ADVANCING PROBABILISTIC MODELS FOR APPROXIMATE AND EXACT INFERENCE

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

ROBERT GIAQUINTO

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

ARINDAM BANERJEE

July 2021 – version 1.0

ACKNOWLEDGMENTS

First and foremost, I want to thank my advisor Professor Arindam Banerjee—I couldn't imagine having a mentor with a greater care for his students and their research. I feel incredibly fortunate that I had Arindam as a role model for conducting research. Pursing a PhD is really hard. Sometimes my good ideas didn't work. Sometimes I only had scattered ideas and unclear path forward. In those difficult situations I knew could count on my discussions with Arindam to bring clarity of thought, renewed optimism, and appreciation for a formidable problem. Thank you Professor Banerjee for always making me feel supported and that the work we did mattered.

Next, I want to thank my kind and generous committee members Professor Catherine Zhao, Professor Junaed Sattar, and Professor Snigdhansu Chatterjee for agreeing to be in my committee. Thank you for your support and encouragement, insightful questions raised, and constructive discussion. I also want to thank Professor Maria Gini and Dr. Genevieve Melton-Meaux for serving on my Master's committee and being someone I could (and repeatedly did) go to for questions, recommendations, and advice.

A big thank you to my UMN friends and lab mates throughout these years: Sijie He, Yingxue Zhou, Xinyan Li, Tiancong Chen, Qilong Gu, Vidyashankar Sivakumar, Konstantina Christakopoulou, Sheng Chen, Farideh Fazayeli, Hardik Goel, Amir Taheri, Chintan Dalal, Somya Sharma, Adam Stewart, Mengdie Wang, and Arun Kumar. I will miss chatting about research, celebrating victories, and spending long days together at our internal workshops to dive-deep into new topics. I also want to thank my friends and collaborators in GroupLens Professor Lana Yarosh, Haiwei Ma, Estelle Smith, and Zachary Levonian for sharing your awesome work and being great people.

I had the pleasure of working at three internships during my PhD study. At Adobe, thank you Hsiang-Yu Yang, Wuyang Dai, and Jun He for selecting me and the opportunity to work on exciting applied problems. At HRL Laboratories, thank you Tsai-Ching Lu for entrusting me on a such a great research project, it was a true pleasure to work with you. At Thomson Reuters, thank you Gayle McElvain and Tonya Custis for the guidance and freedom to work on a delightful summer internship project.

Before pursuing a PhD I was very fortunate to work with Dr. Alex Ushveridze on his research team at Capella Education Company. Thank you Alex for mentoring me, your infectious love of research inspired me to pursue graduate school and ultimately a PhD. As an undergraduate at St. Olaf College, Professor Julie Legler advised me on my very first research opportunity. Thank you Julie for selecting me, for making that first research experience such a positive impact on me, and for giving me the encouragement to pursue a career in research.

I also wouldn't be where I am today without the love and support of my family. I've been incredibly lucky to be surrounded by people who are intensely passionate about my well-being. To my wife Lindsey, thank you for encouraging me to pursue my dream, celebrating the successes, and supporting me during the challenges. To my mom Ann and my dad Jeff, thank you for instilling in me the will to work hard, and to care deeply for those you love. To Charlie, Ella, Tom, Jake, Blaize, Judy, Dave, Nik, and Linda—I love you all.

The research in this thesis was supported by NSF grants OAC-1934634, IIS-1908104, IIS-1563950, IIS-1447566, IIS-1447574, IIS-1422557, CCF-1451986, and we thank the University of Minnesota Supercomputing Institute (MSI) for technical support.

DEDICATION

The greatest moments in life are not concerned with selfish achievements but rather with the things we do for the people we love and esteem.

— Walt Disney

For my parents and Lindsey. Thank you for your never-ending love and support.

ABSTRACT

Probabilistic models have a rich history in machine learning, offering a theoretical and practical framework for learning from observed data. Probabilistic models describe relationships between observed data and latent variables in terms of probability distributions. Practitioners in many fields of science have long been attracted to probabilistic methods as a way to quantify uncertainty in predictions and models, query models via inference, and estimating latent variables. In this thesis, *Advancing Probabilistic Models for Approximate and Exact Inference*, we connect foundational ideas in machine learning, such as probabilistic inference and ensemble learning, with deep learning. More specifically, the focus lies on the design of generative models with likelihood-based objective functions, which offer a solution for overcoming many broader challenges in machine learning—namely, explaining all of the data, efficient data usage, and quantifying uncertainty.

For over two decades graphical models were the predominant paradigm for composing probabilistic models in machine learning. By composing probability distributions as building blocks for larger models, graphical models offer a comprehensible model-building framework that can be tailored to the structure of the data. As a build up to further work, we introduce a novel probabilistic graphical model for analyzing text datasets of multiple authors writing over time. In the era of big data, however, it is necessary to scale such models to large datasets. To that end, we propose an efficient learning algorithm that allows for training and probabilistic inference on text datasets with billions of words with general-purpose computing hardware.

Recently, breakthroughs in deep learning have ushered in an explosion of new successes in probabilistic modeling, with models capable of modeling enormous collections of complex data and generating novel yet plausible data (e.g. new images, text, and speech). One promising direction in likelihood-based probabilistic deep learning is normalizing flows. Normalizing flows use invertible transformations to translate between simple and complex distributions, which allows for exact likelihood calculation and efficient sampling. In order to remain invertible and provide exact likelihood calculations, normalizing flows must be composed of differentiable bijective functions. However, bijections require that the inputs and outputs have the same dimensionality—which can pose significant architectural, memory, and computational costs for high-dimensional data. We introduce *compressive normalizing flows* that are, in the simplest case, equivalent to the probabilistic principal components analysis (PPCA). The PPCA-based compressive flow relaxes the bijective constraints and allows the model to learn a compressed latent representation, while offering parameter updates that are available analytically. Drawing on the connection between PPCA and

Variational Autoencoders (VAE)—a powerful deep generative model, we extend our framework to VAE-based compressive flows for greater flexibility and scalability.

Up until now the trend in normalizing flow literature has been to devise deeper, more complex transformations to achieve greater flexibility. We propose an alternative: *Gradient Boosted Normalizing Flows* (GBNF) model a complex density by successively adding new normalizing flow components via gradient boosting so that each new component is fit to the residuals of the previously trained components. Because each flow component is itself a density estimator, the aggregate GBNF model is structured like a mixture model. Moreover, GBNFs offer a wider, as opposed to strictly deeper, approach that improves existing NFs at the cost of additional training—not more complex transformations.

Lastly, we extend normalizing flows beyond their original unsupervised formulation, and present an approach for learning high-dimensional distributions conditioned on low-dimensional samples. In the context of image modeling, this is equivalent to image super-resolution—the task of mapping a low-resolution (LR) image to a single high-resolution (HR) image. Super-resolution, however, is an ill-posed problem since there are infinitely many HR samples that are compatible with a given LR sample. Approaching super-resolution with likelihood-based models, like normalizing flows, allows us to learn a distribution over all possible HR samples. We present *probabilistic super-resolution (PSR) using normalizing flows* for learning conditional distributions as well as joint PSR where the high- and low-dimensional distributions are modeled simultaneously. However, our approach is not solely for image modeling. Any dataset can be formulated for super-resolution, and using a PSR architecture alleviates challenges commonly associated with normalizing flows such—as the information bottleneck problem, and the inductive biases towards modeling local correlations.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AVI Amortized Variational Inference

CDE Conditional Density Estimation

CDF Cumulative Density Function

DE Density Estimation

DGM Deep Generative Model

ELBO Evidence Lower Bound

EMD Earth Mover's Distance

FID Fréchet Inception Distance

GAN Generative Adversarial Network

GBNF Gradient Boosted Normalizing Flows

GP Gaussian Process

HR High-resolution (image)

KL Kullback–Leibler Divergence

LPIPS Learned Perceptual Image Patch Similarity

LR Low-resolution (image)

MFVI Mean-Field Variational Inference

MLE Maximum Likelihood Estimation

NF Normalizing Flow

PCA Principal Components Analysis

PDF Probabilisty Density Function

PGM Probabilistic Graphical Model

PMF Probabilisty Mass Function

PPCA Probabilistic PCA

PSNR Peak Signal to Noise Ratio

PSR Probabilistic Super-Resolution

RealNVP Real Non-volume Preserving Flow

RMSE Root Mean Squared Error

SR Image Super-Resolution

SSIM Structural Similarity Index Measure

VAE Variational Autoencoder

VI Variational Inference

Part I

PROBABILISTIC REASONING AND DEEP LEARNING

"What I cannot create, I do not understand."

—Richard Feynman

1

INTRODUCTION

The great mathematician John von Neumann said, "the sciences do not try to explain, they hardly even try to interpret, they mainly make models." Throughout science the designing of models of the real-world is an essential activity that helps us clarify complex processes and study observed phenomenon. Modeling in the field of machine learning is no different in spirit, but the focus is on designing algorithms that build models given training data.

GENERATIVE MODELS Probabilistic machine learning models, in particular, closely mimic the more general scientific notation of a model—they allow us to reason about relationships between observed evidence and unobservable *latent* variables (Ghahramani, 2015). Latent variables are important as data rarely tell the entire story, and are often noisy—requiring us to maintain a level of uncertainty in the conclusions we draw from the data. Probabilistic models are not, however, limited to applications that demand probabilistic guarantees on our predictions. With posterior inference we can estimate the values of the latent variables—for example, to identify the major topics within a corpus of documents (Blei, Ng, and Jordan, 2003) or profile a user's preference in movies (Lim and Teh, 2007). Because probabilistic models reflect the process that generated the training data, they are the premier method for synthesizing novel yet plausible data (e. g. new texts, images, and speech), and as such are often referred to as *generative models*. Generating novel and plausible samples is intrinsically valuable, while also beneficial when training data is difficult or expense to obtain, as well as for agents that plan and simulate interactions with their environment.

GENERATIVE VS. DISCRIMINATIVE MODELS Unlike a discriminative model, which learns to map inputs to outputs, a generative model attempts to learn the more general joint distribution over all variables. Generative models, however, have long been considered inferior predictors relative to their discriminative counterparts. Early generative models for classification and regression were considered best deployed in small data settings (Bishop and Lasserre, 2007; Ng and Jordan, 2001). When the task at hand is not solely to discriminate, the generative approach offers a number of advantages: they can be highly intuitive and interpretable, queried to confirm or reject theories, express a causal relationship (Kingma and Welling, 2019), they are more robust to adversarial attacks (Li, Bradshaw, and Sharma, 2019), and have stronger assumptions

on the data and hence a higher asymptotic bias when the model is incorrect (Banerjee, 2007). Combining generative and discriminative models can capture the robustness and data generating process of a generative model, but retain the predictive perform of the discriminative counterpart (Bishop and Lasserre, 2007; Che et al., 2020).

CURRENT STATE: DEEP GENERATIVE MODELS Breakthroughs in deep learning lead to many new successes in generative modeling. These new *deep generative models* can learn rich latent representations of data that benefit downstream tasks, and provide a mechanism for sampling new data that is indistinguishable from real data. Variational autoencoders (VAE, Kingma and Welling 2014; Kingma and Welling 2019; Rezende, Mohamed, and Wierstra 2014) and generative adversarial networks (GAN, Goodfellow, Pouget-Abadie, and Mirza 2014) represent two major paradigms in the generative model revolution. Despite both being deep generative models, GANs and VAEs differ significantly. First, GANs are an *implicit* density estimator which is a stochastic procedure that directly generates the data (Mohamed and Lakshminarayanan, 2017). On the other hand, VAEs represent an *explicit* method to approximate complex posteriors that (1) specify a distribution over the latent variables (e.g. a factorial Gaussian), and (2) provide a log-likelihood function. Early VAEs lacked capacity and were overly entropic which resulted in generating blurry looking images. GANs, on the other hand, grew in popularity quickly because of their ability to generate sharp, realistic images (Creswell et al., 2018; Goodfellow, 2016). However, GANs cannot calculate likelihoods, and exhibit *mode-seeking* behavior wherein the GAN attempts to only explain a single mode of the data really well—as opposed to being a *complete* model of the data. Lastly, GANs can be challenging to train due to the mode collapse problem (Arjovsky and Bottou, 2017; Arjovsky, Chintala, and Bottou, 2017). Models with likelihood-based objectives, like VAEs, must explain all of the data (i. e. *mode-covering* behavior), can calculate the likelihood of samples, are more stable to train (Kingma, 2017), and generate sharp and realistic samples when using more recent techniques and higher capacity models (Vahdat and Kautz, 2020).

Normalizing flows (NF) recently emerged as an alternative to these paradigms, offering a tractable method for both explicit density estimation (Dinh, Krueger, and Bengio, 2015; Dinh, Sohl-Dickstein, and Bengio, 2017; Tabak and Turner, 2013; Tabak and Vanden-Eijnden, 2010) as well as posterior approximation (Rezende and Mohamed, 2015). By mapping a simple base density through a series of invertible transformations, flows provide an exact method of sampling and density evaluation. Flows are stable to train and elicit a maximum likelihood objective function, unlike the VAE which maximizes a lower bound to the log-likelihood of the evidence. Even when using transformations that are individually simple, flows can model complex distributions by chaining sufficiently many transformations together (Papamakarios et al., 2019). More recent work parameterizing the flow transformations with deep neural networks results in a powerful generative model capable of sampling novel and realis-

tic data (Dinh, Sohl-Dickstein, and Bengio, 2017; Ho et al., 2019; Kingma and Dhariwal, 2018), while still retaining the attractive qualities of normalizing flows.

1.1 CONTRIBUTIONS

Probabilistic models are ubiquitous in machine learning, and the techniques for designing models for complex high-dimensional data like text and images continue to evolve. *Directed probabilistic graphical models* (PGM) are a foundational approach with a long history (Jordan et al., 1999b; Koller and Friedman, 2009; Wainwright and Jordan, 2007), that allows practitioners to design a probabilistic model inspired by the generative process that created the data. The rise of deep learning, and the marriage of deep learning with probabilistic modeling has resulted in new paradigms like the variational autoencoder and normalizing flows.

The focus of this dissertation is on designing probabilistic models for complex high-dimensional data. The contributions of this thesis, *Advancing Probabilistic Models for Approximate and Exact Inference*, span both foundational techniques like PGMs, as well as more recent breakthroughs using deep generative models.

In Chapter 2 we review classic techniques in probabilistic modeling as well as how they lead in to modern approaches, and provide notation and background for subsequent chapters. In Part (ii) we contribute to foundational probabilistic models like PGMs with new models and algorithms for efficiently training our model on massive datasets.

Part (iii) shifts focus towards more modern approaches and advancing probabilistic deep learning. In Chapter 4 we advance the state-of-the-art for cutting-edge variational autoencoders and normalizing flows on the more general problem of modeling complex, high-dimensional data. In Chapter 5 we introduce compressive normalizing flows to learn compact latent representations of the data. In Chapter 6 we introduce probabilistic super-resolution using normalizing flows as a general technique to generate high-dimensional data given a subset of the available features, similar to self-supervised learning (Jing and Tian, 2019; Kolesnikov, Zhai, and Beyer, 2019). Below we introduce each of the chapters in more detail and clarify the core problem solved.

1.1.1 *Chapter 3: Probabilistic Models for Massive Text Datasets*

New inference algorithms, and a graphical model for uniquely structured datasets

In Chapter 3 we begin with contributions to *directed probabilistic graphical models* (PGM), or *Bayesian networks*. A PGM organizes all variables topologically into a directed acyclic graph. The power in PGMs lies in the ability to explicitly parameterize the variables in the graph to follow specific distributions and connecting the variables in the graph such that the graph reflects a generative process the produces the observed data. In short, we can tailor a model to reflect the unique structure of our data.

For this work our focus is on designing a topic model for a unique text corpora. A topic model is a PGM designed to model text data and the themes running through the text. We introduce the Dynamic Author Persona (DAP) topic model for text corpora with multiple authors writing over time. The advantage of PGMs, like the DAP topic model, is that it allow us to introduce latent variables based on the specific structure of the data. For instance, in the DAP topic model we hypothesize that (1) authors have a latent *persona* which influences the types of topics they tend to write about over time, and (2) each *topic* is described as a cluster of likely words. We train the model on data in order to learn about the latent *persona* and *topic* variables. By introducing and learning about latent variables in our model we create a high-level view of our data, with the ability to quickly zoom in and out on authors or documents based on the latent themes running through them.

Lastly, we derive a new training algorithm that allows the DAP model to scale to massive text datasets, and introduce *regularized variational inference* (RVI) as a technique to guide a latent variable model to seek out certain kinds of solution. In the DAP model, for example, we use RVI to encourage the model to find unique and diverse personas that describe the authors in our dataset.

1.1.2 *Chapter 4: Gradient Boosted Normalizing Flows*

More flexible normalizing flows that resemble a mixture model

In Chapter 4 we introduce a state-of-the-art approach in deep generative modeling by combining gradient boosting—a classic idea in machine learning, with variational autoencoders and normalizing flows. Unlike the PGM introduced in 3 which are specifically designed for text datasets with a unique structure, our approach is for more general modeling of high-dimensional data, such as images.

The trend, thus far, in normalizing flow literature has been to design deeper, more complex flow transformations. We introduce a "wider", not "deeper" approach to normalizing flows: *gradient boosted normalizing flows* (GBNF) iteratively adds new normalizing flow components to a model based on gradient boosting, where each new component is fit to the *residuals* of the previously trained components. A weight is learned for each component, and hence the combination of the components resembles a mixture model.

As a meta-algorithm, GBNF may be used to augment many existing flow-based models. Practitioners incorporating GBNF trade additional training time to incorporate new components in exchange for added model flexibility. Moreover, adding new components improves model performance without increasing the complexity of the predictions. Because each component is independent, sampling from the model is trivially-parallelizable and highly scalable.

Unlike recent developments in boosted variational inference (Guo et al., 2016; Miller, Foti, and Adams, 2017), the flow-based GBNF enhances posterior approximation in VAEs, and is even flexible enough to model data directly for density estimation tasks.

We demonstrate the effectiveness of GBNF for density estimation on high-dimensional benchmark datasets. We demonstrate GBNF's ability to improve posterior approximation in variational inference tasks. Specifically, by augmenting the VAE with a GBNF variational posterior, we show image modeling results exceeding many state-of-the-art models. However, our analysis highlights the need for *analytically* invertible flows in order to efficiently boost flow-based models for approximate inference tasks. We explore the "decoder shock" phenomenon—a challenge unique to VAEs that model the approximate posterior with GBNF. When GBNF begins training a new component the distribution of samples passed to the VAE's decoder changes abruptly, causing a temporary increase in loss. We propose a training technique to combat "decoder shock" and show that performance steadily improves as more components are added to the model.

1.1.3 *Chapter 5: Compressive Normalizing Flows*

Normalizing flows that change dimensions to learn compressed representations

Normalizing flows are powerful generative models for defining expressive probability distributions using diffeomorphic transformations—that is, transformations that are bijections with differentiable inverses. However, bijections require that the models have the same dimensionality at each flow step. As a result, distilling all the information contained in complex high-dimensional data (like natural images) and finding a compact but useful representation is a major challenge. Thus, there is a growing interest in relaxing the bijective constraint and constructing normalizing flows that map complex high dimensional data to low dimensional, easy to evaluate base distributions.

In Chapter5 we introduce compressive normalizing flows for changing the dimensionality of a normalizing flows latent space. In the simplest case a compressive normalizing flow is based on the classic dimensionality reduction approach of probabilistic principal components analysis (PPCA, (Tipping and Bishop, 1999)). By leveraging PPCA's connection to linear and nonlinear variational autoencoders (VAE), we extend our approach to more flexible compressive flows. A compressive flow can be added as the final step in a model or, alternatively, intermixed throughout the sequence of normalizing flow steps to learn varying levels of compression in the latent space.

In our experiments we compare various change of dimension transformations and architectural choices, and show that the latent space learned by normalizing flows can be compressed heavily without a significant drop in performance.

1.1.4 *Chapter 6: Probabilistic Super-Resolution with Normalizing Flows*

Learning high-dimensional distributions conditioned on low-resolution samples

In Chapter 6 we present approaches for conditional probabilistic super-resolution (PSR) which construct high-dimensional distributions conditioned on low-resolution samples, as well as joint PSR which model the high- and low-dimensional distributions simultaneously. Our PSR models are based on normalizing flows (NFs), in particular two NFs for joint PSR, one marginal flow corresponding to the low-dimensional distribution and one conditional flow for the high-dimensional distribution. By combining state-of-the-art techniques for training flows with our improved method for upsampling low-resolution samples into the conditional flow steps, we present a flexible framework for PSR flows.

Despite their advantages, unconditional normalizing flows face a number of significant challenges. For instance, the information bottleneck problem, which stems from the requirement that flows maintain their dimensionality, limits the amount of information that one flow step can pass on to the next (Chen et al., 2020; Huang, Dinh, and Courville, 2020). Additionally, flows exhibit an inductive bias towards modeling local pixel correlations—as opposed to modeling high-level semantic information (Kirichenko, Izmailov, and Wilson, 2020). Our conditional PSR flow architecture provides effective solutions to these challenges. We design a highly controlled student-teacher evaluation setup to measure how well PSR-based flows learn complex, multi-modal output distributions. In our experiments with benchmark datasets, we show extensive qualitative and quantitative results to evaluate PSR flows on a variety of image datasets and illustrate improvements over the state-of-the-art.

2

PRELIMINARIES AND BACKGROUND

2.1 MODELS FOR PROBABILISTIC REASONING

Probabilistic models seek to learn a joint distribution over the data and latent variables. In this thesis we explore various approaches for probabilistic modeling—namely, (i) probabilistic graphical models (PGM) where the joint distribution factorizes according to the graphical structure of the model, and (ii) deep generative models which aim to learn a process that *generates* the training data.

PROBABILISTIC GRAPHICAL MODELS By composing probability distributions as building blocks for larger models, PGMs form a unified framework for designing models with explicitly defined dependencies. With the model defined we use probabilistic reasoning for inference, prediction, and decision making. The flexibility to design PGMs specific to the structure of the data make them indispensable for scientists seeking to explain their data. Moreover, probabilistic graphical models are robust to overfitting, making them a dependable choice for smaller datasets.

GENERATIVE MODELS Along with discriminative models, generative models represent the two major modeling paradigms in machine learning (Bishop and Lasserre, 2007). Whereas discriminative models learn to map inputs to outputs, a generative model is probabilistic and learns a simulator of the training data. As such, generative models align with the broader scientific notion of developing models of the real world—hypothesize a model for how the world works and then test it.

A generative model examines all dependencies between variables, modeling the full joint probability distribution. By making stronger assumptions about the data than discriminative models, generative models often produce models with greater bias. Moreover, because the model is forced to account for all patterns with the data it is useful in numerous applications related to prediction, interpolation and extrapolation, synthesis (of new images, texts, etc.), and even beneficial for agents that plan and simulate interactions with their environment. The unification of deep learning and generative models, known as *deep generative models*, represents an important recent development in machine learning, yielding significant progress in building models for massive, complex, high-dimensional data.

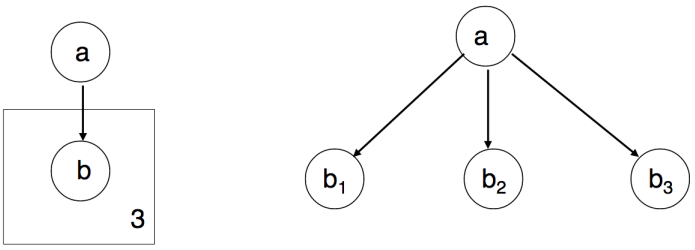


Figure 1: Plate notation provides a compact representation of a graphical model. Figure taken from (Banerjee, 2010)

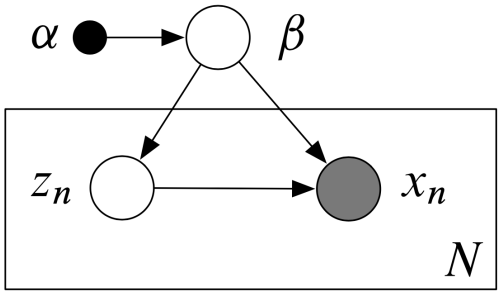


Figure 2: Example of a simple graphical model with prior α, hidden variables β and z_n, and observation $\mathbf{x}_n$.

