Learning Structured Gaussians to Approximate Deep Ensembles

Ivor J.A. Simpson
University of Sussex, UK
i.simpson@sussex.ac.uk

Sara Vicente
Niantic, UK
svicente@nianticlabs.com

Neill D.F. Campbell
University of Bath, UK
n.campbell@bath.ac.uk

Abstract

This paper proposes using a sparse-structured multivariate Gaussian to provide a closed-form approximator for the output of probabilistic ensemble models used for dense image prediction tasks. This is achieved through a convolutional neural network that predicts the mean and covariance of the distribution, where the inverse covariance is parameterised by a sparsely structured Cholesky matrix. Similarly to distillation approaches, our single network is trained to maximise the probability of samples from pre-trained probabilistic models, in this work we use a fixed ensemble of networks. Once trained, our compact representation can be used to efficiently draw spatially correlated samples from the approximated output distribution. Importantly, this approach captures the uncertainty and structured correlations in the predictions explicitly in a formal distribution, rather than implicitly through sampling alone. This allows direct introspection of the model, enabling visualisation of the learned structure. Moreover, this formulation provides two further benefits: estimation of a sample probability, and the introduction of arbitrary spatial conditioning at test time. We demonstrate the merits of our approach on monocular depth estimation and show that the advantages of our approach are obtained with comparable quantitative performance.

1. Introduction

Single prediction neural networks are ubiquitous in computer vision and have demonstrated extensive capability for a variety of tasks. However, researchers are increasingly interested in capturing the uncertainty in estimation tasks to combat over-confidence and ambiguity; such concerns are important when building robust systems that connect computer vision approaches to down-stream applications. The deployment of neural networks for safety-critical tasks, such as autonomous driving, requires an accurate measure of uncertainty. While Bayesian Neural Networks [17] are often a model of choice for uncertainty estimation, ensembles [14] have been proposed as a simple alternative. Empirically, ensembles have been shown to produce good measures of uncertainty for vision tasks [15, 19] and allow practitioners to exploit associated application-specific inductive biases, for example established architectures, directly.

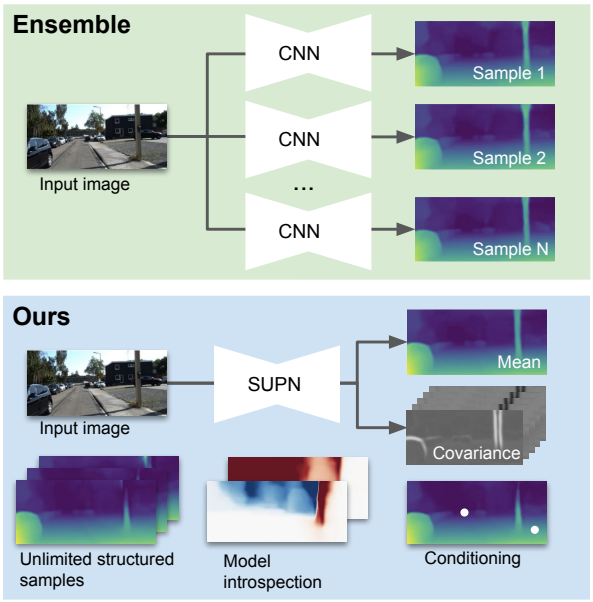


Figure 1. Our method is trained to approximate the output of an ensemble, by using structured uncertainty prediction networks (SUPN) to predict a mean and covariance for a multivariate Gaussian distribution. This explicit distribution enables a variety of tasks including: sampling, conditioning and model introspection.

Limitations of implicit approaches Despite their popularity, ensembles have a number of drawbacks that we group into three themes. Firstly, they come at an increased cost compared with deterministic networks. At training time, they require training multiple deep models, while at test time, multiple inference passes are required. MC-dropout [5] saves computation at training time, but it still requires multiple passes at inference time. Secondly, these approaches only provide an implicit distribution over probable model outputs. Any uncertainty captured is only accessible through ancestral sampling. Accordingly, one cannot

calculate conditional samples, or assess the likelihood of a new sample given the learned model. Finally, and of increasing importance to the community, introspection of the trained models is very difficult.

When combining computer vision with larger systems, there is virtue to summarising the posterior distribution in a formal and compact form that can be visualised and appropriately used to inform downstream tasks. The computational challenges have prompted work on producing a single model to approximate the output of an ensemble; so called "ensemble distillation" [2, 15, 18, 21].

Lack of structure in distillation methods Previous methods focus on: classification problems [15, 18], approximating only the mean of the ensemble [2], or modelling independent per-pixel variance [21]. In contrast, while we also adopt a single model to reduce the computational cost, we propose to learn a model that approximates the ensemble by formally capturing structure in the output space; this is more appropriate for dense prediction tasks. When making per-pixel predictions, it is common to use models that capture spatial correlation in the output space. In particular, models such as Markov or Conditional Random Fields [20], which capture correlations between neighbouring pixels, have been extensively used in computer vision. However, capturing the structure of the output space is less explored in the context of modelling uncertainty. Previous work has focused on per-pixel heteroscedastic uncertainty, by using a Gaussian [11, 21] or Laplace [13] likelihood model with diagonal covariance. Since these models do not capture correlations between pixels, samples suffer from salt-and-pepper (independent) noise.

Capturing structure explicitly Previously, adopting per-pixel uncertainty representations usually follows from the expectation that direct estimation of a full covariance structure is intractable in both storage $\mathcal{O}(\text{pixels}^2)$ and computation $\mathcal{O}(\text{pixels}^3)$. Recently, however, Dorta *et al*. [4] introduced Structured Uncertainty Pprediction Networks (SUPN) for generative models. The paper extended a Variational Auto Encoder (VAE) [12] with a likelihood model that is Gaussian with a full covariance matrix. The authors show how this can be predicted efficiently by using a sparse approximation of the Cholesky decomposition of the *precision* matrix. Working in the domain of the precision allows a dense covariance structure to be obtained whilst also respecting our prior that long range structure is derived from the propagation of local image statistics. By encoding a full covariance matrix, the samples obtained from such a model capture these long range correlations in the image domain and are free from salt-and-pepper (independent) noise.

