it matters
WSI tiling with context	Each whole-slide image is down-sampled ×4 and split into 512 × 512 px tiles with a 256 px "expansion border."	Enables GPU-friendly batches while the overlapping border prevents stitching artefacts along tile edges.
Blank-tile pruning	Tiles whose saturation ≤ 40 or total intensity ≤ 1 000 pixels are skipped.	Avoids wasted compute on empty background (≈ 30 % of tiles).
SAM-guided fourth channel and SAM fusion of features	• Detect LoG blobs in the RGB tile → use their centres as point prompts. • Run Segment Anything (ViT-B) on-device; merge, denoise, and hole-fill the masks. • Stack the 0/1 SAM mask as channel #4 alongside RGB.	Supplies a coarse localisation prior that consistently lifts Dice by ~0.02 without extra training cost.
Mixed-precision batched forward pass	Tiles are streamed through an ensemble of thirteen checkpoints (5 ResNeXt-50, 5 ResNeXt-100 + 3 ResNeXt-101-Mamba) under torch.cuda.amp.autocast() (FP16).	Cuts VRAM ≈ 40 % and keeps total inference < 8 h on Kaggle's P100 while preserving numeric fidelity.
8-way dihedral TTA	For each tile we evaluate {0°, 90°, 180°, 270°} × {mirror, none}. Logits are averaged before sigmoid to maintain calibration.	Reduces prediction variance.
Model-level ensembling	Logits from all checkpoints are averaged (equal weights).	Exploits complementary errors between depths and Mamba blocks.
Probability→mask post-processing	Best global threshold of 0.31 (through trial-and error)	Balances precision/recall and filters noise.

7. Comparing Public LB Scores of Different Architectures

While our Mamba-style architectures outperformed the early fusion models during local cross-validation, this trend did not fully carry over to the public leaderboard, likely due to differences in test distribution or dataset shifts.

We now report the best public leaderboard scores achieved by each architecture at submission time:

Model	SAM Integration Strategy	Public LB Dice
ResNeXt-50	No SAM guidance	0.9178
ResNeXt-101	No SAM guidance	0.9202
ResNeXt-50	4th Channel (Early Fusion)	0.9236
ResNeXt-101	4th Channel (Early Fusion)	0.9247
ResNeXt-Mamba (ResNeXt-50 encoder)	Late Fusion (SAM Feature Merging)	0.9237
ResNeXt-Mamba (ResNeXt-101 encoder)	Late Fusion (SAM Feature Merging)	0.9240
Ensemble (ResNeXt-Mamba (ResNeXt-50 encoder + 101)	Late Fusion (SAM Feature Merging)	0.9244
Ensemble (ResNeXt-50 + 101)	Early Fusion (4th Channel)	0.9255
Ensemble all 4 model configurations (ResNeXt-50 + 101) + (ResNext-Mamba ResNeXt-50 encoder + 101)	Early Fusion (4th Channel) and Late Fusion (SAM Feature Merging)	0.9270

These results suggest that:

- Our Models with SAM guidance outperform the models without SAM guidance in the public leaderboard as well.
- Late fusion Mamba models, although superior in local CV (e.g., up to 0.943 Dice), may be more sensitive to distributional shifts.
- Early fusion ResNeXt models, while slightly behind locally, generalize better to unseen data.
- Ensembling all models, including both early and late fusion strategies, yields the best of both worlds—achieving the highest public leaderboard Dice score of 0.9270.

This validates our hypothesis that SAM guidance improves performance regardless of integration method, and further highlights the value of architectural diversity when ensembling for robust real-world generalization.

8. Challenges

Throughout this project, we encountered several challenges that required careful tuning and experimentation to overcome:

1. **Low score on outlier image (d488c759a)**: One of the biggest issues was the poor performance on a specific outlier image in the public test set. Based on Kaggle forum discussions, this image appears to have been insufficiently labeled, significantly lowering its Intersection-over-Union (IoU) score and skewing our leaderboard performance. To mitigate this, we incorporated manually corrected hand labels (refer to Section 2.1.1), which improved robustness on edge cases.
2. **Discrepancy between local CV and leaderboard scores**: Despite achieving Dice scores as high as 0.943 on local validation folds, our public leaderboard scores were generally in the 0.91–0.92 range. This discrepancy reflects domain shift between validation and test distributions, emphasizing the need for strong generalization in medical segmentation tasks.
3. **Thresholding and expansion tuning during inference**: Post-processing choices—such as the segmentation threshold and expansion size—had a non-trivial impact on final results. We observed fluctuations of up to ±0.02 Dice based on these settings. After extensive experimentation, a threshold of 0.31 and tile expansion of 256 pixels were selected as the optimal combination.
4. **Inability to deploy large SAM models on Kaggle due to resource constraints:** While we initially considered leveraging the ViT-H variant of the Segment Anything Model (SAM) for improved pseudo-mask quality, we encountered out-of-memory (OOM) errors on Kaggle's P100 GPU environment. This limitation constrained us to using ViT-B, which was more memory-efficient but less expressive.