2.2 PROBABILISTIC GRAPHICAL MODELS

A probabilistic graphical model (PGM) describes the relationship between observed evidence $\mathbf{x}$ and latent variables $\mathbf{z}$ with a graph, where dependencies between variables are denoted as edges on the graph. Edges connecting random variables can be either directed or undirected, however we limit our attention to directed graphical models. Graphical models with directed edges are also known as Bayesian networks. In order to compactly represent complex networks, graphical models we often use plate notation as in Figure 1, where repeated nodes are combined into a single node within a plate that represents repetition.

Graphical models on a graph $G = (V, E)$ is defined as a collection of vertices $V = \{1, 2, \ldots, m\}$ and edges $E \subset V \times V$, where each edge is denoted by a pair of vertices with a direction $(s \to t)$. Additionally, the notation $(s \to t)$ specifies that t is a child of s. The reason graphical models are so useful is that their structure succinctly encodes the relationship between all variables. For example, consider the simple graphical model in Figure 2. Small dark circles on graphical models refer to known priors, open circles represent latent variables, and large filled in circles represent observed information.

In the graphical model in Figure 2, the network compactly represents the joint distribution and its factorization:

$$p(\mathbf{x}, \mathbf{z}, \beta \mid \alpha) = p(\beta \mid \alpha) \prod_{n=1}^{N} p(\mathbf{x}_n, \mathbf{z}_n \mid \beta)$$

where α is a prior, β a global hidden variable, $\mathbf{x}_n$ the observed evidence that is conditioned on each corresponding local hidden variable $\mathbf{z}_n$ and the global variable.

2.2.1 *Approximate Inference*

Multiple options exist for performing inference on probabilistic models. For simple Bayesian networks in which each node has only one parent, recursive message passing algorithms can perform exact inference (Wainwright and Jordan, 2007). In complex networks with intractable posteriors, approximate techniques are required. Most commonly Markov Chain Monte Carlo are employed, but these approaches can be slow and computationally expensive (Blei, Kucukelbir, and McAuliffe, 2017).

An increasingly popular technique for approximate inference is *variational inference*, which attempts to approximate a difficult to compute posterior distribution with a simpler distribution called the variational distribution, denoted $q_\phi(\mathbf{z} \mid \mathbf{x})$ where $\mathbf{x}$ are the observed data, $\mathbf{z}$ the latent variables, and ϕ are learned parameters (Blei, Kucukelbir, and McAuliffe, 2017; Jordan et al., 1999b; Wainwright and Jordan, 2007). This simple idea is powerful, casting the inference problem as an optimization problem. The optimization algorithm's goal is to find parameters to the variational distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$ so that it closely approximates the true posterior $p(\mathbf{z} \mid \mathbf{x})$. For the simple graphical model model in Figure 2, the approximate inference means finding a distribution $q \in \Omega$ such that:

$$q^*(\mathbf{z}, \beta) = \arg\min_{q \in \Omega} \mathrm{KL}(q(\mathbf{z}, \beta) \parallel p(\beta, \mathbf{z}|\mathbf{x})), \tag{2.1}$$

where $\mathrm{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$ is the KL-Divergence metric.

KL-Divergence is a natural choice to framing the approximation of p by q because it measures the *distance* between two distributions. Hence, when the KL-Divergence $\mathrm{KL}(q(\mathbf{z}, \beta) \parallel p(\beta, \mathbf{z} \mid \mathbf{x}))$ is small, then q will be nearly equal to p. Moreover, KL-divergence results in the best lower bound on the probability of the evidence $p(\mathbf{x})$ in the family of approximations made by $q(\mathbf{z}, \beta)$ (Jordan et al., 1999a), a fact that will be useful later when solving this optimization problem.

Variational inference is a powerful technique because q can be chosen to follow some known, easy to compute distribution. To do this, consider the posterior p to the graphical model in Figure 2 and a variational distribution q. First, define parameters corresponding to each of the posterior's parameters: endow the variational distribution with a free variational parameters ϕ to approximate $\mathbf{z}$, and λ to approximate β, where each of these is a parameter to a known distribution. Most importantly, assumptions about q's structure can be made to further simplify the calculation. The simplest form of variational inference assumes that each of the parameters are independent of one another—eliminating the difficult computation of interdependent random variables. This form of variational inference is referred to as *mean field variational inference*.

Under mean field variational inference, the variational distribution would factorize as a product of the components:

$$q(\mathbf{z}, \beta \mid \phi, \lambda) = q(\beta \mid \lambda) \prod^{N} q(\mathbf{z} \mid \phi)$$

After defining the structure and parameters of q, the problem is framed as an optimization problem. Since a good generative model will maximize evidence, the log-likelihood of the data $p(\mathbf{x})$ is re-written in terms of the difference between the true posterior p and the approximation q:

$$\begin{aligned}
\log p(\mathbf{x}) &= \log \int p(\mathbf{x}, \mathbf{z}, \beta) d\mathbf{z} d\beta \\
&= \log \int p(\mathbf{x}, \mathbf{z}, \beta) \frac{q(\mathbf{z}, \beta)}{q(\mathbf{z}, \beta)} d\mathbf{z} d\beta \\
&= \log \left(\mathbb{E}_q \left[\frac{p(\mathbf{x}, \mathbf{z}, \beta)}{q(\mathbf{z}, \beta)} \right] \right) \\
&\geqslant \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}, \beta)] - \mathbb{E}_q[\log q(\mathbf{z}, \beta)] \\
&= \mathcal{L}(\mathbf{z}, \beta, \phi, \lambda)
\end{aligned}$$

where $\mathcal{L}(\mathbf{z}, \beta, \phi, \lambda)$ is referred to as the Evidence Lower Bound (ELBO). Note, the last step is given by Jensen's Inequality: $\log \mathbb{E}[f(y)] \geqslant \mathbb{E}[\log f(y)]$. Note, it can be shown that $KL(q \parallel p) = -\mathcal{L}(\mathbf{z}, \beta, \phi, \lambda) + \text{const}$. Thus, maximizing the ELBO is equivalent to the original objective in Equation 2.1 of minimizing KL-divergence, but written in terms of q (which was defined such that it is easy to compute).

Optimizing the ELBO can be done in a straight-forward way. First, the expectation over the joint distribution $\mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}, \beta)]$ and the entropy term $\mathbb{E}_q[\log q(\mathbf{z}, \beta)]$ are expanded based on the factorization of the model and the assumptions made for variational inference (e.g. mean field variational inference). Second, each of the terms in the ELBO is expanded according to how the random variable is parameterized (e.g. if $\beta \sim \mathcal{N}(\mu, \Sigma)$, then use the probability density function of a Gaussian distribution). Third, for each of the parameters, compute a gradient update via by taking the derivative of the ELBO and set it to zero. Note, if a parameter is tied to a constraint—for example, β must sum to one in the case of $\beta \sim$ Dirichlet, then Lagrangian terms must be included. Optimal values of the parameters can then be found using a variational EM algorithm, where updates to the variational parameters are computed first, and then using these values make updates to the model parameters (for more details see Blei (2002)).

In PGMs the dependency between nodes, and their corresponding probability distributions, can form either conjugate or non-conjugate pairs. Non-conjugate pairs occur in a number of famous models, such as in the correlated topic model (CTM, (Blei and Lafferty, 2006a, 2007)) and dynamic topic models (DTM, (Blei and Jordan, 2006)). The challenge with non-conjugate priors, however, is that the posterior does not belong to the same family as the prior, and often the posterior cannot be obtained in a

closed form analytically (Wainwright and Jordan, 2007). As a result, models with non-conjugate terms often require the introduction of additional variational parameters and gradient based optimizations which significantly slows down training.

2.2.2 *Probabilistic Reasoning and Deep Learning*

The PGM framework builds probabilistic reasoning into our model building, allows practitioners to progress beyond only associating inputs to outputs. By modeling latent variables we extend our tools for reasoning and decision making. Quantifying uncertainty helps us anticipate and hypothesize future outcomes and plans. Moreover, PGMs typically excel at being data efficient in their learning and more robust to overfitting, and can provide useful insights in data even in the unsupervised setting where ground-truth labels are expensive or difficult to obtain.

Despite the many advantages of classical probabilistic modeling approaches like PGMs, there are many disadvantages. In particular, they are often linear or limited to modeling with conjugate probability distributions, as opposed to the rich non-linear models found in deep learning. Despite advances, training many PGMs is more computationally challenging and may require running expensive simulations, whereas breakthroughs in deep learning scale well using stochastic gradient methods (Robbins and Monro, 1951) that are conceptually simple and easy to implement with automatic differentiation libraries (Paszke et al., 2019).

These two schools of machine learning—probabilistic models and deep learning, are not odds though. As we will see probabilistic reasoning can be married with deep learning techniques, where the advantages and disadvantages of each complement one another.

2.3 DEEP GENERATIVE MODELS

Prior to work by Kingma and Welling (2014) and Rezende, Mohamed, and Wierstra (2014), a significant application of variational inference was on training Bayesian and probabilistic graphical models. However, the introduction of what is now known as *amortized variational inference* (AVI) marries probabilistic reasoning with deep learning (Kingma and Welling, 2019), and has brought renewed interest and new developments in latent variable models, inference, and probabilistic modeling.

For amortized variational inference we use the familiar ELBO objective from (Blei, Kucukelbir, and McAuliffe, 2017; Jordan et al., 1999b; Wainwright and Jordan, 2007). However, we specify the approximate posterior as conditional on the evidence, denoted $q_\phi(\mathbf{z} \mid \mathbf{x})$ where $\mathbf{x}$ are the observed data, $\mathbf{z}$ the latent variables, and ϕ are learned parameters.

As noted in Section 2.2.1, the ELBO is equal to $\mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z} \mid \mathbf{x})\right)$ up to a constant, a fact that we show next as an alternative derivation that is more illustrative of how tight the bound is to the evidence:

$$\begin{aligned}\mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z} \mid \mathbf{x})\right) &= \mathbb{E}_q\left[\log q_\phi(\mathbf{z} \mid \mathbf{x})\right] - \mathbb{E}_q\left[\log p(\mathbf{z} \mid \mathbf{x})\right] \\ &= \underbrace{\mathbb{E}_q\left[\log q_\phi(\mathbf{z} \mid \mathbf{x})\right] - \mathbb{E}_q\left[\log p(\mathbf{z}, \mathbf{x})\right]}_{\text{(ELBO)} \quad \mathcal{L}_\phi(\mathbf{x})} + \underbrace{\mathbb{E}_q\left[\log p(\mathbf{x})\right]}_{\text{Evidence}}. \end{aligned} \tag{2.2}$$

Here, the evidence term $\log p(\mathbf{x})$ is revealed—which is not computable (and the reason approximate inference is sought in the first place). Since KL-divergences are non-negative, we refer to the first two terms on the RHS of (2.2) as the Evidence Lower Bound (ELBO). Since $\log p(\mathbf{x})$ is a constant with respect to $q_\phi(\mathbf{z} \mid \mathbf{x})$, then the ELBO is the negative of the KL-divergence up to a constant. Hence, maximizing the ELBO is equivalent to minimizing the KL-divergence.

2.3.1 *Amortized Variational Inference*

In AVI the variational posterior $q(\mathbf{z} \mid \mathbf{x})$ is a conditional Bayesian network, but each conditional (stochastically) maps the input $\mathbf{x}$ and random noise $\epsilon \sim \mathcal{N}(0, 1)$ through a neural network f_ω, i. e. $\mathbf{z} \mid \mathbf{x} \sim f_\omega(\mathbf{x}, \epsilon)$. By learning one set of parameters, namely the weights ω to the neural network, we are able to *amortize* the cost of learning variational parameters during training. Traditional approaches like *mean-field* VI, on the other hand, require variational parameters to be learned per-datapoint—which can be very inefficient on large datasets. Thus, mapping inputs to latent variables in the AVI setting is as fast as a forward pass through the neural network f_ω.

2.3.1.1 *Stochastic Gradients of the ELBO*

Amortized variational inference is not exempt from the "no free lunch" principle: the amortized approach introduces added noise in the gradients during training. Taking stochastic gradients of the ELBO is not a new challenge, however. Below we highlight key prior developments in solving the noisy gradient problem, then we conclude by reviewing the crucial reparameterization trick for implementing AVI.

SCORE FUNCTION ESTIMATORS Some of the first attempts at taking stochastic gradients of an objective with respect to a random variable involve score function estimators. The advantage of the score function estimator is it's generally easy to

compute and applicable to discrete and continuous latent variables. Here gradients of the objective are computed by:

$$\begin{aligned}\nabla_\phi \mathbb{E}_{q_\phi}\left[f_\theta(\mathbf{z})\right] &= \nabla \int q_\phi(\mathbf{z} \mid \mathbf{x}) f_\theta(\mathbf{z}) d\mathbf{z} \\ &= \mathbb{E}_{q_\phi}\left[f_\theta(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z} \mid \mathbf{x})\right] \\ &\simeq f_\theta(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z} \mid \mathbf{x}) \end{aligned} \tag{2.3}$$

where $\mathbf{z} \sim q_\phi(\mathbf{z} \mid \mathbf{x})$ and f_θ is some function parameterized by θ—such as the joint likelihood, that re-weights the gradient. This is also known as the likelihood ratio estimator (Glynn, 1990). The REINFORCE algorithm uses a single Monte Carlo sample from the estimator in (2.3) in order to approximate a gradient (Williams, 1992).

Paisley, Blei, and Jordan (2012) provide one of the first modern solutions: taking unbiased stochastic approximations of the gradient term. To control the variance of the gradient approximations, *control variates* are introduced. A control variate is a function that matches another function in expectation, but has a lower variance. A second-order Taylor approximation effectively replaces the original gradient functions and lowers the variance of the gradient approximations. The approach in Paisley, Blei, and Jordan (2012) leads to the popular *stochastic* variational inference algorithm (Hoffman et al., 2013) and more general inference algorithms like black-box variational inference (Ranganath, Gerrish, and Blei, 2014), as well as neural variational inference (Mnih and Gregor, 2014).

WAKE-SLEEP ALGORITHM The wake-sleep algorithm is a two-phase algorithm designed for inference on deep generative models that cleverly avoids the challenge associated with taking gradients from the ELBO (Hinton et al., 1995). In the wake phase the parameters θ to the generator $p_\theta(\mathbf{x} \mid \mathbf{z})$ are trained by minimizing the KL-divergence term in (2.2). Note that, gradients with respect to θ are computable. On the other hand, gradients with respect to the variational distribution's parameters ϕ are intractable since $\mathbf{z}$ is a random variable. Thus, during the sleep phase the algorithm "flips" the KL-divergence term in (2.2), i. e. $\mathrm{KL}\left(p(\mathbf{z} \mid \mathbf{x}) \parallel q_\phi(\mathbf{z} \mid \mathbf{x})\right)$ and takes gradients w.r.t. ϕ.

REPARAMETERIZATION TRICK Score function estimators ignore gradient information from $\mathbf{p}_\theta(\mathbf{x}, \mathbf{z})$, and the wake-sleep algorithm's "flipped" KL-divergence results in an approximate posterior that lacks full support over the data. The reparameterization trick, on the other hand, provides an efficient, unbiased estimator for optimizing the ELBO with respect to both parameters ϕ and θ. Despite the recent popularity of the reparameterization trick for its role in training deep generative models (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014), the original idea dates back many decades (Bonnet, 1964; Price, 1958). In short, the reparameterization trick uses a change of variables: reparameterizing the ELBO's expectation to be with respect to

randomly sampled noise—yielding an ELBO that is differentiable and amenable to optimization via stochastic gradients and back-propagation.

More formally, Kingma and Welling (2014) reparameterize the random variable $\tilde{\mathbf{z}} \sim q_\phi(\mathbf{z} \mid \mathbf{x})$ using a differentiable transformation, namely: $\tilde{\mathbf{z}} = g_\phi(\boldsymbol{\epsilon}, \mathbf{x})$ where $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. In other words instead of having $\mathbf{z}$ be a random variable, we treat $\mathbf{z}$ as a transformed version of the data plus random noise $\boldsymbol{\epsilon}$.

The choice of noise $\boldsymbol{\epsilon}$ and g_ϕ depend on the distribution of the hidden nodes $\mathbf{z}$. For example if $\mathbf{z}$'s distribution is in the "location-scale" family of distributions (e.g. Laplace, Elliptical, Student's T, Logistic, Uniform, and Gaussian distributions), then ϵ is drawn from standard location = 0, scale = 1 and $g(\cdot) = \mathtt{location} + \mathtt{scale} \cdot \epsilon$.

Thus, $\boldsymbol{\epsilon}$ is easy to sample and the transformation function $g(\cdot)$ is differentiable, and easy to compute. Using this reparameterization, we form the following simple Monte Carlo estimator and take the gradient:

$$\begin{aligned}\nabla_\phi \mathbb{E}_{q_\phi}[f(\mathbf{z})] &= \nabla_\phi \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f\left(g_\phi(\boldsymbol{\epsilon}, \mathbf{x}^{(i)})\right)\right] \\ &\overset{(a)}{=} \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_\phi f\left(g_\phi(\boldsymbol{\epsilon}, \mathbf{x}^{(i)})\right)\right] \\ &\simeq \frac{1}{L}\sum_{l=1}^{L} f\left(g_\phi(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(l)})\right)\end{aligned}$$

where $\boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$, the indices i denote individual data-points, and (a) follows from theorems by Bonnet (1964) and Price (1958). Applying this idea to the individual-datapoint ELBO, we derive a generic stochastic variational Bayes estimator $\tilde{\mathcal{L}}(\theta, \phi; \mathbf{x}^{(i)})$ of the ELBO $\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$:

$$\tilde{\mathcal{L}}(\theta, \phi; \mathbf{x}^{(i)}) = \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_\phi(\mathbf{z}^{(i,l)} \mid \mathbf{x}^{(i)})$$

again, where $\mathbf{z}^{(i,l)} = g_\phi(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)})$ and $\boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$.

Finally, we highlight the computation of $\log q_\phi(\mathbf{z} \mid \mathbf{x})$ under the reparameterization trick. Since the reparameterization trick is an application of the change of variables formula and $g_\phi(\boldsymbol{\epsilon}, \mathbf{x})$ is invertible and differentiable, it follows that:

$$\log q_\phi(\mathbf{z} \mid \mathbf{x}) = \log p(\boldsymbol{\epsilon}) - \log \left| \det \frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} \right| , \tag{2.4}$$

where $\log p(\boldsymbol{\epsilon})$ is easily computable since we define it to be random noise (e.g. the density for uniform or a standard normal distribution). The second term, the log determinant Jacobian captures the change in "volume" induced by the transformation from $\boldsymbol{\epsilon}$ to $\mathbf{z}$ using $g(\cdot)$. For the above transformation $g(\cdot)$ mapping $\boldsymbol{\epsilon}$ to a location-scale distribution the log determinant Jacobian is trivial. However, a major focus of our work is more complex transformations g which produce more faithful representations of the true posterior. Thus, in general case the Jacobian matrix will be computed by:

$$\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} = \frac{\partial(z_1, \ldots, z_K)}{\partial(\epsilon_1, \ldots, \epsilon_K)} = \begin{bmatrix} \frac{\partial z_1}{\partial \epsilon_1} & \cdots & \frac{\partial z_1}{\partial \epsilon_K} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_K}{\partial \epsilon_1} & \cdots & \frac{\partial z_K}{\partial \epsilon_K} \end{bmatrix} \tag{2.5}$$

2.3.2 *Variational Autoencoders*

In additional to the reparameterization trick, Kingma and Welling (2014) structure the inference problem as an autoencoder, introducing the variational autoencoder (VAE). The VAE includes an "encoder" neural network which outputs the parameters ϕ to the approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$, and reconstructs the input with a "decoder" neural network $\mathbf{p}_\theta(\mathbf{x} \mid \mathbf{z})$. Parameters to both networks are amortized and optimized jointly by reparameterizing the the ELBO. Specifically, re-writing the negative of the ELBO in (2.2) by replacing the joint probability with a conditional-likelihood and prior that gives the negative-ELBO:

$$\mathcal{F}_{\phi,\theta}(\mathbf{x}) = \underbrace{\mathbb{E}_{q_\phi}\left[-\log p_\theta(\mathbf{x} \mid \mathbf{z})\right]}_{\text{Decoder}} + \underbrace{\mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z})\right)}_{\text{Regularized Encoder}}, \tag{2.6}$$

which shows the probabilistic decoder $p_\theta(\mathbf{x} \mid \mathbf{z})$, and highlights how the VAE encodes the latent variables $\mathbf{z}$ with the variational posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$, but we regularize to limit complexity by keeping $q_\phi(\mathbf{z} \mid \mathbf{x})$ close to the prior $p(\mathbf{z})$.

VAEs along with Generative Adversarial Networks (GAN, Goodfellow, Pouget-Abadie, and Mirza (2014)) represent the two main paradigms in deep generative modeling, with GANs being implicit density models. While GANs tend to generate sharper images, they lack full support over the data (Grover, Dhar, and Ermon, 2018). GANs optimize the reverse KL-divergence relative to VAEs (Hu et al., 2018). As a result, GANs suffer from mode-collapsing behavior and are generally harder to train.

CHALLENGES WITH VAES Despite employing a neural network to encode the latent variables $\mathbf{z}$, the VAE's encoder does not produce a richer approximate posterior than other variational inference techniques. This surprising fact is because despite the complexity of the encoder neural network, the latent variables are still chosen as draws from a fully factorized Gaussian (although other distributions are possible), hence even with a *perfect* encoder the VAE is limited to be only as flexible as a fully factorized Gaussian.

The VAE's encoder bottleneck has lead many researchers to witness "posterior collapse" in VAEs—a phenomenon where the model ignores the latent code $\mathbf{z}$ and relies entirely on the decoder (Casale et al., 2018; Chen et al., 2017c; Huang et al., 2018a; Kingma et al., 2016; Miller, Foti, and Adams, 2017; Rezende and Mohamed, 2015; Tomczak and Welling, 2018; van den Berg et al., 2018). Posterior collapse can be mitigated

by annealing the KL regularization term in (2.6) during training. Proper annealing allows the model more flexibility during the initial stages of training, and an opportunity to find a useful latent representation before being more thoroughly regularized (Bowman et al., 2016; Higgins et al., 2017). Finally, while VAEs are explicit density models, they do not offer exact density estimation—a requirement in many statistical procedures.

2.3.3 *Normalizing Flows*

Normalizing flows (NF) play an important role in the recent developments of both density estimation and variational inference (Rezende and Mohamed, 2015). Normalizing flows are smooth, invertible transformations with tractable Jacobians, which define a mapping between a complex data distribution and a simple distribution, such as a standard normal Tabak and Turner (2013) and Tabak and Vanden-Eijnden (2010). Parameterizing flows with deep neural networks (Dinh, Krueger, and Bengio, 2015; Dinh, Sohl-Dickstein, and Bengio, 2017; Rippel and Adams, 2013) has popularized the technique for density estimation and variational inference (Papamakarios et al., 2019).

2.3.3.1 *Approximate Inference*

In the context of variational inference, a normalizing flow transforms a simple, known base distribution into a more faithful representation of the true posterior. Here, the normalizing flow's role is to improve the posterior approximation (Hasenclever et al., 2017; Kingma et al., 2016; Rezende and Mohamed, 2015; Tomczak and Welling, 2016, 2017; van den Berg et al., 2018). The flow-based approximate posterior can be used in conjunction with a generative model like the VAE—where samples of the latent variable are drawn from the flow-based approximate posterior and "decoded" by the VAE's decoder to generate realistic looking images and plausible texts.

