Flow Stochastic Segmentation Networks

Fabio De Sousa Ribeiro[†] Omar Todd Charles Jones Avinash Kori Raghav Mehta Ben Glocker[†]
Imperial College London

Abstract

We introduce the Flow Stochastic Segmentation Network (Flow-SSN), a generative segmentation model family featuring discrete-time autoregressive and modern continuous-time flow variants. We prove fundamental limitations of the low-rank parameterisation of previous methods and show that Flow-SSNs can estimate arbitrarily high-rank pixel-wise covariances without assuming the rank or storing the distributional parameters. Flow-SSNs are also more efficient to sample from than standard diffusion-based segmentation models, thanks to most of the model capacity being allocated to learning the base distribution of the flow, constituting an expressive prior. We apply Flow-SSNs to challenging medical imaging benchmarks and achieve state-of-the-art results. Code available: https://github.com/biomedia-mira/flow-ssn.

1. Introduction

Semantic image segmentation consists of producing pixel-wise predictions that reflect objects and their boundaries. Traditional methods typically approach this as a deterministic pixel-wise classification task, while overlooking the inherent *uncertainty* of the spatial structures involved [11, 36, 98]. Uncertainty generally relates to unknown or imperfect information, often arising from a lack of knowledge, partial observations, and/or inherently stochastic events. Uncertainty is commonly decomposed into two distinct parts [18, 36]: (i) *Epistemic* uncertainty, which relates to a lack of knowledge, and it can be reduced in principle by observing more data; (ii) *Aleatoric* uncertainty, which relates to inherent unknowns that differ each time we run the same experiment (e.g. flipping a coin), and it cannot be reduced by observing more data. Although conceptually attractive, the utility and validity of this decomposition is actively contested [41, 79].

Inherent ambiguities are particularly prevalent in medical imaging, where medical opinions can vary significantly across different experts [5, 9, 35, 43]. In this context, uncertainty can arise from several factors, including indistinct boundaries, occlusion, poor image acquisition quality, and the intrinsic variability of the underlying pathology. Therefore, to effectively model these ambiguities and reflect the real-world variability of expert opinions, segmentation models ought to capture a rich *distribution* of plausible segmentation outcomes [14, 43]. Furthermore, providing uncertainty estimates is useful for revealing whether a model has sufficient knowledge to provide a reliable assessment, which is important in safety-critical real-world settings [7, 47, 63].

There is a growing interest in using probabilistic methods for estimating uncertainty in image segmentation. Most existing methods handle uncertainty by factorising the output posterior into per-pixel marginal distributions, thereby ignoring any correlations between pixels. Therefore, pixel-wise independent uncertainty estimates are incapable of fully capturing spatially structured uncertainty [46, 58]. To address this, Monteiro et al. [55] proposed Stochastic Segmentation Networks (SSNs), which can explicitly model spatially correlated aleatoric uncertainty without requiring variational approximations or latent variable assumptions. Their method involves placing a low-rank multivariate Gaussian distribution over the logit space (i.e. before the softmax), then using Monte Carlo integration to marginalise out the logits and compute pixel-wise joint likelihoods [36, 37]. Although promising, the trouble with SSNs is three-fold: (i) The assumed rank of the low-rank approximation is typically kept small due to computational constraints (e.g., $\simeq$10), and it is almost surely underspecified relative to the true rank of high-dimensional, pixel-wise covariances; (ii) They often require an expensive mean pre-training stage to ensure proper convergence, as jointly optimising poor initial estimates of the mean and covariance can lead to getting trapped in suboptimal minima; (iii) They suffer from training instabilities, partly due to a lack of guarantee that the low-rank covariance matrix remains positive definite throughout training.

Contributions. We propose Flow Stochastic Segmentation Networks (Flow-SSNs), a generative segmentation model class featuring discrete-time autoregressive and modern continuous-time flow parameterisations. Flow-SSNs can estimate arbitrarily high-rank pixel-wise covariances without assuming the rank a priori, storing distributional parameters, or assuming a lower-dimensional latent space as in VAEs.

[†]Email: {f.de-sousa-ribeiro, bglocker}@imperial.ac.uk

Flow-SSNs are more efficient to sample from than typical diffusion-based segmentation models, as most of the model capacity is dedicated to learning a flow's *prior*, while the flow itself is lightweight. In summary, our contributions are:

§4 We prove fundamental limitations of the low-rank parameterisation of SSNs by showing that the effective rank grows sublinearly with the assumed rank;

§5 We introduce Flow-SSNs, a generative segmentation model capable of modelling complex covariance structures efficiently by learning a flow's *prior* and using a lightweight flow to model pixel-wise dependencies;

§6 Applying Flow-SSN to a toy problem and two real-world medical image segmentation benchmarks, we show state-of-the-art results with fewer parameters.

2. Related Work

Existing work on stochastic segmentation can be broadly categorised as: (i) Bayesian methods which approximate a posterior over neural network parameters [36]; (ii) latent variable generative models [6, 43, 89]; and (iii) distributional/evidential methods, which estimate a complex joint distribution directly in pixel space [53, 55, 77, 84]. Mackay [52], Neal [57] and Hinton and Van Camp [30] laid the foundations for modern-day Bayesian Neural Networks (BNNs), which approximate neural network parameter posteriors and enable uncertainty estimation. More recently, Kendall and Gal [36], Kwon et al. [45], and others [15, 16, 37, 90], used these techniques for classification/segmentation, but handled uncertainty by factorising the output posterior into per-pixel marginal distributions, thereby ignoring pixel correlations.

Stochastic segmentation methods based on the Variational Autoencoder (VAE) [39, 70] framework implicitly assume that the data resides in a lower-dimensional latent manifold and hope that the pixel-wise independent decoder will learn to translate uncorrelated latent variables into meaningful spatial variation in pixel space [6, 43, 93]. Selvan et al. [76], Valiuddin et al. [86] also use a VAE for segmentation but apply a normalising flow [81] to the latent variables to make them more expressive. Since providing latent variable identifiability guarantees is challenging for most problems [34], one often resorts to unfalsifiable assumptions about both the functional form and dimensionality of the latent space. Although VAEs can work well for certain segmentation tasks [43, 54], they sometimes underperform in high-dimensional settings, and are subject to the *prior hole* problem [27, 50, 69, 71].

Diffusion models [31, 80] are a promising viable alternative which has been recently explored for producing stochastic segmentations [3, 66, 89, 92]. However, their high inference costs can restrict their usability for medical experts to edit segmentation annotations in real-time. Recent work on Continuous Normalising Flows (CNFs) [12] provides efficient (simulation-free) ways to learn straighter paths between distributions compared to diffusion paths [2, 48, 49, 82]. CNFs that induce straighter paths are computationally more efficient to solve, making them an attractive option for real-time editing of medical imaging annotations. There is limited prior work on exploring CNFs for segmentation; [8] combine Flow Matching (FM) [48] with the signed distance function (SDF) for image segmentation, but their investigation is restricted to binary data. Our approach differs substantially in its formulation as it is defined within an SSN-like [55] paradigm, and naturally extends to categorical data. Specifically, our model generates multiple segmentations by sampling from an expressive, learned base distribution (i.e. prior) conditional on the image, rather than random noise.

Limited prior work exists on using autoregressive models for stochastic segmentation tasks. Zhang et al. [95] proposed an autoregressive approach using a PixelCNN [74, 87], which can learn full rank pixel-wise covariances. However, it requires each pixel to be generated sequentially, which is slow. To mitigate this, they use a downsampled resolution, discarding input information. SSNs [55] provide a simpler alternative for learning the joint distribution over pixel-wise label maps that does not require making latent variable assumptions or variational approximations. Multiple works build on SSNs to enable fine-grained sample control [59], learning mixtures of stochastic experts [24] and conditioning on label style [93]. However, SSNs are subject to training instabilities and make strong assumptions about the rank of the true pixel-wise covariance being quite small, which is under-determined for most problems. Flow-SSNs make no such assumptions and can estimate arbitrarily high-rank covariances. Concurrently, [28, 83, 94] revisit classical flow models [21, 38, 61] for generation, but not for segmentation.

3. Preliminaries

Let $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ be a dataset of n image $\mathbf{x}_i \in \mathbb{R}^{c\times d}$ and one-hot label map $\mathbf{y}_i \in \{0,1\}^{k\times d}$ pairs, with number of channels c, height times width $hw = d$, and categories k. We denote $\mathrm{softmax}_k(\cdot)$ as the row-wise softmax applied after reshaping the input from $\mathbb{R}^{kd}$ to $\mathbb{R}^{k\times d}$. To avoid cluttered notation, we may use lowercase symbols to denote both random variables and their realisations when context permits.

Stochastic Segmentation Networks. Stochastic Segmentation Networks (SSNs) [55] model joint distributions over label maps to generate spatially coherent segmentations. Pixel-wise dependencies are modelled by placing a low-rank multivariate Gaussian distribution over the logit space, and marginalising[1] the logits $\boldsymbol{\eta} \in \mathbb{R}^{kd}$ to compute likelihoods:

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{x}) &= \int p(\mathbf{y} \mid \boldsymbol{\eta}) p(\boldsymbol{\eta} \mid \mathbf{x}) \, \mathrm{d}\boldsymbol{\eta}, \\ p(\boldsymbol{\eta} \mid \mathbf{x}) &= \mathcal{N}(\boldsymbol{\eta}; \boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\Sigma}(\mathbf{x})), \\ p(\mathbf{y} \mid \boldsymbol{\eta}) &= \mathrm{Categorical}(\mathbf{y}; \mathrm{softmax}_k(\boldsymbol{\eta})), \end{aligned} \tag{1}$$

[1] A Monte Carlo estimator of this (intractable) integral is typically used.

where $\boldsymbol{\mu}(\mathbf{x}) \in \mathbb{R}^{kd}$ and $\boldsymbol{\Sigma}(\mathbf{x}) \in \mathbb{R}^{kd \times kd}$ are predicted by a neural network given an input $\mathbf{x}$. Dependencies in logit space manifest in pixel space through the conditional dependence of $\mathbf{y}$ on $\boldsymbol{\eta}$. Note that $\boldsymbol{\Sigma}(\mathbf{x})$ is not only spatial but also class-wise. Due to the large number of pixels involved, estimating $\boldsymbol{\Sigma}(\mathbf{x})$ is typically computationally infeasible, thus, Monteiro et al. [55] use a low-rank approximation of the form:

$$\boldsymbol{\Sigma}(\mathbf{x}) = \mathbf{D}(\mathbf{x}) + \mathbf{P}(\mathbf{x})\mathbf{P}(\mathbf{x})^\top, \tag{2}$$

where $\mathbf{D}(\mathbf{x}) \in \mathbb{R}_+^{kd \times kd}$ denotes the diagonal matrix containing pixel-wise (including class-wise) independent variances, whereas the covariance factor matrix specifying r as the rank of the approximation is denoted by $\mathbf{P}(\mathbf{x}) \in \mathbb{R}^{kd \times r}$.

Normalising Flows. Normalising Flow (NF) [20, 62, 68, 81] models construct a complex probability distribution p_X of a target variable X by applying a parameterised transformation $\phi : \mathcal{U} \to \mathcal{X}$ to a simple base distribution p_U. If the transformation $\mathbf{x} = \phi(\mathbf{u})$ is both invertible and differentiable (i.e. *diffeomorphic*), the density of the target variable is readily given by the change-of-variables formula:

$$p_X(\mathbf{x}) = p_U(\mathbf{u}) \left|\det \mathbf{J}_\phi(\mathbf{u})\right|^{-1}, \quad \mathbf{u} = \phi^{-1}(\mathbf{x}), \tag{3}$$

where $(\mathbf{J}_\phi(\mathbf{u}))_{ij} = \partial\phi_i/\partial u_j$ is the Jacobian matrix. The design space of transformations ϕ is often restricted to cases where the Jacobian determinant is efficient to compute. Autoregressive flows [40, 61] use a chain of invertible transformations $\phi_1 \circ \phi_2 \circ \cdots \circ \phi_T$, each given by an autoregressive model. Autoregressive flows have been used to increase the flexibility of the approximate posterior in VAEs [40, 85].

Continuous Normalising Flows. Continuous Normalising Flows (CNFs) [12] define a time-dependent (for time $t \in [0, 1]$) continuous flow mapping $\phi_t(\mathbf{x})$ from a simple base density $\mathbf{x}_0 \sim p_0$ to a desired data distribution $\mathbf{x}_1 \sim p_{\text{data}}$ governed by an ordinary differential equation (ODE):

$$\frac{\mathrm{d}\phi_t(\mathbf{x})}{\mathrm{d}t} = v_t(\phi_t(\mathbf{x}); \theta), \qquad \phi_0(\mathbf{x}) = \mathbf{x}_0, \tag{4}$$

where $v_t(\phi_t(\mathbf{x}); \theta)$ is a vector field parameterised by a deep neural network with parameters θ. A vector field v_t is said to generate a *probability density path* p_t that transports p_0 to $p_1 \approx p_{\text{data}}$ if its flow ϕ_t satisfies the continuous-time analogue of the change-of-variables formula in Equation 3. To sample from the model, noise is mapped to data by solving the following differential equation using an ODE solver:

$$\phi_1(\mathbf{x}) = \mathbf{x}_1 = \mathbf{x}_0 + \int_0^1 v_t(\phi_t(\mathbf{x}); \theta)\, \mathrm{d}t. \tag{5}$$

Flow Matching (FM) [2, 48, 49] provides a simulation-free way of training CNFs by regressing a velocity field u_t, inducing a desired probability path p_t. However, both p_t and u_t are generally unknown; there exist many p_t's which generate p_{data}, and we do not know the u_t that generates p_t. Lipman et al. [48] showed that a chosen p_t and corresponding u_t can be constructed by marginalising *conditional* probability paths and vector fields over p_{data}. A conditional probability path $p_t(\mathbf{x} \mid \mathbf{x}_1)$ is a time-dependent distribution that satisfies the following marginal constraints at the endpoints:

$$p_0(\mathbf{x} \mid \mathbf{x}_1) = p_0(\mathbf{x}), \quad p_1(\mathbf{x} \mid \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_1). \tag{6}$$

The marginal probability path and vector field are given by:

$$\begin{aligned} p_t(\mathbf{x}) &= \mathbb{E}_{\mathbf{x}_1 \sim p_{\text{data}}}\left[p_t(\mathbf{x} \mid \mathbf{x}_1)\right] \\ u_t(\mathbf{x}) &= \mathbb{E}_{\mathbf{x}_1 \sim p_{\text{data}}}\left[u_t(\mathbf{x} \mid \mathbf{x}_1)\frac{p_t(\mathbf{x} \mid \mathbf{x}_1)}{p_t(\mathbf{x})}\right], \end{aligned} \tag{7}$$

where the *conditional* vector field $u_t(\mathbf{x} \mid \mathbf{x}_1)$ is defined by the time derivative $\mathrm{d}\phi_t/\mathrm{d}t$ of the chosen flow map ϕ_t, which transports samples from p_0 to $p_t(\mathbf{x} \mid \mathbf{x}_1)$. A common choice is to set the flow to $\phi_t(\mathbf{x}_0 \mid \mathbf{x}_1) = \sigma_t(\mathbf{x}_1)\mathbf{x}_0 + \mu_t(\mathbf{x}_1)$, where $p_t(\mathbf{x} \mid \mathbf{x}_1) = \mathcal{N}(\mathbf{x}; \mu_t(\mathbf{x}_1), \sigma_t^2(\mathbf{x}_1)I)$ is Gaussian.

