ACTIVATION-SPACE UNCERTAINTY QUANTIFICATION FOR PRETRAINED NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reliable uncertainty estimates are crucial for deploying pretrained models; yet, many strong methods for quantifying uncertainty require retraining, Monte Carlo sampling, or expensive second-order computations and may alter a frozen backbone's predictions. To address this, we introduce **Gaussian Process Activations (GAPA)**, a post-hoc method that models uncertainty over activation outputs rather than over network weights. GAPA replaces standard nonlinearities with Gaussian-process activations whose posterior mean *exactly* matches the original activation, preserving the backbone's point predictions by construction while providing closed-form activation-space variances that act as support-based uncertainty estimates. To scale to modern architectures, we use a sparse inducing-point approximation over cached reference pre-activations, combined with *local* k-nearest-neighbor subset conditioning, enabling deterministic single-pass uncertainty propagation. Across regression, classification, image segmentation, and language modeling, GAPA is competitive with strong post-hoc baselines and often improves calibration or out-of-distribution detection while remaining efficient at test time.

1 INTRODUCTION

Reliable uncertainty quantification (UQ) is crucial in risk-sensitive deployments, yet many effective research methods remain impractical in modern settings (Abdar et al., 2021). Weight-space Bayesian approaches (e.g., variational BNNs) often require retraining, labeled data or multi-sample evaluation, ensembles multiply compute, and Laplace-style methods rely on curvature estimates that scale poorly as models and output spaces grow (Blundell et al., 2015; MacKay, 1992; Bergna et al., 2024; Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017; Ritter et al., 2018; Ortega et al., 2023). The gap is most pronounced for pretrained backbones, where weights are not expected to be modified and test-time budgets favor *single-pass* inference. In this regime, a practical post-hoc method should be single-pass, prediction-preserving, provide an epistemic uncertainty signal that reflects lack of support under a reference distribution, and scale to foundation models. The resulting gap is summarized in the comparison below.

Method	Post-hoc	Single-pass	Preserves mean	Epistemic signal	LLM compatible
BNNs	✗	✗	✗	✓	✗
Ensembles	✗	✗	—	✓	✗
MC Dropout	✗	✗	✗	✓	✗
LL-Laplace	✓	✓	✗	✓	✗
Temperature Scaling	✓	✓	✗	✗	✓
GAPA (Ours)	✓	✓	✓	✓	✓

We address this by shifting uncertainty modeling from weights to *activations*. We introduce **Gaussian Process Activations (GAPA)**, a drop-in module that replaces deterministic activations with Gaussian-process activations. Because the GP posterior mean is set to the original activation function, the mean forward pass is identical to that of the frozen backbone. GAPA adds a variance to each activation: this variance is small when a test pre-activation is close to cached pre-activations obtained by running a reference set, typically the training data, through the frozen backbone, and larger when it lies far from this cached activation set. We interpret this as a support-based epistemic signal in activation space (Figure 1). For scalability, GAPA conditions on cached reference pre-activations using a sparse approximation (compression + local kNN conditioning), and propagates the resulting uncertainty

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

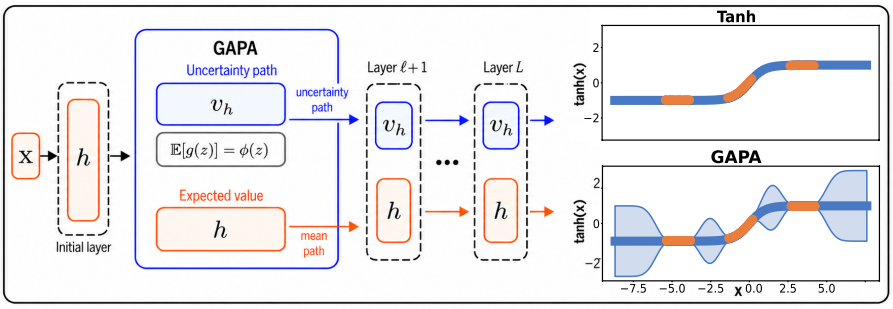


Figure 1: **GAPA overview.** GAPA replaces selected deterministic activations in a frozen network with Gaussian-process activations. The GP posterior mean matches the original activation, preserving the mean path, while a parallel activation-space uncertainty path is propagated forward. Right: cached reference pre-activations anchor the GP; the GAPA-tanh mean matches tanh, and uncertainty increases away from training support.

through the network via deterministic variance-propagation rules, enabling *single-pass* predictive uncertainty. Our contributions are

1. **Mean-preserving post-hoc UQ:** GAPA provides activation-space epistemic uncertainty estimates for pretrained networks while preserving point predictions.
2. **Scalable conditioning:** we combine inducing points and local k-NN conditioning for practical inference at modern scales.
3. **Deterministic propagation:** we derive single-pass variance propagation from activation space to output space.
4. **Empirical validation:** across regression, classification, segmentation, and language modelling, GAPA often improves OOD detection and achieves favorable calibration–efficiency trade-offs.

2 Method

At a high level, GAPA augments a frozen neural network with activation-space uncertainty while strictly preserving its original deterministic predictions. Figure 1 provides a structural overview. In the following sections, we formalize our uncertainty perspective (Sec. 2.1), define the GP activation layer (Sec. 2.2), introduce a scalable inference mechanism (Sec. 2.3), and derive rules for single-pass variance propagation through deep architectures (Sec. 2.4).

Method pipeline. GAPA operates in two phases. **(i) Offline collection:** we run a single forward pass of a reference dataset through the pre-trained backbone and cache pre-activations at selected layers. Optionally, we compress the cache into a smaller inducing set via k-means, yielding inducing inputs for a sparse GP-style approximation. **(ii) Test-time inference:** we replace deterministic activations with Gaussian-process (GP) activations that return epistemic variances in activation space. These variances are then propagated forward through the remaining frozen network using closed-form variance propagation rules, enabling deterministic single-pass predictive uncertainty while preserving the backbone's point predictions.

2.1 Uncertainty Modeling Perspective

We position GAPA by stating *what is random* in the predictive distribution. Let $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$ denote the training data, and let x^* be a test input with corresponding unknown output y^*. We denote by $f(x^*)$ the latent predictor evaluated at x^*. Predictive uncertainty can be written as marginalization over this latent predictor:

$$p(y^* \mid x^*, \mathcal{D}) = \int p\big(y^* \mid f(x^*)\big)\, p\big(f(x^*) \mid x^*, \mathcal{D}\big)\, df(x^*). \tag{1}$$

Weight-space uncertainty. Weight-space methods, such as BNNs and Laplace approximations, parameterize $f(x^*) = f(x^*; \mathbf{w})$ and infer a posterior over weights, $\mathbf{w} \sim p(\mathbf{w} \mid \mathcal{D})$. Epistemic uncertainty is then induced by variability of the predictor under draws from the weight posterior.

Scope of the epistemic interpretation. Unlike BNNs or Laplace methods, GAPA does not infer a posterior over weights or task-trained functions. Its uncertainty comes from an auxiliary GP posterior over activation outputs conditioned on cached reference pre-activations, and is therefore interpreted as a *support-based epistemic uncertainty signal in activation space*: variance increases when a test representation lies in a region weakly supported by the reference cache. Throughout the paper, "epistemic" refers to this operational, reducible lack-of-support notion rather than classical Bayesian posterior uncertainty over weights.

Activation-space uncertainty (GAPA). GAPA keeps the frozen weights deterministic and instead introduces an auxiliary GP posterior over hidden-layer *activation outputs*. The core idea is simple: when a test input produces pre-activations in regions that are weakly supported by the cached reference pre-activations, the GP activation model expresses this lack of support as increased posterior variance. Rather than applying a fixed nonlinearity, GAPA replaces it with a Gaussian-process activation whose posterior mean reproduces the original activation exactly—preserving the backbone's point predictions—while its posterior variance reflects local support in cached pre-activation space.

Formally, for a chosen layer ℓ, decompose the frozen network as $f(x^*) = h_\ell\big(\boldsymbol{\phi}(\mathbf{z}_\ell(x^*))\big)$, where $\mathbf{z}_\ell(x^*)$ is the layer-ℓ pre-activation vector, $\boldsymbol{\phi}$ is the element-wise nonlinearity, and $h_\ell(\cdot)$ is the remaining frozen computation. We replace $\boldsymbol{\phi}$ with a GP activation $\mathbf{g}_\ell$ satisfying $\boldsymbol{\mu}_\ell(\mathbf{z}) = \boldsymbol{\phi}(\mathbf{z})$ (Sec. 2.2), so the mean path is unchanged. Let $\mathbf{a}_\ell^* := \mathbf{g}_\ell(\mathbf{z}_\ell(x^*))$ denote the resulting random activation, and let $\mathcal{D}_\ell$ be the layer-ℓ activation cache obtained by running the training inputs $\{x_n\}_{n=1}^N$ through the frozen backbone, i.e., stored pre-activations paired with their deterministic activation values. GAPA marginalizes over this latent activation:

$$p_{\mathrm{GAPA}}(y^* \mid x^*, \mathcal{D}) = \int p\big(y^* \mid h_\ell(\mathbf{a}_\ell^*)\big)\, p\big(\mathbf{a}_\ell^* \mid \mathbf{z}_\ell(x^*), \mathcal{D}_\ell\big)\, d\mathbf{a}_\ell^*. \tag{2}$$

The GP posterior $p(\mathbf{a}_\ell^* \mid \mathbf{z}_\ell(x^*), \mathcal{D}_\ell)$ is the source of uncertainty in GAPA: it is tight when $\mathbf{z}_\ell(x^*)$ falls near cached training pre-activations and widens in unsupported regions of pre-activation space. We use this posterior variance as an *activation-space epistemic proxy*, because it reflects limited coverage of the cached training/reference pre-activations rather than irreducible observation noise. In this sense, GAPA quantifies epistemic uncertainty operationally as lack of activation-space support, rather than as posterior uncertainty over weights or task-trained functions. Any aleatoric component is handled separately: in regression, through a learned output-noise term in the likelihood; in classification, we do not introduce a learned aleatoric noise head. A formal epistemic–aleatoric decomposition is given in Appendix E.1.

2.2 Gaussian Process Activation Function

We now define the core GAPA module. The goal is to augment a frozen pretrained network with support-based epistemic uncertainty without changing its deterministic predictions. GAPA achieves this by replacing a deterministic activation function with a Gaussian-process activation whose posterior mean matches the original activation and whose posterior covariance provides distance-aware uncertainty in activation space.

Setup. Consider a frozen neural network and a hidden layer ℓ of width d_ℓ. Let $\mathbf{h}_{\ell-1} \in \mathbb{R}^{d_{\ell-1}}$ denote the input to the layer, $\mathbf{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ its frozen weight matrix, and $\mathbf{b}_\ell \in \mathbb{R}^{d_\ell}$ its frozen bias vector. The deterministic layer computes

$$\mathbf{z}_\ell = \mathbf{W}_\ell \mathbf{h}_{\ell-1} + \mathbf{b}_\ell, \qquad \mathbf{h}_\ell = \boldsymbol{\phi}(\mathbf{z}_\ell),$$

where $\mathbf{z}_\ell \in \mathbb{R}^{d_\ell}$ is the pre-activation vector and $\boldsymbol{\phi}(\mathbf{z}) = (\phi(z_1), \ldots, \phi(z_{d_\ell}))^\top$ is an element-wise nonlinearity, such as ReLU or $\tanh$.

Constructing the GP training set. To obtain uncertainty in activation space, we first define where the activation function has been "observed" under the training distribution. We do this without

additional labels: we run the frozen backbone once on a reference training set and store the resulting layer-ℓ pre-activations

$$\tilde{\mathbf{Z}}_\ell = \{\tilde{\mathbf{z}}_\ell^{(n)}\}_{n=1}^{N_\ell}, \qquad \tilde{\mathbf{z}}_\ell^{(n)} \in \mathbb{R}^{d_\ell}.$$

For each stored pre-activation, the corresponding GP target is simply the original deterministic activation value,

$$\tilde{\mathbf{y}}_\ell^{(n)} = \boldsymbol{\phi}(\tilde{\mathbf{z}}_\ell^{(n)}), \qquad n = 1, \dots, N_\ell.$$

Thus, at layer ℓ, GAPA forms the GP training set $\mathcal{D}_\ell = \{(\tilde{\mathbf{z}}_\ell^{(n)}, \tilde{\mathbf{y}}_\ell^{(n)})\}_{n=1}^{N_\ell}$.

These targets are not task labels; they are evaluations of the frozen activation function itself. The purpose of $\mathcal{D}_\ell$ is to define the region of activation space supported by the training distribution: the GP activation will be confident near these stored pre-activations and uncertain away from them.

GP activation. The key idea is to quantify uncertainty in the activation space. Instead of placing a distribution over weights, GAPA places a GP posterior over activation outputs, while keeping its posterior mean equal to the original deterministic activation. Concretely, GAPA replaces the deterministic activation map $\boldsymbol{\phi}(\cdot)$ with a vector-valued Gaussian process

$$\mathbf{g}_\ell(\cdot) : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_\ell}, \qquad \mathbf{g}_\ell(\cdot) \sim \mathcal{GP}(\boldsymbol{m}_\ell(\cdot), \mathbf{K}_\ell(\cdot,\cdot)).$$

Once conditioned on the layer-wise GP training set introduced next, a test pre-activation $\mathbf{z}$ induces a distribution over the activation output,

$$\mathbf{a}_\ell = \mathbf{g}_\ell(\mathbf{z}), \qquad \mathbf{a}_\ell \mid \mathbf{z}, \mathcal{D}_\ell \sim \mathcal{N}(\boldsymbol{\mu}_\ell(\mathbf{z}), \boldsymbol{\Sigma}_\ell(\mathbf{z})).$$

For scalability, we use a diagonal output kernel,

$$\mathbf{K}_\ell(\mathbf{z}, \mathbf{z}') = \mathrm{diag}(k_{\ell,1}(\mathbf{z}, \mathbf{z}'), \dots, k_{\ell,d_\ell}(\mathbf{z}, \mathbf{z}')),$$

which is equivalent to modelling each activation dimension with an independent scalar GP, while using the full pre-activation vector $\mathbf{z}$ as the GP input. Section 2.4 shows how these activation-level variances are propagated deterministically through the frozen network.

Mean preservation. To preserve the deterministic mean path, we set the GP prior mean to the original activation, $\boldsymbol{m}_\ell(\mathbf{z}) = \boldsymbol{\phi}(\mathbf{z})$. The GP targets are obtained by evaluating the same activation on stored training pre-activations, so $\tilde{\mathbf{Y}}_\ell = \boldsymbol{\phi}(\tilde{\mathbf{Z}}_\ell) = \boldsymbol{m}_\ell(\tilde{\mathbf{Z}}_\ell)$.

Under standard GP regression, the posterior mean at a test pre-activation $\mathbf{z}_\ell^*$ is

$$\boldsymbol{\mu}_\ell(\mathbf{z}_\ell^*) = \boldsymbol{m}_\ell(\mathbf{z}_\ell^*) + \mathbf{K}_\ell(\mathbf{z}_\ell^*, \tilde{\mathbf{Z}}_\ell)\big(\mathbf{K}_\ell(\tilde{\mathbf{Z}}_\ell, \tilde{\mathbf{Z}}_\ell) + \sigma_n^2\mathbf{I}\big)^{-1} \underbrace{\big(\tilde{\mathbf{Y}}_\ell - \boldsymbol{m}_\ell(\tilde{\mathbf{Z}}_\ell)\big)}_{=\mathbf{0}}.$$

Hence,

$$\boldsymbol{\mu}_\ell(\mathbf{z}_\ell^*) = \boldsymbol{m}_\ell(\mathbf{z}_\ell^*) = \boldsymbol{\phi}(\mathbf{z}_\ell^*) \qquad \text{for all } \mathbf{z}_\ell^*.$$

Therefore, GAPA preserves the original activation along the posterior mean path. When propagated through the frozen network, this preserves the pretrained model's deterministic predictions exactly, while the GP posterior covariance provides the additional support-based epistemic uncertainty in activation space.