In the original formulation, Rezende and Mohamed (2015) modify the VAE's posterior approximation, applying a chain of K transformations to the inference network's output $\mathbf{z}_0 \sim q_0(\mathbf{z}_0 \mid \mathbf{x})$, giving:

$$\mathbf{z}_K = f_K \circ \cdots \circ f_2 \circ f_1(\mathbf{z}_0) \,, \tag{2.7}$$

where f_i are the transformations and $\mathbf{z}_K$ is the final transformed sample.

We choose the density transformation f to be a smooth, invertible mapping. Then, by the chain rule and inverse function theorem, a random variable $\mathbf{z}' = f(\mathbf{z})$ has a computable density (Tabak and Turner, 2013; Tabak and Vanden-Eijnden, 2010):

$$q(\mathbf{z}') = q(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathbf{z}'} \right| = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1} \tag{2.8}$$

Thus, a NF-based approximate posterior seeks to minimize:

$$\mathcal{F}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_0}\left[-\log p_\theta(\mathbf{x}, \mathbf{z}_K) + \log q_0(\mathbf{z}_0 \mid \mathbf{x}) - \sum_{k=1}^{K} \log \left|\det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}}\right|\right], \tag{2.9}$$

where $q_0(\mathbf{z}_0 \mid \mathbf{x})$ is a known base distribution. Thus the ELBO in (2.9) for normalizing flows is the same ELBO from (2.6) as for the VAE except we've shown the log determinant Jacobian term (just as in (2.4)) associated with the K-step flow transformation.

2.3.3.2 *Density Estimation*

Given a set of samples $\{\mathbf{x}_i\}_{i=1}^n$ from a target distribution p^*, our goal is to learn a flow-based model $p_\phi(\mathbf{x})$, which corresponds to minimizing the forward KL-divergence: $\mathcal{F}^{(ML)}(\phi) = KL(p^*(\mathbf{x}) \,||\, p_\phi(\mathbf{x}))$ (Papamakarios et al., 2019). A normalizing flow formulates $p_\phi(\mathbf{x})$ as a transformation $\mathbf{x} = f(\mathbf{z})$ of a base density $p_Z(\mathbf{z})$ with $f = f_K \circ \cdots \circ f_1$ as a K-step flow (Dinh, Krueger, and Bengio, 2015; Dinh, Sohl-Dickstein, and Bengio, 2017; Papamakarios, Pavlakou, and Murray, 2017). Thus, to estimate the expectation over p^* we take a Monte Carlo approximation of the forward KL, yielding:

$$\mathcal{F} \approx -\frac{1}{n}\sum_{i=1}^{n}\left[\log p_Z\left(f^{-1}(\mathbf{x}_i)\right) + \sum_{k=1}^{K} \log \left|\det \frac{\partial f_k^{-1}}{\partial \mathbf{x}_i}\right|\right], \tag{2.10}$$

which is equivalent to fitting the model to samples $\{\mathbf{x}_i\}_{i=1}^n$ by maximum likelihood estimation (Papamakarios et al., 2019).

2.3.3.3 *Flexible Flows with Efficiently Computable Jacobians*

Normalizing flows use a chain of bijective transformations f to map the base distribution $p_Z(\mathbf{z})$ to the complex distribution $p_\phi(\mathbf{x})$. In order to create flexible mappings we consider K-step normalizing flow, denoted $f = f_K \circ \cdots \circ f_1$, which allows us to compute:

$$\log p_\phi(\mathbf{x}) = \log p_Z(f^{-1}(\mathbf{x})) + \sum_{k=1}^{K} \log \left|\det \frac{\partial f_k^{-1}(\mathbf{x};\phi)}{\partial \mathbf{x}}\right| \tag{2.11}$$

However, to estimate very complex, high-dimensional distributions—such as text or images, it is not sufficient to choose f to be simple affine or linear transformations even by chaining together K flow steps. Moreover, in order to computing the log-determinant Jacobian term in (2.11) requires $O(D^3)$ computations, where D is the dimensionality of the variables. The challenge then is constructing flow transformations that are both flexible and allow us to compute (2.11) efficiently.

Dinh, Krueger, and Bengio (2015) and Dinh, Sohl-Dickstein, and Bengio (2017) demonstrated that using arbitrarily complex neural networks to learn parameters to affine function along with a non-linear activation can produce flexible transformations while maintaining the bijective constraint. The technique, referred to as a coupling

layer, splits the input $z = [z^{[a]}, z^{[b]}]$ and applies a scale $s(\cdot)$ and shift $t(\cdot)$ transformation to half the inputs, while the other half of the inputs are given as inputs to scale and shift transformations. Thus, in the forward direction $z_1 = f(z_0)$ and the inverse $z_0 = f^{-1}(z_1)$ are shown on the left and right, respectively:

$$\begin{aligned} z_1^{[a]} &= z_0^{[a]} \odot \exp(s(z_0^{[b]})) + t(z_0^{[b]}) & \qquad z_0^{[a]} &= \left(z_1^{[a]} - t(z_1^{[b]})\right) / \exp(s(z_0^{[b]})) \\ z_1^{[b]} &= z_0^{[b]} & z_0^{[b]} &= z_1^{[b]} \\ z_1 &= [z_1^{[a]}, z_1^{[b]}] & z_0 &= [z_0^{[a]}, z_0^{[b]}] \end{aligned}$$

where in the final step the split latent variables are concatenated back together. Unlike the general Jacobian matrix shown in (2.5), the Jacobian for the coupling layer is lower triangular and can therefore be computed as the product of the elements on the diagonal. Hence, we can define the $s(\cdot)$ and $t(\cdot)$ functions as parameterized by an arbitrarily complex deep neural networks, allowing us to apply flexible transformations on the variables but compute the density efficiently.

In addition to coupling layers, flow transformations defined by autoregressive models (Germain et al., 2015; Ma et al., 2019; Papamakarios, Pavlakou, and Murray, 2017) also result in fast, easy to compute lower triangular Jacobians. Autoregressive flows are often more flexible than coupling layers, however they are D times slower in the inverse direction than the forward direction (Durkan et al., 2019), making them a poor choice if sampling is an important application for the model.

Lastly, while thus far we've only discussed flows with a *discrete* number of flow steps, there is a continuous extension of normalizing flows based on neural ordinary differential equations (ODE) (Chen et al., 2018). That is, given an ODE defined by the parametric function:

$$\frac{d\mathbf{z}}{dt} = f(\mathbf{z}(t), t) \ ,$$

where the log-density under this model follows the *instantaneous* change of variables formula (Chen et al., 2017a):

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\operatorname{Tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right) \ ,$$

and hence the total change in log-density across time steps t_0 to t_1 is given by integrating:

$$\log p(\mathbf{z}(t)) = \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right) dt \ .$$

Similar to residual flows (Chen et al., 2019), continuous normalizing flows have the advantage that the Jacobian term can be free-form—allowing for much more flexibility and the trace term in the instantaneous change of variables can be computed in $O(D)$ time (Grathwohl et al., 2018). Continuous normalizing flows, however, depend

on the use of an ODE solver during training to compute the integral, which is computed empirically and hence requires setting a tolerance level that balances accurately computing the integral and completing the computation in finite time. While neural ODE based normalizing flows are an exciting direction, they are computationally challenging after long training runs where the neural ODE progressively more and more complex transformations.

2.3.3.4 *Recent Trends in Normalizing Flows*

Building on the planar and radial flows from (Rezende and Mohamed, 2015), Sylvester flows generalize planar flows into a more expressive framework (van den Berg et al., 2018). Inverse autoregressive flows (IAF, Kingma et al. 2016) and masked autoregressive flows (MAF, Papamakarios, Pavlakou, and Murray 2017) scale to higher dimensions by exploiting the ordering of variables in the flow. Neural autoregressive flows (NAF, Huang et al. 2018b) and the more compact Block NAF (De Cao, Titov, and Aziz, 2019), replace affine transformations with autoregressive neural networks.

While all NFs are invertible, flows based on coupling layers like NICE (Dinh, Krueger, and Bengio, 2015) and successor RealNVP (Dinh, Sohl-Dickstein, and Bengio, 2017) are analytically invertible. Glow replaced RealNVP's permutation operation with a 1×1 convolution (Kingma and Dhariwal, 2018). Neural spline flows provide a method for increasing the flexibility of both coupling and autoregressive transforms using monotonic rational-quadratic splines (Durkan et al., 2019). Non-linear squared flows (Ziegler and Rush, 2019) offer a highly multi-modal transformation and are analytically invertible.

Part II

PROBABILISTIC GRAPHICAL MODELS

Probabilistic graphical models (PGM) are graphical representations of probability distributions. More than probabilistic reasoning, PGMs offer a unified framework for model building, inference, prediction, and decision making that can handle uncertainty. Here we use the modular nature of PGMs to customize models for uniquely structured datasets—such as the CaringBridge dataset of health journals, and estimate latent variables within massive corpora of texts.

3

PROBABILISTIC MODELS FOR MASSIVE TEXT DATASETS

3.1 INTRODUCTION

Topic models can compactly represent large collections of documents by the themes running through them. We introduce a topic model designed for the unique challenges presented by the CaringBridge (CB) dataset. The CB dataset includes journals written by patients and caregivers during a health crisis. CB journals function like a blog, and are shared to a private community of friends and family. The full dataset includes 9 million journals written by approximately 200,000 authors ($\approx$ 1 billion words) between 2006 and 2016. From the CB dataset we're interested in capturing health journeys, that is, authors writing about the same topics over time.

The challenges in topic modeling on the CB dataset stem from the asynchronous nature of author's posts. Specifically, authors start and stop journaling at different times—both in terms of calendar dates and how far along they are in their health journey. Additionally, authors post at irregular frequencies. While about 15% of CB authors post nearly everyday, the majority of authors typically post less frequently, often corresponding to a major update, event, or anniversary.

State-of-the-art topic models can identify topics (Blei, Ng, and Jordan, 2003), track how topics change over time (Blei and Lafferty, 2006b; Wang, Blei, and Heckerman, 2008; Wang and McCallum, 2006; Wei, Sun, and Wang, 2007), or associate authors with certain topics (McCallum, Corrada-Emmanuel, and Wang, 2005; Mimno and McCallum, 2007; Rosen-Zvi et al., 2004; Steyvers et al., 2004). These models cannot, however, describe common narratives and the author's sharing them. We present the Dynamic Author-Persona topic model (DAP), a novel approach that represents authors by latent *personas* (Giaquinto and Banerjee, 2018b). Personas act as a soft-clustering on authors based their propensity to write about similar topics over time. Our approach is unique in multiple respects. First, unlike other temporal topic models, the words making up a topic don't evolve over time—rather, DAP's personas reflect the flow of conversation from one topic to next. Second, we introduce a regularized variational inference (RVI) algorithm, an approach we developed to encourage personas to be distinct from one another. Lastly, we adapt new ideas in approximate inference to drastically speed up training of DAP compared to conventional approaches to training complex topic

models, resulting in the DAP Performed Exceedingly Rapidly (DAPPER) topic model (Giaquinto and Banerjee, 2018a).

Our results show that the DAPPER model outperforms competing topic models, producing better likelihoods on heldout data. We demonstrate that using RVI further improves our model's performance, and results in significantly more diversity in the personas discovered. Finally, we demonstrate the scalability of the DAPPER model by training it to the CB and Signal Media One-Million News Article corpora, and share the compelling narratives found by DAPPER's latent personas.

3.2 BACKGROUND: ADVANCES IN VARIATIONAL INFERENCE

In recent years tremendous progress has been made in improving both the speed, quality, and ease of application of variational inference. In Hoffman et al. (2013) the authors introduce stochastic variational inference (SVI): a method that reparameterizes the gradient of the Expected Lower BOund (ELBO) in terms of the natural parameters in order to derive a fast stochastic gradient descent (SGD) algorithm for variational inference (Hoffman et al., 2013; Hoffman, Blei, and Bach, 2010). The SVI approach is an important contribution because of the tremendous speed-up that results. Further, reparameterizing the ELBO so as to derive natural gradients, which leads to stochastic optimization, is closely related to traditional coordinate ascent variational inference. Natural gradients provide more stable learning, as opposed to gradient methods that are better suited for optimization in Euclidean geometry (Amari, 1998). SVI is limited in that it requires the model's parameters to have an exponential family form, and hence is not directly applicable to non-conjugate models like CTM (Blei and Lafferty, 2006a, 2007) or DTM (Blei and Lafferty, 2006b).

Stochastic updates have been shown to speed up convergence of approximate inference algorithms significantly. One reason for this is that full batch gradient updates during the initial iterations inefficiently attempt to infer local variational parameters over the entire dataset given random initializations of the global parameters. Mandt et al. proposed averaging over a fixed number of previous biased stochastic gradients (Mandt and Blei, 2014), as opposed to following unbiased stochastic gradients as is done in SVI. The bias in the smoothed stochastic gradients follows from the size of fixed window of averaged gradients. While introducing a bias, the averaged gradients have the advantage of reducing the variance.

Recent advances, like Black Box Variational Inference (BBVI), demonstrate a promising new way to speed-up updates to non-conjugate terms (Ranganath, Gerrish, and Blei, 2013). The approach introduced in BBVI uses stochastic gradient updates, where the noisy stochastic observations are computed using Monte Carlo techniques. BBVI results in a variational inference algorithm that is faster than conventional approaches and eliminates the need to derive inference algorithms for new models.

3.2.1 *Conjugate-Computation Variational Inference*

The computational downside of BBVI is that does not take advantage of conjugate terms with closed form updates. Khan and Lin introduce Conjugate-computation Variational Inference (CVI) which cleverly allows inference on models with non-conjugate terms to be computed as a *conjugate computation* (Khan and Lin, 2017). A conjugate computation is simply the adding of the natural parameters of a prior to the sufficient statistics of the likelihood. In short, the CVI approach allows for fast updates to complex PGMs. Moreover, unlike SVI which takes gradients of the ELBO in the natural-parameter space, CVI uses stochastic mirror descent in the mean-parameter space that eschews Euclidean geometry (i.e. squared loss) in favor of a Bregman divergence defined by the convex-conjugate of the log-partition function. Khan and Lin (Khan and Lin, 2017) demonstrate that this approach leads to inference in a conjugate model, where non-conjugate terms have been replaced by exponential family approximations. Further, CVI lets models be trained with stochastic mini-batches—in the style of SVI. As a result, even complex models with difficult non-conjugate terms can be trained quickly and efficiently.

The goal of the CVI algorithm is to maximize a lower bound to the marginal likelihood:

$$\arg\max_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} \mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_q[\log p(\mathbf{y},\mathbf{z}) - \log q(\mathbf{z}\mid\boldsymbol{\lambda})]$$

where $\boldsymbol{\Lambda}$ is the set of valid variational parameters, $\boldsymbol{\lambda}$ the variational parameter, and $q(\mathbf{z}\mid\boldsymbol{\lambda})$ the variational approximation. Traditionally, the bound is optimized via gradient descent, i.e. $\boldsymbol{\lambda}_{i+1} \leftarrow \boldsymbol{\lambda}_+ \rho_i \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_i)$, where ρ_i is the learning rate. An equivalent formulation of this gradient, which highlights the divergence function, is:

$$\boldsymbol{\lambda}_{i+1} \leftarrow \arg\max_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} \langle \boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_i)\rangle - \frac{1}{2\rho_i}\|\boldsymbol{\lambda}-\boldsymbol{\lambda}_i\|_2^2$$

CVI assumes distributions are minimal exponential families, meaning there is a one-to-one mapping between $\boldsymbol{\lambda}$ and the mean parameters $\boldsymbol{\mu}\in\mathcal{M}$. The lower bound is reparameterized in terms of $\boldsymbol{\mu}$ such that $\tilde{\mathcal{L}}(\boldsymbol{\mu}) = \mathcal{L}(\boldsymbol{\lambda})$, and mirror descent gradient updates for this bound are derived. Additionally, in making the mean-field assumption, which assumes that the parameters are *posteriori* independent, the gradient update can be expressed as a summation over all nodes $k \in 1,\ldots,M$:

$$\max_{\boldsymbol{\mu}} \sum_{k=1}^{M} \left[\left\langle \boldsymbol{\mu}_k, \hat{\nabla}_{\boldsymbol{\mu}} \tilde{\mathcal{L}}(\boldsymbol{\mu}_i) \right\rangle - \frac{1}{\rho_i} \mathbf{B}_{A^*}(\boldsymbol{\mu}_k \| \boldsymbol{\mu}_{k,i}) \right], \tag{3.1}$$

where i refers to the iteration number, and $\mathbf{B}_{A^*}$ is a Bregman divergence—such as KL divergence, defined over the convex-conjugate of the log-partition A^*. The choice of divergence function is to account for the geometry of the parameter space. Khan et al. prove convergence for the general case of Bregman divergences, even in the stochastic

gradient setting (Khan et al., 2016). The maximization in (3.1) only requires optimizing for a single node k and hence can be done either in parallel, or as a doubly stochastic scheme by randomly picking a term in the summation.

One of the primary results proved by Khan and Lin (Khan and Lin, 2017) is that (3.1) can be implemented as Bayesian inference in a conjugate model. Their method hinges splitting the joint distribution into non-conjugate and conjugate terms (denoted $\tilde{p}_{nc}(\mathbf{y}, \mathbf{z})$ and $\tilde{p}_c(\mathbf{y}, \mathbf{z})$, respectively) and replacing the difficult non-conjugate term with an exponential family approximation whose natural parameter is $\tilde{\boldsymbol{\lambda}}_i$. Hence, the posterior is approximated with a variational distribution defined by:

$$q(\mathbf{z} \mid \boldsymbol{\lambda}_{i+1}) \propto \exp(\phi(\mathbf{z}), \tilde{\boldsymbol{\lambda}}_i) \tilde{p_c}(\mathbf{y}, \mathbf{z}) \ ,$$

where $\tilde{\boldsymbol{\lambda}}_i$ is the natural parameter of the exponential-family approximation to $\tilde{p}_{nc}$, computed as a weighted sum of the gradients of the non-conjugate term. Khan and Lin (Khan and Lin, 2017) show that the exponential-family approximation's parameter $\tilde{\boldsymbol{\lambda}}_i$ and the variational posterior's parameter $\boldsymbol{\lambda}$ are updated by:

$$\tilde{\boldsymbol{\lambda}}_{k,t} = \sum_{a \in \mathbb{N}_k} \mathbb{E}_{q/k,i}[\boldsymbol{\eta}_{a,k}(\mathbf{z}_{a/k}, \mathbf{y}_{a/k})] + \nabla_{\mu_k} \mathbb{E}_{q_i}[\log \tilde{p}_{nc}^{\tilde{a,k}}] \tag{3.2a}$$

$$\boldsymbol{\lambda}_{i+1} = (1 - \rho_i)\boldsymbol{\lambda}_i + \rho_i \tilde{\boldsymbol{\lambda}}_i \tag{3.2b}$$

where $\boldsymbol{\eta}_{a,i}(\mathbf{z}_{a/i}, \mathbf{y}_{a/i})$ are simply the natural parameters for the conjugate parts of the model, and $\mathbb{N}_k$ the local neighborhood containing $\mathbf{z}_i$ and its children. By replacing non-conjugate terms with exponential family approximations, CVI allows even complex models to be trained quickly and efficiently.

3.2.2 *CVI as Expectation Propagation*

While DAPPER's inference algorithm is based on CVI—a recent advance in approximate inference, CVI itself has theoretical connections to the well known expectation propagation (EP) algorithm(Minka, 2001a; Minka, 2001b). The EP algorithm, which is an extension of Assumed Density Filtering, infers the approximate posterior q using localized inferences. With posterior p, hidden parameters $\mathbf{z}$, and observations $\mathbf{y}$, we assume p can be written as a product of terms: $p(\mathbf{z} \mid \mathbf{y}) \propto \prod_{i=0}^{N} f_i(\mathbf{z})$, where $f_0(\mathbf{z}) = p(\mathbf{z})$ expresses the prior density and $f_i(\mathbf{z}) = p(\mathbf{y} \mid \mathbf{z})$ the likelihood. The EP algorithm then approximates the posterior, choosing an approximating family with density $q(\mathbf{z}) \propto \prod_{i=1}^{N} q_i(\mathbf{z})$ and iteratively incorporating $q_i(\mathbf{z})$ into $q(\mathbf{z})$. First, EP computes the cavity distribution—that is, deleting $q_i(\mathbf{z})$ from $q(\mathbf{z})$, by $q_{-i}(\mathbf{z}) \propto q(\mathbf{z})/q_i(\mathbf{z})$. Second, a true Bayesian update incorporates $f_i(\mathbf{z})$:

$$\hat{p}(\mathbf{z}) = Z_i^{-1} q_{-i}(\mathbf{z}) f_i(\mathbf{z}) \ , \quad Z_i = \mathbb{E}_{\mathbf{z} \sim q_{-i}}[f_i(\mathbf{z})]$$

where $\hat{p}(\mathbf{z})$ is a tilted exponential family. The exact posterior is approximated, for exponential families minimizing KL divergence between the posteriors,

$$q^{new}(\mathbf{z}) = \arg\min \mathrm{KL}(\hat{p}(\mathbf{z}) \| q(\mathbf{z})) \,, \tag{3.3}$$

corresponds to matching the moments of p and q. Finally, we update the $q_i^{new}(\mathbf{z})$ by $q_i^{new}(\mathbf{z}) \propto q(\mathbf{z}) q_{-i}(\mathbf{z})$. Note, that the update to $q_i(\mathbf{z})$ is the local minimization and can be formulated as:

$$q_i^{new}(\mathbf{z}) = \arg\min \mathrm{KL}(f_i(\mathbf{z}) q_{-i}(\mathbf{z}) \| q_i(\mathbf{z}) q_{-i}(\mathbf{z})).$$

From here two connections to CVI appear. First, EP also computes an exponential family approximation in its approximation of $\hat{p}(\mathbf{z})$, the tilted distribution induced by $f_i(\mathbf{z})$ (Seeger, 2003). While EP does this computation using moment matching, moment matching corresponds to minimizing the Kullback-Leibler divergence from the tilted distribution to the new approximated marginal distribution (Gelman et al., 2014), and moments can be computed as derivatives of the log normalizer, hence:

$$\begin{aligned} \mu^{new} = \mathbb{E}_{\hat{p}}[\phi(\mathbf{z})] &= \nabla_{q_{-i}} \log Z_i + \mu_{-i} \\ &= \nabla_{q_{-i}} \log \mathbb{E}_{\mathbf{z} \sim q_{-i}}[f_i(\mathbf{z})] + \eta_{-i} \end{aligned}$$

For exponential families $\nabla \log Z = \mathbb{E}[\phi(\mathbf{y})]$, and therefore this moment matching can be viewed as similar to conjugate computations, here we add expectations of sufficient statistics of q to the corresponding expectations of $\mathbf{z}$ in $q_{-i}(\mathbf{z}) f_i(\mathbf{z})$. This shows that EP's creation of the tilted distribution moment parameter is analogous to CVI's computation of the natural parameter to the exponential family approximation, i.e. (3.2a).

The second connection to CVI lies in the "damping" technique used to improve the convergence of EP. Damping replaces the generic update $\lambda_i \to \lambda_{i+1}$ by a convex combination, which reduces the step size so that only a partial update is applied. CVI also updates the natural parameters with a convex combination: $\lambda_{t+1} = (1-\beta)\lambda_t + \beta\tilde{\lambda}_t$ in CVI is essentially just "damped" updates in EP (Gelman et al., 2014). This form of updating is analogous to minimizing an alpha divergence (which includes directed KL as a special case).

