HCNAF: Hyper-Conditioned Neural Autoregressive Flow and its Application for Probabilistic Occupancy Map Forecasting

Geunseob (GS) Oh
University of Michigan
gsoh@umich.edu

Jean-Sébastien Valois
Uber ATG
jsvalois@uber.com

Abstract

We introduce Hyper-Conditioned Neural Autoregressive Flow (HCNAF); a powerful universal distribution approximator designed to model arbitrarily complex conditional probability density functions. HCNAF consists of a neural-net based conditional autoregressive flow (AF) and a hyper-network that can take large conditions in non-autoregressive fashion and outputs the network parameters of the AF. Like other flow models, HCNAF performs exact likelihood inference. We conduct a number of density estimation tasks on toy experiments and MNIST to demonstrate the effectiveness and attributes of HCNAF, including its generalization capability over unseen conditions and expressivity. Finally, we show that HCNAF scales up to complex high-dimensional prediction problems of the magnitude of self-driving and that HCNAF yields a state-of-the-art performance in a public self-driving dataset.

1. Introduction

Recent autoregressive flow (AF) models [1–4] have achieved state-of-the-art performances in density estimation tasks. They offer compelling properties such as *exact* likelihood inference and expressivity. Of those, [3, 4] successfully unified AF models and neural networks, and demonstrated an ability to capture complex multi-modal data distributions while universally approximating continuous probability distributions.

However, due to scalability limitation, existing neural AF models are ineffective at tackling problems with arbitrarily high-dimensional conditional terms. Scene prediction for autonomous driving is such a task where the benefits of AF models can be leveraged but where the contextual information (conditional terms) is too large (i.e. $C > \mathbb{R}^{1,000,000}$ due to using many multi-channel spatio-temporal maps). In contrast, the biggest experiment neural AF models reported is BSDS300 ($\mathbb{R}^{63}$) [5]. This may explain their limited use in common problems despite demonstrating excellent performance in density estimations.
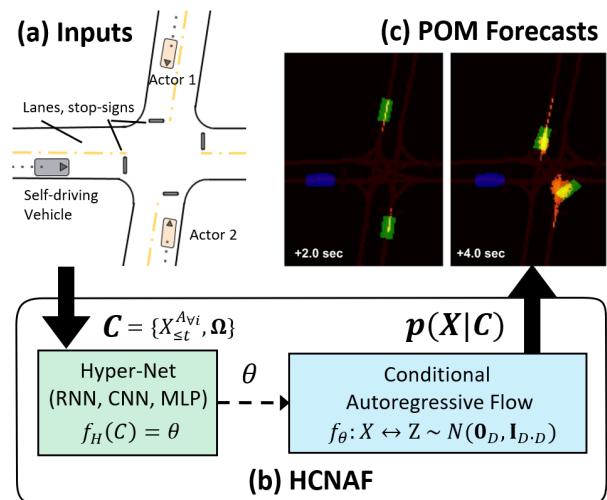

Figure 1: HCNAF used for probabilistic occupancy map (POM) forecasting, demonstrating the network's use of high-dimensional conditions ($C = \mathbb{R}^{>1mil}$). a) Inputs (conditions) are the spatio-temporal scene data. b) HCNAF consists of two neural-net based modules: a hyper-network f_H and a conditional AF f. f_H can take arbitrarily large inputs and produces the network parameters for f, which produces the conditional probability $p(X|C)$ precisely. c) Resulting POMs for agent vehicle centers at t=2 and t=4 secs.

We propose a novel conditional density approximator called Hyper-Conditioned Neural Autoregressive Flow (HCNAF) to address the aforementioned limitation. HCNAF performs an *exact* likelihood inference by precisely computing probability of complex target distributions $p_{model}(X|C) \approx p(X|C)$ with arbitrarily large C. By taking advantage of the design, HCNAF grants neural AFs the ability to tackle wider range of scientific problems; which is demonstrated by the autonomous driving prediction tasks.

Prediction tasks in autonomous driving involve transforming the history of high dimensional perception data up to the current time into a representation of how the environment will evolve [6–15]. To be effective, advanced predic-

tions models should exhibit the following properties:

1. *probabilistic*: reflecting future state uncertainties,
2. *multimodal*: reproducing the rich diversity of states,
3. *context driven*: interactive & contextual reasoning, and
4. *general*: capable of reasoning unseen inputs.

To incorporate the above requirements, we leverage HC-NAF's powerful attributes such as the expressivity to model arbitrarily complex distributions and the generalization capability over unseen data. Furthermore, we opted for probabilistic occupancy maps (POMs) (see figure 1) over a more widely used trajectory-based prediction approach [6–14]. As POM naturally encodes uncertainty, a POM represents all possible trajectories; thus removes the need to exhaustively sample trajectories like in trajectory-based methods.

Before presenting results on self-driving scenarios, we first introduce HCNAF and report results from a number of density estimation tasks to investigate HCNAF's expressivity and generalization capability over diverse conditions.

2. Background

Flow, or normalizing flow, is a type of deep generative models which are designed to learn data distribution via the principle of maximum likelihood [16] so as to generate new data and/or estimate likelihood of a target distribution.

Flow-based models construct an invertible function $f(z) = x$ between a latent variable z and a random variable x, which allows the computation of exact likelihood of an unknown data distribution $p(x)$ using a known pdf $\pi(z)$ (e.g. normal distribution), via the change of variable theorem:

$$p(x) = \pi(f^{-1}(x)) \left| det \frac{df^{-1}(x)}{dx} \right|. \qquad (1)$$

In addition, flow offers data generation capability by sampling latent variables $z \sim \pi()$ and passing it through f. As the accuracy of the approximation $f(z) = x$ increases, the modeled pdf $p_{model}(x)$ converges to the true $p(x)$ and the quality of the generated samples also improves.