Crucially, Lipman et al. [48] showed that if $u_t(\mathbf{x} \mid \mathbf{x}_1)$ generates $p_t(\mathbf{x} \mid \mathbf{x}_1)$ then $u_t(\mathbf{x})$ generates $p_t(\mathbf{x})$ and the following simple regression objective can be used to train a CNF that generates the marginal probability path $p_t(\mathbf{x})$:

$$\mathbb{E}_{t, p_{\text{data}}(\mathbf{x}_1), p_t(\mathbf{x}|\mathbf{x}_1)}\left[\left\|u_t(\mathbf{x} \mid \mathbf{x}_1) - v_t(\mathbf{x}; \theta)\right\|^2\right], \tag{8}$$

with $t \sim \mathcal{U}(0, 1)$, thus approximating the unknown data distribution $p_1 \approx p_{\text{data}}$ at time $t = 1$, as intended.

4. Theoretical Analysis: Effective Rank

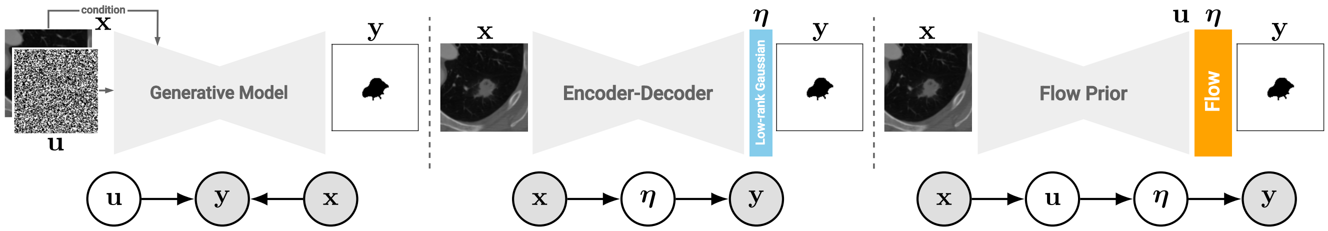
As outlined in Section 1, the low-rank assumption in SSNs is restrictive. Even if scaling the number of covariance factors (e.g. $r \gg 10$) were practical, it would still impose a Gaussian assumption on the pixel-space distribution. We now proceed with a theoretical analysis of the rank assumption in SSNs, to provide a more nuanced argument in favour of our flow-based approach. In short, we prove that the expected likelihood SSNs use results in a rank *increase* relative to the initially assumed rank, but the final effective rank only grows sublinearly w.r.t. the initial rank, limiting expressivity[2].

Lemma 4.1 (Rank Increase)**.** *Let the logits $\boldsymbol{\eta}$ be low-rank Gaussian distributed $\boldsymbol{\eta} \mid \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\Sigma}(\mathbf{x}))$, where the covariance $\boldsymbol{\Sigma}(\mathbf{x}) \in \mathbb{R}^{kd \times kd}$ has rank $\operatorname{rank}(\boldsymbol{\Sigma}(\mathbf{x})) = r$. Given that $\mathbf{y} = \operatorname{softmax}_k(\boldsymbol{\eta})$, the following holds:*

$$\operatorname{rank}(\operatorname{Cov}(\mathbf{y})) > r \iff r < d(k-1). \tag{9}$$

This result reveals that the low-rank approximation is not as restrictive as anticipated, as the non-linear pushforward by the softmax induces a rank *increase*. However, we now prove that the *effective* rank remains low, limiting expressivity.

[2]Proofs for all formal results are provided in Appendices A and B.



(a) Standard Generative Segmentation (b) Stochastic Segmentation Net (SSN) (c) **Proposed:** Flow-SSN

Figure 1. **Graphical models for stochastic segmentation**. **(a)** The typical setup, a generative model of label maps $\mathbf{y}$ conditioned on the image $\mathbf{x}$ (e.g. diffusion-based segmentation). **(b)** Stochastic segmentation network [55], a Markov chain with a low-rank Gaussian parameterisation of the logits $\boldsymbol{\eta}$, where $\mathbf{y} = \mathrm{softmax}(\boldsymbol{\eta})$. **(c)** A Flow-SSN (discrete or continuous-time), which consists of: (i) a parameterised conditional base distribution $p_{U|X}(\mathbf{u} \mid \mathbf{x}; \lambda)$ serving as an expressive flow *prior*; (ii) a lightweight flow $\phi : \mathcal{U} \to \mathcal{Y}$ to model pixel-wise dependencies.

Definition 4.2 (Effective Rank [73]). The effective rank $\mathrm{erank}(A) \in \mathbb{R}$ of a matrix $A \in \mathbb{R}^{d\times d}$ is given by:

$$\mathrm{erank}(A) = e^{H(p)}, \quad H(p) = -\sum_{i=1}^{d} p_i \log p_i, \tag{10}$$

where $H(p)$ is the Shannon entropy of the singular value distribution: $p_i = \sigma_i / \sum_{j=1}^{d} \sigma_j$, for $i \in \{1, \dots, d\}$, with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d \geq 0$ denoting the singular values of A.

Remark 4.3. Intuitively, the effective rank represents the average number of significant dimensions in a matrix's range.

Theorem 4.4 (Sublinear Growth of the Effective Rank). *Given a low-rank Gaussian covariance matrix* $\boldsymbol{\Sigma}(\mathbf{x}) \in \mathbb{R}^{kd\times kd}$ *with initial rank* $\mathrm{rank}(\boldsymbol{\Sigma}(\mathbf{x})) = r < d(k-1)$. *The increase in the effective rank* $\mathrm{erank}(\mathrm{Cov}(\mathbf{y}))$, *in the sense of Lemma 4.1, grows sublinearly w.r.t.* r.

Theorem 4.4 shows that despite the induced rank increase as per Lemma 4.1, the final effective rank $\mathrm{erank}(\mathrm{Cov}(\mathbf{y}))$ only grows sublinearly w.r.t the initial rank r. Thus, the low-rank assumption is still highly restrictive for high-dimensional, pixel-wise covariances. This result may be of independent interest to other low-rank approximation techniques [32, 91].

5. Flow Stochastic Segmentation Networks

Motivated by the theoretical analysis in Section 4, we present Flow Stochastic Segmentation Networks (Flow-SSNs). Unlike standard SSNs, Flow-SSNs can estimate arbitrarily high-rank pixel-wise covariances without needing to assume the rank apriori or store distributional parameters. We begin our exposition by: (§5.1) designing *discrete-time* flows for learning pixel-wise covariances from an autoregressive perspective, then (§5.2) develop a generalisation of the approach to modern *continuous-time* flows, which admit free-form Jacobians. Flow-SSNs are also significantly more efficient to sample from than other diffusion-based segmentation models, as the majority of the model capacity is allocated to learning a flow's prior (normally fixed), while the flow network is lightweight, thereby reducing sampling cost substantially.

5.1. Discrete-Time Autoregressive Flow-SSNs

The motivation for using flows to model pixel-wise covariances in stochastic segmentation is simple. Since pixel-wise Gaussian covariances only represent linear dependencies between components, a *linear* autoregressive flow model is sufficient to transform a Gaussian distributed random variable with diagonal covariance into one with *full* covariance.

Proposition 5.1 (Full Covariance Flow Transformation). *Let* $\mathbf{u} = (u_1, u_2, \dots, u_d)^\top$ *be a Gaussian distributed random vector with diagonal covariance* $\mathbf{u} \sim \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$. *A linear autoregressive model is sufficient to transform* $\mathbf{u}$ *into a new variable* $\boldsymbol{\eta} \in \mathbb{R}^d$ *with full covariance* $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$.

This result underpins our key model proposal, which consists of the following two components: (i) learn an expressive, but pixel-wise independent, flow base distribution conditional on $\mathbf{x}$ to act as an 'initial guess' logit distribution; (ii) use a lightweight autoregressive flow to model logit-space dependencies, thereby inducing *full* covariance structure as per Proposition 5.1, and refining the initial guess. Concretely, a Flow-SSN consists of the following two components:

(i) A conditional base distribution $p_{U|X}$, parameterised by a neural encoder-decoder $f_\lambda : \mathbb{R}^{c\times d} \to \mathbb{R}^{kd} \times \mathbb{R}^{kd}$:

$$p_{U|X}(\mathbf{u} \mid \mathbf{x}; \lambda) = \mathcal{N}\left(\mathbf{u}; \boldsymbol{\mu}(\mathbf{x}), \mathrm{diag}(\boldsymbol{\sigma}^2(\mathbf{x}))\right); \tag{11}$$

(ii) A lightweight autoregressive flow $\phi : \mathbb{R}^{kd} \to \mathbb{R}^{kd}$:

$$\forall i \;:\; \eta_i = \phi_i(\mathbf{u}_{\leq i}; \theta), \quad \mathbf{u}|\mathbf{x} \sim p_{U|X}. \tag{12}$$

The likelihood $p(\mathbf{y} \mid \mathbf{x})$ of a Flow-SSN can be obtained by marginalising the logit variables $\boldsymbol{\eta}$ similar to an SSN:

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{x}) &= \int p(\mathbf{y} \mid \boldsymbol{\eta}) p(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)\, \mathrm{d}\boldsymbol{\eta}, \\ p(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta) &= p_{U|X}(\mathbf{u} \mid \mathbf{x}; \lambda) \left|\det \mathbf{J}_\phi(\mathbf{u})\right|^{-1}, \\ p(\mathbf{y} \mid \boldsymbol{\eta}) &= \mathrm{Categorical}(\mathbf{y}; \mathrm{softmax}_k(\boldsymbol{\eta})), \end{aligned} \tag{13}$$

but $p(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)$ is now more expressive than the low-rank Gaussian in SSNs, as it can model *full* covariance structure.

5.1.1. Designing a Flow & Objective

In this section, we explore the design space of autoregressive Flow-SSNs. Affine autoregressive flows are of the form $\eta_i = \phi_{\mu_i}(\boldsymbol{\eta}_{<i}) + \phi_{\sigma_i}(\boldsymbol{\eta}_{<i})u_i$, and have a tractable, lower triangular Jacobian: $\partial u_i/\partial \eta_j = 0, \forall j > i$. Using a Masked Autoregressive Flow (MAF) [61] for modelling $p(\boldsymbol{\eta} \mid \mathbf{x})$ is not ideal, as it requires slow, pixel-wise sequential sampling. Conversely, Inverse Autoregressive Flows (IAFs) [40] are fast to sample from, but require pixel-wise sequential scoring[3]. Crucially, IAFs can still score their *own* samples efficiently, as intermediate outputs can be cached.

This important fact opens up multiple design options for building discrete-time autoregressive Flow-SSNs, some of which we outline next and detail further in Appendix B.

Inverse Autoregressive Flow-SSN. A simple approach is to choose an IAF, and use a Monte Carlo estimator of the categorical likelihood in Eq. (13) analogous to standard SSNs:

$$\max_{\lambda,\theta} \mathbb{E}_{\mathbf{u}\sim p_{U|X}(\mathbf{u}|\mathbf{x};\lambda)} \left[\log p(\mathbf{y} \mid \boldsymbol{\eta} = \phi(\mathbf{u};\theta))\right]. \tag{14}$$

Dual Flow-SSN. A *dual* Flow-SSN comprised of an IAF $p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)$ and an MAF $p^{\text{MAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \hat{\lambda}, \hat{\theta})$ can be trained concurrently by maximising: $\log p(\mathbf{y} \mid \mathbf{x}) \geq$

$$\mathbb{E}_{\boldsymbol{\eta}\sim p^{\text{IAF}}(\boldsymbol{\eta}|\mathbf{x};\lambda,\theta)} \left[\log p(\mathbf{y} \mid \boldsymbol{\eta})\right] - D_{\text{KL}}\left(p^{\text{IAF}} \parallel p^{\text{MAF}}\right).$$

If we choose p^{MAF} as improper uniform: $\forall \boldsymbol{\eta}, p(\boldsymbol{\eta}) = \text{const}$, then we can avoid training the MAF by optimising:

$$\mathbb{E}_{\boldsymbol{\eta}\sim p^{\text{IAF}}(\boldsymbol{\eta}|\mathbf{x};\lambda,\theta)} \left[\log p(\mathbf{y} \mid \boldsymbol{\eta})\right] + \beta H(p^{\text{IAF}}), \tag{15}$$

where setting the hyperparameter $\beta > 0$ helps prevent p^{IAF} from collapsing to a deterministic model.

5.2. Continuous-Time Flow-SSNs

Relaxing the autoregressive structure of discrete-time Flow-SSNs induces a more expressive *free-form* Jacobian, but it complicates the computation of its determinant, as it is no longer simply the product of its diagonal elements. This calls for adapting continuous-time flows [2, 12, 48, 49] to build Flow-SSNs, as they can be trained efficiently via FM, admit free-form Jacobians, and can therefore model arbitrary pixel-wise covariances in stochastic segmentation tasks.

Interpolation Path. Continuous-time Flow-SSNs share the same design principle as their discrete-time counterparts, comprising: (i) an expressive conditional base distribution; (ii) a lightweight flow to model pixel-wise dependencies. However, the flow ϕ_t is now a continuous-time mapping from the conditional base distribution $\mathbf{u}|\mathbf{x} \sim p_{t=0} = p_{U|X}$ to the label data distribution $\mathbf{y} \sim p_{t=1} = p_{\text{data}}$, and it can be defined as a simple deterministic interpolation path:

$$\mathbf{y}_t = (1-t)\mathbf{u} + t\mathbf{y} \implies \mathrm{d}\mathbf{y}_t = (\mathbf{y} - \mathbf{u})\,\mathrm{d}t. \tag{16}$$

[3] As per Figure 2, the inverse transform of an affine autoregressive flow $\mathbf{u} = (\boldsymbol{\eta} - \boldsymbol{\mu}(\boldsymbol{\eta}))/\boldsymbol{\sigma}(\boldsymbol{\eta})$ is parallelisable since $u_i \perp\!\!\!\perp u_{j\neq i} \mid \{\eta_i, \mu_i, \sigma_i\}$.

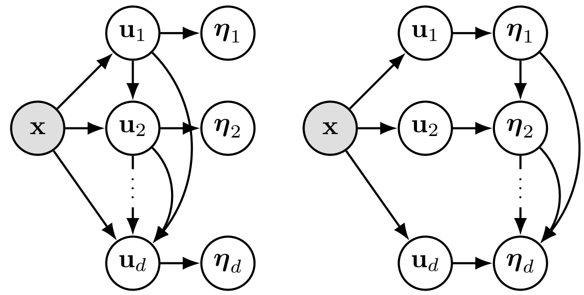


Figure 2. **Graphical models of autoregressive Flow-SSNs**. *Left*: inverse autoregressive; *Right*: masked autoregressive. The target variable $\mathbf{y}$ is omitted here for simplicity, noting that the logits $\boldsymbol{\eta} \to \mathbf{y}$ in all cases, e.g. via a deterministic transform $\mathbf{y} = \text{softmax}_k(\boldsymbol{\eta})$.