Despite a number of similarities, EP and CVI differ critically in convergence guarantees. Khan et al. show that CVI converges under fairly mild assumptions, namely that q is a minimal exponential family and the model's conditional distribution can be split into conjugate and non-conjugate terms. EP, on the other hand, is not guaranteed to converge. EP minimizes KL divergence for each local observation, but does not directly minimize $\mathrm{KL}(p \,\|\, q)$.

3.3 DYNAMIC AUTHOR PERSONA TOPIC MODEL

The design of the DAP model was made with journaling behavior in mind (Giaquinto and Banerjee, 2018b). Consider a CB author journaling about their surgery: initially they may write about topics related to the surgical procedure, but as time progresses the author is more likely to discuss recovery, physical therapy, or returning to normal life. In other words, the likelihood of a topic for some document depends where the document's author is in their health journey. As such, DAP assumes that (1) a state space model controls the likelihood of a topic at each time step, (2) each persona represents a different flow of topics over time, and (3) each author has a distribution over personas.

The DAP's approach for modeling topics in a document, and words in a topic follows the correlated topic model (CTM) and LDA, respectively (Blei and Lafferty, 2006a; Blei, Ng, and Jordan, 2003). The idea of modeling latent personas was originally proposed by Mimno and McCallum (2007), however in their Author-Persona Topic model (APT) personas differ significantly from those proposed in the DAP model. First, in the APT model each author may have a different number of personas, whereas DAP models each author as a distribution over a fixed number of personas—which acts as a soft clustering on authors. Second, APT assigns a persona to documents, which indirectly defines a topic distribution for each cluster of documents, whereas we model documents as each having their own distribution over topics. Lastly, while DAP's personas also correspond to a distribution over topics, DAP evolves these topic distributions over time—thereby capturing the inherent temporal structure resulting from an author writing multiple documents.

The DAP model directly addresses the challenges presented by the CB dataset. First, the asynchronous nature of health journals is handled by: (1) transforming each journal's timestamp to the time elapsed since the author's first post, and (2) learning multiple personas to account for a wide variety in topic trajectories. Second, irregular posting behavior is managed by employing the Brownian motion model, originally used in topic modeling by (Wang, Blei, and Heckerman, 2008), to model topic variance as proportional to the gap in time between documents.

The generative process of the model is described below. The model assumes that each document d in the corpus has a timestamp s_t associated with it. Similar to the CDTM (Wang, Blei, and Heckerman, 2008), timestamps are used in a continuous Brownian motion model to capture an increase in topic variance as time between observations increases. More formally, if s_i and s_j are timestamps at steps $j > i > 0$, then Δ_{s_j,s_i} is the difference in time between s_j and s_i. We use the shorthand Δ_{s_t} to denote the difference in time between timestamps s_t and s_{t-1}. For brevity the variance $\sigma\Delta_{s_t}I$ is denoted $\boldsymbol{\Sigma}_t$, where σ is a known process noise in the state space model.

1. Draw distribution over words $\boldsymbol{\beta}_k \sim \mathrm{Dir}(\eta)$ for each topic k.

2. Draw distribution over personas $\boldsymbol{\kappa}_a \sim \mathrm{Dir}(\omega)$ for each author a.

3. For each persona p, draw initial distribution over topics:

$$\boldsymbol{\alpha}_{0,p} \sim \mathcal{N}(\mu_0, \boldsymbol{\Sigma}_0), \forall p \in \{1, \ldots, P\} .$$

4. For each time step t, where $t \in \{1, \ldots, T\}$:
   - Draw distribution over topics:

$$\boldsymbol{\alpha}_{t,p} \sim \mathcal{N}(\boldsymbol{\alpha}_{t-1,p}, \boldsymbol{\Sigma}_{t-1}), \forall p \in \{1, \ldots, P\} .$$

   - Update $\boldsymbol{\Sigma}_t$ according to Brownian motion model:

$$\boldsymbol{\Sigma}_t - \boldsymbol{\Sigma}_{t-1} \sim \mathcal{N}(0, \sigma\Delta_{s_t} I).$$

   - For each document d, where $d \in \{1, \ldots, D_t\}$:
     (a) Choose persona indicator $\mathbf{x}_{t,d} \sim \text{Mult}(\boldsymbol{\kappa}_a)$ where a corresponds to the author of document d_t.
     (b) Draw topic distribution $\boldsymbol{\theta}_{t,d} \sim \mathcal{N}(\boldsymbol{\alpha}_t x_{t,d}, \boldsymbol{\Sigma}_t)$ for document d_t.
     (c) For each word $w_{t,d,n}$, where $n \in \{1, \ldots, N_{d_t}\}$:
        i. Choose word topic indicator $\mathbf{z}_n \sim \text{Mult}(\pi(\boldsymbol{\theta}_{t,d}))$.
        ii. Choose word $w_{t,d,n}$ from $p(\mathbf{w}_{t,d,n} \mid \boldsymbol{\beta}_{\mathbf{z}_n})$, a multinomial probability conditioned on the topic indicator $\mathbf{z}_n$.

Following the approach in the CTM and DTM, we use the function $\pi(\cdot)$ to map the Logistic Normal $\boldsymbol{\theta}_{t,d}$, parameterized by a mean $\boldsymbol{\alpha}_{t,k,p}$ and covariance $\sigma\Delta_{s_t} I$, to the multinomial's natural parameters via $\pi(\boldsymbol{\theta}_{t,d}) = \frac{\exp(\boldsymbol{\theta}_{t,d})}{\sum_d^D \exp(\boldsymbol{\theta}_{t,d})}$ in order to obey the constraint that the parameters lie on the simplex.

The graphical model corresponding to this process is shown in Figure 3. In LDA and its extensions the parameter $\boldsymbol{\alpha}$ represents a prior probability of each topic. In the DAP model, $\boldsymbol{\alpha}_{t,1:K,p}$ takes on an expanded role: it's a distribution over K topics at time step t for persona p. The choice of letting $\boldsymbol{\alpha}$ evolve over time, as opposed to $\boldsymbol{\beta}$ like in the DTM, is that in a collection of journals there is less interest in changes to topics themselves. In other words, we model the words associated with a topic as static in time, but the topics an author writes about will change over time.

Scaling the DAP model to massive corpora is challenging due to its non-conjugate terms—for which there are no fast, closed form updates. To derive parameter updates the DAP model's intractable posterior is approximated with a variational posterior under the mean-field assumption. In standard fashion, the Evidence Lower BOund (ELBO) is maximized, which is equivalent to minimizing the KL divergence between the variational in true posteriors. Once the ELBO is specified, updates are derived for each parameter by deriving gradient updates with respect to each parameter. However, due to the non-conjugate terms, this standard approach doesn't yield fast, closed-form updates. In Section 3.5 we derive standard updates for the conjugate terms, as well as

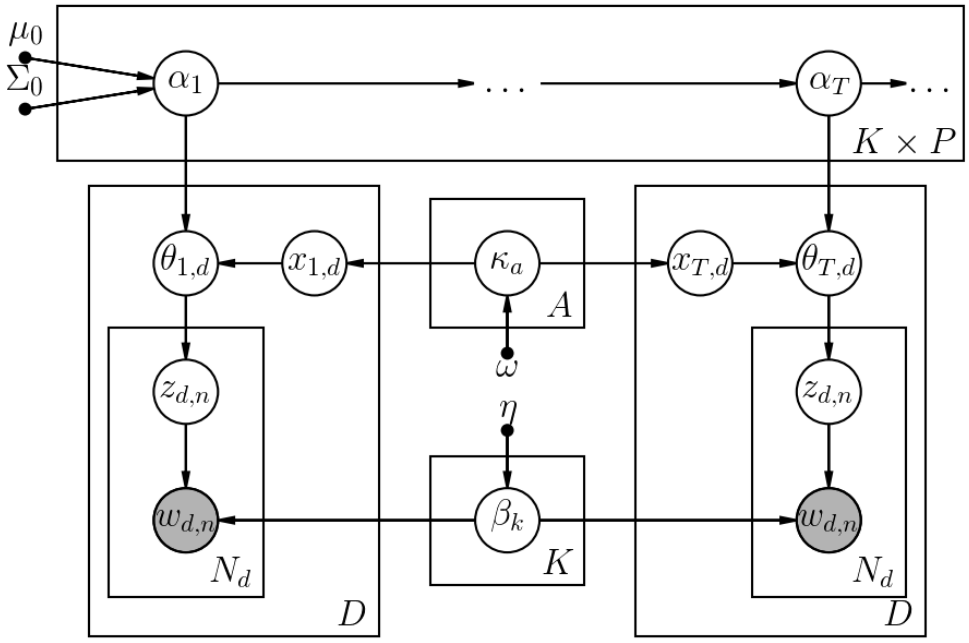


Figure 3: Graphical representation of the Dynamic Author-Persona topic model (DAP). On top, topic distributions for each persona evolve over time: $\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1} \sim \mathcal{N}(\boldsymbol{\alpha}_{t-1}, \boldsymbol{\Sigma})$. The distribution over words for each topic, $\boldsymbol{\beta} \sim \mathrm{Dir}(\eta)$, is fixed in time. Each author $a \in \{1, \ldots, A\}$ is represented by a distribution over personas, that is $\boldsymbol{\kappa}_a \sim \mathrm{Dir}(\omega)$. The distribution over topics for each document is dependent on the persona distribution $\mathbf{x}_{t,d}$ for that document's author, and the evolving topic distribution $\boldsymbol{\alpha}_t$.

the results of the adapting CVI to the DAP model which give updates for the difficult non-conjugate terms.

3.4 ESTIMATING DAP'S PARAMETERS

Given the DAP topic model's structure, next we derive an inference algorithm used to estimate the model's latent parameters. Much like LDA and its extensions, the DAP model's posterior:

$$p(\boldsymbol{\kappa}, \mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{z} \mid \mathbf{w}, \omega, \eta) = \frac{p(\boldsymbol{\kappa}, \mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{z}, \mathbf{w} \mid \omega, \eta)}{p(\mathbf{w} \mid \mu_0, \sigma_0, \eta, \omega)} ,$$

is intractable due to the normalization term. In order learn optimal values to the model's parameters we use a form of variational inference (VI), which approximates the difficult to compute posterior distribution p with a simpler distribution q (see Blei, Kucukelbir, and McAuliffe (2017)). Variational inference casts an inference problem as an optimization problem with the goal of finding parameters to the variational distribution such that $q = q(\boldsymbol{\kappa}, \mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{z}, \boldsymbol{\beta})$ closely approximates $p = p(\boldsymbol{\kappa}, \mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{z}, \boldsymbol{\beta} \mid \mathbf{w})$. Our regularized variational inference (RVI) algorithm (Giaquinto and Banerjee, 2018b) seeks a distribution $q \in \mathcal{Q}$ such that

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q \parallel p) + \rho r(q) , \tag{3.4}$$

Parameter	Variational	Description
P		Number of personas
K		Number of topics
A		Number of authors
T		Number of time steps
D		Number of documents
N_d		Number of words in document d
$\mathbf{w}_{t,d}$		Words in document d_t
$\mathbf{z}_n$	$\boldsymbol{\phi}_n$	Assigns word n to a topic
$\boldsymbol{\theta}_{t,d}$	$\boldsymbol{\gamma}_{t,d}$	Distribution over topics for d_t
$\mathbf{v}_{t,d}$	$\hat{\mathbf{v}}_{t,d}$	Covariance between topics for d_t
μ_0		Prior for mean of $\boldsymbol{\alpha}_0$
Σ_0		Prior for covariance of $\boldsymbol{\alpha}_0$
$\boldsymbol{\alpha}_t$	$\hat{\boldsymbol{\alpha}}_t$	$\forall p$ distribution over topics
$\boldsymbol{\Sigma}_t$	$\hat{\boldsymbol{\Sigma}}_t$	Covariance in topic distributions
ω		Prior for $\boldsymbol{\kappa}_a$
$\boldsymbol{\kappa}_a$	$\boldsymbol{\delta}_a$	Author's distribution over personas
$\mathbf{x}_{t,d}$	$\boldsymbol{\tau}_{t,d}$	Assigns author of d_t to a persona
η		Prior parameter for β_k
$\boldsymbol{\beta}_k$	$\boldsymbol{\lambda}_k$	$\forall k$ distribution over words

Table 1: Notation and parameters used in the DAP model. Variational refers to the corresponding parameter in the mean-field variational inference algorithm.

where $\mathrm{KL}(\cdot)$ is KL-Divergence. The added term $r(q)$ is a regularization function we've introduced to discourage similar personas (further detail given in Section 3.6), and ρ the corresponding hyperparameter.

To make q easy to compute, we apply *mean field* variational inference which assumes that the parameters are *posteriori* independent. Under the mean field assumption the variational distribution factorizes as:

$$\prod_{k=1}^{K} q(\boldsymbol{\beta}_k \mid \boldsymbol{\lambda}_k) \prod_{a=1}^{A} q(\boldsymbol{\kappa}_a \mid \boldsymbol{\delta}_a) \prod_{p=1}^{P} q(\alpha_{1:T,k,p} \mid \hat{\alpha}_{1:T,k,p}) \times \prod_{t=1}^{T} \prod_{d=1}^{D_t} q(x_{t,d,p} \mid \tau_{t,d,p}) q(\boldsymbol{\theta}_{t,d} \mid \boldsymbol{\gamma}_{t,d}) \prod_{n=1}^{N_{d_t}} q(z_n \mid \phi_n) \tag{3.5}$$

where we have introduced the following variational parameters: the persona for each author $\boldsymbol{\kappa}_a$ is endowed with a free Dirichlet parameter $\boldsymbol{\delta}_a$; each assignment of a persona to an author $\mathbf{x}_{t,d}$ is endowed with a free Multinomial parameter $\boldsymbol{\tau}_{t,d}$; in the variational distribution of $\alpha_{1:T,k,p}$ the sequential structure is kept intact with variational observations $\hat{\alpha}_{1:T,k,p}$; each document-topic proportion vector $\boldsymbol{\theta}_{t,d}$ is endowed with a free $\boldsymbol{\gamma}_d$.

The variance for the document-topic parameters are $v_{t,d}$ and $\hat{v}_{t,d}$, for the model and variational parameter, respectively; each word-topic indicator is endowed with a free multinomial parameter $\phi_{t,d,n}$.

An optimal q cannot be computed directly, but following Jordan et al. (1999a) an optimization of the variational parameters proceeds by maximizing a bound on the log-likelihood of the data. In the DAP model, data is observed in words $\mathbf{w}$ for each document d at time step t, hence we re-write the log-likelihood of the data $\log p(\mathbf{w}_{t,d})$ by:

$$\begin{aligned}
\log p(\mathbf{w}_{t,d}) &= \log \iiiint \sum_{\mathbf{z}} \sum_{\mathbf{x}} p \;\; \partial\boldsymbol{\theta}_{t,d}, \partial\boldsymbol{\alpha}_t, \partial\boldsymbol{\beta}, \partial\boldsymbol{\kappa} \\
&= \log \iiiint \sum_{\mathbf{z}} \sum_{\mathbf{x}} \frac{pq}{q} \\
&\geqslant \iiiint \sum_{\mathbf{z}} \sum_{\mathbf{x}} q \log p - \iiiint \sum_{\mathbf{z}} \sum_{\mathbf{x}} q \log q \\
&= \mathbb{E}_q[\log p] - \mathbb{E}_q[\log q]
\end{aligned} \tag{3.6}$$

The inequality in (3.6) follows from Jensen's inequality. Moreover, it can be shown that the difference between $\log p(\mathbf{w}_{t,d})$ and $\mathbb{E}_q[\log p] - \mathbb{E}_q[\log q]$ is $\mathrm{KL}(q \parallel p)$. Hence maximizing the bound in (3.6) is equivalent to minimizing the KL divergence between the variational and true posteriors. We denote the Evidence Lower BOund (ELBO) by $\mathcal{L}(\boldsymbol{\delta}_a, \boldsymbol{\tau}_{t,d}, \boldsymbol{\gamma}_{t,d}, \phi_n, \boldsymbol{\lambda}_k) = \mathbb{E}_q[\log p] - \mathbb{E}_q[\log q]$. Since our objective defined in (3.4) includes a regularization term, we therefore maximize a surrogate likelihood consisting of the ELBO minus the regularization term (see Wainwright and Jordan (2007) for a review of penalized surrogate likelihoods). Hence our objective function $\mathcal{L}_\rho$ for some regularization ρ is defined as:

$$\mathcal{L}_\rho(\boldsymbol{\delta}_a, \boldsymbol{\tau}_{t,d}, \boldsymbol{\gamma}_{t,d}, \phi_n, \boldsymbol{\lambda}_k) \triangleq \mathbb{E}_q[\log p] - \mathbb{E}_q[\log q] - \rho r(q) \;, \tag{3.7}$$

where $\mathbb{E}_q[\log p]$ expands for each term in the model, that is:

$$
\begin{aligned}
\mathbb{E}_q[\log p] = &\sum_{k=1}^{K}\sum_{v=1}^{V}\mathbb{E}_q\left[\log p(\boldsymbol{\beta}_{k,v} \mid \eta)\right] \\
&+\sum_{a=1}^{A}\sum_{p=1}^{P}\mathbb{E}_q\left[\log p(\boldsymbol{\kappa}_{a,p} \mid \omega)\right] \\
&+\sum_{t=1}^{T}\left[\sum_{p=1}^{P}\sum_{k=1}^{K}\mathbb{E}_q\left[\log p\left(\alpha_{t,k,p} \mid \alpha_{t-1,k,p}\right)\right]\right. \\
&\quad+\sum_{d=1}^{D_t}\left[\mathbb{E}_q\left[\log p(\mathbf{x}_{t,d} \mid \boldsymbol{\kappa}_a)\right] + \mathbb{E}_q\left[\log p(\boldsymbol{\theta}_{t,d} \mid \boldsymbol{\alpha}_t \mathbf{x}_{t,d}, \boldsymbol{\Sigma}_t)\right]\right. \\
&\quad\left.\left.\left.+\sum_{n=1}^{N_{d_t}}\left[\mathbb{E}_q\left[\log p(z_n \mid \pi(\boldsymbol{\theta}_{t,d}))\right] + \mathbb{E}_q\left[\log p(w_{t,d,n} \mid z_n, \boldsymbol{\beta})\right]\right]\right]\right]
\end{aligned}
\tag{3.8}
$$

And, similarly $-\mathbb{E}_q[\log q]$ is the entropy term associated with each of the parameters. Some terms in (3.8) are simple, and well-known from foundational topic models like LDA and CTM (Blei and Lafferty, 2006a; Blei, Ng, and Jordan, 2003). For example, the topic distributions over words $\mathbb{E}_q[\log p(\boldsymbol{\beta}_{k,v} \mid \eta)]$ term is found in LDA, and in the DAP model the distributions over personas for each author $\mathbb{E}_q[\log p(\boldsymbol{\kappa}_{a,p} \mid \omega)]$ follows a similar structure. Similarly, the non-conjugate pairs for word-topic assignment $\mathbb{E}_q[\log p(z_n \mid \pi(\boldsymbol{\theta}_{t,d}))]$ has been studied in the CTM.

Expanding the objective function $\mathcal{L}_\rho$ according to the distribution associated with each parameter allows updates to be derived for each parameter. The parameters are optimized using a variational expectation-maximization algorithm, the details of the algorithm are given in Section 3.5.

3.5 DAPPER: SCALING DAP TO BILLION WORD CORPORA

Here we derive a variational-EM algorithm for the Dynamic Author Persona topic model (DAP). In our original paper we take a standard approach for estimating the model parameters (Giaquinto and Banerjee, 2018b), however here we derive a faster variational-EM algorithm that scales the DAP model to larger corpora using conjugate computation variational inference. The result is the Dynamic Author Persona Performed Exceedingly Rapidly topic model (DAPPER) (Giaquinto and Banerjee, 2018a). Similar to the standard parameter updates, training DAPPER alternatives between the expectation step (E-Step) where we estimate local variational parameters, and the maximization step (M-step) where update the model's global parameters.

3.5.1 *E-Step*

During the E-step the model estimates variational parameters for each document and saves the sufficient statistics required to compute global parameters like $\boldsymbol{\alpha}, \boldsymbol{\kappa}$, and $\boldsymbol{\beta}$.

The complexity of the DAP model results in many non-conjugate terms. For simplicity we show the DAPPER approach, which derives a CVI based updates for each parameter.

DOCUMENT TOPIC PROPORTIONS Each document is given a hidden parameter $\boldsymbol{\theta}_d$ representing the proportions of each topic. We begin by identifying the conjugate and non-conjugate terms involving $\boldsymbol{\theta}_d$. For $\boldsymbol{\theta}_d$ we have a conjugate term: $\tilde{p}_c^{\theta} = \mathcal{N}(\boldsymbol{\theta}_d \mid \boldsymbol{\alpha}_t \mathbf{x}_d, \Sigma_t)$. Here $\mathcal{N}(\boldsymbol{\theta}_d \mid \boldsymbol{\alpha}_t \mathbf{x}_d, \Sigma_t)$ is a Gaussian conditioned on a Gaussian because the mean parameter is the dot product between two fixed terms $\boldsymbol{\alpha}_t \mathbf{x}_d$, giving a natural parameter:

$$\begin{bmatrix} \Sigma_t^{-1}(\boldsymbol{\alpha}_t \mathbf{x}) & -\frac{1}{2}\Sigma_t^{-1} \end{bmatrix}^\top \tag{3.9}$$

The second term involving $\boldsymbol{\theta}$ required is the non-conjugate term, which is $\tilde{p}_{nc}^{\theta} = \text{Mult}(\mathbf{z}_n \mid \sigma(\boldsymbol{\theta}_d))$

The variational distribution is defined $q(\theta_{d,k}) = \mathcal{N}(\theta_{d,k} \mid \gamma_{d,k})$ where $\gamma_{d,k} = \{m_k, v_k\}$ for topic k has sufficient statistics:

$$\text{ss}(\boldsymbol{\theta}_{d,k}) = \begin{bmatrix} \theta_{d,k} & \theta_{d,k}^2 \end{bmatrix}^\top$$

Writing the approximate posterior $q(\boldsymbol{\theta})$ as a product of the conjugate and non-conjugate parts gives:

$$q_{i+1}(\boldsymbol{\theta}) \propto \Big[\prod_{k=1}^{K} \exp(\text{ss}(\theta_{d,k}) \tilde{\boldsymbol{\Theta}}_{k,i}) \Big] \mathcal{N}(\boldsymbol{\theta}_d \mid \boldsymbol{\alpha}_t \mathbf{x}_d, \Sigma_t)$$

where $\tilde{\boldsymbol{\Theta}}_{k,i}$ are the natural parameters to the approximated exponential family at iteration i.