Posterior covariance. The previous paragraph shows that GAPA preserves the activation mean exactly. The additional uncertainty comes from the GP posterior covariance, which assigns a variance to each activation dimension, corresponding to each neuron in a standard hidden layer. As in standard GP regression, this variance is small near stored training pre-activations and increases as the test pre-activation moves away from them. For activation dimension i, the posterior variance at $\mathbf{z}_\ell^*$ is

$$[\boldsymbol{\Sigma}_\ell(\mathbf{z}_\ell^*)]_{ii} = k_{\ell,i}(\mathbf{z}_\ell^*, \mathbf{z}_\ell^*) - \mathbf{k}_{\ell,i}^{*\top}\big(\mathbf{K}_{\ell,i} + \sigma_n^2\mathbf{I}\big)^{-1}\mathbf{k}_{\ell,i}^*,$$

where

$$[\mathbf{K}_{\ell,i}]_{mn} = k_{\ell,i}(\tilde{\mathbf{z}}_\ell^{(m)}, \tilde{\mathbf{z}}_\ell^{(n)}), \qquad [\mathbf{k}_{\ell,i}^*]_m = k_{\ell,i}(\mathbf{z}_\ell^*, \tilde{\mathbf{z}}_\ell^{(m)}),$$

for $m, n = 1, \dots, N_\ell$. Collecting these activation-wise variances gives the diagonal layer covariance

$$\boldsymbol{\Sigma}_\ell(\mathbf{z}_\ell^*) = \mathrm{diag}([\boldsymbol{\Sigma}_\ell(\mathbf{z}_\ell^*)]_{11}, \dots, [\boldsymbol{\Sigma}_\ell(\mathbf{z}_\ell^*)]_{d_\ell d_\ell}) \in \mathbb{R}^{d_\ell \times d_\ell}. \tag{3}$$

Thus, GAPA leaves the activation mean equal to $\boldsymbol{\phi}(\mathbf{z}_\ell^*)$ while augmenting it with a distance-aware GP posterior covariance.

Why diagonal covariance? We model activation dimensions as conditionally independent, yielding a diagonal output covariance. A full multi-output covariance would require storing and propagating dense $d_\ell \times d_\ell$ matrices, which is prohibitive for modern wide networks. The diagonal approximation keeps GAPA tractable while still capturing distance-aware, support-based activation-level epistemic uncertainty, as supported by the empirical results in Sec. 3.

2.3 Local Inducing-Point Approximation

The exact GP activation in Eq. (3) is not scalable when the activation cache is large. At layer ℓ, GAPA stores N_ℓ training pre-activations $\tilde{\mathbf{Z}}_\ell \in \mathbb{R}^{N_\ell \times d_\ell}$. Exact conditioning would require solving an $N_\ell \times N_\ell$ linear system for each activation dimension, with time $\mathcal{O}(d_\ell N_\ell^3)$ and memory $\mathcal{O}(d_\ell N_\ell^2)$. This is prohibitive for modern networks. We therefore use a two-stage approximation. First, offline, we compress the stored pre-activations into a smaller inducing set. Second, at test time, we condition only on the K nearest inducing points to the query pre-activation.

Stage 1: Inducing-point construction (offline). We construct inducing inputs $\mathbf{Z}_\ell \in \mathbb{R}^{M_\ell \times d_\ell}$, with $M_\ell \ll N_\ell$, by running k-means on the cached pre-activations $\tilde{\mathbf{Z}}_\ell$ and taking the M_ℓ centroids. These inducing inputs provide a compressed representation of the training-data activation cache for scalable GP conditioning. Appendix D formalizes the resulting sparse GP-style conditional approximation, and Appendix O.4 studies sensitivity to the number of inducing points.

Stage 2: Local K-nearest-neighbour conditioning (test time). At test time, using all M_ℓ inducing points would require an $M_\ell \times M_\ell$ solve per query, which is still expensive. We therefore adopt a local GP approximation (Gramacy & Apley, 2015): for each query pre-activation $\mathbf{z}_\ell^* \in \mathbb{R}^{d_\ell}$, we form a small local subset of inducing inputs by K-nearest neighbours in activation space.

Concretely, let $\mathcal{N}_K(\mathbf{z}_\ell^*) \subseteq \{1, \dots, M_\ell\}$ denote the indices of the K nearest inducing points to $\mathbf{z}_\ell^*$ under Euclidean distance, and define

$$\mathbf{Z}_{\ell,K}(\mathbf{z}_\ell^*) \;=\; \mathbf{Z}_\ell^{\mathcal{N}_K(\mathbf{z}_\ell^*)} \;\in\; \mathbb{R}^{K \times d_\ell},$$

where $\mathbf{Z}_\ell^{\mathcal{N}_K(\cdot)}$ denotes the submatrix formed by selecting the corresponding rows of $\mathbf{Z}_\ell$. Neighbour retrieval is performed using FAISS (Douze et al., 2025) with approximate nearest-neighbour search, yielding sublinear query time in practice. Appendix O.7 further studies sensitivity to K.

Approximate posterior covariance. Given the local inducing inputs $\mathbf{Z}_{\ell,K}(\mathbf{z}_\ell^*) \in \mathbb{R}^{K \times d_\ell}$, we approximate the posterior variance of neuron $i \in \{1, \dots, d_\ell\}$ by applying the standard GP conditional-variance formula restricted to this local subset:

$$\sigma_{\ell,i}^2(\mathbf{z}_\ell^*) \approx k_{\ell,i}(\mathbf{z}_\ell^*, \mathbf{z}_\ell^*) - \mathbf{k}_{\ell,i}^{*\top} \big(\mathbf{K}_{\ell,i} + \sigma_n^2 \mathbf{I}\big)^{-1} \mathbf{k}_{\ell,i}^*. \tag{4}$$

where σ_n^2 is a small jitter term for numerical stability. Let $\{\mathbf{z}_{\ell,K}^{(m)}\}_{m=1}^K$ denote the rows of $\mathbf{Z}_{\ell,K}(\mathbf{z}_\ell^*)$. Then the $K \times K$ kernel matrix and K-vector are defined as

$$[\mathbf{K}_{\ell,i}]_{mn} = k_{\ell,i}(\mathbf{z}_{\ell,K}^{(m)}, \mathbf{z}_{\ell,K}^{(n)}), [\mathbf{k}_{\ell,i}(\mathbf{z}_\ell^*)]_m = k_{\ell,i}(\mathbf{z}_\ell^*, \mathbf{z}_{\ell,K}^{(m)}).$$

Collecting neuron-wise variances yields the diagonal layer covariance

$$\mathbf{\Sigma}_\ell(\mathbf{z}_\ell^*) = \mathrm{diag}\big(\sigma_{\ell,1}^2(\mathbf{z}_\ell^*), \dots, \sigma_{\ell,d_\ell}^2(\mathbf{z}_\ell^*)\big) \in \mathbb{R}^{d_\ell \times d_\ell}.$$

Conservative uncertainty (monotonicity). Intuitively, conditioning a GP on fewer points cannot reduce posterior uncertainty. Fix kernel hyperparameters and observation noise $\sigma_n^2 > 0$, and let $A \subseteq C$ be two conditioning sets. Then for any test input $\mathbf{z}^*$, $\mathrm{Var}\big(f(\mathbf{z}^*) \mid \mathcal{D}_A\big) \;\geq\; \mathrm{Var}\big(f(\mathbf{z}^*) \mid \mathcal{D}_C\big)$. Consequently, conditioning on the local subset $\mathbf{Z}_{\ell,K}(\mathbf{z}_\ell^*) \subseteq \mathbf{Z}_\ell$ cannot underestimate the GP posterior variance relative to using the full inducing set. A formal statement and proof are provided in Appendix C; we additionally ablate sensitivity to K in Appendix O.7.

Computational complexity. Offline preprocessing comprises caching $\tilde{\mathbf{Z}}_\ell$ (one forward pass), constructing $\mathbf{Z}_\ell$ via k-means, and building a FAISS index over $\mathbf{Z}_\ell$, where typically $K \ll M_\ell \ll N_\ell$. At test time, each query requires (i) neighbour search in $\mathcal{O}(\log M_\ell)$ and (ii) local $K \times K$ solves for

activation coordinates. A direct neuron-wise implementation costs $\mathcal{O}(d_\ell K^3)$, although the small local systems are highly vectorizable. With fixed K, the local linear algebra remains constant-size per layer apart from the width factor, and the dominant dependence on the inducing set size is $\mathcal{O}(\log M_\ell)$; memory is linear in $M_\ell d_\ell$.

2.4 Single-Pass Variance Propagation

After a GAPA module produces activation-space uncertainty, we propagate this uncertainty through the remaining frozen network in a single deterministic forward pass. At each layer, we maintain a Gaussian summary of the hidden state: a mean $\boldsymbol{\mu}_{\mathbf{h}}$ and a diagonal variance vector $\mathbf{v}_{\mathbf{h}} := \mathrm{diag}(\boldsymbol{\Sigma}_{\mathbf{h}})$, where $\boldsymbol{\Sigma}_{\mathbf{h}}$ is the hidden-state covariance. The mean follows the original deterministic computation, thereby preserving the pretrained model's point prediction, while the variance is propagated using closed-form moment rules.

For a linear layer $\mathbf{z} = \mathbf{W}\mathbf{h} + \mathbf{b}$ and an element-wise nonlinearity g, respectively, we use

$$\mathbf{v}_{\mathbf{z}} = (\mathbf{W} \odot \mathbf{W})\mathbf{v}_{\mathbf{h}}, \qquad \mathbf{v}_{g(\mathbf{z})} \approx \big(g'(\boldsymbol{\mu}_{\mathbf{z}})\big)^{\odot 2} \odot \mathbf{v}_{\mathbf{z}},$$

where $\odot$ denotes element-wise products and $\mathbf{a}^{\odot 2} := \mathbf{a} \odot \mathbf{a}$. The first rule propagates diagonal uncertainty through affine transformations, while the second is a first-order delta-method approximation for nonlinearities.

A key feature of GAPA is that multiple GP activation modules can be stacked. In this case, the input to a downstream GAPA layer is no longer a point query to the GP, but an uncertain pre-activation summarized by a Gaussian $(\boldsymbol{\mu}_{\mathbf{z}}, \mathbf{v}_{\mathbf{z}})$. We therefore evaluate the local GP posterior variance at the mean input $\boldsymbol{\mu}_{\mathbf{z}}$, and add a noisy-input correction that propagates uncertainty from earlier layers. For output coordinate i of a GAPA layer,

$$v_{\ell,i}^{\mathrm{out}} \approx \underbrace{\sigma^2_{\mathrm{epi},\ell,i}(\boldsymbol{\mu}_{\mathbf{z}})}_{\text{support-based epistemic}} + \underbrace{\nabla_{\mathbf{z}}\mu_{\ell,i}(\boldsymbol{\mu}_{\mathbf{z}})^\top \,\mathrm{diag}(\mathbf{v}_{\mathbf{z}})\nabla_{\mathbf{z}}\mu_{\ell,i}(\boldsymbol{\mu}_{\mathbf{z}})}_{\text{propagated input uncertainty}}.$$

Because GAPA is mean-preserving, $\mu_{\ell,i}(\mathbf{z}) = \phi_\ell(z_i)$ for element-wise activations, so this correction reduces to

$$v_{\ell,i}^{\mathrm{out}} \approx \sigma^2_{\mathrm{epi},\ell,i}(\boldsymbol{\mu}_{\mathbf{z}}) + \big(\phi'_\ell(\mu_{\mathbf{z},i})\big)^2 v_{\mathbf{z},i}.$$

Thus, each GAPA layer both propagates uncertainty already present in the network and adds a new support-based epistemic variance term.

This gives deterministic single-pass predictive uncertainty without test-time backpropagation, network-level Monte Carlo sampling, or second-order computations. Full derivations of the stacked-GAPA rule, together with propagation rules for RMSNorm, self-attention, and other transformer components, are provided in Appendix E and Appendix M. Ablations on layer placement and propagation choices are reported in Appendix O.

2.5 Hyperparameter Strategy

A key design choice in GAPA is to *fix* GP hyperparameters rather than optimize them. GAPA is designed as a post-hoc module for frozen pretrained networks. After the backbone has been trained, we attach GAPA by running a reference training set through the frozen backbone once and caching the selected layer pre-activations. The core GAPA module requires no backpropagation, fine-tuning, or task labels: its GP hyperparameters are set once from simple empirical statistics of the cached pre-activations. Because these hyperparameters are estimated from activation statistics rather than from a task-specific objective, the same procedure applies across regression, classification, segmentation, and token prediction. The caching step is a one-off offline preprocessing step; after it is complete, test-time inference uses the fixed GAPA module without updating the backbone or GP hyperparameters.

We use an RBF kernel $k_{\ell,i}(\mathbf{z}, \mathbf{z}') = c_i^2 \exp(-\|\mathbf{z} - \mathbf{z}'\|^2/(2\ell_i^2))$, with the signal scale c_i^2, length scale ℓ_i, and numerical jitter set from cached activation statistics. Details of the empirical construction are provided in Appendix I.1. For classification, we keep the backbone and GAPA module fixed and convert propagated logit means and variances into predictive probabilities using the Laplace-bridge approximation in Appendix K. For regression, we optionally train a separate aleatoric likelihood head while keeping the backbone and GAPA hyperparameters fixed; details are given in Appendix I.2.

3 RESULTS

We evaluate GAPA as a *post-hoc* uncertainty module for *frozen* pretrained backbones across regression (Sec. 3.1), classification (Sec. 3.2), image segmentation (Appendix Sec. G), and language modeling (Sec. 3.3). Ablations on layer placement, inducing-set size M, local subset size K, and uncertainty-score choices are provided in Appendix O.

3.1 REGRESSION

Setup and baselines. We evaluate on three regression benchmarks using the original train/test splits: YEAR Prediction MSD, Airline (Hensman et al., 2013), and Taxi (Salimbeni & Deisenroth, 2017). We compare against: **MAP** (backbone model), **Dropout** (MC Dropout with multiple stochastic forward passes; (Gal & Ghahramani, 2016)), **Ensemble** (independently trained models; (Lakshminarayanan et al., 2017)), and post-hoc last-layer Bayesian baselines (**LLA** variants, **ELLA**, **VaLLA**; (Ortega et al., 2023)).

Metrics. Performance is evaluated using Negative Log-Likelihood (NLL), Continuous Ranked Probability Score (CRPS), and the Centered Quantile Metric (CQM), with definitions in Appendix L.1.

Table 1: Regression results. Stds $< 10^{-4}$ in all cases; omitted.

Model	Airline			Year			Taxi		
	NLL	**CRPS**	**CQM**	**NLL**	**CRPS**	**CQM**	**NLL**	**CRPS**	**CQM**
MAP	5.121	18.695	0.148	3.673	5.023	0.134	3.775	3.755	0.211
LLA Diag	5.125	18.648	0.143	3.647	4.917	0.088	3.722	3.990	0.257
LLA KFAC	5.127	18.631	0.142	3.648	4.915	0.086	3.706	3.986	0.256
LLA*	5.127	18.631	0.141	3.648	4.915	0.086	3.726	3.985	0.256
LLA*KFAC	5.127	18.631	0.141	3.648	4.914	0.086	3.726	3.985	0.256
ELLA	5.388	21.671	0.413	4.020	6.049	0.424	3.885	3.680	0.219
VaLLA100	4.963	18.814	0.099	3.515	5.004	0.047	3.235	3.999	0.149
VaLLA200	4.965	18.788	0.098	3.485	4.970	0.041	3.232	3.979	0.142
Dropout	5.102	19.066	0.938	3.689	5.128	0.939	3.849	4.592	0.951
Ensemble	5.053	18.205	0.933	3.639	4.833	0.938	3.631	3.384	0.961
GAPA	4.946	18.068	0.103	3.470	4.663	0.014	3.112	4.035	0.104

This table shows a consistent pattern across all three regression benchmarks: GAPA achieves the best NLL, suggesting the strongest overall predictive distribution. It also obtains the best CRPS on Airline and Year, and the best CQM on Year and Taxi. On Taxi, Ensemble gives the best CRPS, while GAPA remains strongest in NLL and CQM.