Contributions In this work, we build on SUPN [4] and show that a deep network can be trained in a regression setting to predict a structured Gaussian distribution that approximates the output distribution of methods that capture model uncertainty, such as ensembles [14] and MC-dropout [5]. We introduce a novel efficient approach to drawing conditioned or unconditioned samples from a structured multivariate Gaussian distribution with a sparsely structured precision matrix. By taking full advantage of the closed form nature of the Gaussian distribution, our method allows introspection and enables conditioning at test time, which proves cumbersome for other methods. Importantly, our approach is not limited to Gaussian likelihoods over the prediction space (see § 3.4).

Evaluation We demonstrate the efficacy of our method for the task of depth estimation. Experiments show that the new advantages can be obtained without sacrificing quantitative performance, with results comparable to the original ensemble; we consider metrics over both accuracy and the capture of uncertainty. The samples are found to follow the ensembles without being limited in the number that can be drawn. The compact representation is capable of encoding a rich distribution with only a modest increase in computation over a single deterministic network. Furthermore, we demonstrate using our explicit representation to perform conditional sampling and illustrate the ability to inspect the model and visualise the correlation structure learned.

2. Background

Our goal is to model $p(\mathbf{d} \,|\, \mathbf{x})$, where $\mathbf{x}$ is the observed image and $\mathbf{d}$ is a per-pixel prediction, *e.g*. a semantic labelling or depth map. While most deterministic deep models can be seen as capturing the mean $\boldsymbol{\mu}(\mathbf{x})$ of this distribution, we are interested in models that also capture the variance $\boldsymbol{\Sigma}(\mathbf{x})$.

2.1. Uncertainty in Deep Models

Previous work for probabilistic modelling using neural networks, can be broadly grouped into three categories: (1) Bayesian approaches that model uncertainty of the network parameters, (2) methods that empirically approximate Bayesian approaches by predicting multiple hypothesis, and (3) approaches that model $p(\mathbf{d} \,|\, \mathbf{x})$ directly by predicting a parametric distribution. The literature on uncertainty modelling in neural networks is vast and we direct the interested reader to a recent review [1].

Modelling uncertainty in the parameters Bayesian neural networks [17] model uncertainty by modelling the probability distribution of the learned weights $\mathbf{w}$ of the network. The resulting posterior $p(\mathbf{d} \,|\, \mathbf{x})$ is then obtained by marginalising over the weights:

$$p(\mathbf{d} \,|\, \mathbf{x}, \mathcal{D}) = \int p(\mathbf{d} \,|\, \mathbf{x}, \mathbf{w})\, p(\mathbf{w} \,|\, \mathcal{D})\, d\mathbf{w}\,, \qquad (1)$$

where $\mathbf{w}$ are the model parameters, and we make explicit the dependence on the dataset $\mathcal{D}$.

While this approach is able to model arbitrary distributions $p(\mathbf{d}|\mathbf{x})$, and generate samples which are correlated in output space, it also suffers some limitations. The majority of approaches rely on mean-field approximations over the weights to maintain tractability. In addition, it is difficult to condition on any of the output values due to the absence of a parametric distribution over the posterior.

MC-dropout [5] approximates Bayesian networks by using dropout at both training and test time. Dropout was first proposed to reduce over-fitting in deep neural networks [22] and it proceeds by randomly setting some of the weights of the network to zero. It has been shown, [5], that this random dropping of weights at test time is akin to sampling from the distribution $p(\mathbf{w} \mid \mathcal{D})$ and may be used to approximate the integral in (1).

Multiple hypothesis Ensemble methods make use of multiple models and combine them to get a single prediction. Deep ensembles can be trained using bootstraping [14], *i.e.* splitting the training set into multiple random subsets and training each model in the ensemble independently. Alternatively, to save computation, a deep ensemble can be trained by taking multiple snapshots [9] from the same training procedure, requiring a cyclic learning rate. Ensembles have been shown to provide good measures of uncertainty [14]. They can be seen as approximating Bayesian networks by replacing the integral in (1) by a sum over a discrete number of models. As discussed in § 1, training and inference procedures can become expensive in terms of maintaining an increasing number of networks; practical approaches are often limited in the number of distinct models that, in turn, restricts the number test-time samples.

Predictive uncertainty via parametric distributions The other alternative to modelling uncertainty is to use a feed-forward neural network to predict the parameters of a parametric distribution [11]. For regression tasks, $p(\mathbf{d}|\mathbf{x})$ is typically described by a Gaussian likelihood, where the mean and variance are outputs of the neural network:

$$p(\mathbf{d} \mid \mathbf{x}) \sim \mathcal{N}\big(\mathbf{d} \mid \boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\Sigma}(\mathbf{x})\big)\,, \tag{2}$$

where $\boldsymbol{\Sigma}(\mathbf{x})$ is usually approximated by a diagonal matrix where the diagonal elements are predicted by the network. Kendall and Gal [11] discuss how the predicted variance can be seen as a loss attenuation factor, reducing the loss for outliers; this predicted per-pixel variance is shown to correlate with error in the predictions. Evaluating predictive uncertainty is more efficient since a single pass of the network at test time is sufficient to fully determine the measure of uncertainty. However, independent per-pixel uncertainty estimates fail to capture spatial correlation that is known to exist in images; samples from these models are destined to be unrealistic and suffer from salt-and-pepper noise.

Ensemble distillation Recently, there has been a growing interest in approximating the probabilistic output of an ensemble by a single model [2, 15, 18, 21]. This process is commonly named "distillation". Most of the focus has been on classification [15, 18], where the goal is to predict the class of the image. While these methods show impressive results in detecting out-of-distribution images, they are not easily extended for dense prediction tasks. Other methods focus on approximating only the mean of the ensemble distribution [2], or modelling independent per-pixel variance [21]. In contrast, our model also does ensemble distillation, but can capture structure in the output space.