In contrast to other classes of deep generative models (namely VAE [17] and GAN [18]), flow is an explicit density model and offers unique properties:

1. Computation of an **exact** probability, which is essential in the POM forecasting task. VAE infers $p(x)$ using a computable term; Evidence Lower BOund (ELBO). However, since the upper bound is unknown, it is unclear how well ELBO actually approximates $p(x)$ and how ELBO can be utilized for tasks that require exact inference. While GAN proved its power in generating high-quality samples for image generation and translation tasks [19, 20], obtaining the density estimation and/or probability computation for the generated samples is non-trivial.

2. The expressivity of flow-based models allows the models to capture complex data distributions. A recently published AF model called *Neural Autoregressive Flow (NAF)* [3] unified earlier AF models including [1, 2] by generalizing their affine transformations to arbitrarily complex non-linear monotonic transformations. Conversely, the default VAE uses *unimodal* Gaussians for the prior and the posterior distributions. In order to increase the expressivity of VAE, some have introduced more expressive priors [21] and posteriors [22, 23] that leverage flow techniques.

The class of invertible neural-net based autoregressive flows, including *NAF* and *BNAF* [4], can approximate rich families of distributions, and was shown to universally approximate continuous pdfs. However, *NAF* and *BNAF* do not handle *external* conditions (e.g. classes in the context of GAN vs cGAN [24]). That is, those models are designed to compute $p(x_t)$ conditioned on previous inputs $x_{1:t-1}$ autoregressively to formulate $p(x_t|x_{1:t-1})$. This formulation is not suitable for taking arbitrary conditions other than the autoregressive ones. This limits the extension of *NAF* to applications that work with conditional probabilities $p(X|C)$, such as the POM forecasting.

MAF and *cMAF* were proposed in [2] to model affine flow transformations with and without additional external conditions. As shown in Equation 2, the transformation between z_d and x_d is affine and the influence of C over the transformation relies on μ, σ, and stacking multiple flows. These may limit the contributions of C to the transformation. This explains the needs for a conditional autoregressive flow that does not have such expressivity bottleneck.

$$cMAF : x_d = \mu(x_{1:d-1}, C) + \sigma(x_{1:d-1}, C)z_d. \qquad (2)$$

Other flavor of normalizing flow methods builds upon invertible convolutions such as 1×1 in (Glow) [25] and $d \times d$ in [26]. The work in [27] modified Glow to work with external conditions for structured output learning, yielding a non-autoregressive normalizing flow model.

3. HCNAF

We propose Hyper-Conditioned Neural Autoregressive Flow (HCNAF), a novel autoregressive flow where a transformation between $X = [x_1, x_2, ..., x_D] \in \mathbb{R}^D$ and $Z = [z_1, z_2, ..., z_D] \in \mathbb{R}^D$ is modeled using a non-linear neural network $f(X; \theta) = Z$ whose parameters θ are determined by arbitrarily complex conditions $C \in \mathbb{R}^{D_c}$ in non-autoregressive fashion, via a separate neural network $f_H(C) = \theta$. $f_H(C)$ is designed to compute the parameters for $f()$, thus being classified as an hyper-network [28]. HCNAF models a conditional joint distribution $p(x_1, x_2, ..., x_D|C)$ autoregressively on $x_{1:D}$, by factorizing it over D conditional distributions $\prod_{d=1}^{D} p(x_d|x_{1:d-1}, C)$.

NAF [3] and HCNAF both use neural networks but those are different in probability modeling, conditioner network structure, and flow transformation as specified below:

$$p(x_1,x_2,...,x_D) = \prod_{d=1}^{D} p(x_d|x_{1:d-1}), \\ \left. f_c(x_{1:d-1}) = \theta_d, \\ f(x_d; \theta_d) = z_d, \right\} \quad \text{NAF} \quad (3)$$

$$p(x_1,x_2,...,x_D|C) = \prod_{d=1}^{D} p(x_d|x_{1:d-1},C), \\ \left. f_H(C) = \theta, \theta_d \in \theta, \\ f(x_d; x_{1:d-1}, \theta_d) = z_d. \right\} \quad \text{HCNAF} \quad (4)$$

In Equations 3, *NAF* uses a conditioner network f_c to obtain the parameters θ_d for the transformation between x_d and z_d, which is parameterized by autoregressive conditions $x_{1:d-1}$. In contrast, in Equations 4, HCNAF models the transformation to be parameterized on both $x_{1:d-1}$, and an arbitrarily large external conditions C in non-autoregressive fashion via the hyper-network f_H. For probability modeling, the difference between the two is analogous to the difference between VAE [17] and conditional VAE [29], and that between GAN [18] and conditional GAN [24].

As illustrated in Figure 1, HCNAF consists of two modules: 1) a neural-net based conditional autoregressive flow, and 2) a hyper-network which computes the parameters of 1). The modules are detailed in the following sub-sections.

3.1. NN-based Conditional Autoregressive Flow

The proposed conditional AF is a bijective neural-network $f(X; \theta) = Z$, which models transformation between random variables X and latent variables Z. The network parameters $\theta := [\mathbf{W}, \mathbf{B}]$ are determined by the hyper-network $f_H(C) = \theta$. The main difference between regular neural nets and flow models is the invertibility of $f^{-1}(Z) = X$ as regular networks are not typically invertible.