3.2 CLASSIFICATION

We now evaluate classification performance in terms of accuracy and calibration, and assess OOD detection via AUC using predictive entropy and BALD. Metric definitions are provided in Appendix L.2.

Baselines. We compare against (i) the deterministic backbone (**MAP**); (ii) sampling-based baselines requiring multiple forward passes or multiple trainings (**MC Dropout** (Gal & Ghahramani, 2016), **Deep Ensembles** (Lakshminarayanan et al., 2017)); (iii) post-hoc last-layer Bayesian baselines that Bayesianize only the head (**LLA/ELLA/VaLLA**; (Ortega et al., 2023)); and (iv) distance-/feature-based alternatives including **SNGP** (Liu et al., 2020), **MF-VI** (Blundell et al., 2015), and a **subset GP** baseline based on the GP interpretation of local linearization. We also include **linear probing** (Alain & Bengio, 2016) and **DDU** (Mukhoti et al., 2023); comparisons with distance and density baselines appear in Appendix A.

MNIST / FMNIST (MLP). We train a 4-layer $\tanh$ MLP with 200 hidden units on MNIST (LeCun et al., 2002) and Fashion-MNIST (Xiao et al., 2017) and, following Ortega et al. (2023), evaluate OOD detection by swapping datasets (Table 2). Since GAPA is mean-preserving, it matches MAP accuracy by construction. GAPA achieves the best BALD AUROC in both directions, the best predictive-entropy AUROC on FMNIST→MNIST, remains competitive on MNIST→FMNIST, and runs near MAP speed (2.05s).

Robustness under distribution shift. We evaluate robustness by rotating the test images by increasing angles and plotting predictive NLL for MNIST and FMNIST in Figure 8. As the rotation angle increases, the inputs move farther from the training distribution, and predictive NLL generally

Table 2: Classification and runtime on MNIST/FMNIST. We report ACC, NLL, ECE, OOD AUROC, BALD, and wall-clock train/test time; "K" denotes 10^3 seconds. Means are over 5 seeds; metric SDs $< 10^{-4}$ are omitted. Best/second-best are purple/teal. GAPA uses the settings in Appendix I.

Model	MNIST							FMNIST						
	Metrics					Time (s)		Metrics					Time (s)	
	ACC	NLL	ECE	OOD	BALD	Train	Test	ACC	NLL	ECE	OOD	BALD	Train	Test
MAP (backbone)	0.978	0.068	0.005	0.919	0.919	—	1.24	0.859	0.392	0.007	0.846	0.821	—	1.20
LLA Diag	0.976	0.177	0.105	0.932	0.941	2.34K	2.39	0.856	0.421	0.057	0.872	0.873	2.34K	2.22
LLA KFAC	0.978	0.102	0.042	0.971	0.971	130.0	2.85	0.858	0.395	0.020	0.909	0.970	129.9	2.84
LLA*	0.978	0.070	0.009	0.924	0.924	42.0	4.6	0.859	0.395	0.019	0.850	0.716	42.0	4.7
LLA* KFAC	0.979	0.070	0.009	0.923	0.928	31.2	17.6	0.859	0.394	0.017	0.849	0.717	31.2	17.4
ELLA	0.978	0.068	0.005	0.919	0.912	821.8	148.7	0.859	0.392	0.007	0.846	0.765	827.1	149.9
VaLLA 100	0.978	0.068	0.005	0.919	0.934	2.19K	16.4	0.865	0.382	0.019	0.925	0.963	495.1	16.4
VaLLA 200	0.978	0.068	0.005	0.919	0.934	3.43K	18.3	0.867	0.378	0.020	0.937	0.970	767.7	19.3
Linear Probing	0.977	0.117	0.015	0.884	0.883	2.78K	3.6	0.858	0.395	0.048	0.785	0.776	2.64K	3.8
GPP	0.978	1.648	0.784	0.934	0.904	5.79K	23.5K	0.857	1.716	0.692	0.867	0.962	5.57K	2.27K
Dropout	0.978	0.072	0.009	0.923	0.944	—	4.3	0.858	0.393	0.009	0.850	0.911	—	4.3
Ensemble	0.979	0.069	0.038	0.936	0.962	—	11.9	0.859	0.373	0.041	0.863	0.938	—	11.9
DDU (est.)	0.978	0.068	0.005	0.921	0.919	202.4	8.2	0.859	0.392	0.007	0.876	0.983	202.3	8.2
GAPA	0.978	0.073	0.016	0.963	0.976	91.9	2.05	0.859	0.390	0.009	0.941	0.993	92.1	2.05

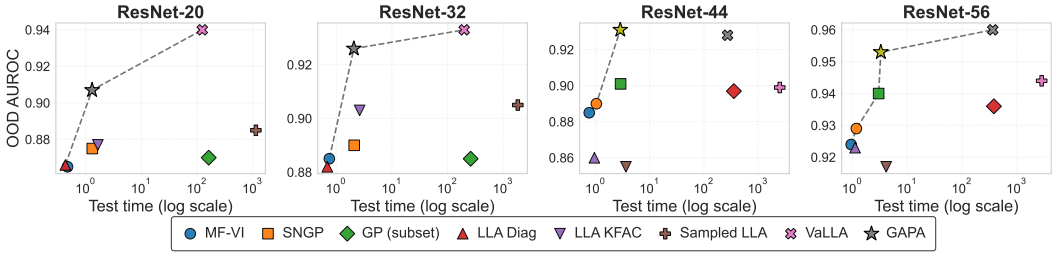


Figure 2: **OOD detection vs inference cost on CIFAR-10.** Dashed curves show Pareto frontiers; GAPA lies on the frontier, achieving strong OOD detection at lower inference cost.

increases for moderate-to-large rotations. GAPA maintains lower NLL than the baselines at large rotation angles, suggesting that its propagated activation-space uncertainty leads to better probabilistic predictions under this controlled distribution shift.

CIFAR-10 (pretrained ResNets). We evaluate CIFAR-10 with pretrained ResNet-20/32/44/56 backbones (He et al., 2016) and use SVHN as the OOD dataset. The central question is the *robustness–efficiency trade-off*: strong OOD baselines can be expensive at inference, while fast methods often sacrifice OOD detection. Figure 2 shows that GAPA lies on the Pareto frontier across ResNet depths, achieving strong OOD detection at substantially lower inference cost than expensive Bayesian baselines such as VaLLA and GP-subset. Full numerical results are provided in Appendix F.

3.3 Language Models

We attach GAPA post hoc to LLaMA-3.2-3B (hidden size 3072). For each chosen transformer block (we report layer indices), we log $\sim$12M pre-activations on WikiText-103 (training split) at sequence length $L = 96$ and build a nearest-neighbor cache for uncertainty propagation. We use KMeans preprocessing; Appendix H ablates inducing-point selection, and Appendix M details propagation through LLM components such as self-attention and RMSNorm. As in classification, after variance propagation the output layer yields mean logits $\mu_{1:k}$ and diagonal logit variances $v_{1:k}$ from a single forward pass. Predictive uncertainty is estimated via a vectorized reparameterization step over the top-k tokens per position ($k = 512$). Specifically, we draw $S = 512$ samples $\ell^{(s)} = \mu_{1:k} + \sqrt{v_{1:k}} \odot \epsilon^{(s)}$ with $\epsilon^{(s)} \sim \mathcal{N}(0, I)$ and compute $p^{(s)} = \mathrm{softmax}(\ell^{(s)})$, to form $\bar{p} = \frac{1}{S}\sum_{s=1}^{S} p^{(s)}$. This procedure introduces no additional network evaluations; sampling occurs only in logit space. Uncertainty is then decomposed using entropy as $\mathrm{TU} = H(\bar{p})$, $\mathrm{AU} = \frac{1}{S}\sum_{s=1}^{S} H(p^{(s)})$, $\mathrm{EU} = \mathrm{TU} - \mathrm{AU}$, where $H(p) = -\sum_v p_v \log p_v$. This is analogous to a mutual-information decomposition under the approximate Gaussian logit distribution; the resulting EU score is the GAPA-based uncertainty.

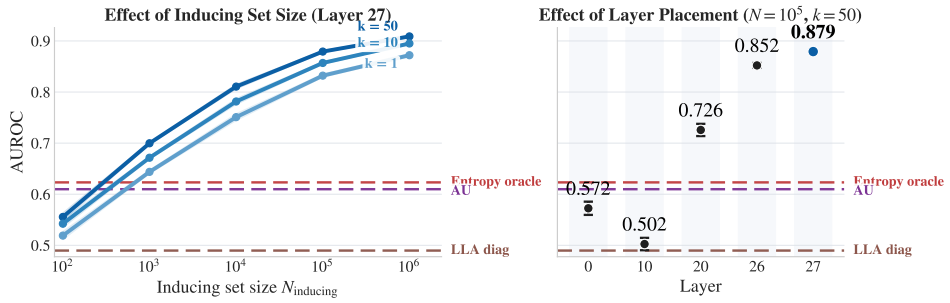


Figure 3: **OOD detection in LLaMA as a function of inducing-set size and layer placement.** Left: AUROC of GAPA-based epistemic uncertainty as the number of inducing points N_{inducing} increases, for different local neighbourhood sizes k, with GAPA attached at layer 27. Right: AUROC as a function of GAPA layer placement, with $N_{\text{inducing}} = 10^5$ and $k = 50$. Results are averaged over 5 runs with 512 sequences each. Dashed horizontal lines show reference baselines, including the oracle logit-temperature bound ℓ/T_{opt}, aleatoric uncertainty $\mathrm{AU}(\ell)$, and last-layer Laplace.

We use WikiText-103 validation as ID and OpenWebText as OOD, treating the latter as OOD relative to the WikiText-103 activation cache rather than to LLaMA itself. After filtering BOS/EOS and WikiText-103 punctuation-spacing artifacts, we average EU and AU over positions and compute AUROC against sequence labels. We include last-layer Laplace trained on WikiText-103 as a baseline, and an unfair temperature-scaling oracle that chooses τ to maximize test AUROC of the entropy of $\mathrm{softmax}(\ell/\tau)$; this oracle is not a fair baseline, but gives an upper bound for global logit rescaling.

Figure 3 shows that GAPA-based EU improves with the number of inducing points and with a larger local neighbourhood size. With enough inducing points, GAPA exceeds the global temperature-scaling oracle, suggesting that activation-space uncertainty captures distributional shift not recoverable by merely rescaling logits. The layer-placement sweep shows that later transformer blocks are most effective, with the strongest performance at layer 27 (last layer).

4 Related Work

Uncertainty quantification methods for deep networks differ primarily in *where* uncertainty is placed and the resulting test-time cost. Since we target *frozen pretrained backbones* with *single-pass* inference, we focus on *post-hoc uncertainty* baselines, including Bayesian last-layer methods, and defer a broader discussion (sampling-based, feature-based, calibration, and activation-space training methods) to Appendix P. Laplace approximations place a Gaussian posterior over weights via a local quadratic approximation of the log posterior (MacKay, 1992; Ritter et al., 2018); last-layer variants Bayesianize only the head while keeping the feature extractor frozen, making them a standard post-hoc baseline. In contrast, GAPA places a support-based epistemic uncertainty signal in *activation space* and uses the original nonlinearity as the GP prior mean, preserving the frozen network's point predictions by construction while enabling deterministic single-pass uncertainty estimation.

5 Conclusion

We introduced *GAPA*, a post-hoc uncertainty quantification method that places uncertainty in activation space rather than weight space. By using the original nonlinearity as the GP prior mean, GAPA preserves the frozen model's deterministic predictions while producing support-based epistemic variances from cached reference activations. A local inducing-point approximation and deterministic variance propagation enable single-pass uncertainty estimation without retraining, repeated forward passes, test-time backpropagation, or second-order computations. Across regression, classification, segmentation, and language modeling, GAPA achieves competitive calibration and OOD detection with favorable robustness–efficiency trade-offs. Its main limitations are the memory cost of cached inducing activations and the diagonal covariance approximation; future work includes compressed indexing and structured covariance propagation.

AI USE STATEMENT

Generative AI tools were used to assist with writing code, proofreading, and drafting and revising the manuscript. The authors take responsibility for the final text, code, analyses, and scientific claims.

ETHICS STATEMENT

GAPA provides post-hoc uncertainty estimates for frozen pretrained models. These estimates may support human review or abstention, but they do not provide formal safety guarantees. Because the uncertainty signal depends on coverage of the reference activation cache, some distribution shifts may remain undetected. Overreliance on these estimates could therefore lead to misplaced confidence in high-stakes applications. We do not deploy GAPA in decision-making systems. Before real-world use, we recommend evaluating calibration, subgroup performance, and failure modes in the intended deployment setting.

REPRODUCIBILITY STATEMENT

The main text specifies the GP activation construction, local conditioning scheme, and variance propagation. The appendices provide supporting derivations, empirical kernel hyperparameter rules, regression likelihood details, and implementation examples for transformer variance propagation. Additional experimental results, ablation studies, metric definitions, and information about computational resources are also provided.

REFERENCES

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Richard Bergna, Sergio Calvo-Ordonez, Felix L Opolka, Pietro Liò, and Jose Miguel Hernandez-Lobato. Uncertainty modeling in graph neural networks via stochastic differential equations. *arXiv preprint arXiv:2408.16115*, 2024.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pp. 1613–1622. PMLR, 2015.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in neural information processing systems*, 34: 20089–20103, 2021.

Zhijie Deng, Feng Zhou, and Jun Zhu. Accelerated linearized Laplace approximation for Bayesian deep learning. *Advances in Neural Information Processing Systems*, 35:2695–2708, 2022.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *IEEE Transactions on Big Data*, 2025.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Robert B Gramacy and Daniel W Apley. Local Gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, 24(2):561–578, 2015.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

James Harrison, John Willes, and Jasper Snoek. Variational Bayesian last layers. *arXiv preprint arXiv:2404.11599*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in neural information processing systems*, 33:7498–7512, 2020.

David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3): 448–472, 1992.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in neural information processing systems*, 32, 2019.

Andrew McHutchon and Carl Rasmussen. Gaussian process training with input noise. *Advances in neural information processing systems*, 24, 2011.

Jishnu Mukhoti, Andreas Kirsch, Joost Van Amersfoort, Philip HS Torr, and Yarin Gal. Deep deterministic uncertainty: A new simple baseline. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24384–24394, 2023.

Luis A Ortega, Simón Rodríguez Santana, and Daniel Hernández-Lobato. Variational linearized Laplace approximation for Bayesian deep learning. *arXiv preprint arXiv:2302.12565*, 2023.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *6th international conference on learning representations, ICLR 2018-conference track proceedings*, volume 6. International Conference on Representation Learning, 2018.

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*, pp. 234–241. Springer, 2015.

Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. *Advances in neural information processing systems*, 30, 2017.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial intelligence and statistics*, pp. 567–574. PMLR, 2009.