9. Future Improvements

While our current pipeline achieved strong performance on both local validation and the public leaderboard, several directions could be explored to further enhance the system:

1. **Expand SAM fusion experiments across all architectures**: We only implemented late fusion of SAM features in our Mamba-based architecture. In future work, applying this technique to other models like ResNeXt-50 or ResNeXt-101 could help isolate and better quantify the value of late fusion versus early fusion strategies.
2. **Improve SAM pseudo-mask quality:** Although SAM pseudo-masks helped boost performance, their quality remains variable. Enhancing the prompting strategy, integrating better blob detection techniques, or using a larger SAM variant (e.g., ViT-L) could produce more reliable spatial priors.
3. **Explore larger and more diverse ViT-based backbones:** Resource limitations prevented us from using heavier encoders like Swin Transformers, ViT-Large, or SAM-H. These

architectures, though more demanding, may provide richer feature representations and better generalization on challenging tiles.

4. **Improve ensembling strategies:** Our ensemble was based on simple averaging. Using more intelligent methods like weighted ensembling based on validation performance, or model stacking with a meta-learner, could further boost performance.

10. Conclusion

We present a robust and scalable approach to histopathological segmentation by combining models with carefully designed architectures. We show that integrating SAM-generated pseudo masks either via - early channel-wise fusion or late-stage feature injection-consistently improves dice performance, offering an effective form of supervision.

Our novel ResNeXt-Mamba architecture combines the representational depth of a ResNeXt101 encoder with the global context modeling of Mamba-inspired Selective State Space Modules and BiFPN for multiscale fusion. This hybrid design outperformed conventional decoder structures in local cross-validation and generalized well under real-world constraints, achieving a top-tier public leaderboard Dice score of 0.9270 through strategic ensembling.

The components introduced here—SAM-guided fusion strategies, lightweight state-space decoding, and cost-aware inference—are highly modular and transferable. They can be extended to other organs, staining protocols, and segmentation tasks in medical imaging where labeled data is scarce but spatial priors can be leveraged.

11. Results on Kaggle

We attained top (239/1200)×100% = **19.9%** in the HuBMAP - Hacking the Kidney competition.

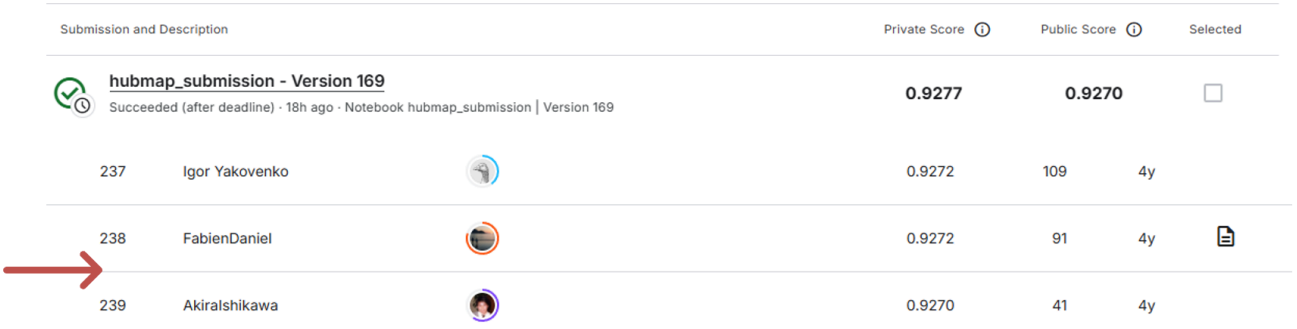

Fig 13: Public leaderboard score achieved using our ResNeXt and ResNeXt-Mamba ensemble models

12. References

[1] mpware. (2021). Handlabels prize. Kaggle. https://www.kaggle.com/code/mpware/handlabels-prize/

[2] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., & Girshick, R. (2023). Segment Anything. arXiv. https://doi.org/10.48550/arXiv.2304.02643

[3] Xie, Y., Zhou, T., Zhou, Y., & Chen, G. (2024). SimTxtSeg: Weakly-Supervised Medical Image Segmentation with Simple Text Cues. arXiv. https://doi.org/10.48550/arXiv.2406.19364

[4] Koleilat, T., Asgariandehkordi, H., Rivaz, H., & Xiao, Y. (2024). MedCLIP-SAMv2: Towards Universal Text-Driven Medical Image Segmentation. arXiv. https://doi.org/10.48550/arXiv.2409.19483

[5] CamelEdge. (2024). Understanding blob detection: Multiscale space filtering and its importance. CamelEdge. https://cameledge.com/post/blob-detection

[6] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In CVPR. arXiv:1512.03385

[7] Gu, A., & Dao, T. (2024). Mamba: Linear-Time Sequence Modeling with Selective State Spaces. arXiv. https://doi.org/10.48550/arXiv.2312.00752

[8] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. arXiv. https://doi.org/10.48550/arXiv.1609.05158

[9] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. arXiv. https://doi.org/10.48550/arXiv.1606.00915

[10] Tan, M., Pang, R., & Le, Q. V. (2020). EfficientDet: Scalable and efficient object detection. arXiv. https://doi.org/10.48550/arXiv.1911.09070

[11] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. arXiv. https://doi.org/10.48550/arXiv.1611.05431