The conditional AF is shown in Figure 2. In each dimension d of the flow, the bijective transformation between x_d and z_d are modeled with a multi-layer perceptron (MLP) with n hidden layers as follows:

$$x_d \leftrightarrow h_d^{l_1} \leftrightarrow h_d^{l_2} \leftrightarrow ... \leftrightarrow h_d^{l_n} \leftrightarrow z_d (= h_d^{l_{n+1}}). \quad (5)$$

The connection between two adjacent hidden layers $h_d^{l_k}$ and $h_d^{l_{k-1}}$ is defined as:

$$h_d^{l_k} = \phi(W_{dd}^{l_k} h_d^{l_{k-1}} + \sum_{r=1}^{d-1} (W_{dr}^{l_k} h_r^{l_{k-1}}) + B_d^{l_k}), \quad (6)$$

where subscript and superscript each denotes flow number and layer number. Specifically, $h_d^{l_k}$ is the hidden layer l_k
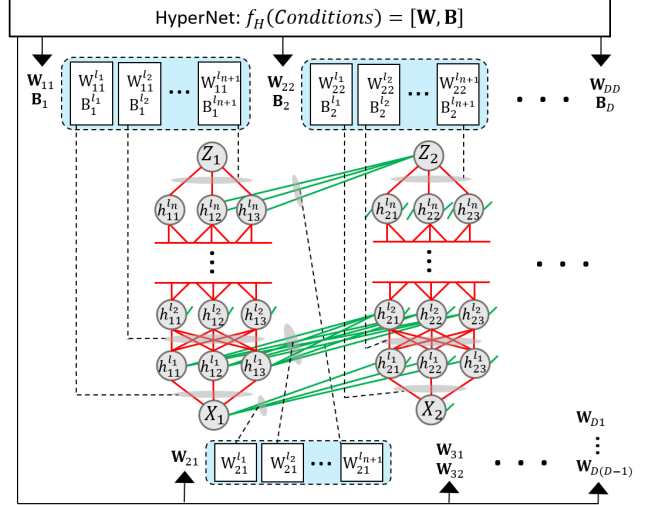

Figure 2: HCNAF's conditional AF model f is a neural-net whose parameters are determined by a hyper-network f_H. The figure describes a D dimensional conditional AF with n hidden layers with 3 nodes. The dash lines refer to connections from f_H to *parameters* of f. Red lines between adjacent hidden layers $h_d^{l_{k-1}}, h_d^{l_k}$ ($\forall d, 1 \leq k \leq n+1$) indicate that $W_{dd}^{l_k}$ is strictly positive. Green lines between layers $h_a^{l_{k-1}}, h_b^{l_k}$ in different flow dimensions ($1 \leq a < b \leq D, 1 \leq k \leq n+1$) have no such constraint (i.e., $W_{ba}^{l_k}$ is unconstrained.)

of the d-th flow. $W_{dr}^{l_k}$ and $B_d^{l_k}$ denote the weight matrix which defines contributions to the hidden layer l_k of the d-th flow from the hidden layer l_{k-1} of the r-th flow, and the bias matrix which defines the contributions to the hidden layer l_k of the r-th flow. Finally, $\phi()$ is an activation function.

The connection between x_d and the first hidden layer, and between the last hidden layer and z_d are defined as:

$$h_d^{l_1} = \phi(W_{dd}^{l_1} x_d + \sum_{r=1}^{d-1} (W_{dr}^{l_1} x_r) + B_d^{l_1}), \\ z_d = W_{dd}^{l_{n+1}} h_d^{l_n} + \sum_{r=1}^{d-1} (W_{dr}^{l_{n+1}} h_r^{l_n}) + B_d^{l_{n+1}}. \quad (7)$$

h^{l_k} are the hidden units at the hidden layer l_k across all flow dimensions $d = 1 : D$ and are expressed as:

$$h^{l_k} = \phi(W^{l_k} h^{l_{k-1}} + B^{l_k}), \quad (8)$$

where W^{l_k} and B^{l_k} are the weights and biases matrices at the hidden layer l_k across all flow dimensions:

$$W^{l_k} = \begin{bmatrix} W_{11}^{l_k} & \mathbf{0} & \dots & \mathbf{0} \\ W_{21}^{l_k} & W_{22}^{l_k} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ W_{D1}^{l_k} & W_{D2}^{l_k} & \dots & W_{DD}^{l_k} \end{bmatrix}, \quad B^{l_k} = \begin{bmatrix} B_1^{l_k} \\ B_2^{l_k} \\ \vdots \\ B_D^{l_k} \end{bmatrix}. \quad (9)$$

Likewise, **W** and **B** denote the weights and biases matrices for all flow dimensions across all the layers. Specifically, $\mathbf{W} := \{\forall k : W^{l_k}\}$ and $\mathbf{B} := \{\forall k : B^{l_k}\}$.

Finally, $Z = f(X)$ is obtained by computing the terms from Equation 8 for all the network layers, from the first $X = h^{l_0}$ to the last layer, $Z = h^{l_{n+1}}$.

We designed HCNAF so that the hidden layer units $h^{l_k}_{1:D}$ are connected to the hidden units of previous layers $h^{l_{k-1}}_{1:D}$, inspired by *BNAF*, as opposed to taking $h^{l_{0:n+1}}_{d}$ as inputs to a separate hyper-network to produce $h^{l_{0:n+1}}_{d+1}$ over $d = 1 : D$, such as presented in *NAF*. This approach avoids running the hyper-network D times; an expensive operation for large hyper-networks. By designing the hyper-network to output $h^{l_{0:n+1}}_{1:D}$ all at once, we reduce the computation load, while allowing the hidden states across all layers and all dimensions to contribute to the flow transformation, as x_d is conditioned not only on $x_{1:d-1}$, but also on all the hidden layers $h^{l_{0:n+1}}_{1:d-1}$.

All Flow models must satisfy the following two properties: 1) monotonicity of $f(X) = Z$ to ensure its invertibility, and 2) tractable computation of the jacobian matrix determinant $\left| det \frac{dZ}{dX} \right|$.