The difficult non-conjugate term $\text{Mult}(\mathbf{z}_n \mid \sigma(\boldsymbol{\theta}))$ has already been approximated in Blei and Lafferty (2006a), specifically:

$$\begin{aligned} f &= \mathbb{E}_q \Big[\log \text{Mult}(\mathbf{z}_n \mid \sigma(\boldsymbol{\theta})) \Big] \\ &\geqslant \sum_{k=1}^{K} m_k \phi_n^{(k)} - \zeta^{-1} \sum_{k=1}^{K} \exp(m_k + \frac{v_k}{2}) - \log(\zeta) + 1 \end{aligned} \tag{3.10}$$

where, again, the parameter ζ is introduced to preserve a lower bound. Thus, we can now take the gradient of $f = \mathbb{E}_q[\log \text{Mult}(\mathbf{z}_n \mid \sigma(\boldsymbol{\theta}))]$ with respect to the mean parameters.

$$\nabla_\mu f = \begin{bmatrix} \frac{\partial f}{\partial \mu^{(1)}} & \frac{\partial f}{\partial \mu^{(2)}} \end{bmatrix}^\top$$

Since the mean variational distribution's mean parameters are

$$\boldsymbol{\mu} = \begin{bmatrix} m_k & m_k^2 + v_k \end{bmatrix}^\top,$$

we can write $m_k = \mu_k^{(1)}$ and $v_k = \mu_k^{(2)} - (\mu_k^{(1)})^2$. By the chain rule the gradient with respect to the mean parameters are $\frac{\partial f}{\partial \mu^{(1)}} = \frac{\partial f}{\partial m} - 2\frac{\partial f}{\partial v}$ and $\frac{\partial f}{\partial \mu^{(2)}} = \frac{\partial f}{\partial v}$. Applying these gradients to the non-conjugate term in (3.10) we can then compute the natural parameter of the variational posterior:

$$\begin{aligned} \frac{\partial f}{\partial m} - 2\frac{\partial f}{\partial v} m = \phi_n^{(k)} \quad \text{and} \quad \frac{\partial f}{\partial v_k} = \frac{1}{2\zeta}\exp(m_k + \frac{v_k}{2}) \\ \implies \nabla_{\boldsymbol{\mu}}(f) = \begin{bmatrix} \phi_n^{(k)} & -\frac{1}{2\zeta}\exp(m_k + \frac{v_k}{2}) \end{bmatrix}^\top \end{aligned} \tag{3.11}$$

where ζ has the same update as before: $\zeta \leftarrow \sum_{k=1}^{K} \exp(m_k + \frac{v_k}{2})$. Thus by the CVI update rules given in (3.2), the natural parameter of the topic proportions is given by a *conjugate computation* adding (3.11) (the sufficient statistics) to (3.9) (the natural parameter of prior):

$$\boldsymbol{\Theta}_{k,i+1} = \rho_i \begin{bmatrix} \sum_{n=1}^{N_d} \phi_n^{(k)} + \Sigma_t^{-1}(\boldsymbol{\alpha}_t \mathbf{x}_d)_k \\ \frac{-N}{2\zeta}\exp(m_k + \frac{v_k}{2}) + (-\frac{1}{2}\Sigma_{t,k,k}^{-1}) \end{bmatrix} + (1-\rho_i)\boldsymbol{\Theta}_{k,i} \tag{3.12}$$

Ideally we want to compute the source parameters to the variational posterior, i.e. m_k and v_k—which is straight-forward[1] given the natural parameter $\boldsymbol{\Theta}_{k,i+1}$ computed in (3.12). The final updates to the variational posterior's source mean and variances are computed:

$$m_{k,i+1} = \frac{-\Theta_{k,i+1}^{(1)}}{2\Theta_{k,i+1}^{(2)}} \quad \text{and} \quad v_{k,i+1} = \frac{-1}{2\Theta_{k,i+1}^{(2)}}$$

TOPIC ASSIGNMENT Each word is assigned to a topic through a hidden parameter $\mathbf{z}_n$. For $\mathbf{z}_n$ the corresponding conjugate term is $\tilde{p}_c^{\mathbf{z}} = \text{Mult}(\mathbf{w}_n \mid \boldsymbol{\beta}_{z_n})$, and non-conjugate term is $\tilde{p}_{nc}^{\mathbf{z}} = \text{Mult}(\mathbf{z}_n \mid \sigma(\boldsymbol{\theta}))$.

Define the variational distribution $q(\mathbf{z}_n) = \text{Mult}(\mathbf{z}_n \mid \phi_n)$ for word n. Writing the approximate posterior $q(\mathbf{z}_n)$ as a product of the conjugate and non-conjugate parts:

$$q_{i+1}(\mathbf{z}_n) \propto \Big[\prod_{n=1}^{N_d} \exp(ss(\mathbf{z}_n)\tilde{\boldsymbol{\Phi}}_i)\Big] \text{Mult}(\mathbf{w}_n \mid \boldsymbol{\beta}_{z_n}) \tag{3.13}$$

where $\tilde{\boldsymbol{\Phi}}_i$ are the natural parameters to the approximated exponential family at iteration i, computed by:

1 These follow from the definitions for converting a Multivariate Gaussian between its source and natural parameters, which can easily be looked up, see for example Nielsen and Garcia (2009).

$$\begin{aligned}\tilde{\boldsymbol{\Phi}}_i &= \begin{bmatrix}\tilde{\Phi}_{1,i} & \dots & \tilde{\Phi}_{K,i}\end{bmatrix}^\top \\ &= \rho_i \nabla_\phi \mathbb{E}_{q_i}\Big[\log \text{Mult}(\mathbf{z}_n \mid \sigma(\boldsymbol{\theta}))\Big]|_{\phi=\phi_i} + (1-\rho_i)\tilde{\boldsymbol{\Phi}}_{i-1}\end{aligned}$$

Using the previously computed approximation to the challenging term $\log \text{Mult}(\mathbf{z}_n \mid \sigma(\boldsymbol{\theta}))$ in (3.11), we now differentiate it with respect to the mean parameter of $q(\mathbf{z}_n)$, i.e. ϕ. This gives:

$$\nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_i}\Big[\log \text{Mult}(\mathbf{z}_n \mid \sigma(\boldsymbol{\theta}))\Big] = \begin{bmatrix} m_1 & \dots & \dots m_K\end{bmatrix}^\top = \mathbf{m} \tag{3.14}$$

where m_k is the mean of $q(\theta_{d,k}) = \mathcal{N}(\theta_{d,k} \mid m_k, v_k)$, for $k \in 1,\dots,K$. To update the natural parameter to the approximated exponential family in (3.13), we use the equations:

$$\begin{aligned}\tilde{\boldsymbol{\Phi}}_i &= \rho_i\Big(\nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_i}[\log \text{Mult}(\mathbf{z}_n \mid \sigma(\boldsymbol{\theta}))]\Big) + (1-\rho_i)\tilde{\boldsymbol{\Phi}}_{i-1} \\ &= \rho_i \mathbf{m} + (1-\rho_i)\tilde{\boldsymbol{\Phi}}_{i-1}\end{aligned} \tag{3.15}$$

Since CVI transforms non-conjugate computations into conjugate computations the update to the variational parameter $\boldsymbol{\phi}_n$ is similar to the LDA case. Specifically, we compute the source parameter of our variational posterior $\boldsymbol{\phi}_n$ by:

$$\phi_{n,k,i+1} \propto \exp(\tilde{\Phi}_{k,i} + \mathbb{E}_q[\log \beta_{k,w_n}]) \tag{3.16}$$

where, as usual, the Dirichlet expectation $\mathbb{E}_q[\log \beta_{k,v}]$, is computed by $\Psi(\lambda_{k,v}) - \Psi(\sum_{j=1}^{V} \lambda_{k,j})$. The first term $\tilde{\Phi}_{k,i}$ is the document's topic distribution computed in the previous section, hence (as expected) the CVI update for $\boldsymbol{\phi}$ results in a simple closed form update with the same form as in the original DAP model.

PERSONA ASSIGNMENT For each document the author is assigned to a persona through a hidden parameter $\mathbf{x}_d$. For $\mathbf{x}_d$ the conjugate term is $\tilde{p}^{\mathbf{x}}_c = \text{Mult}(\mathbf{x}_a \mid \kappa_{d_a})$, and its non-conjugate term is $\tilde{p}^{\mathbf{x}}_{nc} = \mathcal{N}(\boldsymbol{\theta} \mid \alpha_t \mathbf{x}_d, \Sigma_t)$, where the coupling in the mean $\alpha_t \mathbf{x}_d$ made closed form updates in the DAP model impossible.

Define the variational distribution $q(\mathbf{x}_d) = \text{Mult}(\mathbf{x}_d \mid \tau_d)$. Writing the approximate posterior $q(\mathbf{x}_d)$ as a product of the conjugate and non-conjugate parts gives:

$$q_{i+1}(\mathbf{x}_d) \propto \Big[\prod_{p=1}^{P} \exp(ss(\mathbf{x}_d)\tilde{\tau}_{d,p,i})\Big] \text{Mult}(\mathbf{x}_d \mid \boldsymbol{\kappa}_{d_a}) \tag{3.17}$$

where $\tilde{\boldsymbol{\tau}}_{d,i}$ are the natural parameters to the approximated exponential family at iteration i.

We compute the variational posterior by first taking the gradient of the non-conjugate terms, where the non-conjugate term $f = \mathbb{E}_{q_i}[\mathcal{N}(\boldsymbol{\theta} \mid \alpha_t \mathbf{x}_d, \Sigma_t)]$ evaluates to:

$$f = \frac{-1}{2}\Big((\gamma_{t,d} - \hat{\alpha}_t \tau_{t,d})^\top \Sigma_t^{-1}(\gamma_{t,d} - \hat{\alpha}_t \tau_{t,d}) + \sum_{p=1}^{P} \operatorname{Tr}\left[\Sigma_t^{-1} \operatorname{diag}\left(\tau_{t,d,p}(\hat{\alpha}_{t,p}\hat{\alpha}_{t,p}^\top + \hat{\Sigma}_t)\right)\right]\Big) + \text{const}$$

Taking the gradient with respect to the mean parameter $\boldsymbol{\tau}$ gives:

$$\nabla_\tau f = \hat{\alpha}_{t,p}\Sigma_t^{-1}(\gamma_{t,d} - \hat{\alpha}_{t,p}\tau_{t,d,p}) - \frac{1}{2}\operatorname{Tr}(\Sigma_t^{-1} \operatorname{diag}(\hat{\alpha}_{t,p}^2 + \hat{\Sigma}_t))$$

Next we use the CVI updates rule from (3.2) to compute the natural parameter of our variational posterior $q(\mathbf{x}_d)$ by:

$$\tilde{\boldsymbol{\tau}}_{d,i} = \rho_i\Big(\mathbb{E}_q[\log \boldsymbol{\kappa}_{d_a}] + \nabla_\tau f\Big) + (1 - \rho_i)\tilde{\boldsymbol{\tau}}_{d,i-1}$$

where $\mathbb{E}_q[\log \boldsymbol{\kappa}_{d_a}]$ is the expected natural parameters from the conjugate term, and is equivalent to a Dirichlet expectation: $\Psi(\delta_{a,p}) - \Psi(\sum_{j=1}^{P} \delta_{a,j})$. In order to map the natural parameter $\tilde{\boldsymbol{\tau}}_{d,i}$ back to the source parameter of the variational posterior $q(\mathbf{x}_d)$, we use

$$\boldsymbol{\tau}_{d,i} = \left[\frac{\exp(\tilde{\boldsymbol{\tau}}_{d,1,i})}{\sum_{p=1}^{P} \tilde{\boldsymbol{\tau}}_{d,p,i}} \quad \cdots \quad \frac{\exp(\tilde{\boldsymbol{\tau}}_{d,P,i})}{\sum_{p=1}^{P} \tilde{\boldsymbol{\tau}}_{d,p,i}}\right]^\top$$

These closed form updates replace the standard approach of the DAP model (Giaquinto and Banerjee, 2018b) which takes the gradient of the ELBO with respect to the author-persona assignment parameter and requires running exponentiated gradient descent for each document.

PERSONA TOPIC DISTRIBUTION The parameter $\hat{\boldsymbol{\alpha}}_t$ represents the noisy estimate of $\boldsymbol{\alpha}_t$, which captures the distribution over topics for each persona and evolves over time. After calculating $\hat{\boldsymbol{\alpha}}_t$, the forward and backward equations will be applied in the M-step to give a final posterior estimate $\boldsymbol{\alpha}_t$. The terms in the ELBO containing $\hat{\boldsymbol{\alpha}}_t$ are found by expanding $\mathbb{E}_q[\log p(\boldsymbol{\alpha}_{t,p}) \mid \boldsymbol{\alpha}_{t-1,p})]$ for (3.18a) and $\mathbb{E}_q[\log p(\boldsymbol{\theta}_{t,d} \mid \boldsymbol{\alpha}_t \mathbf{x}_{t,d}, \boldsymbol{\Sigma}_t)]$ for (3.18b) and (3.18c):

$$\mathcal{L}(\hat{\boldsymbol{\alpha}}) = \sum_{t=1}^{T}\sum_{p=1}^{P} -\frac{1}{2}(\hat{\boldsymbol{\alpha}}_{t,p} - \hat{\boldsymbol{\alpha}}_{t-1,p})^\top \Sigma_t^{-1}(\hat{\boldsymbol{\alpha}}_{t,p} - \hat{\boldsymbol{\alpha}}_{t-1,p}) \tag{3.18a}$$

$$+ \sum_{t=1}^{T}\sum_{d=1}^{D_t} -\frac{1}{2}\Big((\boldsymbol{\gamma}_{t,d} - \hat{\boldsymbol{\alpha}}_t\boldsymbol{\tau}_{t,d})^\top \boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\gamma}_{t,d} - \hat{\boldsymbol{\alpha}}_t\boldsymbol{\tau}_{t,d}) \tag{3.18b}$$

$$+ \sum_{t=1}^{T}\sum_{d=1}^{D_t}\sum_{p=1}^{P} \operatorname{Tr}\left[\boldsymbol{\Sigma}_t^{-1}\operatorname{diag}\left(\tau_{t,d,p}(\hat{\boldsymbol{\alpha}}_{t,p}\hat{\boldsymbol{\alpha}}_{t,p}^\top + \hat{\boldsymbol{\Sigma}}_t)\right)\right]\Big) \tag{3.18c}$$

Taking the derivative with respect to the mean term for each persona gives the closed form update:

$$\hat{\boldsymbol{\alpha}}_{t,p} = \frac{\hat{\boldsymbol{\alpha}}_{t-1,p} + \sum_{d=1}^{D_t}(\boldsymbol{\gamma}_{t,d} - 1)\tau_{t,d,p}}{1 + \sum_{d=1}^{D_t}\tau_{t,d,p}^2} \tag{3.19}$$

We solve for $\hat{\boldsymbol{\alpha}}_{t,p}$ sequentially over time steps. For the initial time step $t = 1$, we use the prior μ_0 in place of $\hat{\boldsymbol{\alpha}}_{t-1,p}$. Note that the summations in (3.19) are collected during the E-step and $\hat{\boldsymbol{\alpha}}_{t,p}$ need only be computed once after performing inference on all documents.

3.5.2 *M-Step*

In the M-step we use sufficient statistics collected from computing document-level variational parameters computed during the E-step to update the global parameters $\boldsymbol{\beta}, \boldsymbol{\kappa}$, and $\boldsymbol{\alpha}$. Because DAPPER makes use of stochastic mini-batches, we use the learning rate defined for SVI and recommended in CVI: $\rho_i = (i + \tau)^{-\kappa}$ where $\tau \geqslant 0$ is the delay and $\kappa \in (0.5, 1.0]$ is the forgetting rate (Hoffman et al., 2013; Hoffman, Blei, and Bach, 2010; Khan and Lin, 2017).

TOPIC'S DISTRIBUTION OVER WORDS The parameter $\boldsymbol{\beta}$ represents the distribution over words for each topic. The $\boldsymbol{\beta}$ term is already conjugate, and hence the variational distribution for the topics, $q_{i+1}(\boldsymbol{\beta}) = \prod_{k=1}^{K}\operatorname{Dir}(\boldsymbol{\beta}_k \mid \boldsymbol{\lambda}_{k,i+1})$, already has a closed form update:

$$\boldsymbol{\lambda}_{k,i+1} = (1 - \rho_i)\boldsymbol{\lambda}_{k,i} + \rho_i(\eta + \sum_{d=1}^{D}\sum_{n=1}^{N_d}\phi_{d,n,k}w_{d,n}) \,.$$

AUTHOR'S DISTRIBUTION OVER PERSONAS The parameter $\boldsymbol{\delta}$ represents the distribution over personas for each author. The closed form update for $\delta_{a,p}$ is:

$$\boldsymbol{\delta}_{d_a,i+1} = (1 - \rho_i)\boldsymbol{\delta}_{d_a,i} + \rho_i(\omega + \sum_{d=1}^{D}\boldsymbol{\tau}_d)$$

shows that $\boldsymbol{\delta}$'s closed form update is an average of the persona assignments, smoothed by the author-persona prior ω, and ρ_i defines the convect combination between the two terms.

PERSONA'S DISTRIBUTION OVER TOPICS Stochastic mini-batch updates are derived for the variational observations $\hat{\boldsymbol{\alpha}}_{t,p,i+1}$ by first computing $\hat{\boldsymbol{\alpha}}_{t,p}^{new}$ from (3.22) for a small mini-batch of documents. Here CVI compliments RVI because closed form updates, such as the $\hat{\boldsymbol{\alpha}}_{t,p}$ update found by the DAP model, are preserved. Then $\hat{\boldsymbol{\alpha}}_{t,p,i+1}$ can be updated via a convex combination between the previous iteration $\hat{\boldsymbol{\alpha}}_{t,p,i}$ and the mini-batch estimate $\hat{\boldsymbol{\alpha}}_{t,p}^{new}$, that is:

$$\hat{\boldsymbol{\alpha}}_{t,p,i+1} = (1-\rho_i)\hat{\boldsymbol{\alpha}}_{t,p,i} + \rho_i \hat{\boldsymbol{\alpha}}_{t,p}^{new}$$

After computing $\hat{\boldsymbol{\alpha}}_{t,p,i+1}$, our approach follows the variational Kalman filtering method from Wang's Continuous Time Dynamic Topic Model. Specifically, we employ the Brownian motion to model time dynamics. However, because our model's time-varying parameter is a distribution over latent topics, it performs best on data discretized in time (resulting in a smaller T). The forward equations mimic a Kalman filter:

$$\begin{aligned} m_{t,p} &= \frac{\hat{\boldsymbol{\alpha}}_{t,p} P_{t,p} + m_{t-1,p}\hat{w}_t}{P_{t,p} + \hat{w}_t} \\ V_{t,p} &= \hat{w}_t \frac{P_{t,p}}{P_{t,p} + \hat{w}_t} \end{aligned}$$

where $\hat{w}_t$ is the known process noise, and $P_{t,p} = V_{t-1,p} + \sigma\Delta_{s_t}$ captures the increase in variance as time between data points grows. Finally, the Kalman filters backward equations give the final update to $\boldsymbol{\alpha}$ and $\boldsymbol{\Sigma}$:

$$\begin{aligned} \boldsymbol{\alpha}_{t-1,p} &= m_{t-1,p}\frac{\sigma\Delta_{s_t}}{P_{t,p}} + \boldsymbol{\alpha}_{t,p}\frac{V_{t-1,p}}{P_{t,p}} \\ \boldsymbol{\Sigma}_{t-1,p} &= V_{t-1,p} + \frac{(V_{t-1,p})^2}{(P_{t,p})^2}(\boldsymbol{\Sigma}_{t,p} - P_{t,p}) \ , \end{aligned}$$

3.6 REGULARIZED VARIATIONAL INFERENCE

Our *regularized variational inference* (RVI) algorithm nudges $\boldsymbol{\alpha}_t$ to find topic distributions that are different for each persona. A natural choice for capturing this idea is an inner product between each of the personas (excluding a persona with itself). Hence, we define the regularization function by:

$$\mathrm{pr}(q) = \sum_{p=1}^{P} \sum_{1 \leqslant q \leqslant P, q \neq p} \frac{D_t}{2} \rho \hat{\boldsymbol{\alpha}}_{t,p}^{\top} \boldsymbol{\Sigma}_t^{-1} \hat{\boldsymbol{\alpha}}_{t,q} \ , \tag{3.20}$$

Figure 4: DAP *without* Regularized Variational Inference. Here, the DAP model fits every topic trajectory to a similar high probability solution. The goal of RVI is to nudge the model to seek out a diverse set of topic trajectories—unlike the homogenous set of solutions shown here.

$$r(q) \propto \sum_{q \neq p} \boldsymbol{\alpha}_p^\top \boldsymbol{\alpha}_q \tag{3.21}$$

The parameter $\boldsymbol{\Sigma}_t^{-1}$ in included in the regularization for two reasons. First, it simplifies the update to $\hat{\boldsymbol{\alpha}}_{t,p}$. In (3.18) the term $\boldsymbol{\Sigma}^{-1}$ appears in every term, which allows it to be factored out and canceled. By including $\boldsymbol{\Sigma}^{-1}$ in the regularization the same cancellation can occur. Second, since $\boldsymbol{\Sigma}_t^{-1} \propto I$ then its inclusion has the effect of encouraging personas to be orthogonal to one another. We include the number of documents D_t at time t in $r(q)$ so that the regularization is applied evenly, regardless of dataset size or a skewed distribution of documents over time. After including the regularization term in (3.20) with the ELBO terms in (3.18), the regularized $\hat{\boldsymbol{\alpha}}_{t,p}$ update is:

$$(1 + \sum_{d=1}^{D_t} \tau_{d,p}^2)\hat{\boldsymbol{\alpha}}_{t,p} + \rho D_t \sum_{q \neq p} \hat{\boldsymbol{\alpha}}_{t,q} = \hat{\boldsymbol{\alpha}}_{t-1,p} + \sum_{d=1}^{D_t} (\boldsymbol{\gamma}_d - 1)\tau_{d,p} \tag{3.22}$$

Since the vector $\sum_{d=1}^{D_t} (\boldsymbol{\gamma}_d - 1)\tau_{d,p}$ (of length K) is computed during the E-step, then the RHS is known. Similarly, the term $(1 + \sum_{d=1}^{D_t} \tau_{d,p}^2)$ is known, and in combination with ρD_t form the weights over the unknown vector $\hat{\boldsymbol{\alpha}}_{t,p}$, also of length K. Therefore, (3.22) can be solved as a system of linear equations. Through experiments we've found an optimal value of $\rho \in [0, 0.5]$. The model exhibits sensitivity to the hyperparameter ρ, if ρ is large (e.g. > 1.0) then model quality drops due to personas overfitting to a single topic. Since $\hat{\boldsymbol{\alpha}}$ is only used to estimate the global parameter $\boldsymbol{\alpha}$ during the M-step, computing $\hat{\boldsymbol{\alpha}}$ isn't necessary for inference on holdout datasets.

3.7 RELATED WORK

Much of the research on topic modeling builds on the latent Dirichlet allocation (LDA) model Blei, Ng, and Jordan, 2003. The LDA model doesn't account for meta-information like authorship or time. Nevertheless, interest in LDA has endured, in part, due its ability to richly describe topics as distributions over words and documents as mixtures of topics. In the years since LDA's introduction, others have extended the idea to compliment corpora with a variety of structures and metadata.

Author information is common in many corpora. A few topic models are designed to identify authors' preferences for certain topics, and the relationships between authors McCallum, Corrada-Emmanuel, and Wang, 2005; Mimno and McCallum, 2007; Mimno and McCallum, 2008; Pathak et al., 2008; Rosen-Zvi et al., 2004; Steyvers et al., 2004. Corpora with a temporal structure, such as scientific journals or newspaper articles, are the focus of many of temporal topic models Blei and Jordan, 2006; Wang, Blei, and Heckerman, 2008; Wang and McCallum, 2006; Wei, Sun, and Wang, 2007.

TEMPORAL TOPIC MODELS. Two topic models set the standard of comparison for topic modeling on corpora with a temporal element: the dynamic topic model (DTM) Blei and Lafferty, 2006b and the topics over time model TOT Wang and McCallum, 2006. These two models represent very different approaches to modeling time in a topic model.

The TOT model defines time as an observed variable, which leads to a continuous treatment of time and the ability to predict timestamps of documents. Alternatively, the DTM evolves topics over time using a Markov process. In many corpora the evolution of topics provides interesting insights. For example, Blei's model of the *Science* corpus shows words associated with a topic on physics changing over a century.

Building directly on the DTM, in 2008 Wang et al. developed the continuous time dynamic topic model (CDTM) which uses continuous Brownian motion to model the evolution of topics over time Wang, Blei, and Heckerman, 2008. This is a major development in temporal topic models because, unlike the DTM, it doesn't require partitioning the data into discrete time periods. Instead, the model assumes that at each time step the variance in the topic proportions increases proportional to the duration since the previous document. Similar to Wang et al., the Dynamic Mixture Model (DMM) is built for continuous streams of text Wei, Sun, and Wang, 2007. In the DMM, however, topics are fixed in time and the model captures the evolution of document-level topic proportions over time.