Uncertainty in depth prediction models The goal of self-supervised depth estimation is to train a network to predict single image depth maps without explicit depth supervision [7, 8]. Instead, self-supervised approaches use geometric constraints between two calibrated stereo cameras to learn depth prediction. At test time, these methods do not require a stereo pair, only a single image. Given the inherit ambiguity of predicting depth from a single image, depth prediction is a natural use case for uncertainty estimation in dense prediction tasks. In [19] the authors review and compare different approaches for uncertainty prediction for self-supervised depth prediction. They focus on methods that predict multiple hypothesis, such as dropout [5] and ensembles [14], methods that predict per-pixel independent heteroscedastic uncertainty [13], and combinations of both.

In the experiments, we use the pre-trained networks provided by [19] to evaluate the efficiency of our method in approximating ensembles. In particular, we use their most successful model, which combines an ensemble with predictive uncertainty. Their ensembles are trained using bootstrapping [14], and they use an uncorrelated Laplace distribution for predictive parametric uncertainty.

Xia *et al*. [25] show how a probabilistic model for depth prediction can be explored by downstream tasks such as inference with additional information. They model uncertainty at a patch level in a model akin to a Markov Random Field. In contrast to our approach, the method requires solving a complex optimization problem at inference time.

2.2. Predicting Structured Gaussian Distributions

To approximate an ensemble, we train a network to predict the parameters of a Gaussian distribution. Given an input image $\mathbf{x}$ the network outputs the parameters of a Gaussian distribution $\boldsymbol{\mu}(\mathbf{x})$ and $\boldsymbol{\Sigma}(\mathbf{x})$. We focus on dense prediction tasks. For these tasks, if N is the number of pixels in the input image, the size of $\boldsymbol{\mu}$ is also N while a full $\boldsymbol{\Sigma}$ matrix has N^2 parameters. The quadratic scaling of the number of parameters of the covariance matrix leads to the common remedy of a diagonal matrix, which requires only N parameters. However, this simplifying assumption prohibits the capture of correlations between pixels.

Structured Uncertainty Prediction Networks Our approach builds on the work, [4, 24], where the parameter-

isation used is the Cholesky decomposition of the precision matrix, *i.e*. the network predicts $\mathbf{L}_\Lambda$ directly, where $\mathbf{L}_\Lambda \mathbf{L}_\Lambda^\top = \mathbf{\Sigma}^{-1}$ and $\mathbf{L}_\Lambda$ is a lower triangular matrix. For completeness, we review some of the properties of the parameterisation presented in [4], which we use in our work.

When choosing a parameterisation, there are a few criteria that should be taken into account: how easy it is to evaluate the likelihood function required for training, how easy it is to sample from the distribution at inference time and how easy is to impose that the covariance matrix (or equivalently precision matrix) is symmetric and positive definite? Direct prediction of the Cholesky factor guarantees that the precision matrix is symmetric. To guarantee that it is positive definite, the diagonal values of the Cholesky decomposition are required to be positive; an easy constraint to enforce in practice. This choice of parameterisation allows for easy computation of the log-likelihood of the multivariate Gaussian distribution. However, sampling is more difficult to perform, since access to the covariance is required. We discuss a new efficient method for sampling in § 3.3.

Sparsity Despite the advantages of using this parameterisation and the fact that the Cholesky is a lower triangular matrix, the number of elements still grows quadratically with respect to the number of pixels, N, making it prohibitive to directly estimate for large images. We follow SUPN [4] in imposing sparsity in the Cholesky matrix $\mathbf{L}_\Lambda$. For each pixel, we only populate the Cholesky matrix for pixels which are in a small neighborhood, while keeping the matrix lower-triangular. We include an illustration in the supplemental material. This sparse Cholesky matrix can be compactly represented by only predicting the non-zero values; for a 3×3 neighborhood, this corresponds to predicting the diagonal map plus 4 off-diagonal maps. Importantly, this representation can be encoded efficiently into popular APIs such as Tensorflow and PyTorch using standard convolutional operations.

Deep Gaussian MRFs Our model can be seen as a Gaussian Markov Random Field, since the sparsity pattern on the precision matrix directly implies the Markov property: a variable is conditional independent of all other variables given its neighbours. Similar to our approach, [3, 10, 23] use a regression model to predict the parameters of a Gaussian Conditional Random Field that captures structure in output space. They show improved results for semantic segmentation. However, they focus on predicting the MAP solution and do not make use of the full probability distribution.

3. Method

Our goal is to train a single network that approximates the multiple outputs of an ensemble. We assume this ensemble is given as a pre-trained network(s), *e.g*. from [14] or [9]. We predict a structured multivariate Gaussian using the sparse representation discussed in § 2.2.

3.1. Training

Given I training images $\{\mathbf{x}_i \,|\, i \in [1, I]\}$, the pre-trained ensemble is run for the full training set, to obtain S distinct predictions per image $\{\mathbf{d}_i^s \,|\, s \in [1, S]\}$, where S is the size of the ensemble or number of MC-dropout samples.

Log-likelihood loss Our network is trained to minimise the negative log-likelihood of the training set:

$$\mathcal{L} = -\sum_{i=1}^{I} \sum_{s=1}^{S} \log \mathcal{N}\big(\mathbf{d}_i^s \,|\, \boldsymbol{\mu}(\mathbf{x}_i), \mathbf{\Sigma}(\mathbf{x}_i)\big)\,, \tag{3}$$

where $\mathcal{N}\big(\mathbf{d}_i^s \,|\, \boldsymbol{\mu}(\mathbf{x}_i), \mathbf{\Sigma}(\mathbf{x}_i)\big)$ is the probability density function of a multivariate Gaussian distribution.