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Appendix Table of Contents

A Comparison with Distance and Density Baselines **14**

B Qualitative GAPA Plots **15**

C Conservativeness of Subset GP Conditioning **15**

D Inducing-Point Conditional Interpretation and Zero-Noise Limit **16**

E Derivation for Stacking GAPA Layers **17**

E.1 Support-Based Epistemic–Aleatoric Decomposition 18

F ResNets Pretrained Neural Networks **18**

G Image Segmentation **19**

H LLaMA-3.2 Additional Results **20**

I GAPA Hyperparameters **20**

I.1 Empirical Kernel Hyperparameters 20

I.2 Regression Likelihood and Aleatoric Head 21

J Nearest–Neighbour Retrieval with `Faiss` **21**

J.1 Index construction 22

J.2 Query procedure 22

J.3 Complexity 22

J.4 Complexity summary 22

K Laplace-Bridge Approximation for Classification **23**

L Metrics **23**

L.1 Regression Metrics 23

L.2 Classification Metrics 23

M Variance Propagation in Transformer Architectures **24**

M.1 Attention 24

M.2 RMSNorm 25

M.3 Softmax 26

N Random Seeds and Rotation Results **27**

O Ablation Studies **27**

O.1 Full Covariance and Monte Carlo Propagation 27

O.2 Sensitivity to Reference Cache Coverage 27
O.3 Where to put GAPA 28
O.4 Number of inducing inputs 28
O.5 Inducing point selection: KMeans vs. farthest-point sampling 29
O.6 Random vs. Farthest-Point Sampling 29
O.7 KNN Sweep: $K = 1$ to 500 29

P Extended Related Work **31**

Q Assets and Licenses **33**

R Compute Resources **34**

A Comparison with Distance and Density Baselines

We compare GAPA with distance and density scores computed in the same activation space to assess whether its GP formulation provides additional value beyond simpler measures of representation support.

Controlled comparison. We evaluate nearest-neighbour distance, mean kNN distance, an RBF outlier score, and Mahalanobis distance over five seeds. Within each seed, every method uses the same frozen backbone, activation layer, cached features, and test examples. GAPA, mean kNN distance, and the RBF score use the same $K = 50$ neighbours. The RBF baseline also uses the same kernel and hyperparameters as GAPA. These controls ensure that differences do not arise from using different learned representations or reference data.

What the GP formulation adds. The difference is that these baselines assign uncertainty using only distances or densities, whereas the GP posterior conditions on the neighbourhood jointly through its covariance matrix. The resulting uncertainty therefore depends not only on how far the query lies from the reference activations, but also on the covariance structure of the local neighbourhood.

For one activation coordinate, GAPA computes

$$v(\mathbf{z}) = k(\mathbf{z}, \mathbf{z}) - \mathbf{k}_{\mathbf{z}}^{\top} \left(\mathbf{K}_{\mathcal{N}} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{k}_{\mathbf{z}}, \tag{5}$$

where $\mathbf{k}_{\mathbf{z}}$ contains similarities between the query and its K neighbours, and $\mathbf{K}_{\mathcal{N}}$ contains similarities among the neighbours themselves. The inverse kernel matrix accounts for redundancy among the conditioning points. For example, two neighbourhoods can have identical distances from the query but different arrangements of their reference points. Their nearest-neighbour, mean-distance, and RBF scores can therefore coincide while their GP posterior variances differ.

Mahalanobis distance incorporates covariance through its distance metric, but does not perform this local GP conditioning. The $K \times K$ covariance among reference points is retained in GAPA even though covariance across activation outputs is approximated as diagonal.

Activation-space uncertainty is then propagated through the remaining frozen network, so the final predictive uncertainty depends both on representation support and on the downstream weights and nonlinearities. Two inputs with similar activation-space support can therefore receive different predictive uncertainties because the network transforms their perturbations differently.

Correlation and OOD detection. Table 3 reports OOD AUROC and Pearson correlation with GAPA. The ID datasets are MNIST, FMNIST, and CIFAR-10; their corresponding OOD datasets are FMNIST, MNIST, and SVHN.

Table 3: OOD detection AUROC and Pearson correlation r between each method's uncertainty score and GAPA's, evaluated over five seeds in the same activation space. Dataset headings identify ID data; OOD data are FMNIST, MNIST, and SVHN, respectively. GAPA's self-correlation is 1 by definition. Best AUROCs are bold.

Method	MNIST		FMNIST		CIFAR-10	
	AUROC ↑	Pearson r	AUROC ↑	Pearson r	AUROC ↑	Pearson r
Nearest-neighbour distance	0.9851	0.958	0.9930	0.931	0.9314	0.933
Mean kNN distance, $K = 50$	0.9827	0.940	0.9925	0.911	0.9299	0.911
RBF outlier score	0.9827	0.944	0.9924	0.929	0.9298	0.930
Mahalanobis distance	0.9556	0.867	0.9779	0.846	0.9198	0.888
GAPA	**0.9905**	1.000	**0.9962**	1.000	**0.9351**	1.000

GAPA is strongly correlated with the distance and density baselines, consistent with their shared dependence on activation-space support. Nevertheless, it achieves higher OOD AUROC in all three comparisons. Against the strongest distance baseline, nearest-neighbour distance, the improvements are 0.0054, 0.0032, and 0.0037, respectively. Thus, strong correlation does not imply equivalent OOD detection performance. These controlled comparisons support the benefit of GAPA's joint

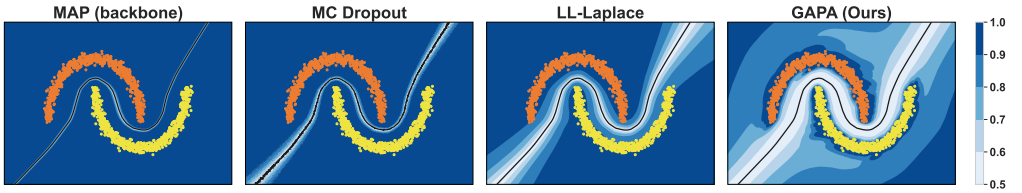


Figure 4: Toy binary classification. Left to right: MAP backbone, MC Dropout, last-layer Laplace, and GAPA. Background shading shows predictive confidence; points show the two classes. GAPA preserves the backbone's mean decision boundary while adding activation-space uncertainty in regions weakly supported by cached reference pre-activations.

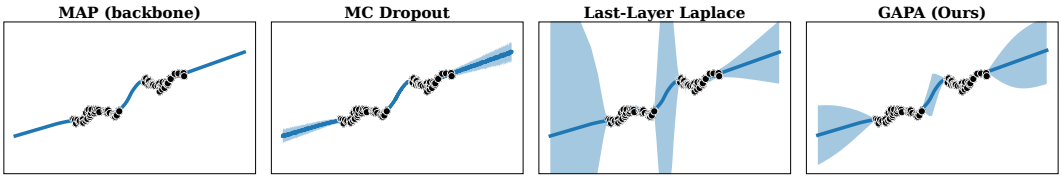


Figure 5: Toy regression. Left to right: MAP backbone, MC Dropout, last-layer Laplace, and GAPA.

conditioning and subsequent variance propagation over directly using the evaluated distance and density scores.

B Qualitative GAPA Plots

This appendix provides qualitative toy examples illustrating how GAPA behaves relative to common post-hoc uncertainty baselines. Figure 4 compares predictive confidence on a binary classification task, while Figure 5 compares predictive means and uncertainty intervals on a regression task. Together, the plots show the intended qualitative behavior of GAPA: it preserves the deterministic backbone's mean prediction while adding activation-space uncertainty in regions weakly supported by the reference activation cache.

C Conservativeness of Subset GP Conditioning

This appendix justifies the monotonicity claim made in Section 2.3: conditioning a GP posterior on a subset of inducing inputs yields posterior variance no smaller than conditioning on the full inducing set.

Lemma 1 (Conservative uncertainty under subset conditioning)**.** *Consider a Gaussian process prior with fixed kernel hyperparameters and Gaussian observation noise. Let $\tilde{Z}$ denote a set of inducing inputs and let $\tilde{Z}_k \subset \tilde{Z}$ be any subset. Then, for any test input z^*, the posterior variance satisfies*

$$\mathrm{Var}\Big(f(z^*) \mid \tilde{Z}\Big) \;\le\; \mathrm{Var}\Big(f(z^*) \mid \tilde{Z}_k\Big)\,.$$

That is, conditioning on a subset of inducing inputs cannot reduce posterior variance relative to conditioning on the full inducing set.

Proof. Let $A = \tilde{Z}_k$ and $B = \tilde{Z} \setminus \tilde{Z}_k$. Under the GP prior with Gaussian observation noise, the random variables

$$\big(f(z^*),\, y_A,\, y_B\big)$$

are jointly Gaussian. The conditional covariance identity for jointly Gaussian variables gives

$$\mathrm{Var}(f(z^*) \mid y_A) - \mathrm{Var}(f(z^*) \mid y_A, y_B) = K_{*B|A}\left(K_{BB|A} + \sigma_n^2 I\right)^{-1} K_{B*|A},$$

where $K_{BB|A}$ and $K_{*B|A}$ denote conditional covariance matrices obtained via the Schur complement.

Since $K_{BB|A} + \sigma_n^2 I$ is positive definite, the right-hand side is positive semidefinite. Therefore,

$$\mathrm{Var}(f(z^*) \mid y_A, y_B) \;\le\; \mathrm{Var}(f(z^*) \mid y_A),$$

which establishes the result. □

This result implies that the local k-nearest-neighbour conditioning strategy used in GAPA yields a conservative approximation of the GP posterior variance: posterior variance may be inflated due to discarded conditioning points, but it is not underestimated relative to the full inducing set.

D Inducing-Point Conditional Interpretation and Zero-Noise Limit

In this appendix, we clarify the inducing-point interpretation used by GAPA. GAPA does not optimize a variational sparse-GP objective. Instead, it uses a fixed set of inducing inputs, obtained from cached pre-activations, and applies the standard GP conditional covariance restricted to those inducing inputs. This is closely related to sparse inducing-point GP approximations, but the inducing locations are chosen by a preprocessing rule such as k-means rather than by ELBO optimization.

Setup. Consider a scalar Gaussian process

$$f(\cdot) \sim \mathcal{GP}\big(m(\cdot), k(\cdot,\cdot)\big),$$

and a set of inducing inputs

$$\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^M, \qquad \mathbf{u} = f(\mathbf{Z}).$$

Let $\mathbf{z}^*$ denote a test input, and write

$$K_{\mathbf{ZZ}} = k(\mathbf{Z},\mathbf{Z}), \qquad k_{*\mathbf{Z}} = k(\mathbf{z}^*,\mathbf{Z}), \qquad k_{\mathbf{Z}*} = k(\mathbf{Z},\mathbf{z}^*).$$

In GAPA, the pseudo-observations at the inducing inputs are evaluations of the prior mean function,

$$\tilde{\mathbf{y}} = m(\mathbf{Z}).$$

For numerical stability, we use a small Gaussian noise term σ_n^2.

Mean preservation. Conditioning the GP on noisy pseudo-observations $(\mathbf{Z}, \tilde{\mathbf{y}})$ gives posterior mean

$$\mu(\mathbf{z}^*) = m(\mathbf{z}^*) + k_{*\mathbf{Z}}\big(K_{\mathbf{ZZ}} + \sigma_n^2 I\big)^{-1}\big(\tilde{\mathbf{y}} - m(\mathbf{Z})\big).$$

Since $\tilde{\mathbf{y}} = m(\mathbf{Z})$, the residual term is zero. Therefore

$$\mu(\mathbf{z}^*) = m(\mathbf{z}^*).$$

In GAPA, $m(\cdot)$ is chosen to be the original activation function, so the posterior mean exactly preserves the deterministic activation.

Posterior covariance. The corresponding GP posterior covariance is independent of the pseudo-target values and is given by

$$\mathrm{Var}\big(f(\mathbf{z}^*) \mid \mathbf{Z}, \tilde{\mathbf{y}}\big) = k(\mathbf{z}^*,\mathbf{z}^*) - k_{*\mathbf{Z}}\big(K_{\mathbf{ZZ}} + \sigma_n^2 I\big)^{-1} k_{\mathbf{Z}*}.$$

This is the covariance used by GAPA when conditioning on an inducing set. In the zero-noise limit,

$$\mathrm{Var}\big(f(\mathbf{z}^*) \mid \mathbf{Z}, \tilde{\mathbf{y}}\big) \;\xrightarrow[\sigma_n^2 \to 0]{}\; k(\mathbf{z}^*,\mathbf{z}^*) - k_{*\mathbf{Z}} K_{\mathbf{ZZ}}^{-1} k_{\mathbf{Z}*},$$

which is the standard noiseless GP conditional covariance given the inducing values.

Relation to posterior uncertainty over inducing variables. Equivalently, one may view the noisy pseudo-observations as inducing a Gaussian posterior over the inducing variables,

$$q(\mathbf{u}) = p(\mathbf{u} \mid \tilde{\mathbf{y}}) = \mathcal{N}(\boldsymbol{\mu}_u, \mathbf{A}),$$

where

$$\boldsymbol{\mu}_u = m(\mathbf{Z}) + K_{\mathbf{ZZ}}\big(K_{\mathbf{ZZ}} + \sigma_n^2 I\big)^{-1}\big(\tilde{\mathbf{y}} - m(\mathbf{Z})\big) = m(\mathbf{Z}),$$

and

$$\mathbf{A} = K_{\mathbf{ZZ}} - K_{\mathbf{ZZ}}\big(K_{\mathbf{ZZ}} + \sigma_n^2 I\big)^{-1} K_{\mathbf{ZZ}}.$$

As $\sigma_n^2 \to 0$, we have $\mathbf{A} \to 0$, reflecting that the inducing values become known exactly in the noiseless limit.

Integrating the conditional GP $p(f(\mathbf{z}^*) \mid \mathbf{u})$ against $q(\mathbf{u})$ gives

$$\mathrm{Var}_q(f(\mathbf{z}^*)) = k(\mathbf{z}^*, \mathbf{z}^*) - k_{*\mathbf{Z}} K_{\mathbf{ZZ}}^{-1} k_{\mathbf{Z}*} + k_{*\mathbf{Z}} K_{\mathbf{ZZ}}^{-1} \mathbf{A} K_{\mathbf{ZZ}}^{-1} k_{\mathbf{Z}*}.$$

In the zero-noise limit, $\mathbf{A} \to 0$, so this reduces to

$$\mathrm{Var}_q(f(\mathbf{z}^*)) = k(\mathbf{z}^*, \mathbf{z}^*) - k_{*\mathbf{Z}} K_{\mathbf{ZZ}}^{-1} k_{\mathbf{Z}*}.$$

Implication for GAPA. For each activation coordinate, GAPA uses the original activation as the GP prior mean and uses cached inducing pre-activations to compute the GP posterior variance. The posterior mean therefore preserves the deterministic activation exactly, while the posterior covariance supplies a support-based epistemic uncertainty estimate in activation space.

E Derivation for Stacking GAPA Layers

When GAPA layers are stacked, the output of a preceding GAPA layer becomes the input to a subsequent GAPA layer. Since each GAPA module produces a Gaussian output, the input to the current layer is itself a random variable. Let

$$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$$

denote the pre-activation input to the current GAPA layer, where $\boldsymbol{\Sigma}_z$ is diagonal under our approximation. We write

$$\mathbf{z} = \boldsymbol{\mu}_z + \boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_z).$$

We consider neuron-wise propagation. For neuron i, the scalar input z_i satisfies

$$z_i = \mu_{z,i} + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma_{z,i}^2).$$

Base support-based epistemic variance (deterministic input). Ignoring input uncertainty, the posterior variance of neuron i under the KNN GAPA approximation is given by the standard GP conditional variance restricted to the local neighbourhood $\mathbf{Z}_{\ell,K}(\boldsymbol{\mu}_z)$:

$$\sigma_{\mathrm{epi},i}^2(\boldsymbol{\mu}_z) = k_i(\boldsymbol{\mu}_z, \boldsymbol{\mu}_z) - \mathbf{k}_i^\top \big(\mathbf{K}_i + \sigma_n^2 I\big)^{-1} \mathbf{k}_i. \tag{6}$$

Here $\mathbf{K}_i \in \mathbb{R}^{K \times K}$ is the kernel matrix over the K nearest inducing pre-activation vectors $\{\mathbf{z}_m\}_{m=1}^K = \mathbf{Z}_{\ell,K}(\boldsymbol{\mu}_z)$, with

$$[\mathbf{K}_i]_{mn} = k_i(\mathbf{z}_m, \mathbf{z}_n), \qquad [\mathbf{k}_i]_m = k_i(\boldsymbol{\mu}_z, \mathbf{z}_m).$$

Thus, although the output is neuron-wise, the GP input remains the full pre-activation vector $\boldsymbol{\mu}_z \in \mathbb{R}^{d_\ell}$.