3.1.1 Invertibility of the Autoregressive Flow

The monotonicity requirement is equivalent to having $\forall d : \frac{dz_d}{dx_d} > 0$, which is further factorized as:

$$\frac{dz_d}{dx_d} = \frac{dz_d}{dh^{l_n}_d} \prod_{k=1}^{n-1} \frac{dh^{l_{k+1}}_d}{dh^{l_k}_d} \frac{dh^{l_1}_d}{dx_d} = W^{l_{n+1}}_{dd} \prod_{k=0}^{n-1} \frac{dh^{l_{k+1}}_d}{dh^{l_k}_d}, \quad (10)$$

where $\frac{dh^{l_{k+1}}_d}{dh^{l_k}_d} \ \forall k \in \{0, ..., n-1\}$ is expressed as:

$$\frac{dh^{l_{k+1}}_d}{dh^{l_k}_d} = \frac{d\phi(A^{l_{k+1}}_d)}{dA^{l_{k+1}}_d} \frac{dA^{l_{k+1}}_d}{dh^{l_k}_d} = \frac{d\phi(A^{l_{k+1}}_d)}{dA^{l_{k+1}}_d} W^{l_{k+1}}_{dd}. \quad (11)$$

$A^{l_k}_d$ denotes the pre-activation of $h^{l_k}_d$. The invertibility is satisfied by choosing a strictly increasing activation function ϕ (e.g. *tanh* or *sigmoid*) and a strictly positive $W^{l_k}_{dd}$. $W^{l_k}_{dd}$ is made strictly positive by applying an element-wise exponential to all entries in $\forall d, k : W^{l_k}_{dd}$ at the end of the hypernetwork, inspired by [4]. Note that the operation is omitted for the non-diagonal elements of $W^{l_k}_{ij}, i \neq j$.

3.1.2 Tractable Computation of Jacobian Determinant

The second requirement for flow models is to efficiently compute the jacobian matrix determinant $\left| det \frac{dZ}{dX} \right|$, where:

$$\frac{dZ}{dX} = \frac{dZ}{dh^{l_n}} \prod_{k=0}^{n-1} \frac{dh^{l_{k+1}}}{dh^{l_k}} = W^{l_{n+1}} \prod_{k=0}^{n-1} \frac{d\phi(A^{l_{k+1}})}{dA^{l_{k+1}}} W^{l_{k+1}}. \quad (12)$$

Since we designed $W^{l_{k+1}}$ to be lower-triangular, the product of lower-triangular matrices, $\frac{dZ}{dX}$, is also lower-triangular, whose log determinant is then simply the product of the diagonal entries: $log \left| det \frac{dZ}{dX} \right| = log \left| \prod_{d=1}^{D} \frac{dz_d}{dx_d} \right| = \sum_{d=1}^{D} log(\frac{dz_d}{dx_d})$, as our formulation states $\forall d : \frac{dz_d}{dx_d} > 0$. Finally, $log(\frac{dz_d}{dx_d})$ is expressed via Equations 10 and 11.

$$log \left(\frac{dz_d}{dx_d} \right) = log \left(W^{l_{n+1}}_{dd} \prod_{k=0}^{n-1} \frac{d\phi(A^{l_{k+1}}_d)}{dA^{l_{k+1}}_d} W^{l_{k+1}}_{dd} \right). \quad (13)$$

Equation 13 involves the multiplication of matrices in different sizes; thus cannot be broken down to a regular log summation. To resolve this issue, we utilize log-sum-exp operation on logs of the matrices in Equation 13 as it is commonly utilized in the flow community (e.g. *NAF* [3] and *BNAF* [4]) for numerical stability and efficiency of the computation. This approach to computing the jacobian determinant is similar to the one presented in *BNAF*, as our conditional AF resembles its flow model.

As HCNAF is a member of the monotonic neural-net based autoregressive flow family like *NAF* and *BNAF*, we rely on the proofs presented *NAF* and *BNAF* to claim that HCNAF is also a universal distribution approximator.

3.2. Hyper-conditioning and Training

The key point from Equation 5 - 13 and Figure 2 is that HCNAF is constraint-free when it comes to the design of the hyper-network. The flow requirements from Sections 3.1.1 and 3.1.2 do not apply to the hyper-network. This enables the hyper-network to grow arbitrarily large and thus to scale up with respect to the size of conditions. The hyper-network $f_H(C)$ can therefore be an arbitrarily complex neural network with respect to the conditions C.

We seek to learn the target distribution $p(X|C)$ using HCNAF by minimizing the negative log-likelihood (NLL) of $p_{model}(X|C)$, i.e. the cross entropy between the two distributions, as in:

$$L := -E_{X \sim p(X|C)}[log\, p_{model}(X|C)] = H(p, p_{model}). \quad (14)$$

Note that minimizing the NLL is equivalent to minimizing the (forward) KL divergence between the data and the model distributions $D_{KL}(p(X|C)||p_{model}(X|C))$, as $H(p, q) = H(p) + D_{KL}(p||q)$ where $H(p)$ is bounded.

4. Probabilistic Occupancy Map Forecasting

In Section 3, we showed that HCNAF can accommodate high-dimensional condition inputs for conditional probability density estimation problems. We leverage this capability to tackle the probabilistic occupancy map (POM) of actors in self-driving tasks. This problem operates on over one million dimensions, as spatio-temporal multi-actor images are part of the conditions. This section describes the design of HCNAF to support POM forecasting. We formulate the problem as follows:

$$p(X_t^{A_i}|C) \quad \text{with} \quad C := \{X_{-\tau:0}^{A_i}, X_{-\tau:0}^{A_{\forall j \neq i}}, \Omega\}, \qquad (15)$$

where $X_{-\tau:0}^{A_i} \in \mathbb{R}^{\tau \times d_i}$ is the past states, with d_i as the dimension of the observed state, over a time span τ. $X_{-\tau:0}^{A_{\forall j \neq i}} \in \mathbb{R}^{\tau \times N_A}$ denotes the past states for all N_A neighboring actors over the same time span. $\Omega \in \mathbb{R}^{N_C \times H \times W}$ encodes contextual static and dynamic scene information extracted from map priors (e.g. lanes and stop signs) and/or perception modules (e.g. bounding boxes for actors) onto a rasterized image of size H by W with N_C channels. However comprehensive, the list of conditions in C is not meant to be limitative; as additional cues are introduced to better define actors or enhance context, those are appended to the conditions. We denote $X_t^{A_i} := [x_t^{A_i}, y_t^{A_i}]$ as the location of an actor A_i over the 2D bird's-eye view (bev) map at time t, by adapting our conditional AF to operate on 2 dimensions. As a result, the joint probability is obtained via autoregressive factorization given by $p(x_t, y_t|C) = p(y_t|x_t, C)p(x_t|C)$.