3.2. Inference

In common with ensembles and MC-dropout, we can use our model to obtain samples from the predictive distribution $p(\mathbf{d} \,|\, \mathbf{x})$. In contrast with ensembles, our model is not restricted on the number of samples that can be taken; we discuss an efficient sampling procedure in § 3.3. More importantly, since our model predicts a closed form probability function, it allows for additional inference tasks which are not possible with ensembles or MC-dropout.

Evaluation of the predictive log-likelihood Our model allows evaluating the log-likelihood for a given dense prediction. This is useful for model comparison.

Conditional distribution The output Gaussian distribution can be used to compute the conditional distribution of some pixel labels, given the label for other pixels. The ability of drawing conditional samples has practical applications, for example: for depth completion, where the depth of a small number of pixels is provided by an external sensor, such as a LIDAR scanner; or for interactive image segmentation, where the label of a few pixels is provided by a user.

3.3. Efficient Sampling

Sampling from a Multivariate Gaussian distribution with a diagonal covariance matrix $\mathbf{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_N)$ can proceed with a straight forward sampling approach where each dimension (pixel) is independent:

$$\tilde{d}_n^{(s)} = \boldsymbol{\mu}_n + \sigma_n \, \tilde{\varepsilon}_n^{(s)}, \quad \tilde{\varepsilon}_n^{(s)} \sim \mathcal{N}(0, 1)\,. \tag{4}$$

If the Gaussian distribution has a general covariance, however, then the sample cannot be computed independently for each pixel and must be drawn through a square root matrix of the covariance, such as the Cholesky factor:

$$\tilde{\mathbf{d}}^{(s)} = \boldsymbol{\mu} + \mathbf{L}_\Sigma \, \tilde{\boldsymbol{\varepsilon}}^{(s)}, \quad \tilde{\boldsymbol{\varepsilon}}^{(s)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_N)\,, \tag{5}$$

where $\mathbf{L}_\Sigma \mathbf{L}_\Sigma^\top = \mathbf{\Sigma}$. Computation of the dense covariance matrix from the sparse precision, followed by the

Cholesky operation would involve a computational complexity of $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ storage making it infeasible.

Efficient calculation via the Jacobi method Fortunately, adopting a sparse structure over the Cholesky precision matrix $\mathbf{L}_\Lambda$ means that we can perform a matrix multiplication efficiently. We can exploit this to take approximate samples using a truncated (to J iterations) version of the Jacobi iterative solver to invert $\mathbf{L}_\Lambda$. This results in a tractable algorithm for obtaining approximate samples of sufficient quality. We can take multiple samples from the same distribution simultaneously while retaining efficiency.

We start with a set of S standard Gaussian samples,

$$\tilde{\boldsymbol{E}} = [\tilde{\boldsymbol{\varepsilon}}^{(1)}, \ldots, \tilde{\boldsymbol{\varepsilon}}^{(S)}], \quad \tilde{\boldsymbol{\varepsilon}}^{(s)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_N)\,. \tag{6}$$

We then note that the transposed, inverse of the precision Cholesky matrix can be used as a sampling matrix since

$$\boldsymbol{\Sigma} = \boldsymbol{\Lambda}^{-1} = (\mathbf{L}_\Lambda \mathbf{L}_\Lambda^\top)^{-1} = \mathbf{L}_\Lambda^{-\top} \mathbf{L}_\Lambda^{-1}\,, \tag{7}$$

indicating that $\mathbf{L}_\Lambda^{-\top}$ is the LHS of a square root matrix for $\boldsymbol{\Sigma}$. Thus we draw low variance Monte Carlo samples as

$$\tilde{\mathbf{D}} = [\tilde{\mathbf{d}}^{(1)}, \ldots, \tilde{\mathbf{d}}^{(S)}] = \boldsymbol{\mu} + \mathbf{L}_\Lambda^{-\top} \tilde{\boldsymbol{E}}\,. \tag{8}$$

To invert $\mathbf{L}_\Lambda^\top$ efficiently, we use J Jacobi iterations; these are particularly efficient to apply with a sparse matrix that is already lower triangular. We initialise $\mathbf{S}^{(0)} = \tilde{\boldsymbol{E}}$ and then, at each iteration, update the samples with

$$\mathbf{S}^{(j+1)} \leftarrow D_\Lambda^{-1}\big(\tilde{\boldsymbol{E}} - U_\Lambda\, \mathbf{S}^{(j)}\big)\,, \tag{9}$$

where $D_\Lambda := \operatorname{diag}(\mathbf{L}_\Lambda^\top)$ and $U_\Lambda := \mathbf{L}_\Lambda^\top - D_\Lambda$, a strictly upper triangular matrix. The final samples are then obtained by the addition of the mean such that $\tilde{\mathbf{D}} = \boldsymbol{\mu} + \mathbf{S}$.

Efficient conditional sampling As we have a closed form representation of the output distribution:

$$\mathbf{d} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),\ \boldsymbol{\Sigma} = \mathbf{L}_\Lambda^{-\top} \mathbf{L}_\Lambda^{-1}\,, \tag{10}$$

we can find the expression for a resulting conditional distribution where we specify values for a subset of the pixels and sample from the resulting distribution over the remaining pixels. Let us partition the pixels into a set of known values $\mathbf{d}_\mathrm{K}$ and unknown values $\mathbf{d}_\mathrm{U}$; pixels (arbitrarily) belong to either one set or the other under a pixel-wise mask:

$$[\mathbf{m}_\mathrm{K}]_n = \begin{cases} 1, & n \in \mathcal{K} \\ 0, & n \in \mathcal{U} \end{cases},\ \mathbf{m}_\mathrm{U} = 1 - \mathbf{m}_\mathrm{K}\,. \tag{11}$$