Importantly, the so-called *Markovian projection* $\mathcal{M}(\cdot)$ [29] of this type of ODE has been previously shown [65, 78] to preserve the marginals p_t for all time points $t \in [0, 1]$:

$$\mathrm{d}\mathbf{y}_t^\star = \mathcal{M}\left(\mathbf{y} - \mathbf{u}\right)\mathrm{d}t = \left(\mathbb{E}[\mathbf{y} \mid \mathbf{y}_t] - \mathbf{u}\right)\mathrm{d}t. \tag{17}$$

Therefore, in practice, we require a model $p(\cdot;\theta)$ to approximate the conditional expectation $\mathbb{E}[\mathbf{y} \mid \mathbf{y}_t]$, which we propose to parameterise using a *much smaller* neural network compared to the one used for learning the base distribution $p_{U|X}$ (i.e. prior). This makes sampling (ODE solving) *significantly* cheaper compared to typical generative segmentation models, where *all* the model parameters are dedicated to learning the score/velocity field [3, 66, 89, 92]. Further, one could then leverage a large foundation model as the prior.

Categorical Likelihood. Since $\mathbf{y}$ is a one-hot map, we have:

$$\begin{aligned}\mathbb{E}[\mathbf{y} \mid \mathbf{y}_t] &= \sum_{\mathbf{y}\in\{0,1\}^{k\times d}} \mathbf{y} \cdot \text{Categorical}(\mathbf{y}; \text{softmax}(\boldsymbol{\eta})) \\ &\approx \text{softmax}(\boldsymbol{\eta}(\phi_t(\mathbf{u} \mid \mathbf{y}); \theta)),\end{aligned} \tag{18}$$

where $\phi_t(\mathbf{u} \mid \mathbf{y}) = \mathbf{y}_t = (1-t)\mathbf{u} + t\mathbf{y}$, and $\boldsymbol{\eta}(\cdot;\theta)$ is a neural network. Thus, like discrete-time Flow-SSNs (cf. Eq. (14)), we train using an expected categorical likelihood:

$$\max_{\lambda,\theta} \mathbb{E}_{\mathbf{u}\sim p_{U|X}(\mathbf{u}|\mathbf{x};\lambda)} \left[\int_0^1 \log p(\mathbf{y} \mid \mathbf{y}_t; \theta)\,\mathrm{d}t\right], \tag{19}$$

where $(\mathbf{x}, \mathbf{y}) \sim p_{\text{data}}(\mathbf{x}, \mathbf{y})$, and we now have $t \sim \mathcal{U}(0, 1)$. This objective differs from the standard FM objective and can be seen as a special case of variational FM [22]. However, the motivation and derivations presented here are distinct, as they are a natural consequence of infinite-depth Flow-SSNs.

Priors. Alternative priors for continuous-time Flow-SSNs can be specified by, e.g., a pushforward of the base distribution $f_\# p_{U|X}$, then defining the interpolant $\mathrm{d}\mathbf{y}_t = (\mathbf{y} - \boldsymbol{\eta})\mathrm{d}t$. For instance, if we choose $\boldsymbol{\eta} = f(\mathbf{u})$ as the log-softmax function, then $f_\# p_{U|X}$ is log-logistic normal. Another promising avenue is to consider mixture distributions (e.g. Gaussian) and/or leverage large foundation models as flexible priors.

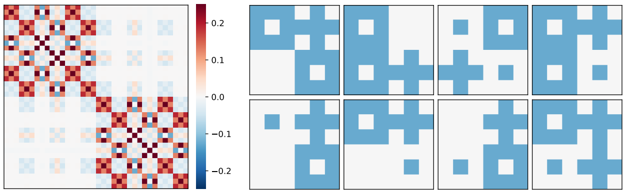


Figure 3. **The *MarkovShapes* dataset**. (*Left*) Ground truth pixel covariance matrix, rank r_{true}=12. (*Right*) Random data samples.

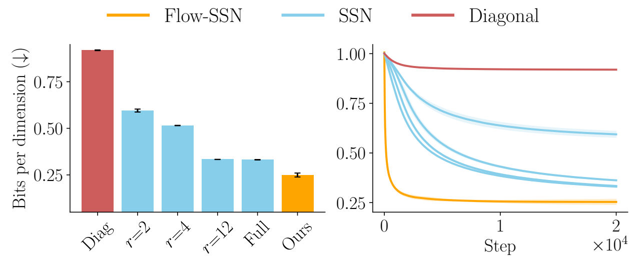
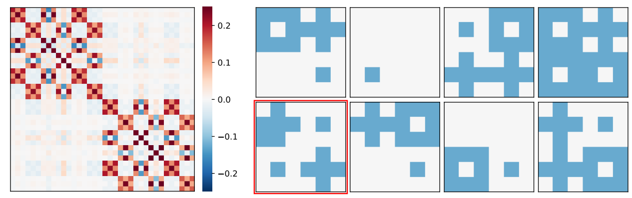
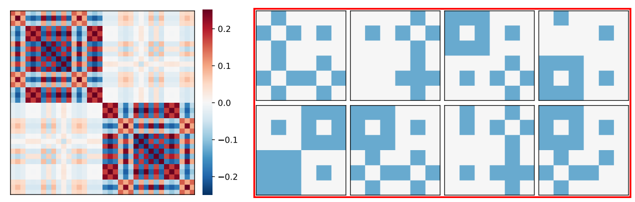


Figure 4. Bits per dimension (BPD) results on *MarkovShapes*.



(a) Cov (Flow-SSN) **(b)** Samples (Flow-SSN)



(c) Cov (SSN r=2) **(d)** Samples (SSN r=2)

Figure 5. **Comparing learnt covariances on *MarkovShapes***. (*Top*) Flow-SSNs make no rank assumptions and approximate the ground truth covariance well. (*Bottom*) The low-rank approximation causes sampling errors, even under a mild underspecification ratio, r=2 relative to r_{true}=12. Sampling errors (hallucinated shapes not faithful to the true covariance structure) are indicated with red boxes.

6. Experiments

6.1. Toy Problem: MarkovShapes

In this experiment, we introduce a synthetic dataset, *MarkovShapes*, where we have full control of the rank of the pixel-space covariance. Comparing Flow-SSN with the low-rank SSN model [55], *MarkovShapes* reveals how the SSN fails when the assumed rank is underspecified and demonstrates how Flow-SSN overcomes this shortcoming.

MarkovShapes consists of images composed of 3 possible shapes: '*square*' ( ), '*plus*' ( ) and '*dot*'( ). Each image quadrant is either filled with a shape or left empty (∅) at random. The data-generating process is a Markov chain, with states corresponding to shapes and the transitions between quadrants governed by the doubly stochastic matrix $\mathbf{T}$:

$$\mathbf{T} = \begin{array}{c} \varnothing \\ \\ \\ \\ \end{array} \begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 3/40 & 27/80 & 27/80 \\ 1/4 & 27/80 & 3/40 & 27/80 \\ 1/4 & 27/80 & 27/80 & 3/40 \end{bmatrix}, \tag{20}$$

where $\mathbf{T}_{ij} = P(X_{t+1} = j \mid X_t = i)$ is the probability of transitioning from shape i to j within a particular image. Figure 3 shows the induced pixel covariance matrix, its empirically calculated true rank r_{true}, and random samples.

We adapt the SSN from the toy problem of Monteiro et al. [55], training four variants from rank 2 to full-rank on *MarkovShapes*. We implement and train a discrete-time autoregressive Flow-SSN with the objective in Eq. (14). We choose a single linear layer with MADE-style masking [26] for the autoregressive flow model. In all cases, we train for 20K steps using the Adam optimiser with 10^{-3} learning rate, batch size 32, and 512 MC samples. Figure 4 plots the performance of Flow-SSN against the baseline SSN, showing how Flow-SSN outperforms all SSN variants. Notably, Flow-SSN converges faster *and* achieves a better final result than even the full-rank SSN. In Figure 5, we show how samples from Flow-SSN faithfully represent each of the underlying shapes (i.e. , , and). In contrast, the rank 2 SSN introduces sampling errors, hallucinating nonexistent shapes due to poorly modelling the true covariance structure.

Rank Underspecification Ratio. The underspecification ratio used for the SSN models on *MarkovShapes* (most extreme is r=2 : r_{true}=12) is mild compared to more complex high-dimensional real-world data, where $r \ll r_{\text{true}}$. Thus, the low-rank assumption of SSNs, typically r=10, is likely even more punishing in real-world settings. As shown in the next experiment, simply increasing the assumed rank for real-world data collapses the SSN to a deterministic model.

6.2. Lung Nodule Segmentation

The LIDC-IDRI dataset [4] is a standard benchmark for evaluating stochastic segmentation models, with multiple ground truth label maps per image reflecting the inherent variability among medical experts. LIDC contains 1018 lung CT scans, with lung nodule annotation masks provided by 4 radiologists (from a pool of 12). We use the preprocessing from Baumgartner et al. [6], Kohl et al. [43], where 128×128 patches are extracted such that each patch is centred on a nodule. The process yields 15,096 slices with 4 segmentations

Table 1. **Quantitative results on LIDC-IDRI**. Flow-SSN achieves SOTA performance with fewer params. (Δ/∞ denote discrete/continuous time variants). Results with (†) are from Zhang et al. [96]. $D^2_{\text{GED}}(M)$ denotes $16<M\leq100$ MC samples used. More baselines in App. E.

	LIDC-IDRI (128×128)					
METHOD	$D^2_{\text{GED}}(16)\downarrow$	$D^2_{\text{GED}}(M)\downarrow$	Diversity ↑	Dice ↑	HM-IoU ↑	#Param
UNet [72]†	-	0.676±.000	-	0.519±.004	0.463±.000	9M
ProbUNet [43]	0.287±N/A	0.252±.004	0.469±.003	0.390±.004	0.500±.030	18M
cFlow [76]†	-	0.225±.002	0.523±.010	0.449±.000	0.584±.000	N/A
PHiSeg [6]	-	0.225±.004	0.496±.003	0.486±.010	0.595±.00†	63M
SSN [55]	-	0.225±.002	0.609±.002	0.436±.004	0.555±.01†	41M
P^2SAM [33]	0.218±N/A	0.216±N/A	-	-	0.679±N/A	N/A
SSN++ (ours)	0.241±.001	0.212±.001	0.575±.005	0.471±.003	-	20M
JProbUNet [96]	-	0.206±.000	0.475±.010	0.511±.010	0.647±.010	N/A
MoSE [25]	0.218±.003	0.189±.002	-	-	0.624±.004	42M
Flow-SSN_Δ : {IAF, 1-step}	0.240±.002	0.212±.001	0.598±.000	0.468±.002	**0.879**±.000	14M
Flow-SSN_∞ : {Dopri5, tol=1e-6}	0.209±.002	0.182±.001	0.521±.005	0.610±.003	0.872±.000	14M
Flow-SSN_∞ : {Euler, T=50}	0.210±.002	0.182±.000	0.518±.006	0.611±.003	0.873±.000	14M
Flow-SSN_∞ : {Euler, T=250}	**0.207**±.000	**0.181**±.000	0.520±.006	**0.611**±.007	0.873±.001	14M

each, with a $60/20/20$ train/valid/test split. As in previous work, we measure performance using Generalised Energy Distance (GED), Dice, and Hungarian-Matched Intersection over Union (HM-IoU) [44, 66, 92]. Our architecture is a streamlined version of the Dhariwal and Nichol [19]'s UNet to parameterise both the prior and the continuous-time flow. We use a *single* autoregressive Transformer layer to parameterise our discrete-time Flow-SSN; the IAF variant trained using Eq. (14). For more details/results, see Appendix D/E.

Table 1 reports our results. Our baseline SSN (SSN++) outperforms the original [55], and the Flow-SSN achieves state-of-the-art performance in all metrics using fewer parameters[4]. Fig. 6 shows ablation results for learning the base distribution (i.e. prior) vs holding it fixed (as typically done). We observe a *significant* reduction in inference time without sacrificing performance, thanks to using a small network for the flow and allocating most of the model capacity to the prior (150K vs 14M params). Flow-SSN is $\simeq10\times$ more efficient than CCDM [92] (cf. App. E), and can perform competitively with just 10 ODE solver steps (c.f. Fig. 7).

6.3. Optical Cup Segmentation

REFUGE2 [23] is a publicly available benchmark dataset for optic disk and optic cup segmentation. Each fundus image in the dataset has multiple associated ground-truth label maps. Specifically, it includes annotations from 7 different independent ophthalmologists, each with an average of 8 years of experience. It contains a total of 1200 images, 400 for training, validation, and testing, respectively. Segmentation of the optic cup is inherently more variable than segmentation of the

[4] We note that diversity is only relevant contextually, as high diversity can be trivially achieved with random noise as a model.

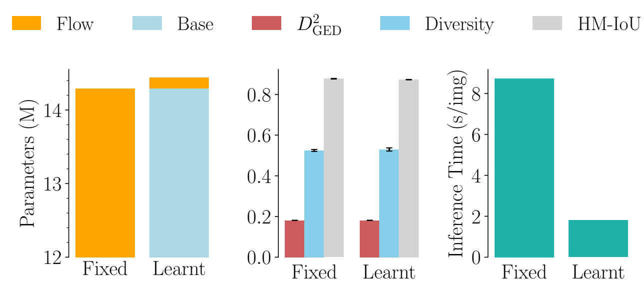


Figure 6. **Comparing learnt vs fixed priors in Flow-SSNs**. We observe a $5\times$ reduction in inference time without sacrificing performance. The flow network used in the learnt prior setup is around $100\times$ smaller than the fixed prior baseline, making ODE solving much cheaper. Note that s/img here includes inferring 100 MC samples per image, effectively performing 100 forward passes.