TOPIC MODELINGS OF HEALTH JOURNEYS. In many topic modeling applications to temporal corpora, the time component is ignored. For example, Wen et al. model cancer event trajectories from users of an online forum for breast cancer support Wen and Rose, 2012. Wen's approach uses LDA to extract cancer event keywords, which are then linked together in time by temporal descriptions mined from the text. This

work demonstrates a quantitative approach to studying the dynamics of social support network, and offer a powerful look at the experiences of users in these support networks.

Numerous studies have shown that support networks, both in person and online, are valuable tools for those suffering from chronic conditions or life-threatening illness and caregivers Beaudoin and Tao, 2007; Rodgers and Chen, 2005; Wen and Rose, 2012. Additionally, online social networks can serve as a way to efficiently disseminate information regarding someone's status to their community. Understanding the health journeys of users in these social support communities is valuable information for improving user experience. Topic models are uniquely suited to succinctly describing and analyzing these health journeys.

3.8 EXPERIMENTS AND RESULTS

3.8.1 *Experimental Setup*

To evaluate the performance of the DAP and DAPPER topic models we perform a quantitative comparison with similar topic models (LDA, DTM, and CDTM), and a qualitative demonstration of DAPPER's scalability and output on the CB[2] and Signal Media One-Million News Article[3] (SM) corpora (Corney et al., 2016). Additionally, the speed and efficiency of DAPPER relative to its predecessor are measured by showing model performance as a function of training time. The qualitative comparison demonstrates the rich and compelling "health journeys" discovered by DAPPER on the CB corpus.

3.8.1.1 *CaringBridge Journals*

The creation of our model is inspired by a desire to discover topics on a unique dataset consisting of 14 million journals posted by half a million authors on the social networking site CaringBridge (CB). Established in 1997, CaringBridge is a 501(c)(3) non-profit organization focused on connecting people and reducing the feelings of isolation that are often associated with a patient's health journey. Due to their content, CB data has been anonymized prior to analysis.

3.8.1.2 *Pre-processing of Texts*

All texts are pre-processed by removing common stopwords and reducing words to their lemma forms. Next, we build a vocabulary consisting of the 10,000 most common

2 CB data were acquired with the permission and collaboration of CB leadership in accordance with CB's Privacy Policy & Terms of Use Agreement. Because of their highly sensitive content the CB dataset has been anonymized, but deidentification techniques are imperfect (Narayanan and Shmatikov, 2010) and hence we cannot publicly release the CB dataset. Those interested in the dataset are encouraged to contact the investigators. All code for training the DAPPER model and running our experiments on the SM dataset, however, are available at https://github.com/robert-giaquinto/dapper.

3 http://research.signalmedia.co/newsir16/signal-dataset.html

remaining words. Timestamps are converted into a discrete, relative measure: for CB we measure the number of weeks since author's first post, and for SM we use the day within span of the corpus (1-30 September, 2015).

From the CB journals we examine only the first year of each authors journals in order to avoid very small sample sizes in the final time points. Finally, we ignore all journals containing less than 10 words. The resulting set of journals, denoted *1st-Year All*, contains 200,388 authors and 9,010,623 journals (with a total of 937,503,945 words). In the *1st-Year All* dataset, authors write, on average, 100 words per journal and 45 journal posts in the first year.

3.8.1.3 *Evaluation Datasets*

QUANTITATIVE CORPORA. To quantitatively evaluate the performance of the DAP and DAPPER topic models we user subsets of the CB and SM datasets.

From the SM dataset we only consider articles written by bloggers who wrote fewer than one blog post per day during the corpus' one month span. Subsetting the data in this way is done to exclude major news organizations and instead focus on bloggers who typically write about a central theme. We refer to the subset as *SM-blogs*. After pre-processing, the *SM-blogs* corpus consists of 97,839 documents for training (15,848 blogs, 19,278,689 total words), and 10,887 documents for testing (same authors, 2,165,634 words).

From the pre-processed CB journals we draw the *CB-subset* dataset by imposing the constraint that authors who post least twice a month during the first year helps identify a set of active users. From the 123K authors meeting these criteria, 2,000 were randomly selected. Journals written by these 2,000 authors total 114,532. From here 90% of the journals ($N = 103,018$) are divided into the training set, and the remaining 10% of each author's journals ($N = 11,728$) make up the test set. Training and test sets contain the same authors because personas distributions are learned for each author during training. Overall, authors in the *CB-subset* corpus journal an average of 57 times, with a mean of 5 days between journal posts.

To compare DAPPER to its predecessor we examine model performance versus (1) training time, as well as (2) number of epochs over the corpus. Showing performance as a function of training time highlights convergence rate of each model. On the other hand, performance as a function of epochs shows how much information the model is able to extract from each pass over the data. Model comparison calculations are performed on a single Haswell E5-2680v3 processor and stopped after 48 hours of training time.

QUALITATIVE CORPUS. We demonstrate scalability and quality of DAPPER's output on the full *1st-Year All* dataset. The *1st-Year All* corpus, consisting of 200,388 authors and 9,010,623 journals, serves to demonstrate the scalability of the DAPPER topic model. From this full dataset, 22,552 documents are set aside as a test set to evaluate the model and track convergence, leaving 8,988,071 journals in the training set. Our

goal in training the DAPPER model on this dataset is to show that the model can find compelling qualitative results even on massive, complex datasets.

3.8.1.4 *Evaluation Metric*

To evaluate the models we compute the per-word log-likelihood (PWLL) on heldout data, which measures how well the model fits the data and is computed by $\text{PWLL} = \frac{\sum_{d=1}^{D} \log p(\mathbf{w}_d)}{\sum_{d=1}^{D} N_d}$. Note that perplexity, another common metric used to compare topic models, is related to PWLL via $\text{perplexity} = \exp(-\text{PWLL})$. It has been shown that perplexity (and hence PWLLs) don't correlate with a model finding topics that human raters agree to be coherent (Chang et al., 2009). Nevertheless, PWLLs provide a fair way to compare how well each model optimizes their objective functions.

3.8.1.5 *Hyperparameters*

To ensure a fair comparison we fix hyperparameters appearing in each of the models, such as number of topics and convergence criteria. Relative differences between model performances don't vary significantly depending on the number of topics chosen, and hence we only report results for models with 25 topics on the CB-subset and 50 topics on the SM-blogs. DAP and DAPPER seek 15 and 25 latent personas for the CB-subset and SM-blogs corpora, respectively, and fix their regularization of personas to $\rho = 0.2$. Because DAPPER can be trained on stochastic mini-batches we report results for various mini-batch sizes and full-batch training.

3.8.2 *Results*

In addition evaluating model fit, we perform a qualitative analysis of the DAP model to highlight the quality and usefulness of the personas discovered. In particular, we establish that the personas are unique from one-another and capture meaningful experiences shared by authors.

3.8.2.1 *Model Performance Comparison*

In Table 2 we list the per-word log-likelihood for each of the competing topic models. There is a significant improvement in the DAP model's performance after regularization. Further analysis of the likelihood computation reveals that the regularization term contributes a relatively small drop in likelihood compared to the total likelihood during training. Nevertheless, these results show that even a small amount of regularization can nudge the model to seek out quality results. In testing additional ρ values we found that, in general, $\rho \in [0.1, 0.3]$ faired comparably. Larger values of ρ can cause model instability and the document likelihoods to have long-tailed distributions. The emergence of outlier document-likelihoods is unsurprising, regularization encourages the personas to focus on different topics—hence, large values of ρ inevitably result in personas that overfit.

Model	PWLL, 24 Hrs, 1 CPU	PWLL Final
DAPPER (full batch)	**-6.73**	**-6.49**
DAPPER (batch size = 512)	-8.19	-8.13
DAP (ρ=0.2)	-8.09	-6.47
DAP (ρ=0.0)	-8.82	-7.22
CDTM	-8.81	-8.81
DTM	-9.59	-9.59
LDA	-9.23	-9.23

Table 2: Overall comparison of model performance, measured by per-word log-likelihoods computed for documents in the test dataset. Separate columns indicate performance after 24 hours training on a single processor, and performance after the model had converged. Despite not converging within 24 hours the DAP model achieves comparable performance to similar models. Moreover, by including a regularization term ($\rho = 0.2$) the performance of the DAP model increases significantly. The DAPPER model, however, shows significant performance improvements over all competing models due to its faster method for handling non-conjugate terms.

In additional to training models to converge we report performance after 24 hours of training. Each model is trained for a maximum of 24 hours on a single Haswell E5-2680v3 processor or until training performance converged—although the DAP model is the only model not to converge within 24 hours. Performance for DAPPER is shown for using $\rho = 0.2$ and a mini-batch size of 512, which achieved the best training set performance after 24 hours, and DAPPER by computing gradients on the full corpus before each M-step (as opposed to mini-batches), which achieved the best overall test performance.

Table 2 highlights three important results: first, the DAP topic model which is designed for the multi-author, temporal structure of the CB dataset achieves competitive performance but clearly suffers by not converging within 24 hours. Second, the DAPPER model benefits from faster training and achieves state-of-the-art performance. Third, smaller mini-batches like 512 result in good training performance that converges quickly but the model does not generalize as well as DAPPER trained on a full batch.

Figure 5 shows mean per-word log-likelihoods at each time step within the dataset. The best performing DAP model shows consistently better results over competing models. However, the unregularized DAP model has a significant drop in performance in the first time step.

3.8.2.2 *Comparison of Model Hyperparameters*

In Table 3 we show the performance of the DAPPER model on the SM-blogs corpus with varying hyperparameter settings. Specifically, we train models with [100, 75, 50, 25] topics, [50, 25, 15] latent personas, and mini-batch sizes of either [256, 512, 1024, 2048] or full gradient training. Full batch training results in the highest per-word log-likelihoods on the test set. Varying the number of personas and the number of topics

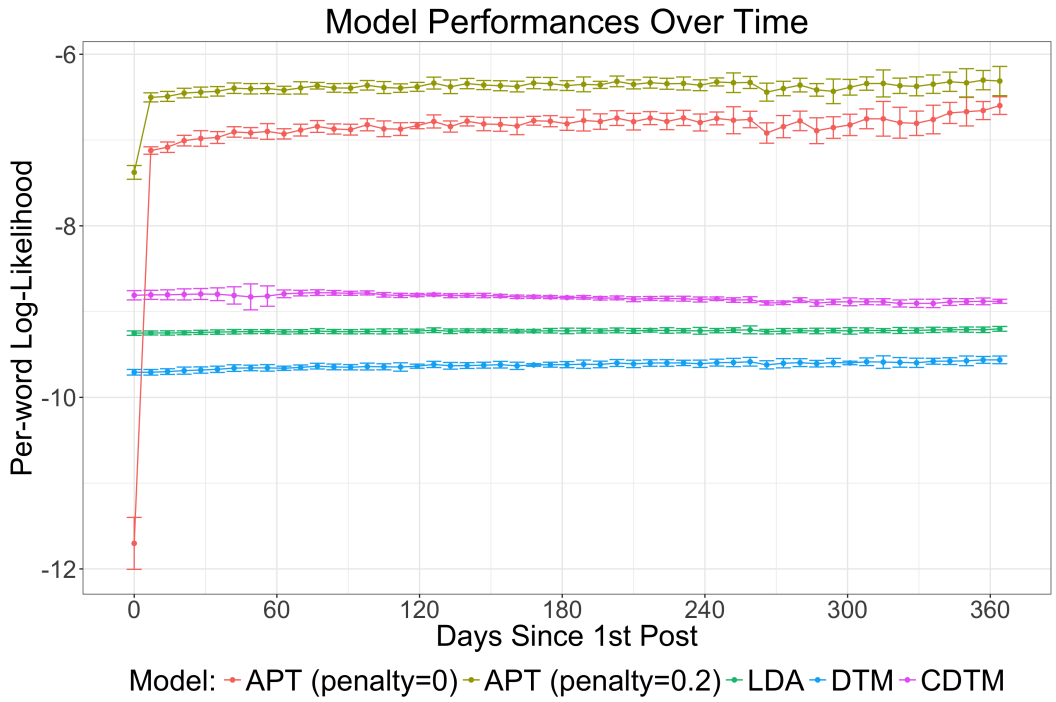


Figure 5: In general the DAP model performs better than competing models over time steps. The regularized DAP model further improves performance and reduces variable results found in the first time step of the unregularized model. Error bars show one standard deviation in document-level PWLL, computed by running 10 fold cross-validation on the *CB-subset* dataset. The DAP models show state-of-the-art performance even on later timesteps in which there are significantly fewer journals per timestep.

has a noticable impact on performance (smaller models tend to do slightly better), however batch size is the most significant factor in achieving optimal performance. Smaller models (in terms of number of personas and topics) tend to do well on the 97,839 document SM-blogs corpus, however the best models used full batch training with 50 topics and either 15 or 25 personas.

3.8.2.3 *Speed and Efficiency*

Test set performance of the DAPPER model varies significantly depending on the batch size. Shown in Figure 6 is the performance of the DAP and DAPPER models trained on the CB-subset corpus and evaluated on the training and test sets after each epoch (one full pass over the training corpus). Each epoch of the DAP model takes an average of 6.7 hours to complete, whereas the DAPPER takes roughly 0.2 hours. The right plot (training set performance) shows that all DAPPER batch sizes begin to converge to a similar value. The training results (left plot in in Figure 6) show that all batch sizes converge to a similar value. On the test set (right plot in in Figure 6), however, larger batch sizes show better generalization.

Smaller batch sizes improve quickly at first but ultimately converge to lower PWLLs. The poor performance of small batch sizes may be due to the mini-batches being too noisy. Conversely, the larger batch sizes achieve the best performances, but improve

Number of Topics	Personas	Batch Size: 256	512	1024	2048	Full Batch
25	15	-6.46	-6.26	-6.17	-6.16	-5.65
25	25	-6.47	-6.26	-6.21	-6.23	-5.68
25	50	-6.55	-6.35	-6.34	-6.35	-5.71
50	15	-6.47	-6.30	-6.11	-6.16	**-4.97**
50	25	-6.50	-6.31	-6.30	-6.39	**-5.07**
50	50	-6.61	-6.38	-6.38	-6.52	-5.47
75	15	-6.87	-6.59	-6.46	-6.53	**-5.08**
75	25	-6.85	-6.61	-6.55	-6.61	-5.42
75	50	-6.96	-6.80	-6.79	-6.95	-6.00
100	15	-7.14	-6.93	-6.90	-6.98	-5.67
100	25	-7.07	-6.90	-6.91	-7.02	-5.99
100	50	-7.33	-7.17	-7.16	-7.31	-6.72

Table 3: Comparison of DAPPER's performance on the SM-blogs test corpus for varying number of topics, personas, and batch sizes. In general, we find the full batch training ultimately leads to higher per-word log-likelihoods. For additional hyperparameters, personas has the smallest impact on performance, and the number of topics has noticable impact. The best three models, in terms of highest test set PWLL, are highlighted in bold.

Batch Size	Hours to Exceed PWLL of -8.65	Speedup
DAP	40.43	Baseline
256	37.32	1.1x
512	2.21	18.3x
1024	1.97	20.5x
2048	2.13	19.0x
4096	4.11	9.8x
Full Batch	7.18	5.6x

Table 4: Hours of training for DAPPER to outperform DAP's best performance (PWLL $= -8.65$) on the CB-subset test dataset.

slowly *at first*. We summarize this phenomenon in Table 4, which reports how quickly DAPPER overtakes the optimal test set performance achieved by DAP. For example, a batch size of 256 converges almost immediately and takes many hours to eventually surpass DAP's best test set result. Whereas a batch size of 1024 improves steadily, and surpasses DAP's best PWLL in a fraction of the time. Despite the implication that the high variance of smaller batch sizes limits performance, we saw no benefit to gradient smoothing techniques, such as those proposed in Mandt and Blei (2014).

3.8.2.4 *Scalability and Qualitative Results*

To evaluate the quality of personas, we focus on three key elements: authors are described by one clear persona; personas are distinct from one another, as shown in

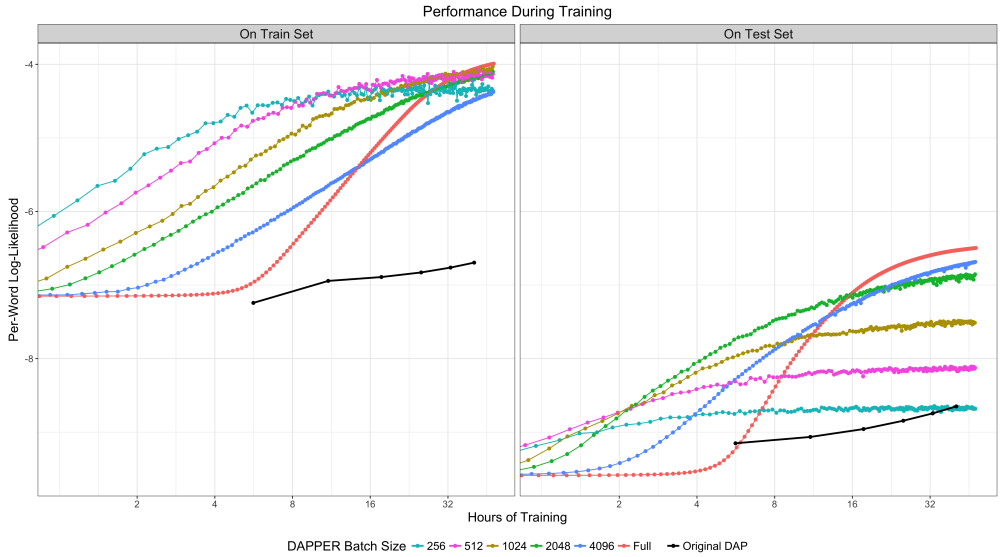


Figure 6: Per-word Log-Likelihood performance on the CB training and test sets (larger is better). Each point represents the performance evaluated at the end of an epoch. Each model was trained for a maximum of 48 hours. DAPPER, which incorporates stochastic CVI updates, achieves better likelihoods and converges faster than the DAP model trained with variational EM. Performance of the DAPPER model varies by mini-batch size.

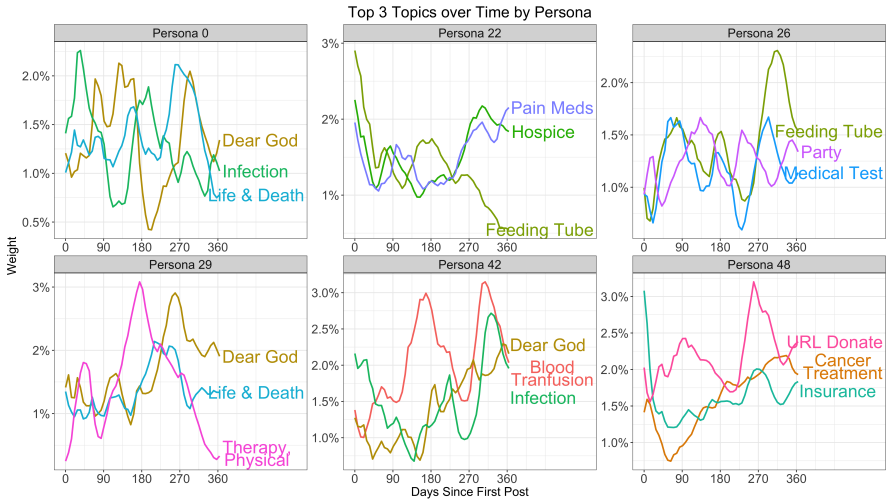


Figure 7: Selected personas learned by the DAPPER model on the full CaringBridge collection of journals. Each plot shows a different persona, and the three topics most strongly associated with that persona. For clarity, topic labels are hand-defined based on the top words in the topic and journals most associated with that topic (see Table 5). Personas show a variety of health journeys. An appeal to a higher power and prayer are common in many journals, and appear in personas 0, 29, and 42. Similarly, a deep reflection on life and death, possibly with respect to one's child appear in 0 and 29. Persona 22 captures a common experience of caring for an aging parent, beginning with intensive care and possibly ending with a hospice or nursing home. Persona 26 shows alternating periods of medical tests and intensive care with times of celebration. Personas 42 and 48 are both associated with cancer, but display very different narratives. Persona 48 includes the pair of topics "Insurance" and "URL Donation" which often appear together, indicating an author struggling with insurance and medical bills and seeking financial support from friends and family.

Infection	Life & Death	Dear God	Pain Medication	Hospice	Feeding Tube	Party	Medical Test	Therapy, Physical
infection	life	god	cause	mom	tube	school	Dr	therapy
fluid	live	lord	pain	dad	feed	birthday	test	physical
lung	child	praise	medication	visit	breathe	fun	scan	leg
remove	die	peace	brain	visitor	weight	_name_	result	therapist
procedure	others	pray	dose	hospice	oxygen	party	drug	arm
chest	moment	trust	increase	nursing	gain	aunt	MRI	rehab
pressure	fear	father	level	facility	rate	game	CT	foot
antibiotic	choose	joy	steroid	phone	ventilator	kid	liver	PT

Blood Transfusion	URL Donate	Cancer Treatment	Insurance	Church	Chemotherapy	Child	ICU	Friend, Memories
blood	_url_	cancer	provide	church	chemo	play	ICU	beautiful
count	_dollars_	radiation	medical	service	treatment	daddy	brain	friend
cell	donate	tumor	information	attend	surgery	mommy	monitor	celebrate
low	money	oncologist	disease	event	port	girl	stable	_name_
bone	benefit	surgeon	insurance	park	infusion	boy	wound	card
transplant	en	chemotherapy	condition	tree	phase	_name_	neck	memory
white	donation	breast	regard	city	spinal	little	doctor	flower
marrow	ha	biopsy	decision	honor	schedule	cute	unit	dance

Table 5: Top eight words associated with the most prevalent topics found by the DAPPER model trained on the full CaringBridge dataset. Topic labels are selected manually in order to aid reference with Figure 7. The words _dollars_, _name_, and _URL_ refer to the result of text pre-processing steps for capturing common patterns like the dollar amounts, anonymized names, and website URLs, respectively.

the combination of topics most associated with that persona; and personas capture a coherent health journey experienced by authors.

1:1 AUTHOR-PERSONA MAPPINGS. Authors are modeled as a distribution over personas; however, to create interpretable results we want these distributions to focus on a single persona. The DAP model achieves this in the majority of cases: 71% of authors are concentrated on a single persona (> 90% probability for that persona), and 27% of authors are evenly split between two personas. This shows that, in general, the model finds personas that generalize well enough to describe the majority of authors.

DISTINCT PERSONAS. The DAP model includes a regularization term specifically for encouraging personas with unique combinations of topics. We examined the top three topics associated with each persona. In the unregularized model, the 15 personas are only a mix of 6 different topics. In fact, a topic on "Life and Death" appears as a common topic for all 15 personas. On the other hand, the regularized DAP model's personas are a mix of 18 different topics. Further, the most frequently appearing topic is "Cancer (general)" (in 6 of 15 personas), which is appropriate given that approximately half of authors report cancer as a health condition.

PERSONAS REFLECT COHERENT HEALTH JOURNEYS. In Figure 7 we show the top three topics evolving over time for selected personas. The labels listed for each topic are created manually based on words and journals most associated with the topic. Words most associated with each topic are listed in Tables 5. The persona plots in Figure 7 paint a compelling picture of common health journeys experienced by CB users.

Scaling the DAPPER model to the full CB corpus makes it possible to build larger, richer models—which in turn can discover a broader range of health journeys. Many of the topics discovered by DAPPER are unrelated health conditions, for example "Eating and Food," and "Life and Death" which highlights how authors blend health and life updates in their journaling. This makes sense, a patient's condition is often known by readers or has been previously publicized on the author's homepage, thus the focus of journals is instead on the patient's current health state. Moreover, health updates tend to focus on procedures (like medical tests and tools), or more general health descriptions like side-effects, infection, pain, or specific body-parts.