It's possible to compute $p(x_{t_{1:T}}, y_{t_{1:T}}|C)$, a joint probability over multiple time steps via Equation 4, but we instead chose to compute $p(x_t, y_t|C)$ (i.e. a marginal probability distribution over a single time step) for the following reasons:

1. Computing $p(x_{t_{1:T}}, y_{t_{1:T}}|C)$ implies the computation of $p(x_t, y_t|x_{1:t-1}, y_{1:t-1}, C)$ autoregressively. While this formulation reasons about the temporal dependencies between the history and the future, it is forced to make predictions on x_t, y_t dependent on unobserved variables $x_{1:t-1}$ and $y_{1:t-1}$. The uncertainties of the unobserved variables have the potential to push the forecast x_t, y_t in the wrong direction.

2. The computation of $p(x_{t_{1:T}}, y_{t_{1:T}}|C)$ is intractable in nature since it requires a marginalization over all variables $t = [0, t-1]$. We note that $p(x_t, y_t|C) = \int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} p(x_1, y_1, \ldots, x_t, y_t) dx_1 \ldots dy_{t-1}$ is practically impossible to integrate over.

In order to predictions predictions all time $t = 0 : T$, we simply incorporate a time variable as part of the conditions.

In addition to POMs, HCNAF can be used to sample trajectories using the inverse transformation $f^{-1} : Z \sim N(\mathbf{0}_D, \mathbf{I}_{DxD}) \to X$. The exact probabilities of the generated trajectories can be computed via Equation 1. However, it is not trivial to obtain the inverse flow since a closed form solution is not available. A solution is to use a numerical approximation or to modify the conditional AF of HCNAF; which is not discussed in this work.

5. Experiments

In this paper, five experiments (including three experiments on publicly available datasets) of various tasks and complexities are presented to evaluate HCNAF. For all, we provide quantitative (NLL, D_{KL}) and qualitative measures (visualizations; except MNIST as the dimension is large). We start the section by demonstrating the effectiveness of HCNAF on density estimation tasks for two *Toy Gaussians*. We then verify the scalability of HCNAF by tackling more challenging, high dimensional ($C > \mathbb{R}^{1,000,000}$) POM forecasting problems for autonomous driving. For POM forecasting, we rely on two datasets: 1) *Virtual Simulator*: a simulated driving dataset with diverse road geometries, including multiple road actors designed to mimic human drivers. The scenarios are based on real driving logs collected over North-American cities. 2) *PRECOG-Carla*: a publicly available dataset created using the open-source Carla simulator for autonomous driving research [10]. Lastly, we run a conditional density estimation task on MNIST which is detailed in the supplementary material.

5.1. Toy Gaussians

We conducted two experiments to demonstrate the performance of HCNAF for density estimations. The first is an experiment from *NAF* paper [3], and aims to show the model's learning ability for three distinct probability distributions over a 2D grid map, $p(x, y)$. The non-linear distributions are spatially distinct groups of gaussians. In the second experiment, we demonstrate how HCNAF can generalize its outputs over previously *unseen* conditions.

5.1.1 Toy Gaussians: Experiment 1

Table 1: NLL for the experiment depicted in Figure 3. Lower values are better.

	AAF	**NAF**	HCNAF (ours)
2 by 2	6.056	**3.775**	3.896
5 by 5	5.289	**3.865**	3.966
10 by 10	5.087	**4.176**	4.278

Results from Figure 3 and Table 1 show that HCNAF is able to reproduce the three nonlinear target distributions, and to achieve comparable results as those using *NAF*, albeit
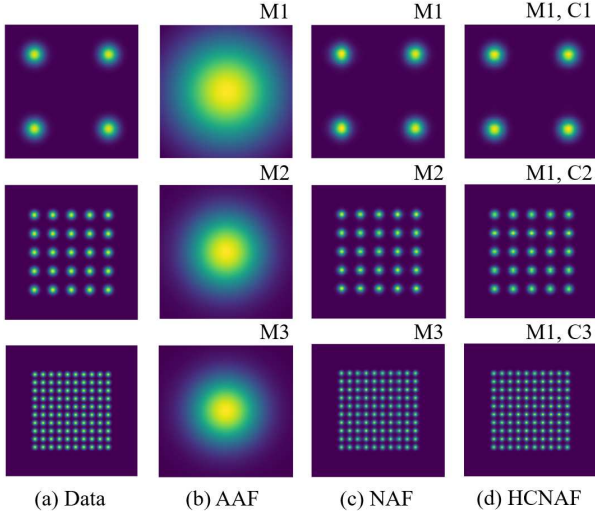
(a) Data   (b) AAF   (c) NAF   (d) HCNAF

Figure 3: Density estimation tasks using three gaussian distributions. In order to reproduce the probability distributions $p(x,y)$, HCNAF uses a **single** model and three conditions, whereas NAF requires three **different** models, i.e. trained separately. In the figure, M: model and C: condition.