Thus, with slight abuse of notation, we recover the full set of values as $\mathbf{d} = \mathbf{m}_\mathrm{K} \odot \mathbf{d}_\mathrm{K} + \mathbf{m}_\mathrm{U} \odot \mathbf{d}_\mathrm{U}$. The conditional distribution for the unknown values, given that the known values $\mathbf{d}_\mathrm{K} = \boldsymbol{\alpha}$, is the Gaussian conditional density:

$$p(\mathbf{d}_\mathrm{U} \,|\, \mathbf{d}_\mathrm{K} = \boldsymbol{\alpha}) \sim \mathcal{N}(\mathbf{b}, \mathbf{B})\,, \tag{12}$$

$$\mathbf{b} := \boldsymbol{\mu}_\mathrm{U} + \boldsymbol{\Sigma}_\mathrm{UK} \boldsymbol{\Sigma}_\mathrm{KK} (\boldsymbol{\alpha} - \boldsymbol{\mu}_\mathrm{K})\,, \tag{13}$$

$$\mathbf{B} := \boldsymbol{\Sigma}_\mathrm{UU} - \boldsymbol{\Sigma}_\mathrm{UK} \boldsymbol{\Sigma}_\mathrm{KK}^{-1} \boldsymbol{\Sigma}_\mathrm{KU}\,, \tag{14}$$

Algorithm 1: Jacobi sampling for the multivariate Gaussian distribution

Result: Samples drawn from a correlated multivariate Gaussian (with sparse precision)

Samples: $\mathbf{S}^{(0)} \leftarrow \tilde{\boldsymbol{E}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_N)$, $N := W \times H$;
Local connection filters: $\mathbf{F} = \{\mathbf{f}_l\}_{l=1}^L$;
Log diagonal terms: $\boldsymbol{\phi} \in \mathbb{R}^N$;
Off diagonal terms: $\boldsymbol{\psi} \in \mathbb{R}^{L \times N}$;

for $j \leftarrow 0$ **to** $J - 1$ **do**
  $\mathbf{V} \leftarrow \text{Conv2D}(\mathbf{S}^{(j)}, \mathbf{F})$;
  $\mathbf{v} \leftarrow \sum_{l=1}^L [\mathbf{V} \odot \boldsymbol{\psi}]_{n,l}$;
  $\mathbf{S}^{(j+1)} \leftarrow \big[\exp(\boldsymbol{\phi})\big]^{-1} \odot (\tilde{\boldsymbol{E}} - \mathbf{v})$
end

Output: $\mathbf{S}^{(J)} \approx \big(\mathbf{L}_\Lambda^{-\top} \tilde{\boldsymbol{E}}\big) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}^{-1})$;

where the subscripts dictate the appropriate partitions of the mean vector or blocks of the covariance matrix.

Evaluating this directly, in matrix form, would again be prohibitively expensive, especially considering the matrix inversions (from precision to covariance matrices). Thankfully we can use a modified form of the Jacobi sampling method combined with Matheron's rule for conditional sampling. Matheron's rule states that if $(\mathbf{a}, \mathbf{b})$ are samples from the joint distribution $p(\mathbf{d}_\mathrm{K}, \mathbf{d}_\mathrm{U})$ then the random variable $\mathbf{b}$ conditioned on $\mathbf{a} = \boldsymbol{\alpha}$ can be found by:

$$(\mathbf{b} \mid \mathbf{a} = \boldsymbol{\alpha}) \leftarrow \mathbf{b} + \boldsymbol{\Sigma}_\mathrm{UK} \boldsymbol{\Sigma}_\mathrm{KK}^{-1} (\boldsymbol{\alpha} - \mathbf{a})\,. \tag{15}$$

We can use straight forward identities to convert Matheron's rule into an update equation in terms of the precision:

$$\begin{bmatrix} \boldsymbol{\Lambda}_\mathrm{KK} & \boldsymbol{\Lambda}_\mathrm{KU} \\ \boldsymbol{\Lambda}_\mathrm{UK} & \boldsymbol{\Lambda}_\mathrm{UU} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\Sigma}_\mathrm{KK} & \boldsymbol{\Sigma}_\mathrm{KU} \\ \boldsymbol{\Sigma}_\mathrm{UK} & \boldsymbol{\Sigma}_\mathrm{UU} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{16}$$

$$\Rightarrow \boldsymbol{\Lambda}_\mathrm{UK}\, \boldsymbol{\Sigma}_\mathrm{KK} + \boldsymbol{\Lambda}_\mathrm{UU}\, \boldsymbol{\Sigma}_\mathrm{UK} = \mathbf{0} \tag{17}$$

$$\Rightarrow \boldsymbol{\Sigma}_\mathrm{UK}\, \boldsymbol{\Sigma}_\mathrm{KK}^{-1} = -\boldsymbol{\Lambda}_\mathrm{UU}^{-1}\, \boldsymbol{\Lambda}_\mathrm{UK}\,. \tag{18}$$

We have ready access to efficient evaluation of the sparse $\mathbf{L}_\Lambda^\top$, as discussed in the Jacobi method. With suitable bookkeeping, we can produce the appropriately shuffled local connection filters $\mathbf{F}_\text{shuff} \leftarrow \text{Shuffle}(\mathbf{F})$ and permuted off-diagonal terms $\boldsymbol{\psi}_\text{shuff} \leftarrow \text{Shuffle}(\boldsymbol{\psi})$ to provide a similar evaluation of the sparse $\mathbf{L}_\Lambda$. This product results in a sparse banded diagonal structure in the precision matrix $\boldsymbol{\Lambda}$. The appropriate blocks of this sparse matrix can be accessed and used to solve for the conditional update of (15) using a precision form of the update (18).

3.4. Extension to Non-Gaussian Likelihoods

For many dense prediction tasks, a multivariate Gaussian distribution is not an appropriate likelihood over the

observations directly. However, SUPN is still applicable for this use case, by fitting the multivariate Gaussian distribution to the logit space, *i.e*. to the layer just before the last non-linear layer. This is then followed by an appropriate activation function. For example, for depth prediction, the outputs of the network should be non-negative and the activation function used is a scaled sigmoid, following monodepth2 [8]. Similarly, for the task of segmentation, the fitting of the SUPN could be done in logit space and soft-max would be used as the activation function.