Input uncertainty correction (NIGP). Because the test input to a downstream GAPA layer is uncertain, we apply a noisy-input GP approximation. We linearize the posterior mean of output coordinate i, denoted $\mu_{\ell,i}(\mathbf{z})$, around the mean input $\boldsymbol{\mu}_z$. This yields the additional variance term

$$\lambda_{\ell,i}(\boldsymbol{\mu}_z) = \nabla_{\mathbf{z}} \mu_{\ell,i}(\boldsymbol{\mu}_z)^\top \,\mathrm{diag}(\mathbf{v}_z) \nabla_{\mathbf{z}} \mu_{\ell,i}(\boldsymbol{\mu}_z). \tag{7}$$

By construction of GAPA, the posterior mean equals the original deterministic activation. For an element-wise activation,

$$\mu_{\ell,i}(\mathbf{z}) = \phi_\ell(z_i),$$

so the gradient has only one nonzero component. Therefore,

$$\lambda_{\ell,i}(\boldsymbol{\mu}_z) = v_{z,i}\big(\phi_\ell'(\mu_{z,i})\big)^2.$$

Table 4: GAPA and baseline results on CIFAR-10 with ResNet backbones. Results are averaged over five random seeds. Standard deviations of ACC (in percentage points), NLL, and OOD AUROC are below 10^{-4} in all cases and are omitted. Best and second-best values in each column are shown in purple and teal, respectively. Higher ACC/OOD and lower NLL/train/test time are better; ties at the displayed precision share the same color.

	ResNet-20					ResNet-32					ResNet-44					ResNet-56				
	ACC	NLL	OOD	Train	Test	ACC	NLL	OOD	Train	Test	ACC	NLL	OOD	Train	Test	ACC	NLL	OOD	Train	Test
MAP	92.6	0.282	0.876	–	–	93.5	0.292	0.909	–	–	94.0	0.275	0.885	–	0.761	94.4	0.252	0.924	–	0.949
MF-VI	92.7	0.231	0.865	0.74	0.47	93.5	0.222	0.885	1.19	0.75	93.9	0.206	0.890	1.63	1.03	94.4	0.188	0.929	1.97	1.18
SNGP	92.4	0.266	0.875	15.9	1.31	93.2	0.256	0.890	25.5	2.10	93.8	0.242	0.901	35.0	2.89	93.8	0.229	0.940	43.5	3.01
GP (subset)	92.6	0.555	0.870	3.75K	162	93.4	0.462	0.885	6.00K	260	93.6	0.424	0.897	8.25K	357	94.4	0.403	0.936	8.42K	382
LLA Diag	92.6	0.260	0.866	18.4	0.43	93.5	0.242	0.882	29.4	0.69	94.0	0.218	0.860	40.4	0.947	94.3	0.195	0.923	40.67	1.12
LLA KFAC	92.6	0.241	0.877	28.7	1.65	93.5	0.229	0.903	45.9	2.63	94.0	0.213	0.855	63.1	3.62	94.4	0.193	0.917	71.3	4.13
Sampled LLA	92.5	0.231	0.885	5.00K	1.14K	93.5	0.217	0.905	8.00K	1.83K	94.0	0.200	0.899	11.0K	2.51K	94.4	0.185	0.944	14.6K	2.84K
VaLLA	92.4	0.231	0.940	7.59K	124	93.2	0.212	0.933	12.2K	199	93.8	0.201	0.928	16.7K	272.9	94.2	0.188	0.960	26.3K	363.8
GAPA (ours)	92.6	0.258	0.907	3.65	1.30	93.5	0.259	0.926	5.84	2.07	94.0	0.230	0.931	8.03	2.85	94.4	0.230	0.953	10.29	3.30

Total predictive variance. Combining the support-based epistemic variance from the local inducing-point posterior, the NIGP correction due to input uncertainty, and optional observation noise $\sigma^2_{y,i}$, the total predictive variance for neuron i is

$$\mathrm{Var}[y_i] = \underbrace{\sigma^2_{\mathrm{epi},i}(\boldsymbol{\mu}_z)}_{\text{support-based epistemic uncertainty}} + \underbrace{v_{z,i}\big(\phi'_\ell(\mu_{z,i})\big)^2}_{\text{propagated input uncertainty}} + \sigma^2_{y,i}. \tag{8}$$

This expression shows that stacking GAPA layers preserves mean predictions exactly while propagating uncertainty forward in closed form, with each layer contributing an additional support-based epistemic variance and attenuating or amplifying uncertainty according to the local slope of the activation function.

E.1 Support-Based Epistemic–Aleatoric Decomposition

The distinction between activation-space epistemic uncertainty and aleatoric likelihood noise can be made explicit through the law of total variance. Under the GAPA predictive distribution, a regression likelihood gives

$$\mathrm{Var}_{\mathrm{GAPA}}(y^* \mid x^*, \mathcal{D}) = \underbrace{\mathbb{E}_{\mathbf{a}^*_\ell}[\mathrm{Var}(y^* \mid h_\ell(\mathbf{a}^*_\ell))]}_{\text{aleatoric / likelihood noise}} + \underbrace{\mathrm{Var}_{\mathbf{a}^*_\ell}(\mathbb{E}[y^* \mid h_\ell(\mathbf{a}^*_\ell)])}_{\text{support-based epistemic uncertainty}}. \tag{9}$$

The expectation and variance over $\mathbf{a}^*_\ell$ are taken with respect to the GP activation posterior

$$p\big(\mathbf{a}^*_\ell \mid \mathbf{z}_\ell(x^*), \mathcal{D}_\ell\big).$$

GAPA contributes to the second term through the auxiliary GP posterior over $\mathbf{a}^*_\ell$; this term reflects reducible uncertainty due to limited activation-space support, while any regression-specific aleatoric component enters through the first term. Dependence on frozen network parameters and any likelihood parameters estimated from $\mathcal{D}$ is suppressed for notational compactness.

F ResNets Pretrained Neural Networks

We report supplementary experiments and visualizations omitted from the main text for space.

ResNet backbones and computational trade-offs. Table 4 reports full CIFAR-10 results for pretrained ResNet backbones, including in-distribution performance (ACC/NLL), OOD detection (AUROC), and train/test runtime. Across all depths, GAPA achieves competitive in-distribution accuracy and NLL while providing strong OOD detection. Notably, methods with the strongest OOD scores (e.g., VaLLA and sampled/GP-subset baselines) often incur substantially higher computational costs, both at test time and train time.

Figure 2 visualises the robustness–efficiency trade-off by plotting OOD AUROC against test-time inference cost (log scale). Dashed lines indicate Pareto frontiers (higher OOD, lower cost), highlighting

methods that offer the best compromise between reliability and efficiency. Across backbones, GAPA lies on the Pareto frontier, achieving strong OOD AUROC while avoiding the 10^2–10^3s inference costs characteristic of more computationally intensive Bayesian baselines.

G Image Segmentation

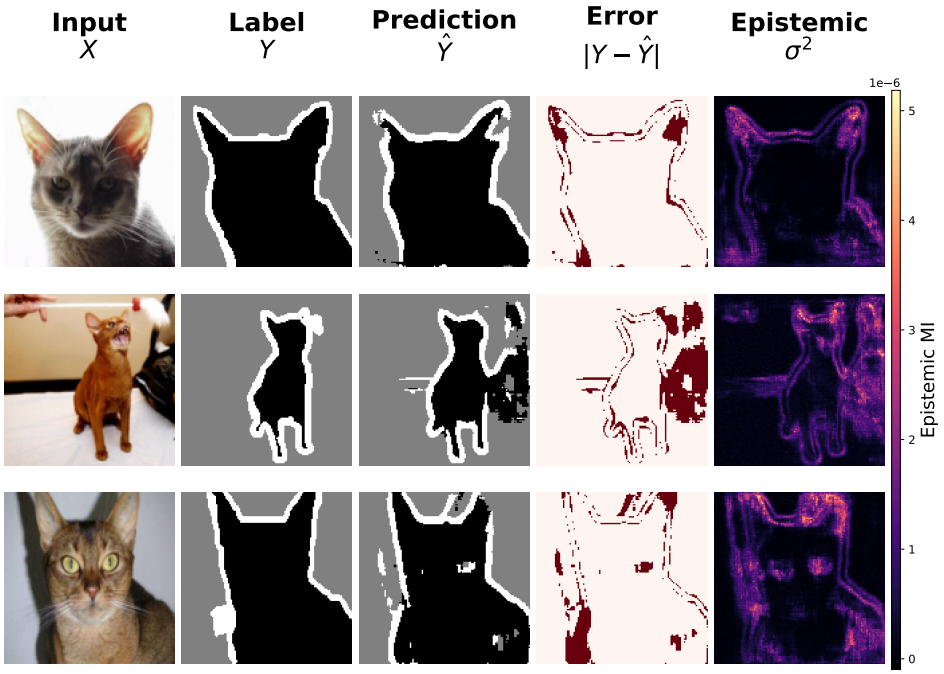


Figure 6: Qualitative segmentation results with pixel-wise error and support-based epistemic uncertainty. **Columns**: (1) Input image X, (2) Ground-truth mask Y, (3) Predicted mask $\hat{Y}$, (4) Error map $|Y - \hat{Y}|$, (5) Support-based epistemic uncertainty (mutual information). **Rows**: three representative validation examples.

As a proof of concept for high-dimensional outputs, we apply GAPA to a U-Net model (Ronneberger et al., 2015) pre-trained on the Oxford-IIIT Pet dataset (Parkhi et al., 2012) for a 3-class segmentation task (background, pet, outline) with input images resized to 128×128. The U-Net architecture features an encoder path with two downsampling stages (32 and 64 channels, using double convolutions and max pooling), leading to a bottleneck with 128 channels. From this bottleneck, an embedding head comprising adaptive average pooling and a linear layer projects the features to a $d = 64$ dimensional embedding vector. Standard skip connections are used in the decoder path.

For these experiments, GAPA was applied to this $d = 64$ dimensional embedding vector at the U-Net bottleneck. This vector represents the most compressed representation in the network; after pooling and flattening, its one-dimensional structure makes it convenient for applying GAPA. The GAPA-processed embedding (mean preserved, variance added) is then reshaped and fed into the decoder to produce the final segmentation map.

The dimensionality of the full segmentation output space (e.g., $128 \times 128 \times 3$) renders methods like full Laplace approximation computationally infeasible due to memory and time constraints. In contrast, applying GAPA at the compressed embedding stage scales efficiently. Figure 6 demonstrates that this approach not only produces accurate segmentation masks but also generates spatially localized support-based epistemic uncertainty maps that closely track regions where prediction errors occur.

H LLaMA-3.2 Additional Results

In Figure 7 we show an additional ablation study where we compare inducing point selection using `Kmeans` or random choice. We find that random underperformed compared to `KMeans`.

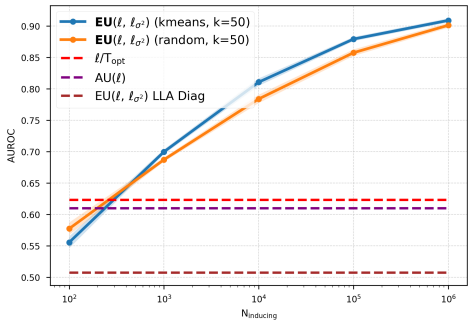


Figure 7: Effect of the number of inducing points N_{inducing} and preprocessing strategy on OOD detection task. We plot the AUC using EU (blue) with GAPA at layer [27]. Results are averaged over 5 runs with 512 sequences each. The horizontal line shows the ℓ/T_{opt} bound as an upper threshold for what can be achieved by global logit scaling.

I GAPA Hyperparameters

This section describes how we set the empirical GP hyperparameters used by GAPA and how we handle the regression likelihood. The GAPA kernel hyperparameters and inducing inputs are fixed from cached activation statistics and are not optimized with labels or backpropagation. In the regression experiments only, we train a separate aleatoric likelihood head; the backbone and GAPA module remain fixed.

I.1 Empirical Kernel Hyperparameters

GAPA deliberately avoids gradient-based optimization of GP hyperparameters. For each selected layer ℓ, we first run the frozen backbone on a reference training set and cache the resulting pre-activations

$$\tilde{\mathbf{Z}}_\ell = \{\tilde{\mathbf{z}}_\ell^{(n)}\}_{n=1}^{N_\ell}, \qquad \tilde{\mathbf{z}}_\ell^{(n)} \in \mathbb{R}^{d_\ell}.$$

The corresponding deterministic activation targets are

$$\tilde{\mathbf{y}}_\ell^{(n)} = \boldsymbol{\phi}(\tilde{\mathbf{z}}_\ell^{(n)}).$$

We then fix the RBF kernel hyperparameters once from these cached quantities.

For activation coordinate i, we use

$$k_{\ell,i}(\mathbf{z}, \mathbf{z}') = c_{\ell,i}^2 \exp\left(-\frac{\|\mathbf{z} - \mathbf{z}'\|_2^2}{2\rho_\ell^2}\right),$$

where ρ_ℓ is a layer-wise length scale and $c_{\ell,i}^2$ is a coordinate-wise signal variance.

Length scale. We set the layer-wise length scale ρ_ℓ to the empirical median of pairwise Euclidean distances between cached pre-activation vectors:

$$d_{mn}^{(\ell)} = \left\|\tilde{\mathbf{z}}_\ell^{(m)} - \tilde{\mathbf{z}}_\ell^{(n)}\right\|_2, \qquad \rho_\ell = \text{Median}\left(\{d_{mn}^{(\ell)}\}\right).$$

In practice, we approximate the median by sampling 10^6 random pairs from the activation cache.

Signal variance. For each activation coordinate i, we set the signal variance to the empirical variance of the deterministic activation values on the cached training set:

$$c_{\ell,i}^2 = \mathrm{Var}\left(\{\phi_i(\tilde{\mathbf{z}}_\ell^{(n)})\}_{n=1}^{N_\ell}\right).$$

We clamp $c_{\ell,i}^2$ to a minimum value of 10^{-6} for numerical stability.

Numerical jitter. We add a small diagonal jitter term $\sigma_n^2 I$ to all local kernel matrices before inversion. Unless otherwise stated, we use $\sigma_n^2 = 10^{-6}$. This term is used only for numerical stability and is not learned.

Inducing inputs. Given an inducing-point budget M_ℓ, we construct a compressed inducing set

$$\mathbf{Z}_\ell = \{\mathbf{z}_{\ell,m}\}_{m=1}^{M_\ell} \subset \mathbb{R}^{d_\ell}$$

from the cached pre-activation vectors. Our default construction uses k-means on $\tilde{\mathbf{Z}}_\ell$, taking the M_ℓ cluster centroids as inducing inputs. We initialize the clustering with k-means++ seeding, where the first center is chosen uniformly at random and subsequent centers are sampled with probability proportional to their squared distance from the closest existing center.