(a) Training data ($C_{1:5}$)   (b) Pdfs generated for **seen** $C_{1:5}$ (c) Pdfs generated for **unseen** $C_{6:9}$
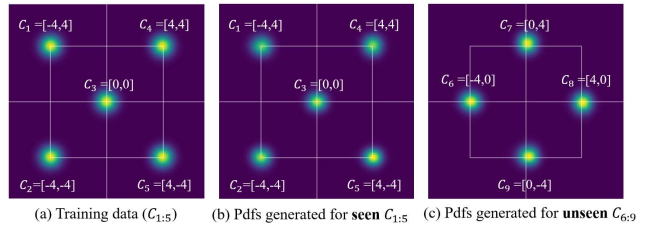
Figure 4: HCNAF model trained with 5 different discrete conditions $C_{train} = \{C_1,...,C_5\}$, where C_i represents the mean of an isotropic bivariate gaussian pdf. a) $p(x,y|C_{train})$, b) $p_{model}(x,y|C_{train})$ c) predictions on previously **unseen** conditions $p_{model}(x,y|C_{unseen})$, $C_{unseen} := \{C_6,...,C_9\}$.

Table 2: Differences between the target and predicted distributions in terms of cross entropy and KL divergence for Figure 4.

	$p(x,y)$	$p_{HCNAF}(x,y\|C_i)$	
C	-	$C_i \in C_{train}$	$C_i \in C_{unseen}$
$H(p)$	1.452	-	-
$H(p,p_{model})$	-	1.489	1.552
$D_{KL}(p\|\|p_{model})$	-	0.037	0.100

with a small increase in NLL. We emphasise that HCNAF uses a **single** model (with a 1-dimensional condition variable) to produce the 3 distinct pdfs, whereas AAF (Affine AF) and *NAF* used 3 distinctly trained models. The autoregressive conditioning applied in HCNAF is the same as for the other two models. The hyper-network of HCNAF uses $C \in \{0,1,2\}$ where each value represents a class of 2-by-2, 5-by-5, and 10-by-10 gaussians.

5.1.2 Toy Gaussians: Experiment 2

From the density estimation experiment shown in Figure 4, we observed that HCNAF is capable of *generalization* over **unseen** conditions, i.e. values in the condition terms that were intentionally omitted during training. The experiment was designed to verify that the model would interpolate and/or extrapolate probability distributions beyond the set of conditions it was trained with, and to show how effective HCNAF is at reproducing both the target distribution $p(x,y|C_i)$ for $C_i \in C_{train}$. As before, we trained a single HCNAF model to learn 5 distinct pdfs, where each pdf represents a gaussian distribution with its mean (center of the 2D gaussian) used as conditions $C := (x_c, y_c) \in \mathbb{R}^2$ and with an isotropic standard deviation σ of 0.5.

For this task, the objective function is the maximization of log-likelihood, which is equivalent to the maximization of the KL divergence $-E_{(x,y)\sim N(C_i, 0.25 \cdot I)}[log\, p_{model}(x,y|C_i)]$ where C_i is uniformly sampled from the set of conditions $C_{train} := \{C_1, C_2,...,C_5\}$. Table 2 provides quantitative results from the cross entropy

$H(p, p_{model})$ and a KL divergence $D_{KL}(p||p_{model})$. Note that $H(p, p_{model})$ is lower-bounded by $H(p)$ since $H(p, p_{model}) = H(p) + D_{KL}(p||p_{model})$. The differential entropy $H(p)$ of an isotropic bi-variate Gaussian distribution $p(x,y)$ and is computed using: $H(p) = 0.5 \cdot ln(2\pi e(\sigma)^2)^2$. The results show that HCNAF is able to generalize its predictions for unseen conditions as shown by the small deviation of $H(p, p_{model})$ from its lower bound $H(p)$.

5.2. Forecasting POM for Autonomous Driving

Through changes in the hyper-network, we show how HCNAF can be scaled up to tackle the POM forecasting problems for autonomous driving. The condition C is now significantly larger when compared to that from the experiments in Section 5.1, as shown in Equation 15. C now includes information extracted from various sensors (lidar, camera), maps (lanes, stop-signs), and perception object detections (expressed as bounding boxes for actors), with a total dimension is in the millions of parameters. As per its design, HCNAF's AF network is unaffected by the increase in conditional dimensions.

Figure 5 depicts the customized hyper-network used for the POM forecasting task. The hyper-network takes perception inputs as the condition C, and outputs a set of network parameters $\mathbf{W}$ and $\mathbf{B}$ for the subsequent HCNAF's conditional AF $f(\cdot;\mathbf{W},\mathbf{B}): X \leftrightarrow Z \sim N(0, I_{2x2})$. The inputs come
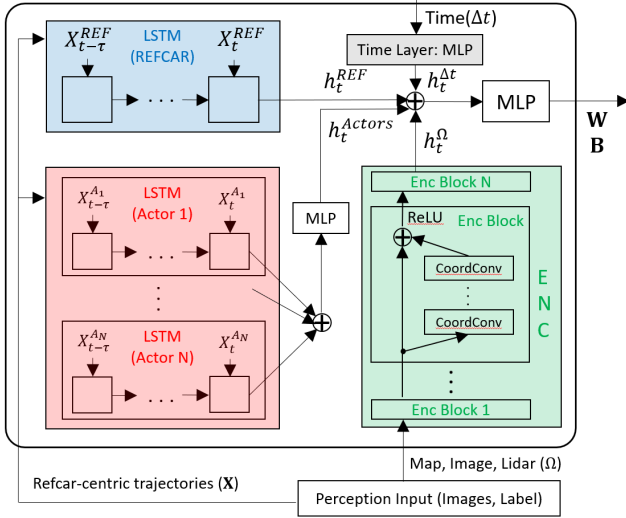
Figure 5: Design for the Hyper-network of HCNAF used in the POM forecasting problem.

from various sensors (lidar or camera) through a perception module and also from prior map information. Specifically, C is formed with 1) the bev images which include lanes, stop-signs, lidar data, and actor bounding boxes in a 2D grid map (see figures presented in the supplementary material and Figure 6), and 2) the states of actors in actor-centric pixel coordinates. The perception module used reflects other standard approaches for processing multi-sensor data, such as [30]. The hyper-network consists of three main components: 1) LSTM modules, 2) an *encoder* module, and 3) a *time* module. The outputs of the three modules $h_t^{REF}, h_t^{Actors}, h_t^{\Omega}, h_t^{\Delta t}$ are concatenated and fed into an MLP, which outputs **W** and **B**, as shown in Figure 5.