3.5. Implementation Details

Architecture We build upon the U-Net architecture used by Monodepth2 [8], *i.e*. an encoder-decoder architecture where the encoder is a ResNet18 and there are skip connections between the encoder and the decoder. We add an additional decoder to predict the Cholesky parameters. This decoder takes skip connections from the mean decoder as input. The additional decoder concatenates coordinate maps in the convolutional blocks [16] to provide additional spatial information. We designed an off-diagonal prediction approach where the scale of the values is initially very small, $\approx \exp(-4)$, but adapts during training. We found this inductive bias, in lieu of formal priors, was required to predict high quality covariances. We use a 5×5 neighborhood for the Cholesky decomposition; please see the supplementary details for architecture details and ablation experiments.

Model size Our model encodes the distillation of an ensemble of large models into a single framework; we use only 24% more parameters than a single network (out of 8 in the ensemble).

Multi-scale loss For depth prediction, we use a multi-scale loss similar to Monodepth2 [8], where the loss in (5) is applied across different scales.

Complexity Fixed sparsity ensures that all operations are $\mathcal{O}(N)$ for both computation and storage. Sampling is $\mathcal{O}(J)$ (we used $J = 1000$); empirically, the total time for a full Jacobi sample was 0.6s.

4. Experiments

For the experiments, we show our method applied to monocular depth estimation. We use the KITTI dataset [6] and base our implementation on the Monodepth2 repository [8] and the repository from [19].

Pre-trained ensembles We use the pre-trained models provided by [19]. In particular, the ensembles created through bootstrapping together with predictive uncertainty. Two different approaches are used for predictive uncertainty. Both use a diagonal multivariate Laplace distribution, but differ in the way they are trained: LOG is trained by directly optimizing the log-likelihood of a self-supervised depth model; while SELF uses a pretrained network for depth prediction, without uncertainty estimation, as the teacher model.

Metrics For evaluating the accuracy of the estimated depth maps we use a subset of the metrics commonly used for the Kitti dataset: absolute relative error, root mean squared error (RMSE) and the A1 metric.

For evaluation of the uncertainty estimates, we use the metrics used in [19]: area under the sparsification error (AUSE) and area under the random gain (AURG). Both these metrics rely on using per-pixel uncertainty estimates to rank pixels from less confident to more confident. For AUSE, this ranking is compared with an oracle ranking that sorts pixels from higher error to lower error, using the different ground truth metrics for ranking. A small AUSE means that the ranking provided by the uncertainty estimate is similar to this oracle ranking. AURG compares the ranking based on estimated uncertainty with a random ranking, large values are preferred for this metric.

Since both these established metrics only consider per-pixel estimates, we also evaluate the posterior log-likelihood of test samples from the ensembles under our model. To provide a baseline, we also train a version of our model with only a diagonal covariance structure (per-pixel), which cannot model structure. Comparing against this baseline allows us to determine if the model has correctly captured the distribution of test samples and avoided overfitting. We also measure the log-likelihood to other ensembles to ensure that the SUPN variants estimate distributions that generalise well to support other plausible samples.

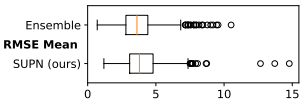
4.1. Quantitative Results

Depth accuracy In Table 1 we show a quantitative comparison between the two variants of ensembles and the corresponding versions of our model, trained to approximate them. We compare the methods using the depth estimation metrics. While the mean performance of the ensembles is slightly superior to our approximate models, the results are comparable within the margin of error. The box plot in Table 1 highlights the strong overlap in the error distribution of the ensemble and SUPN models, indicating that despite the significant reduction in the number of parameters, SUPN is able to approximate the performance of the ensemble.

Our models compare favourably with a diagonal only model. This is particularly noticeable in the metrics for the best sample. Samples from our model consistently outperform samples from a diagonal only Gaussian.

Uncertainty estimation Table 2 provides a quantitative comparison in terms of uncertainty metrics. SUPN consistently outperforms the teacher ensemble model for both LOG and SELF. The log-likehood values demonstrate that the correlated structure capture by SUPN is better able to explain the test outputs of the ensembles that the baseline

Table 1. **Accuracy comparison**: Quantitative comparison of quality on commonly used depth metrics (see supplement for the remaining metrics in [8]). The "Best" metrics sample 40 different predictions for our model, and from the 8 ensembles for the baseline, and choose the best under each metric. Standard deviations are given in brackets. The box plot illustrates the substantial overlap in distributions.



Box plot illustrating the strong distribution overlap between the original ensemble and the trained SUPN model for Boot+Log RMSE mean.

Model name	AbsRel Mean ↓	AbsRel Best ↓	RMSE Mean ↓	RMSE Best ↓	A1 Mean ↑	A1 Best ↑
MD2 Boot+Log	0.092 (0.035)	0.084 (0.031)	3.850 (1.370)	3.600 (1.260)	0.911 (0.064)	0.923 (0.055)
MD2 Boot+Self	0.088 (0.034)	0.083 (0.031)	3.795 (1.397)	3.574 (1.323)	0.918 (0.060)	0.929 (0.051)
Diagonal	0.101 (0.044)	0.103 (0.043)	4.000 (1.457)	4.020 (1.444)	0.896 (0.076)	0.894 (0.074)
SUPN Boot+Log	0.104 (0.047)	0.095 (0.039)	4.071 (1.489)	3.577 (1.191)	0.892 (0.080)	0.909 (0.069)
SUPN Boot+Self	0.103 (0.049)	0.096 (0.046)	4.091 (1.442)	3.800 (1.396)	0.894 (0.078)	0.906 (0.073)

Table 2. **Pixelwise uncertainty metrics**: AUSE (area under the sparsification error), lower is better. AURG (area under the random gain), higher is better. Uncertainy for SUPN estimated from std-deviation of 10 samples. Results marked with a * differ from the published work by [19], as to make it comparable we do not use the Monodepth 1 post-processing. LL (Log-Likelihood) columns provide the log-likelihood of samples from the respective ensembles under the diagonal (baseline) and SUPN models. Standard deviations are given in brackets.