SCALABILITY. To demonstrate the scalability of the DAPPER model, we train DAPPER on the *1st-Year All* corpus. Figure 7 presents selected personas discovered by a DAPPER model with 100 topics and 50 personas. The model is trained using a 24 processor machine for 94 hours, using a regularization of $\rho = 0.15$ and a batch size of 4096. We find that the increased speed of the DAPPER model makes it possible to scale it to massive datasets on standard commercial hardware. Additionally, DAPPER's stochastic updates require a constant amount of memory, whereas full batch

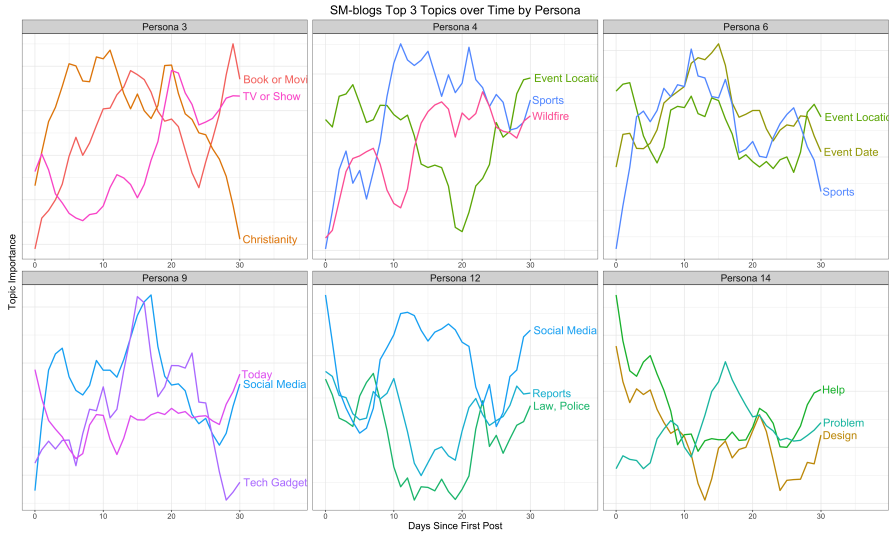


Figure 8: Selected personas learned by the DAPPER model on the SM-blogs corpus. Each plot shows a different persona, and the three topics most strongly associated with that persona. For clarity, topic labels are hand-defined based on the top words in the topic and blog posts most associated with that topic (see Table 6). Results produce by DAPPER trained with full batch gradients, 50 topics, and 15 personas, which was found to perform best during compared to other hyperparameter settings (see Table 3 for results on additional settings).

updates can require $O(N)$ memory which can be a concern for large datasets. The personas shown in Figure 7 highlight a variety of health journeys experienced by CB authors. In Table 5 we list the most likely words associated with each topic as well as the hand-defined labels assigned to each topic.

Finally, in Figure 8 we share qualitative results from DAPPER on the SM-blogs when trained with full batch gradients, 50 topics, and 15 personas—which was found to perform best during compared to other hyperparameter settings. Figure 8 shows the top three weighted topics for selected personas, highlighting how DAPPER discovers groups of authors whose writing blends unique topics over time. For instance, authors in Persona 12 tend to talk predominantly about the law and police in additional to reports and public records. Like a number of other personas, Persona 12 often references social media, which is associated with discussing something the author discovered through social media or the author encouraging readers to share and comment on their blog.

3.9 CONCLUSION

The Dynamic Author-Persona topic model is uniquely suited to modeling text data with a temporal structure and written by multiple authors. Unlike previous temporal topic models, DAP discovers latent personas—a novel component that identifies authors with similar topics trajectories. Our RVI algorithm further improves the DAP model's performance over competing models and results in the discovery of distinct personas.

Christianity	Tech Gadgets	Sports	TV or Show	Social Media	Wildfire	Book or Movie	Problem
life	apple	game	show	post	area	story	may
god	feature	season	live	share	water	book	change
word	phone	play	night	free	fire	movie	number
heart	device	team	star	comment	north	film	deal
church	user	against	news	video	land	full	result
pope	plus	player	series	photo	west	character	allow
father	update	football	special	click	near	author	note
christian	version	yard	tv	link	south	title	problem
son	iphone	coach	award	facebook	california	director	within
lord	app	ball	fan	twitter	local	writer	step

Law, Police	Today	Event Date	Design	Reports	Event Location	Help	Systems & Security
case	new	_year_	include	report	city	use	service
law	best	september	add	plan	event	need	system
police	today	th	design	issue	center	help	data
court	next	watch	create	member	st	different	technology
claim	open	online	large	public	street	small	customer
charge	month	date	image	accord	art	easy	network
officer	late	october	view	continue	park	save	access
against	hour	august	base	national	friday	important	solution
act	york	episode	space	action	sept	type	security
judge	check	july	form	official	monday	choose	provide

Table 6: Top words associated with the most prevalent topics found by the DAPPER model trained on the full SM-blogs corpus. Topic labels are selected manually in order to aid reference with Figure 8.

DAP model's inference algorithm is further improved by adapting CVI, which eliminates the bottlenecks introduced by difficult non-conjugate terms. The faster version, DAPPER, surpasses its predecessor in terms of speed (35x faster), memory (constant requirements for mini-batch training), and performance in likelihood calculations on holdout data. DAPPER scales to massive datasets on commodity hardware, which in turn allows for deeper insights into topics, and common narratives hidden in the data. Additionally, we show that Regularized Variational Inference, which is applied to the DAPPER model to encourage distinct personas, integrates with CVI cleanly because CVI preserves closed form updates. The success of DAPPER demonstrates that CVI can be applied to complex, temporal graphical models—eliminating the need to run multiple optimization procedures on each document, and instead replace all parameter updates with fast, closed form updates and stochastic mini-batch training.

In evaluating the proposed topic models, we introduce the CaringBridge corpus: a massive collection of journals written by patients and caregivers, many of who face serious, life-threatening illnesses. From this corpus our proposed topic models extract compelling descriptions of health journeys.

While the work presented here demonstrates the DAPPER topic model's readiness for industrial-sized problems, there exist opportunities for further research. For one, our results show that too noisy of updates resulting from small mini-batches lead to poor performance. However, as briefly mentioned in the results, simple attempts to reduce variance through gradient averaging did not yield performance improvements. Further research is needed to find gradient updates that improve model performance in early iterations (as with small to medium batch sizes), but converge to better PWLLs (as with the larger batch sizes).

Part III

PROBABILISTIC DEEP LEARNING

Normalizing flows can simulate novel and plausible data (e.g. new images, texts, or speech) and compute likelihoods exactly, making them one of the most intriguing directions in probabilistic deep learning. Here, we present works for (1) building more flexible normalizing flows, (2) changing the dimensionality of normalizing flows in order to learn compressed latent representations of data, and (3) extending normalizing flows for probabilistic super-resolution to learn high-dimensional distributions conditioned on low-dimensional samples.

4

GRADIENT BOOSTED NORMALIZING FLOWS

4.1 INTRODUCTION

Deep generative models seek rich latent representations of data, and provide a mechanism for sampling new data. Beyond their wide-ranging applications, generative models are an attractive class of models that place strong assumptions on the data and hence exhibit higher asymptotic bias when the model is incorrect (Banerjee, 2007). A popular approach to generative modeling is with variational autoencoders (VAEs) (Kingma and Welling, 2014). A major challenge in VAEs, however, is that they assume a factorial posterior, which is widely known to limit their flexibility (Casale et al., 2018; Chen et al., 2017c; Huang et al., 2018a; Kingma et al., 2016; Miller, Foti, and Adams, 2017; Rezende and Mohamed, 2015; Tomczak and Welling, 2018; van den Berg et al., 2018). Further, VAEs do not offer exact density estimation, which is a requirement in many settings.

Normalizing flows (NF) are an important recent development and can be used in both density estimation (Dinh, Krueger, and Bengio, 2015; Rippel and Adams, 2013; Tabak and Turner, 2013) and variational inference (Rezende and Mohamed, 2015). Normalizing flows are smooth, invertible transformations with tractable Jacobians, which can map a complex data distribution to simple distribution, such as a standard normal (Papamakarios et al., 2019). In the context of variational inference, a normalizing flow transforms a simple, known base distribution into a more faithful representation of the true posterior. As such, NFs complement VAEs, providing a method to overcome the limitations of a factorial posterior. Flow-based models are also an attractive approach for density estimation (De Cao, Titov, and Aziz, 2019; Dinh, Krueger, and Bengio, 2015; Dinh, Sohl-Dickstein, and Bengio, 2017; Grathwohl et al., 2018; Ho et al., 2019; Huang et al., 2018b; Kingma and Dhariwal, 2018; Papamakarios, Pavlakou, and Murray, 2017; Papamakarios et al., 2019; Salman et al., 2018; Tabak and Turner, 2013) because they provide exact density computation and sampling with only a single neural network pass (in some instances) (Durkan et al., 2019).

Recent developments in NFs have focused of creating deeper, more complex transformations in order to increase the flexibility of the learned distribution (Behrmann et al., 2019; Chen et al., 2020; Chen et al., 2019; Ho et al., 2019; Huang et al., 2018b; Kingma and Dhariwal, 2018; Ma et al., 2019). With greater model complexity comes

a greater risk of overfitting while slowing down training, prediction, and sampling. Boosting (Freund and Schapire, 1997; Friedman, 2001, 2002; Friedman, Hastie, and Tibshirani, 2000; Mason et al., 1999) is flexible, robust to overfitting, and generally one the most effective learning algorithms in machine learning (Hastie, Tibshirani, and Friedman, 2001). While boosting is typically associated with regression and classification, it is also applicable in the unsupervised setting (Campbell and Li, 2019; Grover and Ermon, 2018; Guo et al., 2016; Locatello et al., 2017; Miller, Foti, and Adams, 2017; Rosset and Segal, 2002).

OUR CONTRIBUTIONS. In this work we propose a *wider*, as opposed to strictly deeper, approach for increasing the expressiveness of density estimators and posterior approximations. Our approach, *gradient boosted normalizing flows* (GBNF), iteratively adds new NF components to a model based on gradient boosting, where each new NF component is fit to the residuals of the previously trained components. A weight is learned for each component of the GBNF model, resulting in a mixture structure. However, unlike a mixture model, GBNF offers the optimality advantages associated with boosting (Bartlett, Jordan, and McAuliffe, 2006), and a simplified training objective that focuses solely on optimizing a single new component at each step. GBNF compliments existing flow-based models, improving performance at the cost of additional training cycles—not more complex transformations. Prediction and sampling are not slowed with GBNF, as each component is independent and operates in parallel.

While gradient boosting is straight-forward to apply in the density estimation setting, our analysis highlights the need for *analytically* invertible flows in order to efficiently boost flow-based models for variational inference. Further, we address the "decoder shock" phenomenon—a challenge unique to VAEs with GBNF approximate posteriors, where the loss increases suddenly coinciding with the introduction of a new component. Our results show that GBNF improves performance on density estimation tasks, capable of modeling multi-modal data. Lastly, we augment the VAE with a GBNF variational posterior, and present image modeling results on par with state-of-the-art NFs.

The remainder of the paper is organized as follows. In Section 4.2 we briefly review normalizing flows. In Section 4.3 we introduce GBNF for density estimation, and Section 4.5 we extend our idea for the approximate inference setting. In Section 4.6 we discuss normalizing flows that are compatible with GBNF, and the "decoder shock" phenomenon. In Section 4.7 we present results. Finally, we conclude the paper in Section 4.8.

4.2 KEY CONCEPTS AND PRELIMINARIES

Tabak and Turner (2013) and Tabak and Vanden-Eijnden (2010) introduce normalizing flows (NF) as a composition of simple maps. Parameterizing flows with deep neural

networks (Dinh, Krueger, and Bengio, 2015; Dinh, Sohl-Dickstein, and Bengio, 2017; Rippel and Adams, 2013) has popularized the technique for density estimation and variational inference Papamakarios et al., 2019.

VARIATIONAL INFERENCE Rezende and Mohamed (2015) use NFs to modify the VAE's (Kingma and Welling, 2014) posterior approximation q_0 by applying a chain of K transformations $\mathbf{z}_K = f_K \circ \cdots \circ f_1(\mathbf{z}_0)$ to the inference network output $\mathbf{z}_0 \sim q_0(\mathbf{z}_0 \mid \mathbf{x})$. By defining $f_k, k = 1, \ldots, K$ as an invertible, smooth mapping, then with the chain rule and inverse function theorem $\mathbf{z}_k = f_k(\mathbf{z}_{k-1})$ has a computable density (Tabak and Turner, 2013; Tabak and Vanden-Eijnden, 2010): $q_k(\mathbf{z}_k) = q_{k-1}(\mathbf{z}_{k-1}) \left|\det \frac{\partial f_k^{-1}}{\partial \mathbf{z}_{k-1}}\right| = q_{k-1}(\mathbf{z}_{k-1}) \left|\det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}}\right|^{-1}$. The VAE maximizes a lower bound on the log-likelihood of the data: the evidence lower bound (ELBO) (Blei, Kucukelbir, and McAuliffe, 2017; Jordan et al., 1999b; Wainwright and Jordan, 2007). Thus, a VAE with a K-step flow-based posterior minimizes the negative-ELBO:

$$\begin{aligned}\mathcal{F}^{(\mathrm{VI})}_{\theta,\phi}(\mathbf{x}) &= \mathbb{E}_{q_K}\left[-\log p_\theta(\mathbf{x}, \mathbf{z}_K) + \log q_K(\mathbf{z}_K \mid \mathbf{x})\right] \\ &= \mathbb{E}_{q_0}\left[-\log p_\theta(\mathbf{x} \mid \mathbf{z}_K) - \sum_{k=1}^{K} \log \left|\det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}}\right|\right] + \mathrm{KL}\left(q_0(\mathbf{z}_0 \mid \mathbf{x}) \,\|\, p(\mathbf{z}_K)\right) ,\end{aligned} \tag{4.1}$$

where $q_0(\mathbf{z}_0 \mid \mathbf{x})$ is a known base distribution (e.g. standard normal) with parameters ϕ.

DENSITY ESTIMATION Given a set of samples $\{\mathbf{x}_i\}_{i=1}^n$ from a target distribution p^*, our goal is to learn a flow-based model $p_\phi(\mathbf{x})$, which corresponds to minimizing the forward KL-divergence: $\mathcal{F}^{(\mathrm{ML})}(\phi) = \mathrm{KL}(p^*(\mathbf{x}) \,\|\, p_\phi(\mathbf{x}))$ (Papamakarios et al., 2019). A NF formulates $p_\phi(\mathbf{x})$ as a transformation $\mathbf{x} = f(\mathbf{z})$ of a base density $p_0(\mathbf{z})$ with $f = f_K \circ \cdots \circ f_1$ as a K-step flow (Dinh, Krueger, and Bengio, 2015; Dinh, Sohl-Dickstein, and Bengio, 2017; Papamakarios, Pavlakou, and Murray, 2017). Thus, to estimate the expectation over p^* we take a Monte Carlo approximation of the forward KL, yielding:

$$\mathcal{F}^{(\mathrm{ML})}(\phi) \approx -\frac{1}{n}\sum_{i=1}^{n}\left[\log p_0\left(f^{-1}(\mathbf{x}_i)\right) + \sum_{k=1}^{K} \log \left|\det \frac{\partial f_k^{-1}}{\partial \mathbf{x}_i}\right|\right] , \tag{4.2}$$

which is equivalent to fitting the model to samples $\{\mathbf{x}_i\}_{i=1}^n$ by maximum likelihood estimation (Papamakarios et al., 2019).

GRADIENT BOOSTING Gradient boosting (Friedman, 2001, 2002; Friedman, Hastie, and Tibshirani, 2000; Mason et al., 1999) considers the minimization of a loss $\mathcal{F}(G)$, where $G(\cdot)$ is a function representing the current model. Consider an additive pertur-

bation around G to $G + \epsilon g$, where g is a function representing a new component. A Taylor expansion as $\epsilon \to 0$:

$$\mathcal{F}(G + \epsilon g) = \mathcal{F}(G) + \epsilon \langle g, \nabla \mathcal{F}(G) \rangle + o(\epsilon^2) \,, \tag{4.3}$$

reveals the functional gradient $\nabla \mathcal{F}(G)$, which is the direction that reduces the loss at the current solution.

Thus, to minimize loss $\mathcal{F}(G)$ at the current model, choose the best function g in a class of functions $\mathcal{G}$ (e.g. regression trees), which corresponds to solving a linear program where $\nabla \mathcal{F}(G)$ defines the weights for every function in $\mathcal{G}$. Underlying Gradient boosting is a connection to conditional gradient descent and the Frank-Wolfe algorithm (Frank and Wolfe, 1956): we first solve a constrained convex minimization problem to choose g, then solve a line-search problem to appropriately weight g relative to the previous components G (Campbell and Li, 2019; Guo et al., 2016).

BOOSTING IN THE UNSUPERVISED SETTING By considering convex combinations of distributions G and g with weight $\rho \in [0, 1]$, boosting can be applied outside of the traditional classification or regression setting (Campbell and Li, 2019; Cranko and Nock, 2019; Grover and Ermon, 2018; Guo et al., 2016; Lebanon and Lafferty, 2002; Locatello et al., 2017; Rosset and Segal, 2002; Tolstikhin et al., 2017). In particular, Rosset and Segal (2002) apply boosting for density estimation—where the model learns a distribution given samples of data. Similarly, boosted generative models (BGM, (Grover and Ermon, 2018)) constructs a density estimator by iteratively combining sum-product networks.

Recently, Guo et al. (2016) and Miller, Foti, and Adams (2017) introduced the idea of *boosting variational inference* (BVI), which improves a variational posterior by iteratively adding simple approximations. The advantage of this approach is that, like MCMC, the practitioner can trade additional processing for higher accuracy. Specifically, by incorporating more components in the mixture the family of variational posteriors that can be specified becomes more flexible.

More formally, BVI is way to find an approximate posterior $q^{(C)}(\mathbf{z}; \lambda)$, which is a mixture of C simpler component distributions:

$$q^{(C)}(\mathbf{z}; \lambda) = \sum_{c=1}^{C} \rho_c \, q_c(\mathbf{z}; \lambda_c) \quad \text{s.t.} \quad p_c \geqslant 0 \quad \text{and} \quad \sum_c \rho_c = 1$$

where the component distributions q_c a combined via a mixture defined by parameters $\lambda = \{\lambda_1, \ldots, \lambda_C\}$ and weights on the mixture $\rho = \{\rho_1, \ldots, \rho_C\}$. Miller, Foti, and Adams (2017) show that their approach works so long as the components are any distributions over $\mathbf{z}$ from which samples can be drawn.

The challenge in implementing variational boosting is that sampling from a mixture is difficult, and hence Monte Carlo gradients of the ELBO via the reparameterization trick are not straight-forward. The solution is to re-write the ELBO as a weighted combination of expectations with respect to the individual mixture components:

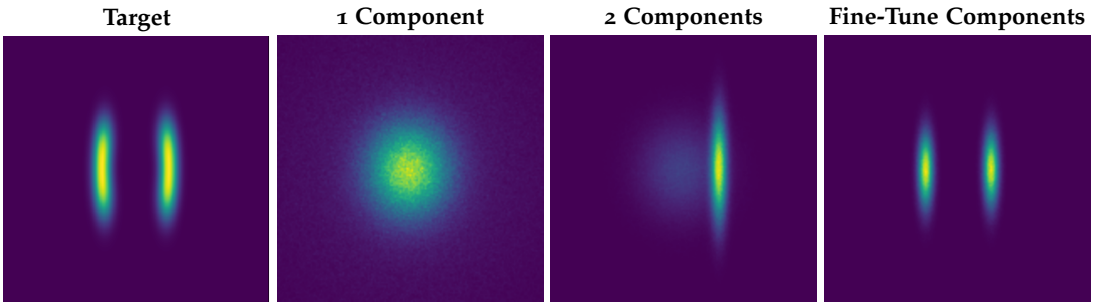


Figure 9: Example of GBNF behavior: A simple affine flow (one scale and shift operation) is not flexible enough to model the target distribution and leads to mode-covering behavior as shown in the **1 Component** figure. In the **2 Components** figure, GBNF introduces a second component, which seeks a region of high probability that is not well modeled by the first component (mass is fit to the right ellipsoid). For this toy problem, fine-tuning the components with additional *boosted* training leads to an even better solution—adjusting the first component to fit the left ellipsoid, and re-weighing the first component appropriately as shown in the **Fine-Tune Components** figure.

$$\mathcal{L}(\lambda, \rho) = \sum_{c=1}^{C} \rho_c \mathbb{E}_{q_c} [\log p(\mathbf{z}, \mathbf{x}) - \log q(\mathbf{z}; \lambda)]$$

now the reparameterization trick can be applied to each component to obtain gradients of the ELBO. In short, the gradients of the ELBO can be taken in a component by component manner, which results in an optimization scheme where the current mixture's parameters are held fixed and the new component mixture's parameters are optimized.

Boosted generative models (BGM, (Grover and Ermon, 2018)) constructs a density estimator by iteratively combining sum-product networks. Unlike BVI and BGM our approach addresses the unique algorithmic challenges of boosting applied to flow-based models—such as the need for analytically invertible flows and the "decoder shock" phenomena when enhancing the VAE's approximate posterior with GBNF.

4.3 DENSITY ESTIMATION WITH GBNF

Gradient boosted normalizing flows (GBNF) build on recent ideas in boosting for variational inference (Guo et al., 2016; Miller, Foti, and Adams, 2017) and generative models (Grover and Ermon, 2018) in order to increase the flexibility of density estimators and posteriors approximated with NFs. A GBNF is constructed by successively adding new components based on forward-stagewise gradient boosting, where each new component $g_K^{(c)}$ is a K-step normalizing flow that is fit to the functional gradient of the loss from the $(c-1)$ previously trained components $G_K^{(c-1)}$.

Gradient boosting assigns a weight ρ_c to the new component $g_K^{(c)}$ and we restrict $\rho_c \in [0, 1]$ to ensure the model stays a valid probability distribution. The resulting density can be written as a mixture model:

$$G_K^{(c)}(\mathbf{x}) = \psi\left((1-\rho_c)\psi^{-1}(G_K^{(c-1)}(\mathbf{x})) + \rho_c\psi^{-1}(g_K^{(c)}(\mathbf{x}))\right)/\Gamma_{(c)}\,, \tag{4.4}$$

where the full model $G_K^{(c)}(\mathbf{x})$ is a monotonic function ψ of a convex combination of fixed components $G_K^{(c-1)}$ and new component $g_K^{(c)}$, and $\Gamma_{(c)}$ is the partition function. Two special cases are of interest: first, when $\psi(a) = a$, which corresponds to a standard additive mixture model with $\Gamma_{(c)} = 1$:

$$\log G_K^{(c)}(\mathbf{x}) = \log\left((1-\rho_c)(G_K^{(c-1)}(\mathbf{x})) + \rho_c(g_K^{(c)}(\mathbf{x}))\right)\,.$$

Second, when $\psi(a) = \exp(a)$ with $\psi^{-1}(a) = \log(a)$, which corresponds to a multiplicative mixture model (Cranko and Nock, 2019; Grover and Ermon, 2018):

$$\log G_K^{(c)}(\mathbf{x}) = \left((1-\rho_c)\log(G_K^{(c-1)}(\mathbf{x})) + \rho_c\log(g_K^{(c)}(\mathbf{x}))\right) - \log\Gamma_{(c)}\,.$$

The advantage of GBNF over a standard mixture model with a pre-determined fixed number of components is that additional components can always be added to the model, and the weights ρ_c for non-informative components will degrade to zero. Since each GBNF component is a NF, we evaluate (4.4) recursively, and at each stage the last component is computed by the change of variables formula:

$$g_K^{(c)} = p_0\left(f_c^{-1}(\mathbf{x})\right)\prod_{k=1}^{K}\left|\det\frac{\partial f_{k,c}^{-1}}{\partial \mathbf{x}}\right|\,, \tag{4.5}$$

where $f_c = f_{c,K}\circ\cdots\circ f_{c,1}$ is the K-step flow transformation for component c, and the base density p_0 is shared by all components. In our formulation we consider c components, where c is fixed and finite.