We also evaluate farthest-point sampling (FPS) as an alternative inducing-input selection rule in Appendix O. FPS greedily selects points that maximize coverage of the cached activation space:

1. Select an initial point $\mathbf{z}_{\ell,1}$ from the cache.
2. For $m = 2, \ldots, M_\ell$, choose the cached pre-activation whose minimum distance to the already selected set $\{\mathbf{z}_{\ell,1}, \ldots, \mathbf{z}_{\ell,m-1}\}$ is maximal.

Both k-means and FPS are task-agnostic preprocessing rules and do not use labels.

I.2 Regression Likelihood and Aleatoric Head

For regression, predictive uncertainty contains both the support-based epistemic component produced by GAPA in activation space and an aleatoric likelihood component. We therefore train a small separate noise head s_ψ to predict data-dependent observation noise:

$$\sigma_{\mathrm{ale}}^2(\mathbf{x}) = \mathrm{softplus}(s_\psi(\mathbf{x})) + \varepsilon,$$

where $\varepsilon = 10^{-6}$ is a variance floor.

The total predictive variance is

$$\sigma_{\mathrm{tot}}^2(\mathbf{x}) = \sigma_{\mathrm{epi}}^2(\mathbf{x}) + \sigma_{\mathrm{ale}}^2(\mathbf{x}),$$

where $\sigma_{\mathrm{epi}}^2(\mathbf{x})$ denotes the support-based epistemic variance obtained from GAPA variance propagation. We train only the aleatoric-head parameters ψ by minimizing the Gaussian negative log-likelihood

$$\mathcal{L}_{\mathrm{reg}} = \frac{1}{N}\sum_{n=1}^{N}\left[\frac{(y_n - \mu_n)^2}{2\sigma_{\mathrm{tot}}^2(\mathbf{x}_n)} + \frac{1}{2}\log\big(2\pi\sigma_{\mathrm{tot}}^2(\mathbf{x}_n)\big)\right],$$

where μ_n is the fixed mean prediction of the frozen backbone.

This supervised step is used only to estimate aleatoric observation noise in regression. It does not update the backbone, the GAPA inducing inputs, or the GAPA kernel hyperparameters, and therefore preserves the backbone's deterministic mean predictions.

J Nearest–Neighbour Retrieval with Faiss

GAPA requires fast retrieval of the K nearest inducing activations in activation space. Given a set of inducing inputs $\mathcal{Z} = \{z_m\}_{m=1}^M \subset \mathbb{R}^d$, for each query activation z^* we compute

$$\mathcal{N}_K(z^*) = \arg\min_{\substack{S\subset\{1,\ldots,M\}\\ |S|=K}} \sum_{m\in S} \|z^* - z_m\|_2.$$

A brute-force search costs $\mathcal{O}(Md)$ per query. Instead, we use `Faiss` to support efficient approximate nearest-neighbour retrieval with sublinear complexity.

J.1 INDEX CONSTRUCTION

1. **Choice of index.** For small to moderate inducing sets we use `IndexFlatL2` (exact search). For large M, we use `IndexIVFPQ`, which partitions the space via coarse k-means clustering and stores PQ-compressed residuals (Douze et al., 2025).
2. **Training (optional).** Quantization-based indices (IVF, PQ) require a one-time offline training step on a representative subset of inducing activations.
3. **Adding vectors.** All inducing activations z_m are added once to the index; their identifiers link back to the cached kernel statistics required for GP inference.

The resulting index requires $\mathcal{O}(M)$ memory and supports K-NN queries in $\mathcal{O}(\log M)$ expected time for IVF-based indices.

J.2 QUERY PROCEDURE

For each test-time activation z^*:

1. Query the Faiss index to retrieve the K nearest inducing inputs:

$$(\mathbf{d}, \mathcal{N}_K) \leftarrow \texttt{index.search}(z^*,\, K).$$

2. Form the local inducing set $\mathbf{Z}_K(z^*) = \{z_m : m \in \mathcal{N}_K\}$.
3. Compute the neuron-wise posterior variance using the standard inducing-point conditional restricted to this local set:

$$\mathrm{Var}[f(z^*)] = k(z^*, z^*) - \mathbf{k}^\top \left(\mathbf{K}_K + \sigma_n^2 I\right)^{-1} \mathbf{k},$$

   where $\mathbf{K}_K$ is the $K \times K$ kernel matrix on $\mathbf{Z}_K(z^*)$ and $[\mathbf{k}]_m = k(z^*, z_m)$.

By construction, the posterior mean remains equal to the original deterministic activation, so nearest-neighbour retrieval affects only the support-based epistemic variance.

J.3 COMPLEXITY

- *Index construction:* one-off $\mathcal{O}(Md)$ time and $\mathcal{O}(M)$ memory.
- *Query:* $\mathcal{O}(\log M)$ approximate neighbour search plus $\mathcal{O}(K^3)$ local linear algebra, with $K \ll M$ fixed.

This design keeps test-time GP inference independent of the total number of cached activations while preserving a principled, distance-aware posterior variance.

J.4 COMPLEXITY SUMMARY

Table 5 summarizes the computational complexity of exact GP conditioning and the local inducing-point approximation used by GAPA. The table reports per-layer costs and suppresses constants associated with vectorized implementation; N_ℓ is the number of cached training pre-activations, M_ℓ the number of inducing points, K the local neighbourhood size, and I_{kmeans} the number of k-means iterations.

Table 5: Computational complexity per layer ℓ. Exact GP conditioning uses all N_ℓ cached activations, while GAPA first compresses them into M_ℓ inducing points and then conditions locally on $K \ll M_\ell$ nearest inducing points.

Operation	**Exact GP**	**GAPA**
Preprocessing	$\mathcal{O}(N_\ell^3)$	$\mathcal{O}(I_{\text{kmeans}} N_\ell M_\ell) + \mathcal{O}(M_\ell \log M_\ell)$
Inference per query	$\mathcal{O}(N_\ell^2)$	$\mathcal{O}(\log M_\ell) + \mathcal{O}(d_\ell K^3)$
Memory	$\mathcal{O}(N_\ell^2)$	$\mathcal{O}(M_\ell)$

K LAPLACE-BRIDGE APPROXIMATION FOR CLASSIFICATION

Given mean logits $\boldsymbol{\mu} \in \mathbb{R}^C$ and per-class variances $\mathbf{v} \in \mathbb{R}^C$ from GAPA propagation, we compute predictive probabilities using:

$$p(y = c \mid \mathbf{x}) \approx \frac{\exp\Big(\mu_c / \sqrt{1 + (\pi/8) v_c}\Big)}{\sum_{c'=1}^{C} \exp\Big(\mu_{c'} / \sqrt{1 + (\pi/8) v_{c'}}\Big)} \tag{10}$$

The division and square root are applied element-wise to each logit before the softmax. This approximation integrates Gaussian logit uncertainty into categorical predictions, derived from the probit approximation $\Phi(x) \approx \sigma(x\sqrt{\pi/8})$ where Φ is the Gaussian CDF and σ is the sigmoid function.

L METRICS

L.1 REGRESSION METRICS

For evaluating performance on regression tasks (Section 3.1), we use several key metrics. First, the **Negative Log-Likelihood (NLL)** measures the quality of the predictive probability distribution. Assuming a Gaussian predictive distribution $p(y|x) = \mathcal{N}(y; \mu(x), \sigma^2(x))$, where $\mu(x)$ is the predicted mean and $\sigma^2(x)$ is the predicted variance, the NLL for a true target value y_{true} is $\frac{1}{2}\log(2\pi\sigma^2(x)) + \frac{(y_{\text{true}} - \mu(x))^2}{2\sigma^2(x)}$. Lower NLL values are better, indicating that the predictive distribution is both accurate and appropriately confident. Second, the **Continuous Ranked Probability Score (CRPS)** (Gneiting & Raftery, 2007) generalizes the Mean Absolute Error (MAE) to probabilistic forecasts. For a predictive cumulative distribution function (CDF) F and a true outcome y_{true}, it is defined as $\text{CRPS}(F, y_{\text{true}}) = \int_{-\infty}^{\infty} (F(y) - \mathbf{1}\{y \geq y_{\text{true}}\})^2 dy$, where $\mathbf{1}\{\cdot\}$ is the indicator function. For a Gaussian predictive distribution $\mathcal{N}(\mu, \sigma^2)$, a closed-form expression exists. Lower CRPS values are better, indicating a sharper and more calibrated predictive distribution. Finally, the **Centered Quantile Metric (CQM)** measures calibration of central predictive intervals. We use the metric of Ortega et al. (2023) with a factor of two to normalize it to $[0, 1]$:

$$\text{CQM} = 2 \int_0^1 |\widehat{c}(\alpha) - \alpha| \, d\alpha.$$

Here $I(x, \alpha) = [F_x^{-1}((1-\alpha)/2), F_x^{-1}((1+\alpha)/2)]$ is the central interval of the predictive CDF F_x, and $\widehat{c}(\alpha)$ is the fraction of test observations y_i falling in $I(x_i, \alpha)$. This normalization is applied to every method; lower CQM indicates better calibration.

L.2 CLASSIFICATION METRICS

For evaluating performance on classification tasks (Section 3.2), we use several key metrics. **Accuracy (ACC)** is the overall proportion of correctly classified samples; we note that GAPA, by design, preserves the mean predictions of the backbone network, so its ACC should match that of the original pre-trained model unless other methods being compared modify these predictions. The **Negative Log-Likelihood (NLL)**, in classification, is equivalent to the cross-entropy loss and measures the quality of the predictive probability distribution. For a given sample with true class label y_{true} (out of C classes) and where the model predicts a probability distribution $p(y|x)$ over the classes, the NLL for that sample is specifically $-\log p(y_{\text{true}}|x)$, which is the negative logarithm of the probability assigned by the model to the correct class; lower values indicate better performance. **Expected Calibration Error (ECE)** measures the discrepancy between a model's predicted confidences and its empirical accuracies. Predictions are typically binned by their confidence scores. For each bin B_m, the accuracy $\text{acc}(B_m)$ and average confidence $\text{conf}(B_m)$ are computed. ECE is then a weighted average of the absolute difference: $\sum_{m=1}^{M} \frac{|B_m|}{N} |\text{acc}(B_m) - \text{conf}(B_m)|$, where N is the total number of samples; lower values indicate better calibration. For **Out-of-Distribution (OOD) Detection**, we report the Area Under the ROC curve (AUC). This evaluates the model's ability to distinguish between in-distribution (ID) and out-of-distribution (OOD) samples based on an uncertainty score. We primarily use the predictive entropy of the softmax distribution as the uncertainty score (denoted

OOD-Entropy or **OOD-AUC**); higher AUC values (closer to 1) indicate better OOD detection performance. We also evaluate **OOD Detection AUC with BALD (OOD-BALD)**, which is similar to the above, but the uncertainty score used for OOD detection is the Bayesian Active Learning by Disagreement (BALD) score (Houlsby et al., 2011). For Bayesian models, BALD measures mutual information between predictions and model parameters. For GAPA, we use the analogous mutual-information-style score induced by the approximate Gaussian logit distribution; this should be interpreted as a support-based epistemic uncertainty score rather than posterior uncertainty over model parameters. A higher AUC indicates better OOD detection using this score.

M Variance Propagation in Transformer Architectures

To implement variance propagation in transformers, in addition to the classical linear layers or activation, we need three additional propagation rules: `RMSNorm`, `CausalSelfAttention` and `Softmax`.

M.1 Attention

Here, we present two variants to propagate the variance through a self-attention layer.

Given an input vector $\mathbf{x} \in \mathbb{R}^d$ with per–feature variances $\mathrm{Var}(x_j) = v_j$, we first form the standard query/key/value projections

$$q = W^Q x, \quad k = W^K x, \quad v = W^V x,$$

with

$$\mathrm{Var}(q_i) = \sum_{j=1}^{d} (W^Q_{ij})^2 \, v_j, \quad \mathrm{Var}(k_i) = \sum_{j=1}^{d} (W^K_{ij})^2 \, v_j, \quad \mathrm{Var}(v_i) = \sum_{j=1}^{d} (W^V_{ij})^2 \, v_j.$$

Variant A. We treat the attention weights a_{ts} as deterministic, and propagate akin to a linear layer propagation:

$$\mathrm{Var}(y_{t,i}) \;=\; \sum_s a_{ts}^2 \; \mathrm{Var}(v_{s,i}).$$

Variant B. Let d_k be the head dimension and define the scaled logits $e_{ts} = d_k^{-1/2} \, q_t^\top k_s$. Under the delta method

$$\mathrm{Var}(a_{ts}) \;=\; \frac{1}{d_k} \sum_{h=1}^{d_k} \Big(q_{t,h}^2 \; \mathrm{Var}(k_{s,h}) + k_{s,h}^2 \; \mathrm{Var}(q_{t,h}) + \mathrm{Var}(q_{t,h}) \; \mathrm{Var}(k_{s,h}) \Big).$$

After masking and applying the soft-max propagation rule of Appendix M.3 we obtain $\mathrm{Var}(a_{ts})$. The variance of the head output is then

$$\mathrm{Var}(y_{t,i}) \;=\; \sum_s \Big[\mathrm{Var}(a_{ts}) \, v_{s,i}^2 + a_{ts}^2 \; \mathrm{Var}(v_{s,i}) + \mathrm{Var}(a_{ts}) \; \mathrm{Var}(v_{s,i}) \Big].$$

While the second method is arguably modeling the overall variance propagation in a more sophisticated way, from a design choice perspective, the decision is not obvious: the first propagation scheme is much faster. Although we weren't directly able to use flash attention, in theory a `FlashAttention` kernel could be modded to calculate the squared attention operation on-the-fly at no additional cost. Secondly we found that the variances can grow quickly the more layer the transformer model has because of the compounding, multiplicative effect of the variance over both the attention scores and the query, key and values. This compounding effect could be addressed with full covariance propagation, however, for transformers the embedding space is large (e.g. for LLaMA-3.2-3B it is 3072), making a full covariance treatment computationally intractable (and low-rank approximations either still exhibit compounding effect (if too low-rank) or again intractable). Based on these reasons in the paper use Variant A: we will assume the model knows where to look (deterministic attention weights), but is uncertain about what it sees there (value variance).

M.2 RMSNorm

Let $\mathbf{x} \in \mathbb{R}^d$ with per-feature variances $\mathrm{Var}(x_j) = v_j$. RMSNorm computes the root mean square

$$\mathrm{RMS}(\mathbf{x}) = \sqrt{\frac{1}{d}\sum_{j=1}^{d} x_j^2 + \varepsilon},$$

and applies the transformation

$$y_i = \gamma_i \cdot \frac{x_i}{\mathrm{RMS}(\mathbf{x})},$$

where γ_i are learned scale parameters and $\varepsilon > 0$ is a small constant for numerical stability.