Model name	AbsRel AUSE ↓	AbsRel AURG ↑	RMSE AUSE ↓	RMSE AURG ↑	A1 AUSE ↓	A1 AURG ↑	LL Boot+Log $\times 10^5$↑	LL Boot+Self $\times 10^5$↑
MD2 Boot+Log	0.038 (0.020)	0.021 (0.019)	2.449 (0.877)	0.820 (0.929)	0.046 (0.048)	0.037 (0.040)		
MD2 Boot+Self	0.029 (0.018)	0.028 (0.019)	1.924 (1.006)	1.316 (1.000)	0.028 (0.041)	0.049 (0.037)		
MD2 Boot+Log*	0.041 (0.019)	0.018 (0.020)	2.927 (1.327)	0.324 (1.019)	0.050 (0.049)	0.032 (0.037)		
MD2 Boot+Self*	0.040 (0.021)	0.017 (0.018)	2.906 (1.458)	0.331 (1.08)	0.045 (0.045)	0.031 (0.035)		
Diagonal	0.085 (0.050)	-0.020 (0.030)	5.075 (1.924)	-1.697 (0.799)	0.138 (0.083)	-0.440 (0.053)	1.77 (11.48)	1.15 (12.78)
SUPN Boot+Log	0.037 (0.027)	0.030 (0.025)	1.555 (1.307)	1.856 (1.355)	0.040 (0.063)	0.058 (0.047)	40.60 (1.35)	38.18 (2.93)
SUPN Boot+Self	0.050 (0.037)	0.017 (0.028)	2.786 (1.796)	0.674 (1.544)	0.062 (0.074)	0.034 (0.055)	36.51 (2.31)	38.87 (1.63)

Figure 2. Visualisation of learned covariances between example pixels (red crosses) and other pixel locations for SUPN BOOT+LOG. Red indicates high positive correlation, blue is strong negative correlation. For clarity, these plots are scaled into the standard deviation range (via a signed square root operation) and plotted over a range [-0.05, 0.05]. These examples illustrate the long range correlations that can be captured from very local structure (5×5 pixel regions) in the precision matrix. For more examples, see the supplementary video.

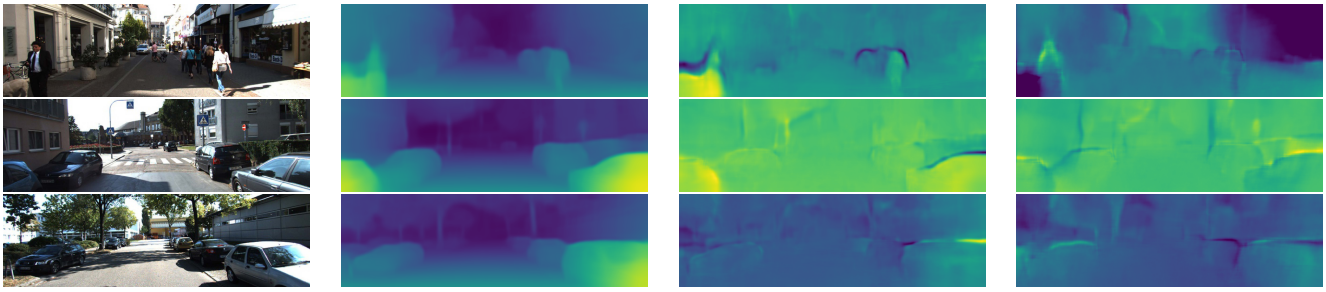
diagonal model. The samples routinely have higher support under the SUPN model which suggests that some of the other measures are not accurately measure the quality of the structure present in the posterior predictions of the model. As expected, the performance of the SUPN approaches on the test set for the corresponding samples are slightly better but we note that overall the values are similar between the two methods indicating that the correlations captured are not overfit to the specific ensembles.

4.2. Qualitative Results

Figure 3 illustrates samples from the BOOT+LOG ensemble, and the SUPN approximation; the samples are visually similar and exhibit considerable long-range structure.

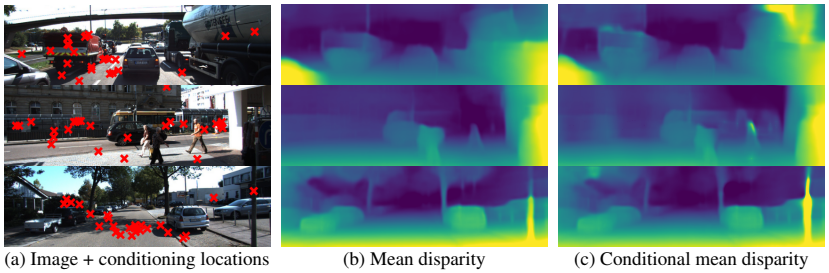
Introspection As discussed in § 1, one of the advantages of an explicit distribution is that it allows for introspection. Figure 2 illustrates how the covariance between a specified pixel, and any other, can be explicitly computed. These visualisations are the corresponding row of the covariance matrix obtained using the sampling process of (9) twice (with $\mathbf{L}_\Lambda$ and then $\mathbf{L}_\Lambda^\top$) to a one-hot vector encoding the pixel of interest on the RHS (instead of $\tilde{\boldsymbol{E}}$) and no mean.

Conditional Distributions Figure 4 illustrates our model's ability to condition samples on arbitrary output pixels, which is not possible in most deep probabilistic models. In this example, we use some samples from the ground truth depth as additional conditioning on the predicted distribution, and show the conditional mean.