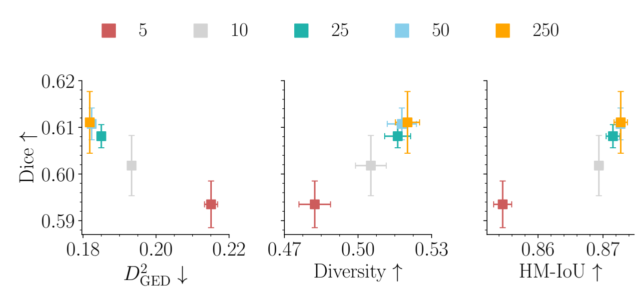


Figure 7. **Ablation study of the number of ODE solving steps**. Results (LIDC-IDRI) obtained using the Euler method. Flow-SSNs can perform competitively with as few as T=10 ODE solver steps.

optic disk [23], therefore, we only consider the former in this work. As before, we measure performance using GED, Dice,

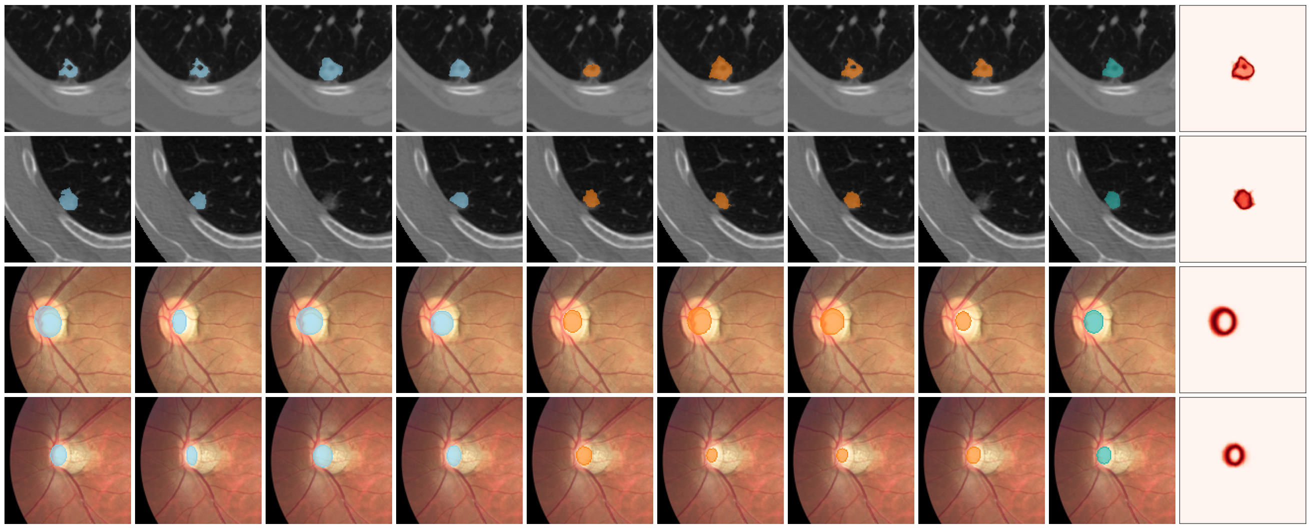


Figure 8. **Qualitative results on LIDC-IDRI (*Rows 1, 2*) and REFUGE-MultiRater (*Rows 3, 4*) using Flow-SSN**. (*Cols. 1-4*) Multiple ground-truth expert segmentations; (*Cols. 5-8*) Non-cherry-picked model samples; (*Cols. 9, 10*) Mean prediction and pixel uncertainty map.

Table 2. **Stochastic segmentation on REFUGE-MultiRater**. We set *new* benchmarks for D^2_{GED}, Diversity and HM-IoU. We use Δ and ∞ to denote discrete and continuous-time Flow-SSN variants.

		REFUGE-MultiRater (256×256)		
METHOD	M	$D^2_{\text{GED}}\downarrow$	Diversity ↑	HM-IoU ↑
Flow-SSN$_\Delta$	16	0.116±.007	0.441±.012	0.766±.006
	100	0.089±.004	0.462±.013	0.851±.001
	512	**0.081**±N/A	0.447±N/A	**0.881**±N/A
Flow-SSN$_\infty$	16	0.112±.003	0.424±.004	0.751±.003
	100	**0.089**±.001	0.453±.020	**0.832**±.004

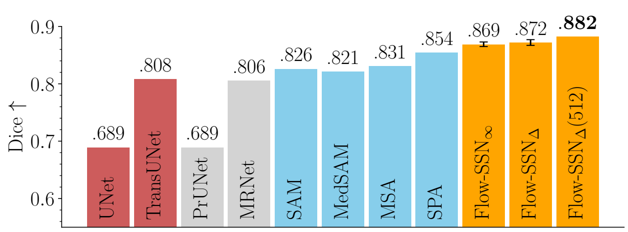


Figure 9. **Optical cup segmentation on REFUGE-MultiRater**. Baselines are from Zhu et al. [99]. We observe impressive Dice performance using modestly-sized Flow-SSNs with only 15M params.

and HM-IoU. For training, we resize the images to 256×256 to match the output size used by Zhu et al. [99] and enable fair comparisons with their reported Dice baselines. Namely, TransUNet [10], MRNet [10], SAM [42], MedSAM [51] and MSA [97]. For training details, please refer to Appendix D, and for extra ablation results see Appendix E. As shown in Fig. 9, Flow-SSNs outperform all baselines by a considerable margin using only 15M parameters. Table 2 provides *new* stochastic segmentation benchmark results, featuring both discrete and continuous-time Flow-SSN variants.

Surprisingly, we find that a discrete-time autoregressive Flow-SSN (a single-step IAF trained as per Eq. (14)) outperforms the continuous-time one on this dataset. We expect continuous-time Flow-SSNs to outperform shallow discrete-time ones in general, but this result encourages further exploration into this model class, as the discrete-time Flow-SSN is ≈200× faster to sample from than the continuous-time one. For this reason, we were able to push the number of MC samples used for evaluation up to M=512 and observed further significant gains in performance on all metrics.

7. Conclusion

We introduce the Flow Stochastic Segmentation Network (Flow-SSN), a generative segmentation model featuring both discrete-time autoregressive and modern continuous-time flow variants. By overcoming the constraints of low-rank parameterisations in standard SSNs, Flow-SSNs enable the estimation of arbitrarily high-rank pixel-wise covariances without requiring prior assumptions about rank or explicit storage of distributional parameters. Moreover, Flow-SSNs significantly improve sampling efficiency compared to standard diffusion-based segmentation models, thanks to a key architectural design that includes learning the base distribution (i.e. the prior) of the flow, which is typically fixed. Experimental results on standard real-world benchmarks demonstrate the efficacy of Flow-SSNs, achieving state-of-the-art performance, and highlighting their potential to advance stochastic segmentation in safety-critical applications with inherent ambiguities, such as medical imaging.

Acknowledgments

F.R. and B.G. acknowledge the support of the UKRI AI programme, and the EPSRC, for CHAI-EPSRC Causality in Healthcare AI Hub (grant no. EP/Y028856/1). O.T. and R.M. are funded by the European Union's Horizon Europe research and innovation programme under grant agreement 101080302. C.J. is supported by Microsoft Research, EPSRC, and The Alan Turing Institute through a Microsoft PhD scholarship and a Turing PhD enrichment award. A.K. is supported by UKRI (grant no. EP/S023356/1), as part of the UKRI Centre for Doctoral Training in Safe & Trusted AI, and acknowledges support from the EPSRC Doctoral Prize. B.G. received support from the Royal Academy of Engineering as part of his Kheiron/RAEng Research Chair.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016. 17

[2] Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 3, 5

[3] Tomer Amit, Tal Shaharbany, Eliya Nachmani, and Lior Wolf. Segdiff: Image segmentation with diffusion probabilistic models. *arXiv preprint arXiv:2112.00390*, 2021. 2, 5, 20

[4] Samuel G Armato III, Geoffrey McLennan, Luc Bidaut, Michael F McNitt-Gray, Charles R Meyer, Anthony P Reeves, Binsheng Zhao, Denise R Aberle, Claudia I Henschke, Eric A Hoffman, et al. The lung image database consortium (lidc) and image database resource initiative (idri): a completed reference database of lung nodules on ct scans. *Medical physics*, 38(2):915–931, 2011. 6

[5] Theodore Barfoot, Luis C Garcia Peraza Herrera, Ben Glocker, and Tom Vercauteren. Average calibration error: A differentiable loss for improved reliability in image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 139–149. Springer, 2024. 1

[6] Christian F Baumgartner, Kerem C Tezcan, Krishna Chaitanya, Andreas M Hötker, Urs J Muehlematter, Khoschy Schawkat, Anton S Becker, Olivio Donati, and Ender Konukoglu. Phiseg: Capturing uncertainty in medical image segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part II 22*, pages 119–127. Springer, 2019. 2, 6, 7

[7] Mélanie Bernhardt, Fabio De Sousa Ribeiro, and Ben Glocker. Failure detection in medical image classification: A reality check and benchmarking testbed. *Transactions on Machine Learning Research*, 2022. 1

[8] Lea Bogensperger, Dominik Narnhofer, Alexander Falk, Konrad Schindler, and Thomas Pock. Flowsdf: Flow matching for medical image segmentation using distance transforms. *arXiv preprint arXiv:2405.18087*, 2024. 2

[9] Hongwei Bran, Fernando Navarro, Ivan Ezhov, Amirhossein Bayat, Dhritiman Das, Florian Kofler, Suprosanna Shit, Diana Waldmannstetter, Johannes C Paetzold, Xiaobin Hu, et al. Qubiq: Uncertainty quantification for biomedical image segmentation challenge. *arXiv preprint arXiv:2405.18435*, 2024. 1

[10] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L. Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021. 8

[11] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 1

[12] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018. 2, 3, 5

[13] Ricky T. Q. Chen. torchdiffeq, 2018. https://github.com/rtqichen/torchdiffeq. 18

[14] Steffen Czolbe, Kasra Arnavaz, Oswin Krause, and Aasa Feragen. Is segmentation uncertainty useful? In *Information Processing in Medical Imaging: 27th International Conference, IPMI 2021, Virtual Event, June 28–June 30, 2021, Proceedings 27*, pages 715–726. Springer, 2021. 1

[15] Fabio De Sousa Ribeiro, Francesco Calivá, Mark Swainson, Kjartan Gudmundsson, Georgios Leontidis, and Stefanos Kollias. Deep bayesian self-training. *Neural Computing and Applications*, 32(9):4275–4291, 2020. 2

[16] Fabio De Sousa Ribeiro, Georgios Leontidis, and Stefanos Kollias. Introducing routing uncertainty in capsule networks. *Advances in neural information processing systems*, 33:6490–6502, 2020. 2

[17] Fabio De Sousa Ribeiro, Tian Xia, Miguel Monteiro, Nick Pawlowski, and Ben Glocker. High fidelity image counterfactuals with probabilistic causal models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 7390–7425, 2023. 17

[18] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009. 1

[19] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 7, 18, 19

[20] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 3

[21] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017. 2

[22] Floor Eijkelboom, Grigory Bartosh, Christian A. Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In *The Thirty-eighth Annual*

Conference on Neural Information Processing Systems, 2024. 5

[23] Huihui Fang, Fei Li, Junde Wu, Huazhu Fu, Xu Sun, Jaemin Son, Shuang Yu, Menglu Zhang, Chenglang Yuan, Cheng Bian, et al. Refuge2 challenge: A treasure trove for multi-dimension analysis and evaluation in glaucoma screening. *arXiv preprint arXiv:2202.08994*, 2022. 7

[24] Zhitong Gao, Yucong Chen, Chuyu Zhang, and Xuming He. Modeling multimodal aleatoric uncertainty in segmentation with mixture of stochastic experts. *International Conference on Representation Learning (ICLR)*, 2022. 2

[25] Zhitong Gao, Yucong Chen, Chuyu Zhang, and Xuming He. Modeling multimodal aleatoric uncertainty in segmentation with mixture of stochastic experts. In *The Eleventh International Conference on Learning Representations*, 2023. 7, 20

[26] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International conference on machine learning*, pages 881–889. PMLR, 2015. 6

[27] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael Black, and Bernhard Scholkopf. From variational to deterministic autoencoders. In *International Conference on Learning Representations*, 2020. 2

[28] Jiatao Gu, Tianrong Chen, David Berthelot, Huangjie Zheng, Yuyang Wang, Ruixiang Zhang, Laurent Dinh, Miguel Angel Bautista, Josh Susskind, and Shuangfei Zhai. Starflow: Scaling latent normalizing flows for high-resolution image synthesis. *arXiv preprint arXiv:2506.06276*, 2025. 2, 21

[29] István Gyöngy. Mimicking the one-dimensional marginal distributions of processes having an itô differential. *Probability theory and related fields*, 71(4):501–516, 1986. 5

[30] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993. 2

[31] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2

[32] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. 4

[33] Yuzhi Huang, Chenxin Li, Zixu Lin, Hengyu Liu, Haote Xu, Yifan Liu, Yue Huang, Xinghao Ding, Xiaotong Tu, and Yixuan Yuan. P2sam: Probabilistically prompted sams are efficient segmentator for ambiguous medical images. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 9779–9788, 2024. 7

[34] Aapo Hyvärinen, Ilyes Khemakhem, and Ricardo Monti. Identifiability of latent-variable and structural-equation models: from linear to nonlinear. *Annals of the Institute of Statistical Mathematics*, 76(1):1–33, 2024. 2

[35] Leo Joskowicz, D Cohen, N Caplan, and Jacob Sosna. Inter-observer variability of manual contour delineation of structures in ct. *European radiology*, 29:1391–1399, 2019. 1

[36] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017. 1, 2

[37] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. 1, 2

[38] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018. 2

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014. 2

[40] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016. 3, 5, 15, 19

[41] Michael Kirchhof, Gjergji Kasneci, and Enkelejda Kasneci. Position: Uncertainty quantification needs reassessment for large language model agents. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025. 1

[42] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 8

[43] Simon Kohl, Bernardino Romera-Paredes, Clemens Meyer, Jeffrey De Fauw, Joseph R Ledsam, Klaus Maier-Hein, SM Eslami, Danilo Jimenez Rezende, and Olaf Ronneberger. A probabilistic u-net for segmentation of ambiguous images. *Advances in neural information processing systems*, 31, 2018. 1, 2, 6, 7, 18

[44] Simon AA Kohl, Bernardino Romera-Paredes, Klaus H Maier-Hein, Danilo Jimenez Rezende, SM Eslami, Pushmeet Kohli, Andrew Zisserman, and Olaf Ronneberger. A hierarchical probabilistic u-net for modeling multi-scale ambiguities. *arXiv preprint arXiv:1905.13077*, 2019. 7, 18

[45] Yongchan Kwon, Joong-Ho Won, Beom Joon Kim, and Myunghee Cho Paik. Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics & Data Analysis*, 142:106816, 2020. 2

[46] James Langley, Miguel Monteiro, Charles Jones, Nick Pawlowski, and Ben Glocker. Structured uncertainty in the observation space of variational autoencoders. *Transactions on Machine Learning Research*, 2022. 1

[47] Yucen Lily Li, Daohan Lu, Polina Kirichenko, Shikai Qiu, Tim G. J. Rudner, C. Bayan Bruss, and Andrew Gordon Wilson. Position: Supervised classifiers answer the wrong questions for OOD detection. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025. 1

[48] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 3, 5

[49] Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 3, 5

[50] James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[51] Jun Ma, Yuting He, Feifei Li, Lin Han, Chenyu You, and Bo Wang. Segment anything in medical images. *Nature Communications*, 15(1):654, 2024. 8

[52] David John Cameron Mackay. *Bayesian methods for adaptive models*. California Institute of Technology, 1992. 2

[53] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31, 2018. 2

[54] Raghav Mehta, Fabio De Sousa Ribeiro, Tian Xia, Melanie Roschewitz, Ainkaran Santhirasekaram, Dominic C Marshall, and Ben Glocker. Cf-seg: Counterfactuals meet segmentation. *arXiv preprint arXiv:2506.16213*, 2025. 2

[55] Miguel Monteiro, Loïc Le Folgoc, Daniel Coelho de Castro, Nick Pawlowski, Bernardo Marques, Konstantinos Kamnitsas, Mark van der Wilk, and Ben Glocker. Stochastic segmentation networks: Modelling spatially correlated aleatoric uncertainty. *Advances in neural information processing systems*, 33:12756–12767, 2020. 1, 2, 3, 4, 6, 7, 18, 21, 22