As a first-order approximation we define the expected RMS squared as

$$s^2 = \mathbb{E}\Big[\frac{1}{d}\sum_{j=1}^{d} x_j^2\Big] + \varepsilon.$$

Using the identity $\mathbb{E}[x_j^2] = \mathrm{Var}(x_j) + \mathbb{E}[x_j]^2$, we can rewrite this as

$$\begin{aligned} s^2 &= \frac{1}{d}\sum_{j=1}^{d} \mathbb{E}[x_j^2] + \varepsilon \\ &= \frac{1}{d}\sum_{j=1}^{d} \big(v_j + \mathbb{E}[x_j]^2\big) + \varepsilon. \end{aligned}$$

Under this approximation, we treat s^2 as deterministic and propagate variance as

$$\mathrm{Var}(y_i) \approx \frac{\gamma_i^2}{s^2}\, v_i.$$

The following PyTorch implementation realizes the simplified scheme:

```python
class RMSNormVar(torch.nn.Module):
    def __init__(self, dim: int, eps: float = 1e-6):
        super().__init__()
        self.eps = eps
        self.weight = nn.Parameter(torch.ones(dim))

    def _norm(self, x):
        return x * torch.rsqrt(x.pow(2).mean(-1, keepdim=True) + self.eps)

    def forward(self, input_mean, input_var):
        """
        Args:
            input_mean (torch.Tensor): Input means
            input_var (torch.Tensor): Input variances

        Returns:
            tuple: (output_mean, output_var)
        """

        output_mean = self._norm(input_mean) * self.weight

        # Compute expected rms squared
        expected_rms_squared = (
```

```
            input_mean.pow(2).mean(dim=-1, keepdim=True)
            + input_var.mean(dim=-1, keepdim=True)
            + self.eps
        )

        # Compute output variances
        output_var = input_var / expected_rms_squared * self.weight.pow(2)

        return output_mean, output_var
```

M.3 SOFTMAX

For softmax we follow the Delta method approach. We note that this method is only used for the second variant of `SelfAttention`, whereas in this paper we use the first variant.

Let $\mathbf{x} \in \mathbb{R}^K$ with per–feature variances $\mathrm{Var}(x_i) = v_i$. The softmax output is

$$s_k = \frac{e^{x_k}}{\sum_{j=1}^{K} e^{x_j}}.$$

The Jacobian of the softmax for fixed k is

$$\frac{\partial s_k}{\partial x_i} = s_k \left(\delta_{ik} - s_i\right).$$

Applying the Delta method with $\Sigma_x = \mathrm{diag}(v_1, \ldots, v_K)$ gives

$$\mathrm{Var}(s_k) \;=\; \sum_{i=1}^{K} \Big(s_k(\delta_{ik} - s_i) \Big)^2 \, v_i.$$

If we split out the $i = k$ term and the $i \neq k$ terms, this expands to

$$\begin{aligned} \mathrm{Var}(s_k) &= s_k^2 \left(1 - s_k\right)^2 v_k \;+\; \sum_{i \neq k} s_k^2 \, s_i^2 \, v_i \\ &= s_k^2 \Big[\left(1 - s_k\right)^2 v_k + \sum_{i \neq k} s_i^2 \, v_i \Big]. \end{aligned}$$

```
def softmax_var(y_mean, x_var, axis=-1):
    y = y_mean.transpose(axis, -1)
    v = x_var.transpose(axis, -1)
    W = y.pow(2) * v
    S = W.sum(dim=-1, keepdim=True)
    sum_excluding_k = S - W
    diag_term = (1 - y).pow(2) * v
    var_last = y.pow(2) * (diag_term + sum_excluding_k)
    return var_last.transpose(-1, axis)
```

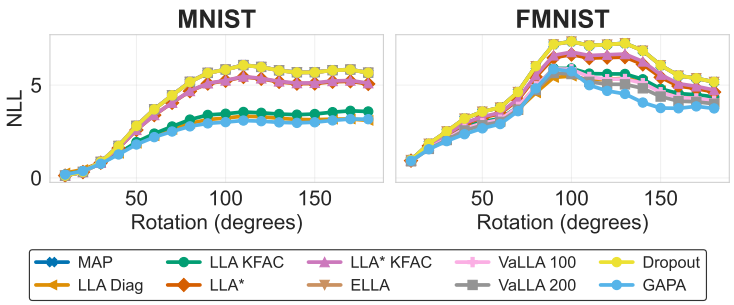


Figure 8: Predictive NLL under rotation corruption for MNIST and FMNIST; lower is better.

N Random Seeds and Rotation Results

For GAPA and post-hoc methods applied to frozen backbones, repeated runs use the same pretrained checkpoint for each dataset and architecture; random seeds vary the stochastic components of the post-hoc procedure. The resulting standard deviations describe variability conditional on that checkpoint, rather than variability across independently trained backbones.

O Ablation Studies

We assess covariance and propagation approximations, reference-cache coverage, layer placement, inducing-set size and selection, and local neighbourhood size.

O.1 Full Covariance and Monte Carlo Propagation

We compare the practical diagonal, single-pass implementation with a reference implementation using full GP covariance and Monte Carlo propagation through the frozen network. The experiment uses an MLP with three linear layers and two $\tanh$ nonlinearities. The reference implementation propagates $1{,}000$ samples through the network; runtimes are measured over approximately $20{,}000$ inputs.

Table 6: Effect of covariance and propagation approximations in the MLP experiment. The Monte Carlo reference uses $1{,}000$ samples; runtimes cover approximately $20{,}000$ inputs. Best AUROCs and the lowest runtime are bold.

Method	**MNIST → FMNIST** AUROC ↑	**FMNIST → MNIST** AUROC ↑	**Runtime** seconds ↓
GAPA (diagonal, single pass)	0.981	0.996	**2.8**
Full covariance + Monte Carlo	**0.983**	**0.999**	17,531

Table 6 shows that the reference implementation improves AUROC by 0.002 and 0.003, respectively, while requiring approximately $6{,}000$ times longer inference time. These results support the practical approximations in this setting: the reduction in computational cost is substantial, while the difference in OOD detection performance is small. This comparison evaluates the covariance and propagation approximations jointly; it does not isolate their individual effects.

O.2 Sensitivity to Reference Cache Coverage

We first reduce the available reference cache while keeping the backbone frozen. Retaining only 1% of cached activations reduces support-score AUROC from 0.991 to 0.968 on MNIST and from 0.996 to 0.985 on FMNIST. At the same time, mean in-distribution variance on MNIST increases from 1.29 to 5.08. Retaining 10% gives intermediate AUROCs of 0.984 and 0.993, respectively, and mean MNIST in-distribution variance of 2.43. Thus, reduced reference support produces higher uncertainty,

Table 7: Comparison of metrics at different GAPA layer placements ($M = 55{,}000$). Best values are **bold**. Lower is better (↓) for NLL; higher is better (↑) for OOD-AUC/BALD.

GAPA layers	MNIST			FMNIST		
	NLL ↓	OOD-AUC ↑	OOD BALD ↑	NLL ↓	OOD-AUC ↑	OOD BALD ↑
[1]	0.072	0.915	0.916	0.326	0.870	0.900
[2]	0.070	0.921	0.920	0.321	0.884	0.907
[3]	**0.068**	0.933	0.933	**0.309**	0.901	0.911
[4]	0.117	0.951	0.957	0.353	0.973	**0.969**
[1, 2]	0.069	0.923	0.922	0.318	0.901	0.921
[1, 3]	0.069	0.934	0.934	**0.309**	0.912	0.924
[1, 4]	0.120	**0.953**	**0.961**	0.357	0.973	**0.969**
[2, 3]	0.070	0.935	0.935	0.310	0.917	0.928
[2, 4]	0.122	**0.953**	**0.961**	0.360	0.973	0.968
[3, 4]	0.134	**0.953**	**0.961**	0.372	0.973	0.966
[1, 2, 3]	0.072	0.936	0.935	0.312	0.924	0.934
[1, 2, 4]	0.125	**0.953**	0.960	0.364	0.973	0.968
[1, 3, 4]	0.137	**0.953**	0.960	0.376	0.973	0.966
[2, 3, 4]	0.139	**0.953**	0.960	0.380	0.973	0.966
[1, 2, 3, 4]	0.142	**0.953**	0.960	0.384	**0.974**	0.966

with a gradual decline in detection performance. This experiment complements the inducing-point ablations by varying the available reference data itself.

We then remove every cached activation belonging to one class, repeating the experiment for each of the ten classes. Using variance alone, inputs from the omitted class are identified with AUROC 0.996 on MNIST and 0.85 on FMNIST. These results show that GAPA responds to gaps in reference coverage and support its interpretation as an uncertainty signal tied to the chosen activation cache.

O.3 Where to put GAPA

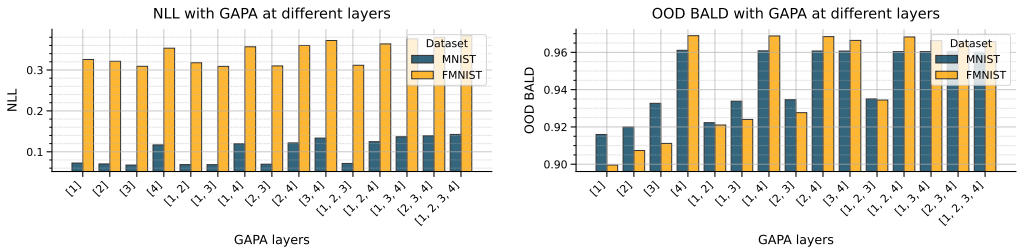


Figure 9: Comparison of metrics at different GAPA layer placements ($M = 55{,}000$)

Table 7 (and Figure 9) examines GAPA placement across the 4-layer MLP described in Section 3.2. For MNIST, placing GAPA at layer 3 achieves the best NLL (0.068), while layer 4 or any combination including layer 4 maximizes OOD detection (0.953 AUC, 0.961 BALD). For FMNIST, layer 3 minimizes NLL (0.309), while using all four GAPA layers achieves the highest OOD-AUC (0.974). Layer 4 alone achieves a nearly identical OOD-AUC (0.973) and the joint-best OOD-BALD (0.969). Interestingly, adding more GAPA layers generally degrades NLL while maintaining strong OOD performance, suggesting a trade-off between calibration and uncertainty awareness. The final layer (closest to output) appears most critical for OOD detection, while intermediate layers better preserve calibration.

O.4 Number of inducing inputs

Table 8 shows performance as M increases from 10 to 55,000. Both datasets exhibit clear saturation: MNIST plateaus around $M = 40,000$ (NLL: 0.119→0.117, OOD: 0.953), while FMNIST shows

Table 8: Metrics across different M values for MNIST and FMNIST, GAPA at the 4th layer.

	MNIST					FMNIST				
M	NLL ↓	OOD ↑	BALD ↑	set up/s ↓	inference/s ↓	NLL ↓	OOD ↑	BALD ↑	set up/s ↓	inference/s ↓
10	0.248	0.897	0.919	**2.733**	**7.517**	0.489	0.957	0.936	**0.257**	**7.584**
100	0.248	0.897	0.919	185.477	7.478	0.489	0.957	0.936	181.340	7.625
1000	0.246	0.898	0.920	184.787	7.674	0.486	0.957	0.937	183.503	7.763
5000	0.219	0.913	0.934	195.889	8.663	0.470	0.960	0.943	194.468	8.702
10000	0.181	0.933	0.950	212.990	10.119	0.442	0.964	0.952	211.333	9.873
20000	0.139	0.947	0.958	247.684	12.498	0.390	0.970	0.964	241.000	12.164
40000	0.119	**0.953**	**0.961**	301.511	16.926	0.355	0.972	0.968	301.086	16.826
55000	**0.117**	**0.953**	**0.961**	455.735	20.445	**0.353**	**0.973**	**0.969**	384.825	20.527

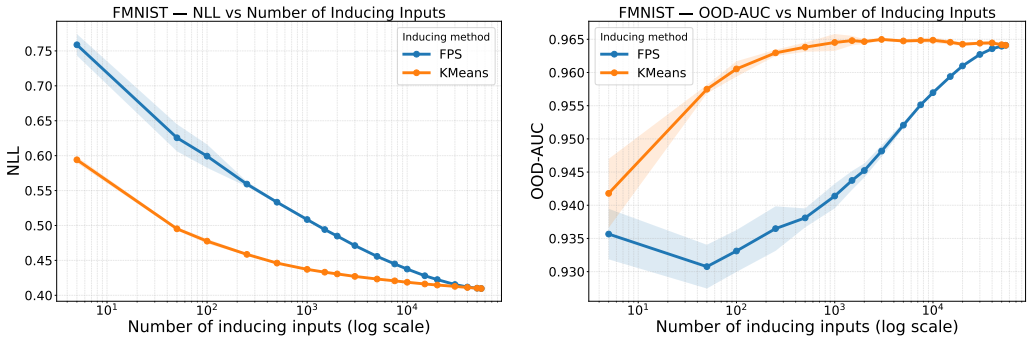


Figure 10: FMNIST: NLL (left) and OOD-AUC (right) for KMeans vs. FPS across M.

similar convergence. Computational costs scale sub-linearly due to FAISS indexing—setup time increases from 2.7s to 455s for MNIST, while inference remains tractable (7.5s→20s). This demonstrates GAPA's efficiency: near-optimal uncertainty quantification is achievable with moderate M values, making the method practical for larger models.

O.5 Inducing point selection: KMeans vs. farthest-point sampling

We compare the KMeans strategy used in the main experiments with farthest-point sampling (FPS) and random sampling alternatives. Figures 10–13 report results for MNIST and FMNIST across a range of inducing-point budgets M.

Overall, both methods exhibit similar behaviour: performance improves monotonically with M and saturates once a sufficient coverage of the activation space is achieved. KMeans, however, provides a more efficient trade-off between coverage and inducing-point count, reaching its plateau at substantially smaller M values than FPS. This makes KMeans a practical alternative when memory, storage, or index construction time is a constraint.

O.6 Random vs. Farthest-Point Sampling

Table 9 reveals that farthest-point sampling (FPS) and random sampling exhibit different strengths. At smaller M (5K-10K), random sampling achieves better NLL and OOD detection, likely because FPS's greedy selection may overfit to specific activation patterns. However, as M increases to 40K, FPS shows marginal improvements, suggesting its structured coverage becomes beneficial with sufficient inducing points. The convergence of both methods at large M indicates that with enough inducing points, the activation space is well-covered regardless of sampling strategy.

O.7 KNN Sweep: $K = 1$ to 500

To evaluate the robustness of the KNN GAPA approximation used in GAPA, we performed a comprehensive KNN sweep over $K = \{1, 2, 3, 5, 10, 20, 50, 100, 150, 200, 300, 400, 500\}$ on both

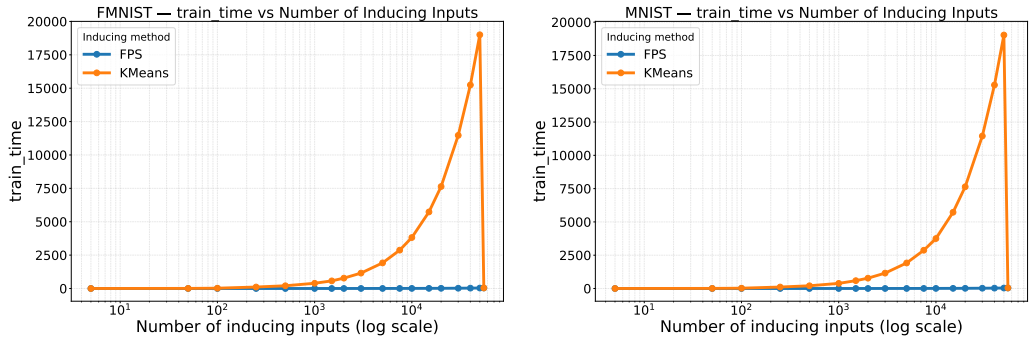


Figure 11: Setup time (FAISS indexing) for KMeans vs. FPS on FMNIST (left) and MNIST (right).

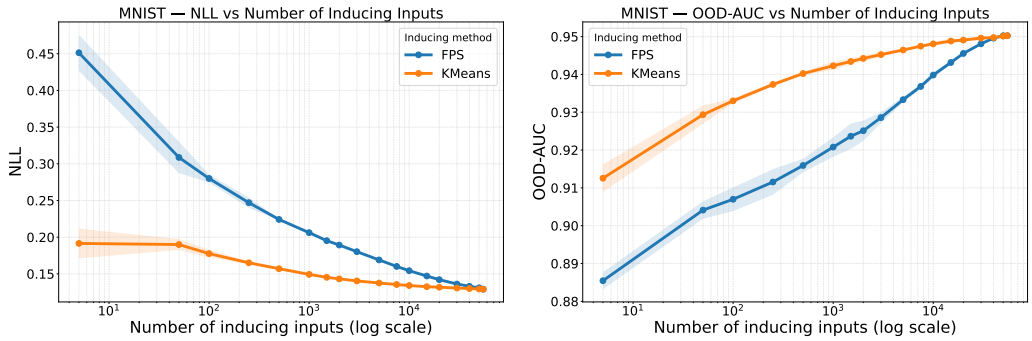


Figure 12: MNIST: NLL (left) and OOD-AUC (right) for KMeans vs. FPS across M.