(a) Input image (b) Mean disparity (c) BOOT+LOG diff (d) SUPN BOOT+LOG diff

Figure 3. Example depth samples (see supplementary video for more). (b) Average normalised disparity predicted by the ensemble models. Difference between the mean and one of the (c) BOOT+LOG ensemble or (d) SUPN BOOT+LOG; the samples appear qualitatively similar.



(a) Image + conditioning locations (b) Mean disparity (c) Conditional mean disparity

Model name	AbsRel ↓	RMSE ↓	A1 ↑
Ensemble	0.088 (0.034)	3.795 (1.397)	0.918 (0.060)
SUPN	0.104 (0.047)	4.071 (1.489)	0.892 (0.080)
SUPN 25*	0.086 (0.036)	3.893 (1.541)	0.922 (0.053)
SUPN 50*	0.076 (0.031)	3.607 (1.433)	0.936 (0.041)
SUPN 100*	0.064 (0.028)	3.251 (1.290)	0.949 (0.033)
SUPN 200*	0.054 (0.022)	2.873 (1.100)	0.960 (0.028)

(d) Quantitative conditioning accuracy

Figure 4. Conditional prediction using sparse ground truth depth information. (a) shows 25 randomly sampled conditioning locations that have valid depth (b) shows the original mean disparity, while (c) shows the conditional mean disparity. d) Quantifies the accuracy improvement of our SUPN Boot+Log model when conditioned on N^* random ground truth depth pixels, repeated 10 times per image.

5. Discussion and Limitations

Similar to other distillation methods, the performance of our method is upper-bounded by the performance of the original ensemble model. This might become an issue when the ensemble is small, and doesn't capture the full diversity of data. We've observed visually that the log-likelihood alone is not always a good predictor of sample quality, *i.e.* a sample might have high log-likelihood while looking implausible, this may be due to a lack of variation in the ensemble predictions. This could potentially be overcome by using priors on the predicted Gaussian distribution, and future work will consider subsequently training the model on the specified task, using the drawn samples.

As a deterministic approximation to the output of an ensemble, we seek to capture all forms of uncertainty captured by the ensemble (e.g. aleatoric and epistemic). We acknowledge that we do not consider the epistemic uncertainty in the approximation separately, however our work may be considered orthogonal to work in this area, e.g. BNNs, and could be readily combined.

Potential negative impact We think that uncertainty estimation is a valuable endeavor in improving deep models, and that our approach of using explicit distributions is a step in the right direction, providing tangible benefits. However, the predicted distributions have yet to be evaluated on out-of-distribution data. As with most machine learning models, we cannot expect generalisation of our SUPN prediction networks to very different data. Clearly, this approach will require extensive validation before deployment in safety-critical applications, such as autonomous driving.

Conclusion We presented a method for uncertainty estimation by distillation of ensemble models. We showed that our structured Gaussian model can be predicted by a single pass of a convolutional neural network, we have proposed an efficient method for drawing samples.

Our method was validated on the task of depth prediction from a single image. Our distilled model is able to perform similarly to the original ensemble on uncertainty metrics, while requiring fewer parameters and allowing arbitrary numbers of samples to be drawn. We have illustrated that the samples capture long-range correlations in the image domain, which is in stark contrast to prior works that use diagonal covariance matrices. We demonstrated the benefit of our predicted distribution in terms of enabling arbitrary test-time conditioning and allowing for direct introspection of the inferred distribution. We hope that our paper sparks interest in predictive uncertainty models that are able to model correlation in the output space, with many practical applications in computer vision and integration with subsequent down-stream tasks.

References

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Abbas Khosravi, U Rajendra Acharya, Vladimir Makarenkov, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *arXiv preprint arXiv:2011.06225*, 2020. 2

[2] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Dropout distillation. In *International Conference on Machine Learning*, pages 99–107. PMLR, 2016. 2, 3

[3] Siddhartha Chandra and Iasonas Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. In *European conference on computer vision*, pages 402–418. Springer, 2016. 4

[4] Garoe Dorta, Sara Vicente, Lourdes Agapito, Neill DF Campbell, and Ivor Simpson. Structured uncertainty prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5477–5485, 2018. 2, 3, 4

[5] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016. 1, 2, 3

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 6

[7] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017. 3

[8] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019. 3, 6, 7

[9] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. In *ICLR*, 2017. 3, 4

[10] Jeremy Jancsary, Sebastian Nowozin, Toby Sharp, and Carsten Rother. Regression tree fields: An efficient, non-parametric approach to image labeling problems. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2012. 4

[11] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017. 2, 3

[12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 2

[13] Maria Klodt and Andrea Vedaldi. Supervising the new with the old: learning sfm from sfm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 698–713, 2018. 2, 3

[14] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *NeurIPS*, 2017. 1, 2, 3, 4

[15] Zhizhong Li and Derek Hoiem. Improving confidence estimates for unfamiliar examples. In *CVPR*, 2020. 1, 2, 3

[16] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *arXiv preprint arXiv:1807.03247*, 2018. 6

[17] David JC MacKay. Probable networks and plausible predictions-a review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 1995. 1, 2

[18] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. *ICLR*, 2020. 2, 3

[19] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3227–3237, 2020. 1, 3, 6, 7

[20] Simon JD Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012. 2

[21] Yichen Shen, Zhilu Zhang, Mert R Sabuncu, and Lin Sun. Real-time uncertainty estimation in computer vision via uncertainty-aware distribution distillation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 707–716, 2021. 2, 3

[22] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 2014. 3

[23] Raviteja Vemulapalli, Oncel Tuzel, Ming-Yu Liu, and Rama Chellapa. Gaussian conditional random field network for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3224–3233, 2016. 4

[24] Peter M Williams. Using neural networks to model conditional multivariate densities. *Neural computation*, 8(4):843–854, 1996. 3

[25] Zhihao Xia, Patrick Sullivan, and Ayan Chakrabarti. Generating and exploiting probabilistic monocular depth estimates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 65–74, 2020. 3