[56] Miguel Monteiro, Fabio De Sousa Ribeiro, Nick Pawlowski, Daniel C. Castro, and Ben Glocker. Measuring axiomatic soundness of counterfactual image models. In *The Eleventh International Conference on Learning Representations*, 2023. 17

[57] Radford M Neal. Probabilistic inference using markov chain monte carlo methods. Technical report, Department of Computer Science, University of Toronto, ON, Canada, 1993. 2

[58] Elias Nehme, Omer Yair, and Tomer Michaeli. Uncertainty quantification via neural posterior principal components. *Advances in Neural Information Processing Systems*, 36:37128–37141, 2023. 1

[59] Frank Nussbaum, Jakob Gawlikowski, and Julia Niebling. Structuring uncertainty for fine-grained sampling in stochastic segmentation networks. *Advances in Neural Information Processing Systems*, 35:27678–27691, 2022. 2

[60] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pages 3918–3926. PMLR, 2018. 16

[61] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017. 2, 3, 5, 15, 19

[62] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. 3

[63] Theodore Papamarkou, Maria Skoularidou, Konstantina Palla, Laurence Aitchison, Julyan Arbel, David Dunson, Maurizio Filippone, Vincent Fortuin, Philipp Hennig, José Miguel Hernández-Lobato, Aliaksandr Hubin, Alexander Immer, Theofanis Karaletsos, Mohammad Emtiyaz Khan, Agustinus Kristiadi, Yingzhen Li, Stephan Mandt, Christopher Nemeth, Michael A Osborne, Tim G. J. Rudner, David Rügamer, Yee Whye Teh, Max Welling, Andrew Gordon Wilson, and Ruqi Zhang. Position: Bayesian deep learning is needed in the age of large-scale AI. In *Proceedings of the 41st International Conference on Machine Learning*, pages 39556–39586. PMLR, 2024. 1

[64] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 18

[65] Stefano Peluchetti. Non-denoising forward-time diffusions, 2022. https://openreview.net/forum?id=oVfIKuhqfC. 5

[66] Aimon Rahman et al. Ambiguous medical image segmentation using diffusion models. In *IEEE/CVF conference on computer vision and pattern recognition*, 2023. 2, 5, 7, 20, 22

[67] Marianne Rakic, Hallee E Wong, Jose Javier Gonzalez Ortiz, Beth A Cimini, John V Guttag, and Adrian V Dalca. Tyche: Stochastic in-context learning for medical image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11159–11173, 2024. 20

[68] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015. 3

[69] Danilo Jimenez Rezende and Fabio Viola. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018. 2

[70] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014. 2

[71] Fabio De Sousa Ribeiro, Ben Glocker, et al. Demystifying variational diffusion models. *Foundations and Trends® in Computer Graphics and Vision*, 17(2):76–170, 2025. 2

[72] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 7, 18

[73] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE, 2007. 4

[74] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: Improving the pixelCNN with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, 2017. 2, 14, 19

[75] John Schulman. The KL approximation, 2017. http://joschu.net/blog/kl-approx.html. 16

[76] Raghavendra Selvan, Frederik Faye, Jon Middleton, and Akshay Pai. Uncertainty quantification in medical image segmentation with normalizing flows. In *Machine Learning in*

Medical Imaging: 11th International Workshop, MLMI 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4, 2020, Proceedings 11, pages 80–90. Springer, 2020. 2, 7

[77] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. *Advances in neural information processing systems*, 31, 2018. 2

[78] Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion schrödinger bridge matching. *Advances in Neural Information Processing Systems*, 36, 2024. 5

[79] Freddie Bickford Smith, Jannik Kossen, Eleanor Trollope, Mark van der Wilk, Adam Foster, and Tom Rainforth. Rethinking aleatoric and epistemic uncertainty. *arXiv preprint arXiv:2412.20892*, 2024. 1

[80] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 2

[81] Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013. 2, 3

[82] Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. Expert Certification. 2

[83] Michael Tschannen, André Susano Pinto, and Alexander Kolesnikov. Jetformer: An autoregressive generative model of raw images and text. In *The Thirteenth International Conference on Learning Representations*, 2025. 2, 21

[84] Dennis Thomas Ulmer, Christian Hardmeier, and Jes Frellsen. Prior and posterior networks: A survey on evidential deep learning methods for uncertainty estimation. *Transactions on Machine Learning Research*, 2023. 2

[85] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020. 3

[86] MM Amaan Valiuddin, Christiaan GA Viviers, Ruud JG van Sloun, Peter HN de With, and Fons van der Sommen. Improving aleatoric uncertainty quantification in multi-annotated medical image segmentation with normalizing flows. In *Uncertainty for Safe Utilization of Machine Learning in Medical Imaging, and Perinatal Imaging, Placental and Preterm Image Analysis: 3rd International MICCAI Workshop*, pages 75–88. Springer, 2021. 2

[87] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016. 2, 14

[88] Aäron Van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016. 14, 19

[89] Junde Wu, Rao Fu, Huihui Fang, Yu Zhang, Yehui Yang, Haoyi Xiong, Huiying Liu, and Yanwu Xu. Medsegdiff: Medical image segmentation with diffusion probabilistic model. In *Medical Imaging with Deep Learning*, pages 1623–1639. PMLR, 2024. 2, 5, 20

[90] Lihe Yang, Wei Zhuo, Lei Qi, Yinghuan Shi, and Yang Gao. St++: Make self-training work better for semi-supervised semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4268–4277, 2022. 2

[91] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*, 2018. 4

[92] Lukas Zbinden et al. Stochastic segmentation with conditional categorical diffusion models. In *International Conference on Computer Vision*, 2023. 2, 5, 7, 20

[93] Kilian Zepf, Eike Petersen, Jes Frellsen, and Aasa Feragen. That label's got style: Handling label style bias for uncertain image segmentation. *International Conference on Representation Learning (ICLR)*, 2023. 2

[94] Shuangfei Zhai, Ruixiang ZHANG, Preetum Nakkiran, David Berthelot, Jiatao Gu, Huangjie Zheng, Tianrong Chen, Miguel Ángel Bautista, Navdeep Jaitly, and Joshua M. Susskind. Normalizing flows are capable generative models. In *Forty-second International Conference on Machine Learning*, 2025. 2, 21

[95] Wei Zhang, Xiaohong Zhang, Sheng Huang, Yuting Lu, and Kun Wang. Pixelseg: Pixel-by-pixel stochastic semantic segmentation for ambiguous medical images. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4742–4750, 2022. 2

[96] Wei Zhang, Xiaohong Zhang, Sheng Huang, Yuting Lu, and Kun Wang. A probabilistic model for controlling diversity and accuracy of ambiguous medical image segmentation. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4751–4759, 2022. 7

[97] Yichi Zhang, Zhenrong Shen, and Rushi Jiao. Segment anything model for medical image segmentation: Current applications and future directions. *Computers in Biology and Medicine*, page 108238, 2024. 8

[98] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537, 2015. 1

[99] Jiayuan Zhu, Junde Wu, Cheng Ouyang, Konstantinos Kamnitsas, and Alison Noble. Spa: Efficient user-preference alignment against uncertainty in medical image segmentation. *arXiv preprint arXiv:2411.15513*, 2024. 8

Appendices

Table of Contents

A Theoretical Rank Analysis: Proofs

Lemma A.1 (Rank Increase)**.** *Let the logits $\boldsymbol{\eta} \in \mathbb{R}^{kd}$ be low-rank Gaussian distributed $\boldsymbol{\eta} \mid \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\Sigma}(\mathbf{x})) = p_{H|X}$, where the covariance matrix $\boldsymbol{\Sigma}(\mathbf{x}) \in \mathbb{R}^{kd \times kd}$ has rank $\operatorname{rank}(\boldsymbol{\Sigma}(\mathbf{x})) = r$. If we define $\mathbf{y} = g(\boldsymbol{\eta})$, where $g = \operatorname{softmax}_k$, the rank of the covariance matrix $\operatorname{Cov}(\mathbf{y})$ of the pushforward distribution $g_{\#}p_{H|X}$ increases under the condition:*

$$\operatorname{rank}(\operatorname{Cov}(\mathbf{y})) > r \iff r < d(k-1). \tag{21}$$

Proof. First recall that we use $\operatorname{softmax}_k(\cdot)$ to denote that the softmax function is applied row-wise across the k dimension (i.e. over the number of categories k), after reshaping the input logits $\boldsymbol{\eta} \in \mathbb{R}^{kd}$ to $\boldsymbol{\eta} \in \mathbb{R}^{k \times d}$. More formally:

$$\mathbf{y}_{:,j} = \operatorname{softmax}(\boldsymbol{\eta}_{:,j}), \qquad \text{where} \qquad y_{ij} = \frac{e^{\eta_{ij}}}{\sum_{l=1}^{k} e^{\eta_{lj}}}, \qquad \text{for} \quad i = 1, \dots, k, \ j = 1, \dots, d. \tag{22}$$

Now let $\mathbf{f} : \mathbb{R}^{kd} \to (0,1)^{kd}$ be the composition of $\operatorname{softmax}_k(\boldsymbol{\eta})$ then flattening $\mathbf{y} \in (0,1)^{k \times d}$ back to $\mathbf{y} \in (0,1)^{kd}$. The resulting Jacobian $\mathbf{J_f} \in \mathbb{R}^{kd \times kd}$ of $\mathbf{f}$ is a sparse matrix given by:

$$\mathbf{J_f} = \begin{bmatrix} \frac{\partial y_1}{\partial \eta_1} & \cdots & \frac{\partial y_1}{\partial \eta_{kd}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_{kd}}{\partial \eta_1} & \cdots & \frac{\partial y_{kd}}{\partial \eta_{kd}} \end{bmatrix}, \qquad \text{where} \qquad \frac{\partial y_i}{\partial \eta_j} \neq 0 \quad \text{if} \quad j \equiv i \pmod{k}. \tag{23}$$

In words, $\mathbf{J_f}$ has a total of dk^2 non-zero entries, including the diagonal and a checkerboard-like pattern with column-wise intervals of stride k, caused by the flattening operation.

Intuitively, $\mathbf{f}$ non-linearly couples each element in $\boldsymbol{\eta}$ with $k-1$ number of other elements in $\boldsymbol{\eta}$ to produce $\mathbf{y}$, i.e. each $\boldsymbol{\eta}_{:,j} \in \mathbb{R}^{k\times 1}$ is independently mapped to the $(k-1)$-dimensional simplex Δ^{k-1} via an element-wise softmax, for $j = 1, 2, \dots, d$. As a result, we can conclude that $\operatorname{rank}(\mathbf{J}_\mathbf{f}) = d(k-1)$.

Considering a Taylor expansion of $\operatorname{Cov}(\mathbf{y})$, we know that the first-order term only contributes linearly to the result:

$$\operatorname{Cov}(\mathbf{y}) = \mathbf{J}_\mathbf{f}\boldsymbol{\Sigma}(\mathbf{x})\mathbf{J}_\mathbf{f}^\top + \cdots, \qquad \text{and} \qquad \operatorname{rank}(\mathbf{J}_\mathbf{f}\boldsymbol{\Sigma}(\mathbf{x})\mathbf{J}_\mathbf{f}^\top) = \min(r, d(k-1)) = r. \tag{24}$$

Since $\mathbf{f}$ is non-linear, the higher-order terms in the Taylor expansion (and their rank) must be non-zero. Thus, by the rank subadditivity property: $\operatorname{rank}(A+B) \leq \operatorname{rank}(A) + \operatorname{rank}(B)$, we conclude that $\operatorname{rank}(\operatorname{Cov}(\mathbf{y})) > r$ iff $r < d(k-1)$. □

Theorem A.2 (Sublinear Growth of the Effective Rank)**.** *Given a low-rank Gaussian covariance matrix* $\boldsymbol{\Sigma}(\mathbf{x}) \in \mathbb{R}^{kd\times kd}$ *with initial rank* $\operatorname{rank}(\boldsymbol{\Sigma}(\mathbf{x})) = r < d(k-1)$*. The increase in the effective rank* $\operatorname{erank}(\operatorname{Cov}(\mathbf{y}))$*, in the sense of Lemma 4.1, grows sublinearly with respect to the initial rank* r*.*

Proof. Recall that the rank of $\boldsymbol{\Sigma}(\mathbf{x}) \in \mathbb{R}^{kd\times kd}$ is by definition given by:

$$\operatorname{rank}(\boldsymbol{\Sigma}(\mathbf{x})) = \sum_{i=1}^{kd} \mathbb{I}(\sigma_i > 0), \qquad \text{where} \qquad \sigma_1 \geq \sigma_2 \geq, \dots, \sigma_{kd} \geq 0 \tag{25}$$

are the singular values of $\boldsymbol{\Sigma}(\mathbf{x})$, and $\mathbb{I}(\cdot)$ is the indicator function. Since $\operatorname{rank}(\boldsymbol{\Sigma}(\mathbf{x})) = r$ we have $\sigma_1 \geq, \dots, \geq \sigma_r > 0$.

Given that the non-linear pushforward operation defined in Lemma 4.1 is rank-increasing (as long as $r < d(k-1)$), we can assert that the final rank, $\operatorname{rank}(\operatorname{Cov}(\mathbf{y})) = r_1$, grows *at least linearly* w.r.t. the initial rank $\operatorname{rank}(\boldsymbol{\Sigma}(\mathbf{x})) = r$. This is intuitive as each non-zero singular value of $\boldsymbol{\Sigma}(\mathbf{x})$ contributes at least one additional rank increment to $\operatorname{rank}(\operatorname{Cov}(\mathbf{y}))$.

This holds for the *effective rank* $\operatorname{erank}(\operatorname{Cov}(\mathbf{y}))$ if and only if the distribution of singular values of $\operatorname{Cov}(\mathbf{y})$, p, is uniform:

$$\operatorname{erank}(\operatorname{Cov}(\mathbf{y})) = \operatorname{rank}(\operatorname{Cov}(\mathbf{y})) \iff p_i = \frac{1}{r_1}, \quad \text{for} \quad i = 1, 2, \dots, r_1, \tag{26}$$

which is straightforward to show by substituting p into the definition of effective rank (c.f. Definition 4.2):

$$H(p) = -\sum_{i=1}^{r_1} \frac{1}{r_1} \log\left(\frac{1}{r_1}\right) = \log r_1 \implies \operatorname{erank}(\operatorname{Cov}(\mathbf{y})) = e^{H(p)} = r_1 = \operatorname{rank}(\operatorname{Cov}(\mathbf{y})). \tag{27}$$

Alternatively, when the singular value distribution p is non-uniform, e.g. skewed, we have that:

$$\sigma_1 \gg \sigma_2 \geq, \dots, \geq \sigma_{r_1} > 0 \implies H(p) < \log r_1 \implies \operatorname{erank}(\operatorname{Cov}(\mathbf{y})) < \operatorname{rank}(\operatorname{Cov}(\mathbf{y})), \tag{28}$$

since the uniform is the maximum entropy distribution and $H(p)$ is bounded above by $\log r_1$. For a random matrix with continuous entries and $kd > 1$, the probability that all its singular values are equal is zero; so the singular value distribution p is almost surely non-uniform. Thus, since linear growth holds iff p is uniformly distributed (c.f. Eq. (26)), the final effective rank $\operatorname{erank}(\operatorname{Cov}(\mathbf{y})) = r_1$ must grow *sublinearly* w.r.t. the initial rank $\operatorname{rank}(\boldsymbol{\Sigma}(\mathbf{x})) = r$, concluding the proof. □

Remark A.3. One way to intuit this result is by recalling that entropy grows logarithmically with the number of dimensions in uniform distributions. For non-uniform distributions, the growth of entropy can be slower, depending on how the additional event (represented by, say, $\sigma_{r+1} > 0$ in our case) relates to the existing probabilities $p_1, p_2, \dots, p_r$.