MNIST and FMNIST. For each K, we recomputed the GP posterior variance using the K nearest cached activations and measured all uncertainty metrics (NLL, ECE, OOD-AUC, OOD-BALD) as well as test-time inference cost.

Across datasets, increasing the local neighbourhood size K improves calibration and NLL, while OOD metrics quickly saturate and can slightly degrade for very large K. Thus, moderate neighbourhood sizes provide the best accuracy–cost trade-off.

Negative Log-Likelihood (NLL). NLL decreases continuously as K increases for both datasets. MNIST improves from ≈ 0.092 at $K{=}1$ to ≈ 0.081 at $K{=}500$, while FMNIST improves from ≈ 0.408 to ≈ 0.390.

Expected Calibration Error (ECE). ECE also improves smoothly with K. MNIST decreases from ≈ 0.062 to ≈ 0.015, and FMNIST shows a similar trend.

OOD-AUC. OOD detection improves with K before saturating. MNIST increases from 0.950 at $K{=}1$ to 0.963 at $K{=}500$. FMNIST improves up to $K \approx 50$, after which performance plateaus or slightly decreases, suggesting mild over-smoothing for very large neighbourhoods.

OOD-BALD. BALD-based OOD detection improves steadily across the sweep, indicating that larger neighbourhoods provide smoother and more informative epistemic estimates.

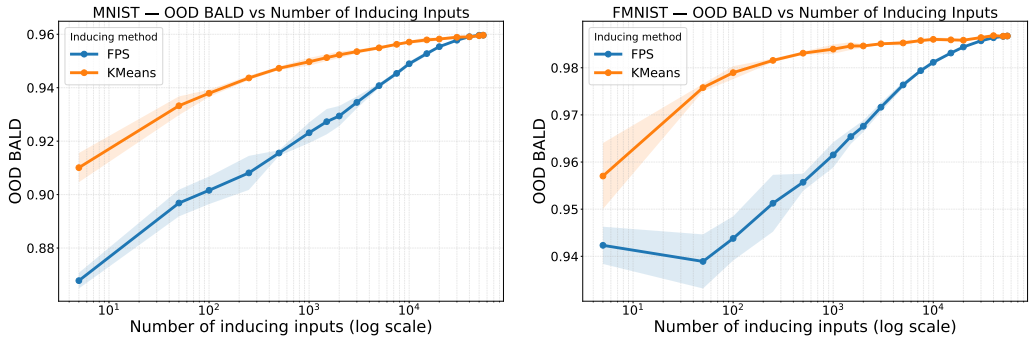


Figure 13: BALD-based OOD detection for MNIST (left) and FMNIST (right).

Table 9: Comparison of NLL and OOD BALD for FPS and three random baselines (FMNIST, gapa_index= last layer).

M	FPS NLL↓	FPS OOD↑	Rand1 NLL↓	Rand1 OOD↑	Rand2 NLL↓	Rand2 OOD↑	Rand3 NLL↓	Rand3 OOD↑
5000	0.470	0.943	0.394	0.957	0.394	0.957	0.394	0.957
10000	0.442	0.952	0.380	0.960	0.380	0.960	0.380	0.960
20000	0.390	0.964	0.369	0.964	0.369	0.964	0.369	0.964
40000	**0.355**	**0.968**	**0.359**	**0.967**	**0.359**	**0.967**	**0.359**	**0.967**

Test-time cost. Inference time increases approximately linearly with K. For MNIST, test time grows from $\approx$ 2.1ms to $\approx$ 16ms, with FMNIST following the same scaling pattern.

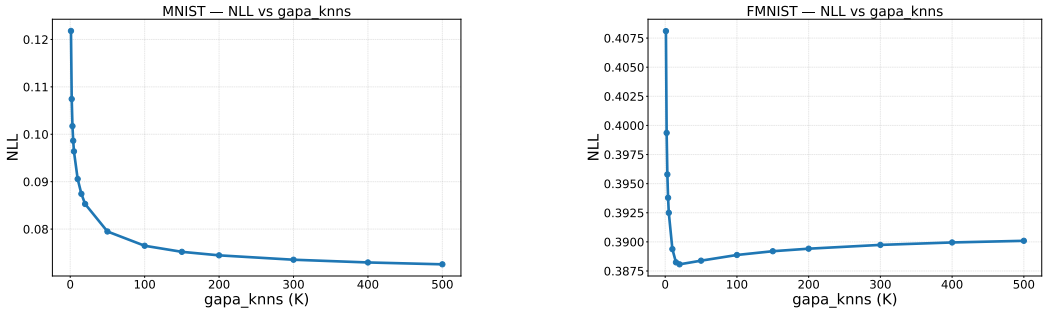
Takeaway. These experiments show that 1-NN is already stable and competitive, especially for OOD detection. Increasing K to roughly 20–50 provides clear gains in calibration and NLL, while very large K gives diminishing returns and increases compute. Overall, the sweep confirms that the local KNN approximation used by GAPA is robust and behaves predictably across a wide range of K.

P Extended Related Work

Uncertainty quantification (UQ) in deep learning differs primarily in *where* uncertainty is placed (e.g., weights, outputs, or representations) and in the resulting training and test-time costs. We focus on the *post-hoc* regime for *frozen pretrained backbones*, where retraining, multiple test-time samples, or full-network second-order computation can be infeasible.

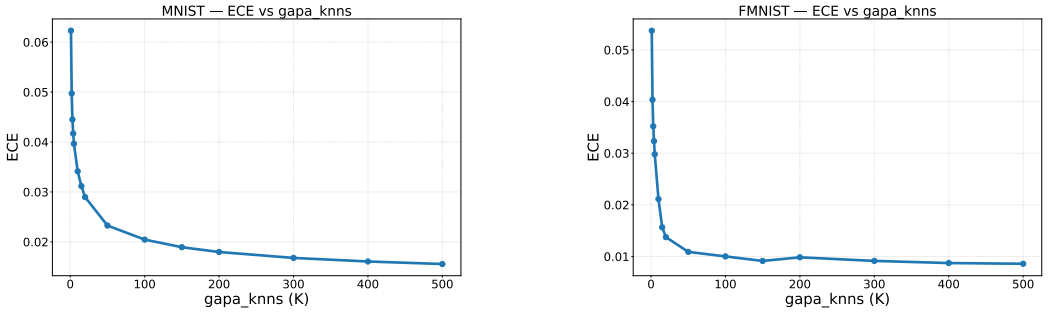
Weight-space Bayesianization and Laplace. Weight-space Bayesian methods model uncertainty directly over parameters, typically requiring either approximate inference during training or posterior sampling at test time. Laplace approximations instead fit a local Gaussian posterior around a trained solution using curvature information, and recent work has revisited Laplace as a competitive and practical Bayesian deep learning baseline, including analyses of when structured approximations are necessary for scalability and fidelity (Blundell et al., 2015; MacKay, 1992; Daxberger et al., 2021; Ortega et al., 2023; Deng et al., 2022). Closely related, stochastic weight-space approximations such as SWAG provide inexpensive posterior samples from SGD trajectories and often serve as strong uncertainty baselines without changing the underlying architecture (Maddox et al., 2019).

Last-layer and single-pass post-hoc methods. A popular compromise Bayesianizes only the final layer(s) while keeping the feature extractor frozen, yielding post-hoc uncertainty at substantially reduced cost. Beyond Laplace-style last-layer treatments, deterministic variational formulations such as variational Bayesian last layers (VBLL) aim to deliver *single-pass* predictive uncertainty for frozen-backbone models (Harrison et al., 2024). These approaches are strong deployment-friendly baselines, but they concentrate Bayesian modeling at the head and do not in general propagate mean-preserving uncertainty through intermediate computations of the frozen network.



(a) MNIST (b) FMNIST

Figure 14: NLL as a function of local neighbourhood size K.



(a) MNIST (b) FMNIST

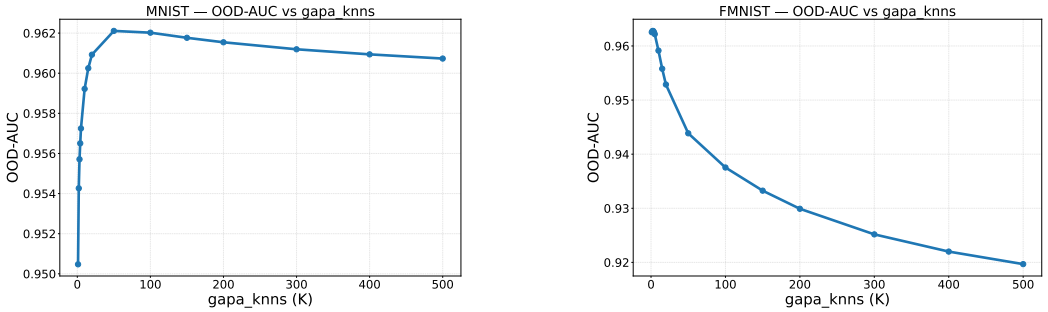
Figure 15: ECE as a function of local neighbourhood size K.

Sampling-based baselines. Deep ensembles approximate epistemic uncertainty through multiple independently trained models (Lakshminarayanan et al., 2017), while MC Dropout relies on multiple stochastic forward passes at inference (Gal & Ghahramani, 2016). These are effective but often incompatible with strict single-pass deployment constraints.

Representation-based and calibration baselines. Distance-/density-aware approaches estimate uncertainty from representations, e.g., spectral-normalized GP-style heads (SNGP) (Liu et al., 2020) or density-based uncertainty on deep features (DDU) (Mukhoti et al., 2023). Calibration-only post-processing such as temperature scaling can improve confidence calibration but does not model epistemic uncertainty (Guo et al., 2017).

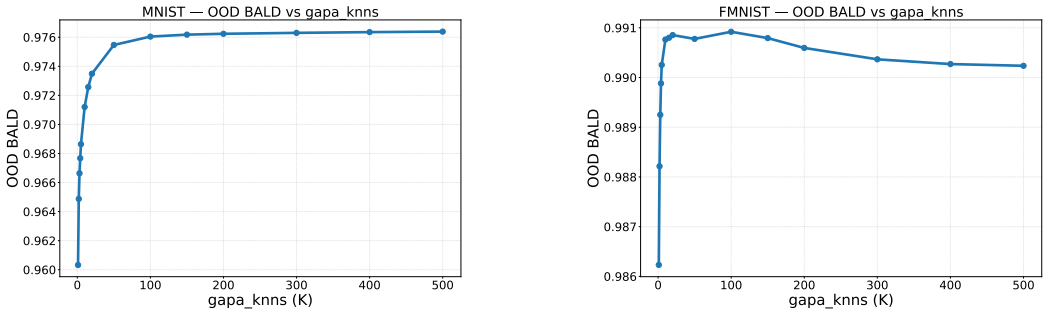
Gaussian processes and function-space views. Gaussian processes (GPs) provide a classical function-space approach to uncertainty and connect naturally to infinite-width neural networks: fully-connected nets converge to an NNGP prior (Lee et al., 2017), and their infinite-width training dynamics are characterized by the NTK (Jacot et al., 2018). For scalability, GP/kernel Inference is commonly accelerated via low-rank approximations such as inducing points (Titsias, 2009) and Nyström methods (Williams & Seeger, 2000), while local GP approximations and noisy-input corrections (e.g., NIGP) provide efficient variance adjustments under locality or input uncertainty (Gramacy & Apley, 2015; McHutchon & Rasmussen, 2011). GAPA builds on these function-space ideas but applies them *inside* frozen networks by placing uncertainty in activation space, preserving the pretrained mean and enabling deterministic single-pass inference.

Summary. Overall, existing approaches trade off between (i) retraining or multi-sample inference, (ii) post-hoc last-layer approximations that concentrate uncertainty at the head, or (iii) representation-based proxies. These gaps motivate mean-preserving, post-hoc activation-space uncertainty with deterministic single-pass inference.



(a) MNIST (b) FMNIST

Figure 16: OOD-AUC as a function of local neighbourhood size K.



(a) MNIST (b) FMNIST

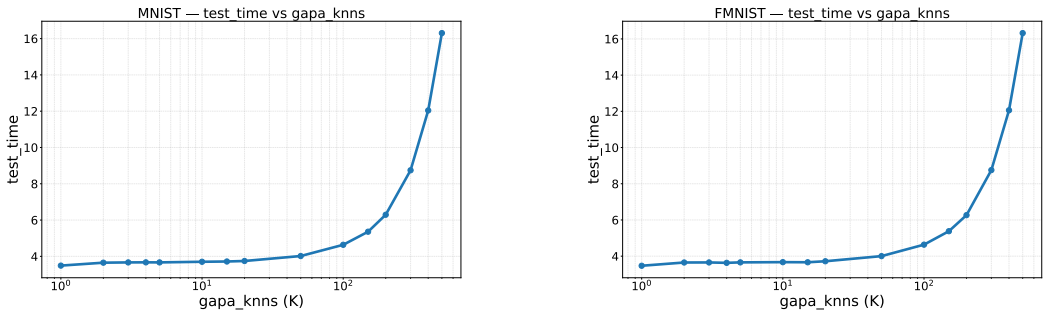
Figure 17: OOD-BALD as a function of local neighbourhood size K.

Q Assets and Licenses

Table 10: External assets used in the paper. Licenses and terms are reported to the best of our knowledge from the official dataset, model, or software sources.

Asset	Use	License / terms
MNIST	Classification benchmark	Public research benchmark; original source cited
Fashion-MNIST	Classification benchmark	MIT license
CIFAR-10	Classification benchmark	Public research benchmark; original source cited
SVHN	OOD benchmark	Public research benchmark; original source cited
Oxford-IIIT Pet	Segmentation benchmark	Creative Commons Attribution-ShareAlike 4.0 International
WikiText-103	LLM reference data	CC BY-SA 3.0 / GFDL
OpenWebText	LLM OOD/reference comparison	CC0 packaging; underlying web text not owned by dataset creators
YEAR Prediction MSD	Regression benchmark	UCI / Million Song Dataset terms; original source cited
Airline	Regression benchmark	Public benchmark used in prior work; original source cited
Taxi	Regression benchmark	Public benchmark used in prior work; original source cited
LLaMA-3.2-3B	Pretrained LLM benchmark	Meta LLaMA 3.2 Community License
Pretrained ResNets	Image classification backbones	Original model/checkpoint source cited
FAISS	Nearest-neighbour retrieval	MIT license
PyTorch	Implementation framework	BSD-style license

We use these assets only for research evaluation and cite the original sources in the main paper or references. We do not redistribute pretrained model weights, datasets, or benchmark assets as part of this submission. For LLaMA-3.2-3B, we use the model only as an evaluation backbone and follow the applicable Meta LLaMA license terms.



(a) MNIST (b) FMNIST

Figure 18: Test-time cost as a function of local neighbourhood size K.

R Compute Resources

All reported experiments were run on GPU workstations or institutional HPC cluster nodes equipped with NVIDIA A100 GPUs. We report wall-clock train/setup and test times in the main experimental tables where relevant. The most computationally demanding experiments are the LLaMA activation-cache and inducing-point experiments, which require storing millions of cached pre-activations and running nearest-neighbour retrieval over the inducing set. We did not include preliminary failed runs in the reported compute budget.