The *LSTM module* takes the states of an actor A_i in the scene $X_{t-\tau:t}^{A_i}$ where $X_t^{A_i} := [x_t^{A_i}, y_t^{A_i}, sin(\theta_t^{A_i}), cos(\theta_t^{A_i}), v^{A_i}]$ to encode temporal dependencies and trends among the state parameters. A total of $N+1$ LSTM modules are used to model the N actors and the reference car for which we produce the POM forecasts. The resulting outputs are $h_t^{REF} \in \mathbb{R}^{d_{REF}=20}$, and $h_t^{Actors} \in \mathbb{R}^{d_{Actors}=18}$.

The *encoder module* takes in the bev images denoted as Ω. The role of this module is to transform the scene contexts into a one-dimensional tensor that is concatenated with other parameters of our conditional AF flow module. We use residual connections to enhance the performance of our encoder as in [30]. Since our hyper-network works with Cartesian (x,y) space and pixel (image) space, we use coordinate convolution (coordconv) layers as in [31] to strengthen the association between the two data. Overall, the *encoder* network consists of 4 encoder blocks, and each encoder block consists of 5 coordconv layers with residual connections, max-pooling layers, and batch-normalization

layers. The resulting output is $h_t^{\Omega} \in \mathbb{R}^{d_{\Omega}=64}$.

Lastly, the *time layer* adds the forecasting time $\Delta t \in \mathbb{R}^1$, i.e. time span of the future t away from the reference (or present) time $t = 0$. In order to increase the contribution of the time condition, we apply an MLP which outputs a hidden variable for the time condition $h_t^{\Delta t} \in \mathbb{R}^{d_t=10}$.

Forecasting POM with a Virtual Simulator Dataset

Using the POM hyper-network, HCNAF was trained on an internal dataset that we call *Virtual Simulator*. The dataset is comprised of bev images of size $N \times 256 \times 256$, where N may include all, or a subset of the following channels: stop signs, street lanes, reference car locations, and a number of actors. We also add the history of actor states in pixel coordinates, as discussed in the previous sub-section. For each of the vehicles/actors, we apply a coordinate transformation to obtain actor-centric labels and images for training. The vehicle dataset includes parked vehicles and non-compliant road actors to introduce common and rare events (e.g. sudden lane changes or sudden stopping in the middle of the roads). We produce POM for all visible vehicles, including parked vehicles, and non-compliant actors, even if those are not labeled as such. Note that the dataset was created out of several million examples, cut into snippets of 5 seconds in duration. We present a figure which depicts POM forecasts for three scenarios sampled from the test set and a table for an ablation study to show the impact of different hyper-networks inputs on the POM forecasting accuracy in the supplementary material.

As discussed in Section 4, HCNAF produces not only POM, but also trajectory samples via the inverse transformation of the conditional AF f^{-1}. As we advocate the POM approach, we do not elaborate further on the trajectory based approach using HCNAF.

Forecasting POM with PRECOG-Carla Dataset

We trained HCNAF for POM forecasting on the PRECOG-Carla Town01-train dataset and validated the progress over Town01-val dataset [10]. The hyper-network used for this experiment was identical to the one used for the *Virtual Simulator* dataset, except that we substituted the bev images with two overhead lidar channels; the above ground and ground level inputs. The *encoder* module input layer was updated to process the lidar image size (200x200) of the PRECOG-Carla dataset. In summary, C included the lidar data, and the history of the reference car and other actors.

To evaluate the performance of the trained models, [10] used the *extra nats* $\hat{e}$ metric for the likelihood estimation instead of NLL. $\hat{e}$ is a normalized, bounded likelihood metric defined as $\hat{e} := [H(p', p_{model}) - H(\eta)]/(T \cdot A \cdot D) \geq 0$, where $H(p', p_{model}), T, A, D$ each represents the cross-entropy between p' (perturbed with an isotropic gaussian
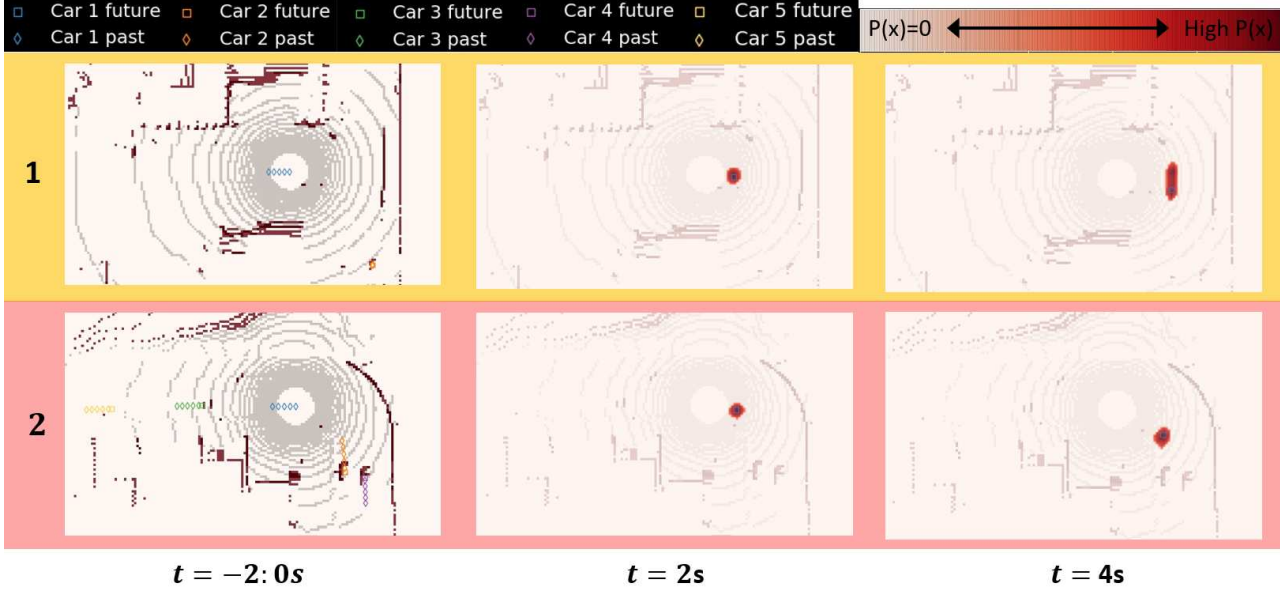
Figure 6: Visualized POM forecasts on *PRECOG-Carla* dataset using the HCNAF model described in Table 3 (with lidar). Left: 2 seconds history of cars. Center and right: probabilistic occupancy predictions for Car 1 at $t = 2$ and 4 secs depicted as red heatmaps, with actor ground truth (blue square) overlayed. Note that we only forecast POMs for the car 1 as the lidar data is only available for the car 1. In the examples 1, the car 1 enters a 3-way intersection, HCNAF uses the road geometry coming from the lidar data and correctly forecasts that there are two natural modes (left-turn & right-turn) and depict the probabilities of positions as heatmaps. In example 2, HCNAF uses the curved road geometry and successfully forecasts the occupancy probabilities of the car 1. More results on POM visualizations are presented in the supplementary material.