B Flow Stochastic Segmentation Networks: Proofs

We begin by reviewing autoregressive image models, then proceed to theoretical results and design options for Flow-SSNs.

B.1 Autoregressive Image Models

Autoregressive models factorise joint probability distributions into a product of conditional distributions using the chain rule of probability. For example, by representing an image $\mathbf{y}$ as a sequence of pixels $y_1, y_2, \dots, y_T$, we can estimate the joint pixel distribution by maximum likelihood estimation on the parameters of a model:

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p(\mathbf{y}_i; \boldsymbol{\theta}), \qquad p(\mathbf{y}_i; \boldsymbol{\theta}) = \prod_{t=1}^{T} p(y_{i,t} \mid y_{i,1}, \dots, y_{i,t-1}; \boldsymbol{\theta}_t). \tag{29}$$

Autoregressive image models [74, 87, 88] can estimate arbitrary joint pixel distributions but are computationally costly to sample from as each pixel must be generated sequentially. Discrete-time autoregressive Flow-SSNs avoid sequential sampling.

Proposition B.1 (Full Covariance Flow Transformation)**.** *Let* $\mathbf{u} = (u_1, u_2, \ldots, u_d)^\top$ *be a Gaussian random vector with diagonal covariance* $\mathcal{N}(\boldsymbol{\mu}, \operatorname{diag}(\boldsymbol{\sigma}^2))$*. A linear autoregressive model is sufficient to transform* $\mathbf{u}$ *into a new variable* $\boldsymbol{\eta} \in \mathbb{R}^d$ *with full covariance* $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$*.*

Proof. First recall that autoregressive models can represent any joint distribution as a product of conditionals: $p(\boldsymbol{\eta}; \theta) = \prod_{i=1}^{d} p(\eta_i \mid \eta_1, \ldots, \eta_{i-1}; \theta)$. Define the autoregressive transformation of $\mathbf{u}$ into $\boldsymbol{\eta}$ as follows:

$$\eta_i = \mu_i(\boldsymbol{\eta}_{1:i-1}) + \sigma_i(\boldsymbol{\eta}_{1:i-1})u_i, \qquad \text{for} \quad i = 1, 2, \ldots, d, \tag{30}$$

where $\mu_i(\cdot)$ and $\sigma_i(\cdot)$ are functions of preceding elements $\boldsymbol{\eta}_{1:i-1}$, thereby inducing a lower triangular dependency structure in $\boldsymbol{\eta}$. Thus, there exists a lower triangular matrix $\mathbf{L} \in \mathbb{R}^{d\times d}$ such that $\mathbf{L}\mathbf{L}^\top = \boldsymbol{\Sigma}$ (via Cholesky decomposition), and the autoregressive transformation can be equivalently expressed as a set of linear equations of the form:

$$\eta_i = \sum_{j=1}^{i-1} L_{ij}u_j + L_{ii}u_i, \qquad \text{for} \quad i = 1, 2, \ldots, d, \tag{31}$$

where L_{ij} determine the linear dependencies between elements, thereby inducing a full covariance structure for $\boldsymbol{\eta}$. $\square$

Remark B.2. Kingma et al. [40] undoubtedly recognised this fact, as it was informally mentioned in their exposition. We provide a simple proof here to make the argument in favour of our proposed approach more rigorous.

B.2 Choosing a Flow & Optimization Objective

The following provides supplementary derivations of the different optimization objectives for training discrete-time Flow-SSNs outlined in the main paper. Recall the discrete-time Flow-SSN is defined as follows:

$$p(\mathbf{y} \mid \mathbf{x}) = \int p(\mathbf{y} \mid \boldsymbol{\eta})p(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)\, \mathrm{d}\boldsymbol{\eta} \tag{32}$$

$$\text{where} \quad p(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta) = p_{U|X}(\mathbf{u} \mid \mathbf{x}; \lambda) \left|\det \mathbf{J}_\phi(\mathbf{u})\right|^{-1}, \quad \text{and} \quad p(\mathbf{y} \mid \boldsymbol{\eta}) = \operatorname{Cat}(\mathbf{y}; \operatorname{softmax}_k(\boldsymbol{\eta})). \tag{33}$$

We then have to choose a flow to model $p(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)$, and for this we revisit affine autoregressive flows, specifically IAFs [40] and MAFs [61] as they are both simple and flexible enough for our needs. They remain relatively underexplored to date, and in this work, we combine them with modern autoregressive Transformers to build Flow-SSNs. With that in mind, in the following subsections, we detail the various design options available for Flow-SSNs.

B.3 Expected Categorical Likelihood: Monte Carlo Estimator

The simplest approach is to use an IAF $p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)$, which is fast to sample form, and use a simple Monte Carlo estimator of the likelihood in Eq. (32) analogous to a standard SSN:

$$p(\mathbf{y} \mid \mathbf{x}) = \mathbb{E}_{\boldsymbol{\eta}\sim p^{\text{IAF}}(\boldsymbol{\eta}|\mathbf{x};\lambda,\theta)} \left[p(\mathbf{y} \mid \boldsymbol{\eta})\right] \tag{34}$$

$$= \mathbb{E}_{\mathbf{u}\sim p_{U|X}(\mathbf{u}|\mathbf{x};\lambda)} \left[p(\mathbf{y} \mid \boldsymbol{\eta} = \phi(\mathbf{u}; \theta))\right] \tag{35}$$

$$\approx \frac{1}{M}\sum_{i=1}^{M} p(\mathbf{y} \mid \phi(\mathbf{u}^{(i)}; \theta)), \qquad \mathbf{u}^{(i)}|\mathbf{x} \sim p_{U|X}, \tag{36}$$

where $(\mathbf{x}, \mathbf{y}) \sim p_{\text{data}}(\mathbf{x}, \mathbf{y})$ and the flow is parameterised by an autoregressive model with parameters θ. Taking logs, we get:

$$\log p(\mathbf{y} \mid \mathbf{x}) \approx \log \frac{1}{M}\sum_{i=1}^{M} p(\mathbf{y} \mid \phi(\mathbf{u}^{(i)}; \theta)) \tag{37}$$

$$= \operatorname{LSE}\left(\log p(\mathbf{y} \mid \phi(\mathbf{u}^{(1)}; \theta)) + \cdots + \log p(\mathbf{y} \mid \phi(\mathbf{u}^{(M)}; \theta))\right) - \log M, \tag{38}$$

where $\operatorname{LSE}(\cdot)$ is the log-sum-exp function, and the individual categorical likelihood terms are given by:

$$\log p(\mathbf{y} \mid \boldsymbol{\eta} = \phi(\mathbf{u}; \theta)) = \log \operatorname{Cat}(\mathbf{y}; \operatorname{softmax}_k(\boldsymbol{\eta})) = \sum_{i=1}^{k}\sum_{j=1}^{d} y_{i,j} \log \operatorname{softmax}(\boldsymbol{\eta}_{:,j})_i. \tag{39}$$

Remark B.3. This approach is the most similar to standard SSNs as it uses the same Monte Carlo setup to integrate out the logits and compute $p(\mathbf{y} \mid \mathbf{x})$. We simply replace the low-rank Gaussian parameterisation with an autoregressive flow that is still cheap to sample from: an IAF. The approach is attractive in that the majority of the model capacity is allocated to learning the base distribution $p_{U|X}$, which only requires a single forward pass to compute. Given the parameters of $p_{U|X}$, it is cheap to sample from it and compute the logits $\boldsymbol{\eta} = \phi(\mathbf{u}; \theta)$, as the flow ϕ is lightweight, e.g. a single linear autoregressive transformation is sufficient (c.f. Proposition 5.1). In practice, it can be beneficial to make ϕ more expressive.

B.4 Dual-Flow: Evidence Lower Bound

An important fact about IAFs is that, although scoring observations is slow and unparallelizable, they can still score their *own* samples efficiently since intermediate outputs can be cached when sampling $\boldsymbol{\eta} \sim p^{\text{IAF}}$, then reused for scoring the sample. This opens up various design options for Flow-SSNs, which we explain in greater detail next.

B.4.1 Dual-Flow

We introduce a *dual-flow* setup comprised of an IAF $p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)$ and an MAF $p^{\text{MAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \hat{\lambda}, \hat{\theta})$, both defined in logit space and trained concurrently, such that we maximize a lower bound on $\log p(\mathbf{y} \mid \mathbf{x})$:

$$\log p(\mathbf{y} \mid \mathbf{x}) = \log \int p(\mathbf{y} \mid \boldsymbol{\eta}) p^{\text{MAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \hat{\lambda}, \hat{\theta}) \, \mathrm{d}\boldsymbol{\eta} \tag{40}$$

$$= \int p(\mathbf{y} \mid \boldsymbol{\eta}) p^{\text{MAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \hat{\lambda}, \hat{\theta}) \frac{p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)}{p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)} \, \mathrm{d}\boldsymbol{\eta} \tag{41}$$

$$\geq \mathbb{E}_{\boldsymbol{\eta} \sim p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)} \left[\log \frac{p(\mathbf{y} \mid \boldsymbol{\eta}) p^{\text{MAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \hat{\lambda}, \hat{\theta})}{p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)} \right] \tag{42}$$

$$= \mathbb{E}_{\boldsymbol{\eta} \sim p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)} \left[\log p(\mathbf{y} \mid \boldsymbol{\eta}) \right] - D_{\text{KL}} \left(p^{\text{IAF}} \parallel p^{\text{MAF}} \right). \tag{43}$$

where $(\mathbf{x}, \mathbf{y}) \sim p_{\text{data}}(\mathbf{x}, \mathbf{y})$. There is no general closed-form solution to this KL term, so we approximate it using samples:

$$D_{\text{KL}} \left(p^{\text{IAF}} \parallel p^{\text{MAF}} \right) \approx \frac{1}{M} \sum_{i=1}^{M} \log \frac{p^{\text{IAF}}(\boldsymbol{\eta}^{(i)} \mid \mathbf{x}; \lambda, \theta)}{p^{\text{MAF}}(\boldsymbol{\eta}^{(i)} \mid \mathbf{x}; \hat{\lambda}, \hat{\theta})}, \qquad \boldsymbol{\eta}^{(i)} | \mathbf{x} \sim p^{\text{IAF}}, \tag{44}$$

which is cheap since sampling from p^{IAF} is parallelizable and computing the base distribution $p_{U|X}$ only requires a single forward pass of $\mathbf{x}$. Furthermore, scoring p^{IAF}'s samples under p^{MAF} is also cheap since scoring in MAFs is parallelizable.

In practice, we recommend using a different (unbiased) estimator which has lower variance, proposed by Schulman [75]:

$$D_{\text{KL}} \left(p^{\text{IAF}} \parallel p^{\text{MAF}} \right) \approx \frac{1}{M} \sum_{i=1}^{M} \operatorname{expm1}(r) - r, \qquad r := \log \frac{p^{\text{IAF}}(\boldsymbol{\eta}^{(i)} \mid \mathbf{x}; \lambda, \theta)}{p^{\text{MAF}}(\boldsymbol{\eta}^{(i)} \mid \mathbf{x}; \hat{\lambda}, \hat{\theta})}, \qquad \boldsymbol{\eta}^{(i)} | \mathbf{x} \sim p^{\text{IAF}}. \tag{45}$$

Remark B.4. This combination of IAFs and MAFs is reminiscent of *probability density distillation* [60], a student-teacher distillation technique used in audio synthesis models, wherein a pre-trained MAF is fixed as a teacher, and an IAF student learns to match it. In our setup, both flow models are trained concurrently to maximise a bespoke lower bound for stochastic segmentation tasks. Furthermore, it is possible for p^{IAF} and p^{MAF} to share the base distribution parameters λ.

B.4.2 Improper Uniform Prior

Alternatively to the above, we can replace p^{MAF} with an improper uniform prior p such that:

$$p(\boldsymbol{\eta}) = \text{const}, \ \forall \boldsymbol{\eta} \implies D_{\text{KL}} \left(p^{\text{IAF}} \parallel p \right) = \mathbb{E}_{p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)} \left[\log \frac{p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)}{\text{const}} \right] \tag{46}$$

$$= -H(p^{\text{IAF}}) - \log \text{const}, \tag{47}$$

which, after dropping the constant term w.r.t the model parameters, yields the objective:

$$\log p(\mathbf{y} \mid \mathbf{x}) \geq \mathbb{E}_{\boldsymbol{\eta} \sim p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x}; \lambda, \theta)} \left[\log p(\mathbf{y} \mid \boldsymbol{\eta}) \right] + \beta H(p^{\text{IAF}}), \tag{48}$$

with a weighting hyperparameter $\beta > 0$. Maximising $H(p^{\text{IAF}})$ prevents the model from collapsing to a deterministic one. The special case where β is set to 0 makes this objective equivalent to the expected categorical likelihood above.

Proposition B.5 (IAF Entropy Estimator)**.** *The entropy $H(p^{\text{IAF}})$ can be efficiently estimated in parallel via Monte Carlo sampling $\mathbf{u}^{(i)}|\mathbf{x} \sim p_{U|X}$ using the following formula:*

$$H(p^{\text{IAF}}) \approx H(p_{U|X}) - \frac{1}{M}\sum_{i=1}^{M} \log \left|\det \mathbf{J}_{\phi^{-1}}(\phi(\mathbf{u}^{(i)};\theta))\right|, \qquad \mathbf{u}^{(i)}|\mathbf{x} \sim p_{U|X}. \tag{49}$$

Proof. We simply start with the definition of differential entropy, then use a change-of-variables and simplify:

$$\begin{aligned}
H(p^{\text{IAF}}) &= -\mathbb{E}_{\boldsymbol{\eta}\sim p^{\text{IAF}}(\boldsymbol{\eta}|\mathbf{x};\lambda,\theta)}\left[\log p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x};\lambda,\theta)\right] && (50)\\
&= -\int p^{\text{IAF}}(\boldsymbol{\eta} \mid \mathbf{x};\lambda,\theta)\Big[\log p_{U|X}(\phi^{-1}(\boldsymbol{\eta};\theta) \mid \mathbf{x};\lambda) + \log\left|\det \mathbf{J}_{\phi^{-1}}(\boldsymbol{\eta})\right|\Big]\,\mathrm{d}\boldsymbol{\eta} && (51)\\
&= -\int p_{U|X}(\mathbf{u} \mid \mathbf{x};\lambda)\Big[\log p_{U|X}(\mathbf{u} \mid \mathbf{x};\lambda) + \log\left|\det \mathbf{J}_{\phi^{-1}}(\phi(\mathbf{u};\theta))\right|\Big]\,\mathrm{d}\mathbf{u} && (52)\\
&= H(p_{U|X}) - \mathbb{E}_{\mathbf{u}\sim p_{U|X}(\mathbf{u}|\mathbf{x};\lambda)}\left[\log\left|\det \mathbf{J}_{\phi^{-1}}(\phi(\mathbf{u};\theta))\right|\right], && (53)\\
&\approx H(p_{U|X}) - \frac{1}{M}\sum_{i=1}^{M}\log\left|\det \mathbf{J}_{\phi^{-1}}(\phi(\mathbf{u}^{(i)};\theta))\right|, \qquad \mathbf{u}^{(i)}|\mathbf{x} \sim p_{U|X}, && (54)
\end{aligned}$$

which is what we wanted to show. □

Remark B.6. The entropy $H(p_{U|X})$ is available in closed-form for typical base distributions (e.g. Gaussian). Computing $p_{U|X}(\mathbf{u} \mid \mathbf{x};\lambda)$ only requires a single forward pass, which is important as it comprises the majority of the model parameters. The flow component $\phi(\cdot)$ is lightweight, since a single (linear) layer is sufficient (c.f. Proposition 5.1). Lastly, recall that autoregressive flows admit a lower triangular Jacobian by design, and as such, their log absolute determinant simplifies to a sum of the diagonal elements, which is easy to compute.

B.5 Logit Bijections

For bijective logit mappings $\mathbf{y} = g(\boldsymbol{\eta})$ with tractable Jacobians determinants, the likelihood term $p(\mathbf{y} \mid \mathbf{x})$ in Flow-SSN models can be evaluated directly using the following change-of-variables formula:

$$p(\mathbf{y} \mid \mathbf{x}) = p_{U|X}((\phi \circ g)^{-1}(\mathbf{y}) \mid \mathbf{x};\lambda)\left|\det \mathbf{J}_{\phi^{-1}}(g^{-1}(\mathbf{y}))\right|\left|\det \mathbf{J}_{g^{-1}}(\mathbf{y})\right|.$$

However, the above does not hold when $g(\cdot)$ is the softmax function, as it is not bijective. TensorFlow Probability [1] provides the somewhat underused function `tfp.bijectors.SoftmaxCentered`, which is an alternative bijective softmax transformation that we can use here, albeit at the cost of having to dequantise $\mathbf{y}$. We must also train with $k+1$ classes, where a newly introduced dummy class acts as a pivot and facilitates the bijective property of the function. Relatedly, De Sousa Ribeiro et al. [17], Monteiro et al. [56] have also recently used this transformation to train discrete causal mechanisms.

C Evaluation Metrics

Throughout this work, we use the following evaluation metrics standard for stochastic segmentation tasks.

Dice Similarity Coefficient. For two label maps $\mathbf{y}, \hat{\mathbf{y}}$ of dimension $h \times w$, the Dice Similarity Coefficient (DSC) is defined as follows:

$$\mathrm{DSC}(\mathbf{y},\hat{\mathbf{y}}) = \frac{2|\mathbf{y}\cap\hat{\mathbf{y}}|}{|\mathbf{y}| + |\hat{\mathbf{y}}|}, \quad \text{where} \quad |\mathbf{y}\cap\hat{\mathbf{y}}| = \sum_{i=1}^{h}\sum_{j=1}^{w} y_{i,j}\times\hat{y}_{i,j}, \quad \text{and} \quad |\mathbf{y}| = \sum_{i=1}^{h}\sum_{j=1}^{w} y_{i,j}. \tag{55}$$

Intersection over Union. Intersection over Union (IoU) is a similar widely used segmentation metric defined as follows:

$$\mathrm{IoU}(\mathbf{y},\hat{\mathbf{y}}) = \frac{|\mathbf{y}\cap\hat{\mathbf{y}}|}{|\mathbf{y}\cup\hat{\mathbf{y}}|}, \quad \text{where} \quad |\mathbf{y}\cup\hat{\mathbf{y}}| = |\mathbf{y}| + |\hat{\mathbf{y}}| - |\mathbf{y}\cap\hat{\mathbf{y}}|. \tag{56}$$

Compared with DSC, IoU is generally a less forgiving metric since it does not weight the overlap as strongly, thereby penalising mismatches more strongly.

Generalised Energy Distance. We use Generalised Energy Distance (GED) to compare the quality of samples from the model with the ground truth labels. Independent samples $\hat{\mathbf{y}}, \hat{\mathbf{y}}' \overset{\text{iid}}{\sim} p_{\text{model}}$ are drawn from the predictive model distribution p_{model} whereas the multiple ground truth annotations $\mathbf{y}, \mathbf{y}' \sim p_{\text{data}}$ are from the data distribution p_{data}, then:

$$D^2_{\text{GED}}(p_{\text{data}}, p_{\text{model}}) = 2\,\mathbb{E}_{\mathbf{y}\sim p_{\text{data}}, \hat{\mathbf{y}}\sim p_{\text{model}}}[d(\mathbf{y}, \hat{\mathbf{y}})] - \mathbb{E}_{\hat{\mathbf{y}}, \hat{\mathbf{y}}'\sim p_{\text{model}}}[d(\hat{\mathbf{y}}, \hat{\mathbf{y}}')] - \mathbb{E}_{\mathbf{y}, \mathbf{y}'\sim p_{\text{data}}}[d(\mathbf{y}, \mathbf{y}')]. \tag{57}$$

We follow Kohl et al. [43], Monteiro et al. [55] by using the distance function $d(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \text{IoU}(\mathbf{y}, \hat{\mathbf{y}})$, which is a metric, and implies D^2_{GED} is also a metric. Lower GED indicates better alignment between the predictive and ground truth distributions. The sample diversity $\mathbb{E}_{\hat{\mathbf{y}}, \hat{\mathbf{y}}'\sim p_{\text{model}}}[d(\hat{\mathbf{y}}, \hat{\mathbf{y}}')]$ is also a quantity of interest, as it measures the average distance between pairs of samples from the predictive distribution, and provides a measure of variability in our samples. We note that diversity is only contextually relevant, as high diversity alone can be trivially achieved with random noise as a model.

Hungarian-matched IoU. We also use Hungarian-matched IoU (HM-IoU) to compare samples between the ground truth and predictive distributions. HM-IoU uses the Hungarian algorithm to find an optimal assignment between the two sets of samples: $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} \sim p_{\text{data}}$ and $\{\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_M\} \overset{\text{iid}}{\sim} p_{\text{model}}$, then using $1 - \text{IoU}$ as the cost matrix for solving a linear sum assignment problem [44]. HM-IoU can be viewed as more robust across sets of ground truth and predictive samples compared to the GED, which can suffer from inflated scores if the predictive samples are very diverse.

D Implementation Details

D.1 Training Setup & Hyperparameters

In this section, we provide the setup and hyperparameters we used to train our Flow-SSN models on the medical datasets. Our implementation is based in `PyTorch` [64]. As shown in Table 3, we use a UNet [19, 72] to parameterise the base distribution of our Flow-SSN (i.e. Prior Network). To parameterise the flow transformation itself (i.e. Flow Network), we use an autoregressive Transformer for the discrete-time Flow-SSN variant and a UNet for the continuous-time variant. For data augmentation on LIDC-IDRI, we used random 90 degree rotations and vertical/horizontal flips with 0.5 probability. For data augmentation on REFUGE-MultiRater, we used random vertical flips with 0.5 probability; random rotations in the range of [-20, 20] degrees, random resized cropping to 256x256 with scale [0.9, 1.1], and applied color jitter of 0.3 to the images.

The remaining hyperparameters we used are largely equal for both datasets, as reasonable initial values performed well enough; we did not perform extensive hyperparameter tuning for each dataset. For the continuous-time Flow-SSN, 8 ODE solving steps (Euler method in `torchdiffeq` [13]) were used on the validation set throughout training for model selection. In all cases, the best checkpoint was selected based on the lowest GED achieved on the validation set during training. The final model artefact we use for evaluation is an exponential moving average (EMA) of the model parameters.

Table 3. Training hyperparameters used across all experiments.

	LIDC-IDRI		**REFUGE-MultiRater**	
CONFIG	Flow-SSN$_\Delta$	Flow-SSN$_\infty$	Flow-SSN$_\Delta$	Flow-SSN$_\infty$
Prior Network	UNet	UNet	UNet	UNet
Flow Network	Transformer	UNet	Transformer	UNet
Optimiser	AdamW	AdamW	AdamW	AdamW
Batch Size	16	16	16	16
Learning Rate	10^{-4}	10^{-4}	10^{-4}	10^{-4}
LR Warmup	Linear 2K	Linear 2K	Linear 1K	Linear 1K
Weight Decay	10^{-4}	10^{-4}	10^{-4}	10^{-4}
EMA Rate	0.9999	0.9999	0.999	0.999
Max Epochs	1001	1001	1001	1001
Eval Freq.	16	16	50	50
MC Samples (train)	16	1	32	1
MC Samples (eval)	16	16	16	16
Prior Dist.	Gaussian	Gaussian	Gaussian	Gaussian

D.2 UNet Architecture

As mentioned in the main text, we reimplemented a streamlined version of Dhariwal and Nichol [19]'s UNet, which uses fewer attention layers. Recall that LIDC-IDRI images are grayscale, whereas REFUGE images are RGB. To parameterise a Flow-SSN prior, the UNet outputs a mean and variance per output pixel, representing the 'initial guess' distribution as a diagonal Gaussian, to be later refined by the choice of flow. The prior's mean can be optionally initialised with a pre-trained network and then fine-tuned alongside the flow network. In our case, we train everything end-to-end for simplicity and find that fixing the prior variance to, e.g. 1, can also help stabilise training in some instances. Table 4 shows the remaining details, including input and output shapes for both the prior and flow networks.

Table 4. UNet hyperparameters used for parameterising both the base distribution (Prior) and the flows in our Flow-SSN models. We use (Δ) and (∞) to denote relation to the discrete and continuous-time version of the associated Flow-SSN.

	LIDC-IDRI		**REFUGE-MultiRater**	
CONFIG	Prior (Δ, ∞)	Flow (∞)	Prior (Δ, ∞)	Flow (∞)
Input Shape	(1, 128, 128)	(2, 128, 128)	(3, 256, 256)	(2, 256, 256)
Model Channels	32	16	32	32
Output Channels	4	2	4	2
Residual Blocks	1	1	2	1
Dropout	0.1	0.1	0.1	0.1
Channel Multipliers	[1, 2, 4, 8]	[1, 1, 1]	[1, 2, 2, 4, 6]	[1, 1, 1, 1]
Attention Resolution	[16]	[16]	[16]	[16]
Num. Heads	1	1	1	1
Head Channels	64	16	64	32
#Parameters	14.4M	150K	14.6M	787K

D.3 Autoregressive Transformer Architecture

To parameterise the discrete-time autoregressive Flow-SSN, we require a lightweight autoregressive model. The challenge is that our flow lives in pixel-space, so autoregressively predicting each pixel can become computationally expensive for large datasets. To overcome this obstacle, we propose two things. The first is that we use an IAF [40] defined in *in pixel-space* and train under the expected likelihood objective in Eq. (14), which avoids pixel-wise sequential likelihood evaluation and enables fast, one-pass sampling at inference time. Recall that MAFs [61] are equally fast to train as likelihood evaluation can be parallelised, but they still require sequential autoregressive sampling at inference time. The second is that we use an autoregressive Transformer to parameterise the flow, with image strips/patches of, e.g., size (1,8) or (8,8) pixels to reduce memory requirements. To reconstruct the patches back to the output size, we simply use a transposed convolution. Grouping pixels this way induces a block covariance structure in pixel space, but as long as the strips/patches are small relative to the full image size, the maximum attainable covariance rank remains high (i.e. in the hundreds/thousands for real images). Furthermore, since Flow-SSNs model pixel covariance in logit space and transform the learned distribution by a non-linear transfer function (softmax), there is a sublinear increase in rank we can benefit from (c.f. theoretical results in Appendix A).

In summary, and perhaps surprisingly, we find that the above combination of design choices works well provided we use enough MC samples during training, e.g. ≥ 8, otherwise, the model can collapse to a near-deterministic state (c.f. Fig. 11). It is worth noting that we briefly experimented with a shallow PixelCNN [74, 88, 88] to parameterise the flow instead of an autoregressive Transformer, but did not have as much success. As shown in Table 5, the final architecture setup for a discrete-time autoregressive Flow-SSN uses just 1 autoregressive flow transformation parameterised by a single Transformer layer! As proven in Appendix A, a single autoregressive transformation is sufficient to transform the diagonal Gaussian prior into a highly expressive full covariance. We confirm this works well in practice on both toy and real medical imaging datasets.

Table 5. Autoregressive Transformer hyperparameters used for our discrete-time autoregressive Flow-SSNs.

CONFIG	**LIDC-IDRI** Flow (Δ)	**REFUGE-MultiRater** Flow (Δ)
Input Shape	(2, 128, 128)	(2, 256, 256)
Num. Flows	1	1
Flow Type	IAF	IAF
Output Channels	4	4
Embed Dim	64	128
MLP Width	256	512
Patch Size	(1, 8)	(8, 8)
Patchify	Conv	Conv
Unpatchify	ConvTranspose	ConvTranspose
Num. Blocks	1	2
Num. Heads	1	1
Pos Embed Init	$\mathcal{N}(0, 0.02)$	$\mathcal{N}(0, 0.02)$
Dropout	0.1	0.1
Activation	GELU	GELU
#Parameters	345K	0.92M

E Extra Results

E.1 Sampling Efficiency & Additional Baselines

As reported in Table 6, our method is $\approx 10\times$ more efficient than CCDM [92], thanks to most of the model parameters being in the flow's *prior* (i.e. base/source distribution), which only requires a single forward pass to start the sampling chain. Afterwards, only the *flow* network is needed to solve the ODE, and it has only 150K parameters[5]. The advantage of our model translates to any other diffusion-based segmentation model that dedicates all its model capacity to learning the score/velocity field [3, 66, 89, 92]. That said, we expect that using large models for both the prior *and* the flow would improve performance even further; though we argue that the latter may not be necessary if the prior is expressive enough, e.g. a foundation model. Table 7 reports further baseline comparisons against recent SOTA methods; Flow-SSN outperforms all previous methods.

Table 6. Comparing Flow-SSN sampling efficiency against CCDM (see Figure 5 in [92]).

	CCDM [92]			Flow-SSN		
Steps	$D^2_{\text{GED}}(16)\downarrow$	HM-IoU $\uparrow$	FLOPS $\downarrow$	$D^2_{\text{GED}}(16)\downarrow$	HM-IoU $\uparrow$	FLOPS $\downarrow$
1	-	-	-	0.240$\pm$.002	0.879$\pm$.000	12G
50	$\simeq$ 0.34$\pm$N/A	$\simeq$ 0.55$\pm$N/A	365G	0.210$\pm$.002	0.873$\pm$.000	**51G**
250	0.212$\pm$.002	0.623$\pm$.002	1.83T	0.207$\pm$.000	0.873$\pm$.001	**207G**

Table 7. Additional comparisons of Flow-SSN against recent SOTA methods on LIDC-IDRI.