noise) and p_{model} [10], prediction horizon, number of actors, and dimension of the actor position. We used the same $\eta = N(\mathbf{0}, 0.01^2 \cdot \mathbf{I})$ as cited, whose differential entropy is analytically obtained using $H(\eta) = 0.5 \cdot T \cdot A \cdot D \cdot ln(2\pi e|\Sigma|)$. We computed $p(x_t, y_t|C)$ over all time-steps available in the dataset. The results are presented in Table 3 and Figure 6.

It is worth mentioning that there exists works including [8], [32] that used the PRECOG-Carla dataset. However, most reported trajectory-based predictions metrics (MSE, MinMSD, etc). To the best of our knowledge, the only available benchmark for *NLL* on the PRECOG dataset is what is presented in this paper (PRECOG-ESP). Since we take the occupancy-based approach, trajectory-based metrics are not applicable to our approach.

Table 3: PRECOG-CARLA Town01 Test, 1 agent, mean $\hat{e}$

Method	Test ($\hat{e}$): Lower is better
PRECOG-ESP, no lidar	0.699
PRECOG-ESP	0.634
HCNAF, no lidar (ours)	0.184
HCNAF (ours)	**0.114** (5+ times lower)

We believe that HCNAF performed better than *PRE-COG-ESP*, which is a state-of-the-arts prediction model in autonomous driving, by taking advantage of HCNAF's expressivity comes from non-linear flow transformations and having condition terms affecting the hidden states of all layers of HCNAF's conditional AF. Note, *PRECOG* utilizes bijective transformations $f : X \leftrightarrow Z$ that is rooted in affine AF, similar to *cMAF* (See Equation 2). We also believe that the HCNAF's generalization capability is a contributing factor that explains how HCNAF is able to estimate probability densities conditioned on previously unseen contexts.

6. Conclusion

We present HCNAF, a novel universal distribution approximator tailored to model conditional probability density functions. HCNAF extends neural autoregressive flow [3] to take arbitrarily large conditions, not limited to autoregressive conditions, via a hyper-network which determines the network parameters of HCNAF's AF. By modeling the hyper-network constraint-free, HCNAF enables it to grow arbitrarily large and thus to scale up with respect to the size of non-autoregressive conditions. We demonstrate its effectiveness and capability to generalize over unseen conditions on density estimation tasks. We also scaled HCNAF's hyper-network to handle larger conditional terms as part of a prediction problem in autonomous driving.

References

[1] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016. 1, 2

[2] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017. 1, 2

[3] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2083–2092, 2018. 1, 2, 3, 4, 5, 8

[4] Nicola De Cao, Ivan Titov, and Wilker Aziz. Block neural autoregressive flow. *arXiv preprint arXiv:1904.04676*, 2019. 1, 2, 4

[5] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001. 1

[6] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 2

[7] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 947–956. PMLR, 29–31 Oct 2018. 1, 2

[8] Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple futures prediction. *arXiv preprint arXiv:1911.00997*, 2019. 1, 2, 8

[9] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 1, 2

[10] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. *arXiv preprint arXiv:1905.01296*, 2019. 1, 2, 5, 7, 8

[11] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1349–1358, 2019. 1, 2

[12] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. 1, 2

[13] Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8483–8492, 2019. 1, 2

[14] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. Interaction-aware multi-agent tracking and probabilistic behavior prediction via adversarial learning. *arXiv preprint arXiv:1904.02390*, 2019. 1, 2

[15] Ajay Jain, Sergio Casas, Renjie Liao, Yuwen Xiong, Song Feng, Sean Segal, and Raquel Urtasun. Discrete residual flow for probabilistic pedestrian behavior prediction. *arXiv preprint arXiv:1910.08041*, 2019. 1

[16] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 2

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2, 3

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2, 3

[19] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2

[20] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018. 2

[21] Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017. 2

[22] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016. 2

[23] Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018. 2

[24] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2, 3

[25] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018. 2

[26] Emiel Hoogeboom, Rianne van den Berg, and Max Welling. Emerging convolutions for generative normalizing flows. *arXiv preprint arXiv:1901.11137*, 2019. 2

[27] You Lu and Bert Huang. Structured output learning with conditional generative flows. *arXiv preprint arXiv:1905.13288*, 2019. 2

[28] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 2

[29] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015. 3

[30] 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*, pages 770–778, 2016. 7

[31] 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. In *Advances in Neural Information Processing Systems*, pages 9605–9616, 2018. 7

[32] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 8