METHOD	Pub. Venue	$D^2_{\text{GED}}(16)\downarrow$	HM-IoU $\uparrow$	#Param
Tyche [67]	CVPR'24	0.400$\pm$.010	-	1.7M
CCDM [92]	ICCV'23	0.212$\pm$.002	0.623$\pm$.002	9M
MoSE [25]	ICLR'23	0.218$\pm$.003	0.624$\pm$.004	42M
CIMD [66]	CVPR'23	0.234$\pm$.005	0.587$\pm$.001	24M
Flow-SSN$_\infty$	ICCV'25	**0.207**$\pm$.000	**0.873**$\pm$.001	14M

[5]For 1-step we use a discrete-time inverse autoregressive Flow-SSN

E.2 Ablation Study: Increasing the Assumed Rank

We first assess the scalability of standard SSNs by ablating an increase in the assumed rank using a replication of Monteiro et al. [55]'s setup. We observe (Figure 10) that SSNs tend to collapse to deterministic models even under mild increases in the assumed rank, which shows that a different approach is needed to capture higher-order interactions between pixels.

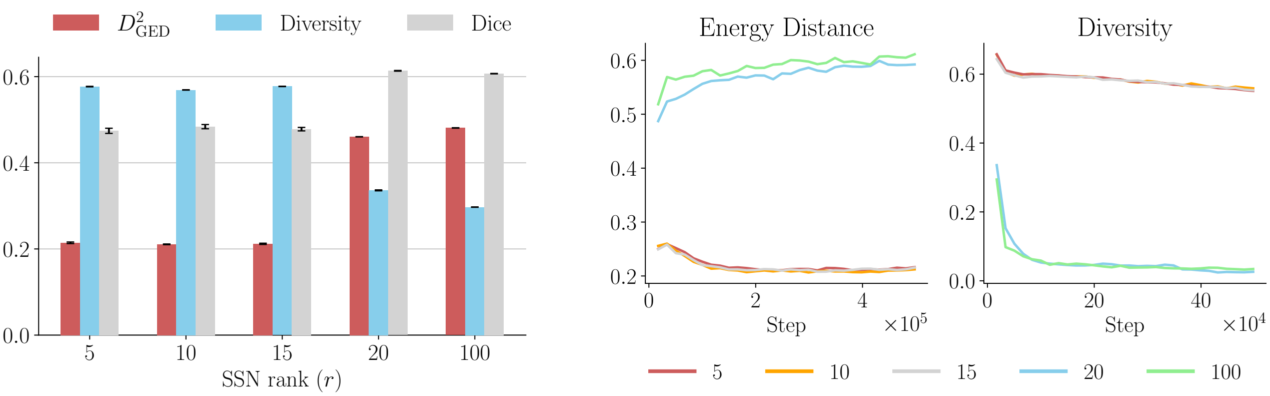


Figure 10. **Scalability ablation study of SSNs on LIDC in terms of the assumed rank**. The results show that mild increases in the assumed rank can cause SSNs to collapse into a near-deterministic state. This warrants a new approach for estimating high-rank covariances.

E.3 Ablation Study: Number of Monte Carlo Samples

As shown in Figure 11, we find that multiple MC samples are needed to properly learn the underlying distribution over outputs when using a discrete-time inverse autoregressive Flow-SSN. With only 1 MC sample, the model enters the near-deterministic regime, and increasing the number of MC samples provides diminishing returns in terms of improved performance. We consider further improving the training stability of IAFs at scale, possibly borrowing tricks from modern MAFs and coupling flows [28, 83, 94], to be fertile ground for future work. The payoff in efficiency is significant as sampling becomes parallelizable.

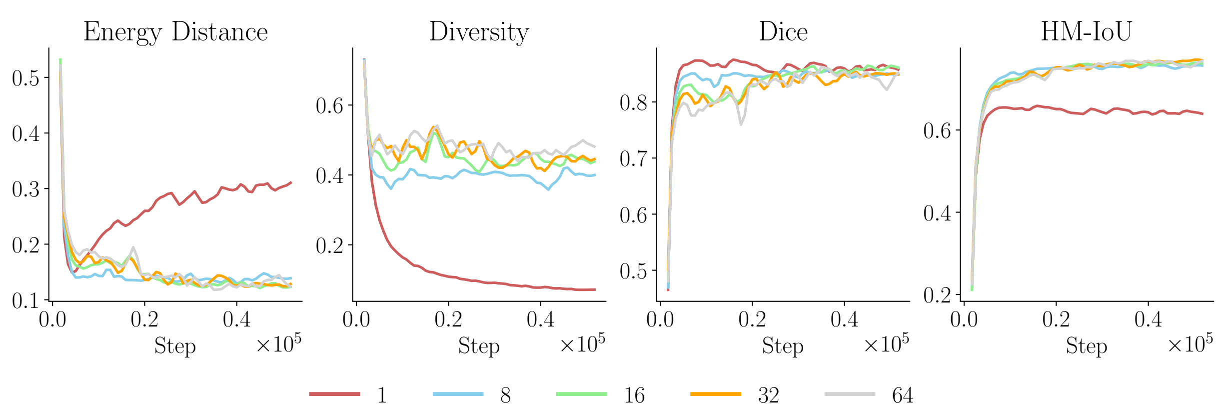


Figure 11. **Ablation analysis of the number of Monte Carlo (MC) samples needed for training**. We look at $\{1, 8, 16, 32, 64\}$ used for training a discrete-time autoregressive Flow-SSN with the objective in Equation (14). The above results were obtained using the REFUGE-MultiRater dataset, and performance on the validation set is shown throughout training.

E.4 Qualitative Results: LIDC-IDRI

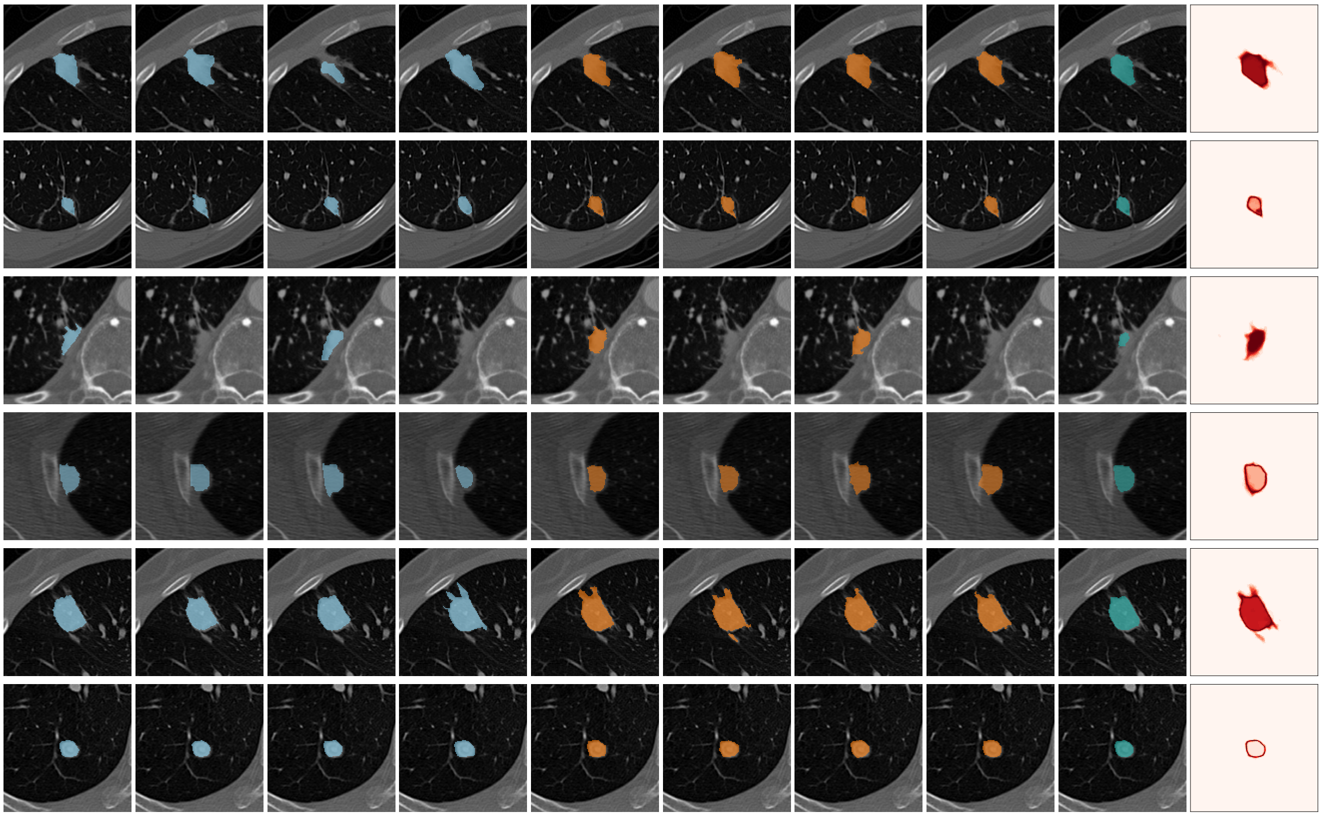


Figure 12. **Extra qualitative results on LIDC-IDRI using our continuous-time Flow-SSN model**. (*Cols. 1-4*) Multiple ground truth segmentations from experts; (*Cols. 5-8*) Non-cherry-picked random samples from our model; (*Cols. 9, 10*) The mean prediction and per-pixel uncertainty map. In all cases, 100 MC samples and 50 ODE solving steps (Euler method) were used for evaluation.

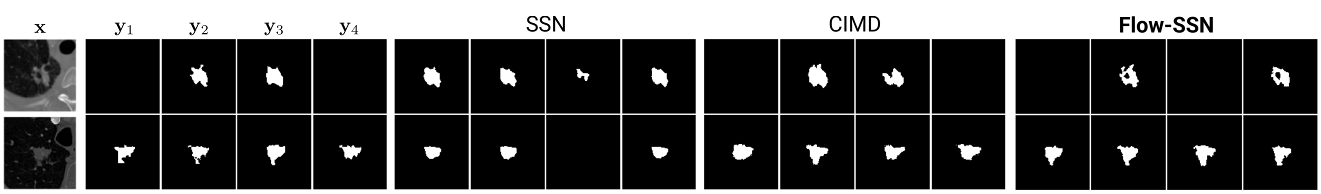


Figure 13. **Qualitative comparison of Flow-SSN against previous methods on LIDC-IDRI**. We used the same images as CIMD [66] for fair comparisons. (*Col. 1*) Input images for segmentation; (*Cols. 2-5*) Ground truth annotations from multiple experts (i.e. four total in this case $\{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4\}$); (*Cols. 6-9*) Random samples from our reproduced SSN [55] model; (*Cols. 10-13*) Random samples taken from CIMD [66], a diffusion-based segmentation model; (*Cols. 14-17*) Random samples from our (continuous-time) Flow-SSN model.

E.5 Qualitative Results: REFUGE-MultiRater

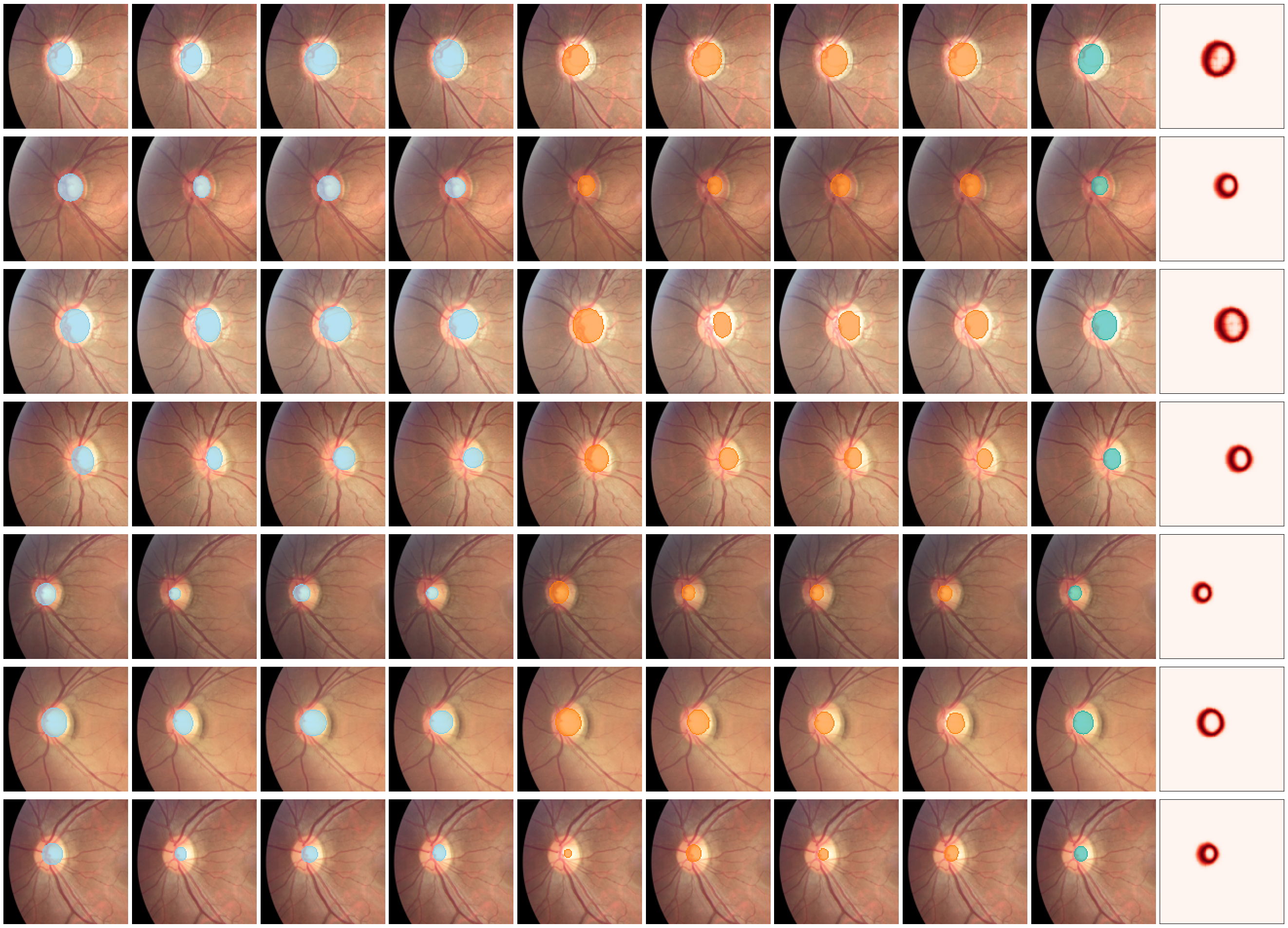


Figure 14. **Extra qualitative results on REFUGE-MultiRater using our discrete-time autoregressive Flow-SSN model**. (*Cols. 1-4*) Multiple ground truth segmentations from experts; in each case, four segmentations were randomly chosen out of the seven available in total; (*Cols. 5-8*) Non-cherry-picked random samples from our discrete-time autoregressive Flow-SSN model; (*Cols. 9, 10*) The mean prediction and per-pixel uncertainty map. In all cases, 512 Monte Carlo samples were used for performing the evaluation.