DENSITY ESTIMATION With GBNF density estimation is similar to (4.2): we seek flow parameters $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_c]$ that minimize $\mathrm{KL}\left(p^*(\mathbf{x}) \,\|\, G_K^{(c)}(\mathbf{x})\right)$, which for finite number of samples $\{\mathbf{x}_i\}$ drawn from $p^*(\mathbf{x})$ corresponds to minimizing:

$$\mathcal{F}^{(\mathrm{ML})}(\boldsymbol{\phi}) = -\frac{1}{n}\sum_{i=1}^{n}\left[\log\left\{\psi\left((1-\rho_c)\psi^{-1}(G_K^{(c-1)}(\mathbf{x}_i)) + \rho_c\psi^{-1}(g_K^{(c)}(\mathbf{x}_i))\right)/\Gamma_{(c)}\right\}\right]\,. \tag{4.6}$$

Directly optimizing (4.6) for mixture model $G_K^{(c)}$ is non-trivial. Gradient boosting, however, provides an elegant solution that greatly simplifies the problem. During training, the first component is fit using a traditional objective function—no boosting is applied[1]. At stages $c > 1$, we already have $G_K^{(c-1)}$, consisting of a convex combination of the $(c-1)$ K-step flow models from the previous stages, and we train a new

1 No boosting during the first stage is equivalent to setting $G_K^{(0)}(\mathbf{x})$ to uniform on the domain of $\mathbf{x}$.

component $g_K^{(c)}$ by taking a Frank-Wolfe (Beygelzimer et al., 2015; Campbell and Li, 2019; Frank and Wolfe, 1956; Guo et al., 2016; Jaggi, 2013; Locatello et al., 2017) linear approximation of the loss (4.6). Since jointly optimizing w.r.t. both $g_K^{(c)}$ and ρ_c is a challenging non-convex problem (Guo et al., 2016), we train $g_K^{(c)}$ until convergence, and then use (4.6) as the objective to optimize w.r.t the corresponding weight ρ_c.

4.3.1 *Updates to New Boosting Components*

ADDITIVE BOOSTING We first consider the special case $\psi(a) = a$ and $\Gamma_{(c)} = 1$, corresponding to the additive mixture model. Our goal is to derive an update to the new component $g_K^{(c)}$ using functional gradient descent. Thus, we take the gradient of (4.6) w.r.t. fixed parameters $\boldsymbol{\phi}_{1:c-1}$ of $G_K^{(c)}$ at $\rho_c \to 0$, giving:

$$\nabla_{\boldsymbol{\phi}_{1:c-1}} \mathcal{F}^{(\mathrm{ML})}(\boldsymbol{\phi})\Big|_{\rho_c \to 0} = -\frac{1-\rho_c}{(1-\rho_c)G_K^{(c-1)} + \rho_c g_K^{(c)}}\Bigg|_{\rho_c \to 0} = -\frac{1}{G_K^{(c-1)}} \tag{4.7}$$

Since $G_K^{(c-1)}$ is fixed, then maximizing $-\mathcal{F}^{(\mathrm{ML})}(\boldsymbol{\phi})$ is achieved by choosing a new component $g_K^{(c)}$ and weighing by the negative of the gradient from (4.7) over the samples:

$$g_K^{(c)} = \arg\max_{g_K \in \mathcal{G}_K} \frac{1}{n} \sum_{i=1}^{n} \frac{g_K(\mathbf{x}_i)}{G_K^{(c-1)}(\mathbf{x}_i)} , \tag{4.8}$$

where $\mathcal{G}_K$ is the family of K-step flows. Note that (4.8) is a linear program in which $G_K^{(c-1)}(\mathbf{x}_i)$ is a constant, and will hence yield a degenerate point probability distribution where the entire probability mass is placed at the minimum point of $G_K^{(c-1)}$. To avoided the degenerate solution, a standard approach adds an entropy regularization term which is controlled by the hyperparameter λ (Campbell and Li, 2019; Guo et al., 2016):

$$g_K^{(c)} = \arg\max_{g_K \in \mathcal{G}_K} \frac{1}{n} \sum_{i=1}^{n} \frac{g_K(\mathbf{x}_i)}{G_K^{(c-1)}(\mathbf{x}_i)} - \lambda \sum_{i=1}^{n} g_K(\mathbf{x}_i) \log g_K(\mathbf{x}_i) . \tag{4.9}$$

MULTIPLICATIVE BOOSTING In this paper, we instead use $\psi(a) = \exp(a)$ with $\psi^{-1}(a) = \log(a)$, which corresponds to the multiplicative mixture model, and, from the boosting perspective, a multiplicative boosting model Cranko and Nock, 2019; Grover and Ermon, 2018. However, in contrast to the existing literature on multiplicative boosting for probabilistic models, we consider boosting with normalizing flow components. In the multiplicative setting, explicitly maintaining the convex combination between $G_K^{(c-1)}$ and $g_K^{(c)}$ is unnecessary: the partition function $\Gamma_{(c)}$ ensures the

validity of the probabilistic model. Thus, the multiplicative GBNF seeks a new component $g_K^{(c)}$ that minimizes:

$$\mathcal{F}^{(ML)}(\boldsymbol{\phi}) = -\frac{1}{n}\sum_{i=1}^{n}\left[\left(\log(G_K^{(c-1)}(\mathbf{x}_i)) + \rho_c \log(g_K^{(c)}(\mathbf{x}_i))\right) - \log \Gamma_{(c)}\right] . \tag{4.10}$$

The partition function is defined as $\Gamma_{(c)} = \int_x \prod_{j=1}^{c}(g_K^{(j)})^{\rho_j}(\mathbf{x})p_0(\mathbf{x})d\mathbf{x}$ and computing $\Gamma_{(c)}$ for GBNF is straightforward since normalizing flows learn self-normalized distributions—and hence can be computed without resorting to simulated annealing or Markov chains (Grover and Ermon, 2018). Moreover, following standard properties (Cranko and Nock, 2019; Grover and Ermon, 2018), we also inherit a recursive property of the partition function. To see the recursive property, denote the un-normalized GBNF density as $\tilde{G}_K^{(c)}$, where $\tilde{G}_K^{(c)} \propto G_K^{(c)}$ but $\tilde{G}_K^{(c)}$ does not integrate to 1. Then, by definition:

$$\Gamma_{(c)}G_K^{(c)}(\mathbf{x}) = g_K^{(c)}(\mathbf{x})^{\rho_c}\tilde{G}_K^{(c-1)}(\mathbf{x}) = \Gamma_{(c-1)}g_K^{(c)}(\mathbf{x})^{\rho_c}G_K^{(c-1)}(\mathbf{x}) ,$$

then, integrating both sides and using $\int_{\mathbf{x}} G_K^{(c)}(\mathbf{x})d\mathbf{x} = 1$ gives

$$\Gamma_{(c)} = \Gamma_{(c-1)}\int_{\mathbf{x}} g_K^{(c)}(\mathbf{x})^{\rho_c}G_K^{(c-1)}(\mathbf{x})d\mathbf{x} = \Gamma_{(c-1)}\mathbb{E}_{G_K^{(c-1)}}\left[g_K^{(c)}(\mathbf{x})^{\rho_c}\right] .$$

and therefore $\Gamma_{(c)} = \Gamma_{(c-1)}\mathbb{E}_{G_K^{(c-1)}}[g_K^{(c)}(\mathbf{x})^{\rho_c}]$, as desired.

The objective in (4.10) represents the loss under the model $G_K^{(c)}$ which followed from minimizing the forward KL-divergence $KL(p^*\|\mathbf{G}_K^{(c)})$, where $\mathbf{G}_K^{(c)}$ is the normalized approximate distribution and p^* the target distribution. To improve (4.10) with

gradient boosting, consider the difference in losses after introducing a new component $g_K^{(c)}$ to the model:

$$\begin{aligned}
\mathrm{KL}(p^*\|\mathbf{G}_K^{(c-1)}) - \mathrm{KL}(p^*\|\mathbf{G}_K^{(c)}) &= \mathbb{E}_{p^*}\left[\log\frac{p^*(\mathbf{x})}{G_K^{(c-1)}(\mathbf{x})} - \log\frac{p^*(\mathbf{x})}{G_K^{(c)}(\mathbf{x})}\right] \\
&= \mathbb{E}_{p^*}\left[\log\frac{G_K^{(c)}(\mathbf{x})}{G_K^{(c-1)}(\mathbf{x})}\right] \\
&= \mathbb{E}_{p^*}\left[\log\frac{(g_K^{(c)}(\mathbf{x}))^{\rho_c}\tilde{G}_K^{(c-1)}(\mathbf{x})}{\Gamma_{c-1}\mathbb{E}_{G_K^{(c-1)}}[(g_K^{(c)}(\mathbf{x}))^{\rho_c}]} \times \frac{\Gamma_{c-1}}{\tilde{G}_K^{(c-1)}(\mathbf{x})}\right] \\
&= \mathbb{E}_{p^*}\left[\log\frac{(g_K^{(c)}(\mathbf{x}))^{\rho_c}}{\mathbb{E}_{G_K^{(c-1)}}[(g_K^{(c)}(\mathbf{x}))^{\rho_c}]}\right] \\
&= \mathbb{E}_{p^*}\left[\log g_K^{(c)}(\mathbf{x}))^{\rho_c}\right] - \log\mathbb{E}_{G_K^{(c-1)}}\left[g_K^{(c)}(\mathbf{x}))^{\rho_c}\right] \\
&\overset{(a)}{\geqslant} \mathbb{E}_{p^*}\left[\log g_K^{(c)}(\mathbf{x}))^{\rho_c}\right] - \log\left(\mathbb{E}_{G_K^{(c-1)}}[g_K^{(c)}(\mathbf{x})]\right)^{\rho_c} \\
&= \rho_c\left\{\mathbb{E}_{p^*}\left[\log g_K^{(c)}(\mathbf{x})\right] - \log\mathbb{E}_{G_K^{(c-1)}}\left[g_K^{(c)}(\mathbf{x})\right]\right\},
\end{aligned} \tag{4.11}$$

where (a) follows by Jensen's inequality since $\rho_c \in [0,1]$. Note that we want to choose the new component $g_K^{(c)}(\mathbf{x})$ so that $\mathrm{KL}(p^*\|\mathbf{G}_K^{(c)})$ is minimized, or equivalently, for a fixed $\mathbf{G}_K^{(c-1)}$, the difference $\mathrm{KL}(p^*\|\mathbf{G}_K^{(c-1)}) - \mathrm{KL}(p^*\|\mathbf{G}_K^{(c)})$ is maximized. Since $\rho_c \geqslant 0$, it suffices to focus on the following maximization problem:

$$g_K^{(c)} = \arg\max_{g_K \in \mathcal{G}_K} \mathbb{E}_{p^*}[\log g_K(\mathbf{x})] - \log\mathbb{E}_{G_K^{(c-1)}}[g_K(\mathbf{x})] \,. \tag{4.12}$$

If we choose a new component according to:

$$g_K^{(c)}(\mathbf{x}) = \frac{p^*(\mathbf{x})}{G_K^{(c-1)}(\mathbf{x})},$$

then, with this choice of $g_K^{(c)}$, we see that (4.12) reduces to:

$$\mathbb{E}_{p^*}\left[\log\frac{p^*(\mathbf{x})}{G_K^{(c-1)}(\mathbf{x})}\right] - \log\mathbb{E}_{G_K^{(c-1)}}\left[\frac{p^*(\mathbf{x})}{G_K^{(c-1)}(\mathbf{x})}\right] = \mathrm{KL}\left(p^*\|\mathbf{G}_K^{(c-1)}\right) - \underbrace{\log\mathbb{E}_{p^*(\mathbf{x})}[1]}_{0}.$$

Our choice of $g_K^{(c)}$ is, therefore, optimal and gives a lower bound to (4.11) with $\mathrm{KL}(p^*\|\mathbf{G}_K^{(c)}) \to 0$. The solution to (4.12) can also be understood in terms of the change-of-measure inequality (Banerjee, 2006; Donsker and Varadhan, 1976), which also forms the basis of the PAC-Bayes bound and a certain regret bounds (Banerjee, 2006).

CONNECTION TO ADDITIVE MODEL Similar to the additive case, the new component chosen in (4.12) shows that $g_K^{(c)}$ maximizes the likelihood of the samples while

discounting those that are already explained by $G_K^{(c-1)}$. Unlike the additive GBNF update in (4.9), however, the multiplicative GBNF update is a numerically stable and does not require an entropy regularization term.

SURROGATE LOSSES Further, our analysis reveals the source of a surrogate loss function (Bartlett, Jordan, and McAuliffe, 2006; Nguyen, Wainwright, and Jordan, 2005; Nguyen, Wainwright, and Jordan, 2009) which optimizes the global objective—namely, when written as as a minimization (4.12) is the *weighted* negative log-likelihood of the samples. Surrogate loss functions are common in the boosting framework (Bartlett, Jordan, and McAuliffe, 2006; Bühlmann and Hothorn, 2007; Freund and Schapire, 1996, 1997; Friedman, Hastie, and Tibshirani, 2000; Rosset and Segal, 2002; Schapire et al., 1998; Tolstikhin et al., 2017). Adaboost (Freund and Schapire, 1996, 1997), in particular, solves a re-weighted classification problem where weak learners, in the form of decision trees, optimize surrogate losses like information gain or Gini index. The negative log-likelihood is specifically chosen as a surrogate loss function in other boosted probabilistic and density estimation models which also have f-divergence based global objectives (Grover and Ermon, 2018; Rosset and Segal, 2002), however here we clarify that the surrogate loss follows from (4.11).

IMPLEMENTATION OF TRAINING OBJECTIVE In practice, instead of weighing the loss by to the reciprocal of the fixed components, we follow Grover and Ermon (2018) and train the new component to perform maximum likelihood estimation over a re-weighted data distribution:

$$g_K^{(c)} = \arg\min_{g_K \in \mathcal{G}_K} \mathop{\mathbb{E}}_{\mathcal{D}^{(c-1)}} [-\log g_K] \tag{4.13}$$

where $\mathcal{D}^{(c-1)}$ denotes a re-weighted data distribution whose samples are drawn with replacement using sample weights inversely-proportional to $G_K^{(c-1)}$. Since (4.13) is the same objective as the generative multiplicative boosting model in Grover and Ermon (2018), where we have set their parameter β from Proposition 1 to one, then (4.13) provides a non-increasing update to the multiplicative objective function (4.10).

CONVERGENCE GUARANTEES Lastly, we note that the analysis of Cranko and Nock (2019) highlights important properties of the broader class of boosted density estimation models that optimize (4.6), of which both the additive and multiplicative forms of GBNF are members. Specifically, Remark 3 in Cranko and Nock (2019) shows a sharper decrease in the loss—that is, for any step size $\rho \in [0, 1]$ the loss has geometric convergence:

$$\mathrm{KL}\left(p^* \,\|\, \mathbf{G}_K^{(c)}|\rho_c\right) \leqslant (1-\rho_c)\mathrm{KL}\left(p^* \,\|\, \mathbf{G}_K^{(c-1)}\right) \tag{4.14}$$

where $\mathbf{G}_K^{(c)}|\rho_c$ denotes the explicit dependence of $\mathbf{G}_K^{(c)}(\mathbf{x})$ on ρ_c. Thus GBNF provides a strong convergence guarantee on the global objective.

4.3.2 *Update to Component Weights*

Component weights ρ are updated to satisfy $\rho_c = \arg\min_\rho \mathcal{F}^{(ML)}(\boldsymbol{\phi})$ using line-search. Alternatively, taking the gradient of the loss $\mathcal{F}^{(ML)}_{\boldsymbol{\phi}}(\mathbf{x})$ with respect to ρ_c gives a stochastic gradient descent (SGD) algorithm (see Section 4.5.3, for example).

Updating a component's weight is only needed once after each component converges. We find, however, that results improve by "fine-tuning" each component and their weights with additional training after the initial training pass. During the fine-tuning stage, we sequentially retrain each component $g^{(i)}_K$ for $i = 1, \ldots, c$, during which we treat $G^{(-i)}_K$ as fixed where $-i$ represents the mixture of all other components: $1, \ldots, i-1, i+1, \ldots c$. Figure 9 demonstrates this phenomenon: when a single flow is not flexible enough to model the target, mode-covering behavior arises. Introducing the second component trained with the boosting objective improves results, and consequently the second component's weight is increased. Fine-tuning the first component leads to a better solution and assigns equal weight to the two components.

4.4 NORMALIZING FLOWS FOR VARIATIONAL INFERENCE

One of the most attractive properties of normalizing flows is the ability to translate between distributions while calculating the likelihood exactly. In Section 4.3 we explored the application of normalizing flows to density estimation by translating between a data distribution and the base distribution. In density estimation we maximize a forward KL-divergence objective function of the form $\mathrm{KL}\left(p^*(\mathbf{x}) \,\|\, p_\phi(\mathbf{x})\right)$, where p^* is the data and p_ϕ the model, and can be optimized exactly with maximum likelihood estimation.

We next consider the application of normalizing flows for approximate inference. In approximate inference the goal is to approximate a complex and intractable posterior distribution $p(\mathbf{z} \mid \mathbf{x})$ over some latent variables $\mathbf{z}$ given the observed data $\mathbf{x}$. Because of a normalizing flow's ability to translate between distributions—such as a base distribution and the posterior, they are a powerful framework for creating flexible approximations of the posterior but are easy to compute and sample from.

To adapt normalizing flows for approximate inference we begin with mean-field variational inference (MFVI)—a simple and ubiquitous approach to approximate inference. In MFVI we introduce an approximation to the true posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ that is easy to compute—such as a factorial Gaussian for example. In order to optimize the parameters ϕ to the approximate posterior, we take a probabilistic approach and

re-write the log-likelihood of the data in terms of the data $\mathbf{x}$, the latent variables $\mathbf{z}$, and the model parameters:

$$\begin{aligned}\log p(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}, \mathbf{z}) dz d\theta \\ &= \log \int p_\theta(\mathbf{x}, \mathbf{z}) \frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{q_\phi(\mathbf{z} \mid \mathbf{x})} dz d\theta \\ &= \log \left(\mathbb{E}_q \left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} \right] \right) \\ &\geqslant \mathbb{E}_q[\log p_\theta(\mathbf{x}, \mathbf{z})] - \mathbb{E}_q[\log q_\phi(\mathbf{z} \mid \mathbf{x})] \\ &= \mathcal{L}_{\theta,\phi}(\mathbf{x}) \,, \end{aligned} \tag{4.15}$$

where the final step follows from Jensen's Inequality and gives a computable lower-bound $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ on the true likelihood, referred to as the Evidence Lower Bound (ELBO). While MFVI results in a quick and easy to compute approximate posterior q, constraining q to a Gaussian is limiting and does not always yield a faithful representation of the true posterior. Normalizing flows, however, are particularly adept at transforming a simple base distribution into a more complex distribution, and hence our focus will be on optimizing the lower-bound in (4.15) where q takes the form of a normalizing flow.

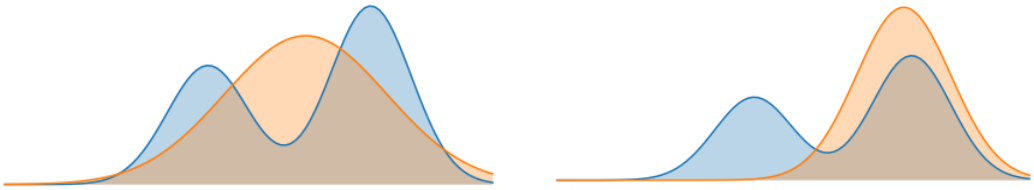
APPROXIMATE INFERENCE OBJECTIVE It can be shown that the *reverse* KL-divergence is equivalent to the ELBO up to a constant. To see this, consider the following:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x})] \\ &= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} \right]\right]}_{\mathcal{L}_{\theta,\phi}(\mathbf{x}) \quad \text{(ELBO)}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \left[\frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{p(\mathbf{z} \mid \mathbf{x})} \right]\right]}_{\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}|\mathbf{x}))} \,, \end{aligned} \tag{4.16}$$

where we have written the log-likelihood of the data in terms of the ELBO (similar to (4.15)) plus a second term representing how *tight* our approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ is to the true posterior $p(\mathbf{z} \mid \mathbf{x})$, which is a reverse KL-divergence. To be clear, in training our model we optimize the objective defined by the ELBO, which is computable. The second term in (4.16) corresponding to the reverse KL-divergence simply establishes that as out approximate posterior approaches the true posterior, then the ELBO will more closely match the true likelihood.

FORWARD VERSUS REVERSE KL-DIVERGENCE OBJECTIVES While in practice we do not optimize the reverse KL-divergence directly, the equivalence with our objective $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ shown in (4.16) helps to explain the behaviour of such models. Specifically, in Section 4.3 we observed the mode covering behavior that is typical for a forward KL-divergence based objectives.

Mode covering is a desirable property in many instances. In particular, mode covering forces the model to explain all the data. In instances where quantifying uncertainty



(a) Forward KL (b) Reverse KL

Figure 10: Forward and reverse KL-divergences result in different model behaviour. The true posterior distribution shown in blue, with the model in orange. The forward KL-divergence causes a mode-covering behavior, in which the model must explain all of the data. The reverse KL-divergence, however, causes mode-seeking, in which the model distribution will fit to the largest mode. Images from (Dibya Ghosh, 2018).

and understanding the tails of distributions is paramount, mode-covering gives a desirable objective function. The drawback with training models with objectives that enforce mode-covering is that the model must be sufficiently flexible to explain all the data. In applications with complex, high-dimensional data—such as image modeling, a lack in model flexibility can lead to poor estimates and result in sampling blurry images.

On the contrary, the reverse KL-divergence based objective results in a mode-seeking behavior. Generative Adversarial Networks (GAN, (Goodfellow, Pouget-Abadie, and Mirza, 2014)) are a popular deep generative model that has been shown to optimize a reverse KL-divergence (Hu et al., 2018). In the case of the GAN, the direction of the KL-divergences helps explain why the model can often generate sharp and realistic looking images, but also offer limited diversity of generated samples. GANs are famously known to be difficult to train and suffer "mode-collapse", resulting in the model assigning a probability of zero to all points except one.

Unlike GANs, however, in the approximate inference setting the reverse KL-divergence characterizes the behavior of the posterior approximation—not the model over the data. Specifically, the approximate posterior seeks to optimize:

$$q_\phi^*(\mathbf{z} \mid \mathbf{x}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}|\mathbf{x})) \,,$$

and from (4.16)

Normalizing flows allow us to progress beyond the constricting limitations of choosing a simple approximate posterior q, and instead construct a flexible approximation to the posterior by mapping a simple base distribution to a faithful representation of the posterior. Here we retain many of the attractive qualities of the normalizing flows: (1) sampling from the posterior is easy—we simply draw a random sample from the known and computable base distribution $\mathbf{z}_0 \sim q_0(\mathbf{z} \mid \mathbf{x})$ and then apply the usual forward flow transformation $f_K \circ \cdots \circ f_1(\mathbf{z}_0) = \mathbf{z}_K$; and (2) unlike other methods for approximate inference (see Section 2.2.1) which can be slow and computationally expensive, variational methods are scalable and amenable to stochastic gradient optimization (Blei, Kucukelbir, and McAuliffe, 2017).

TIGHTER VARIATIONAL BOUNDS Normalizing flows have the attractive property that they produce more faithful approximations of the true posterior (Rezende and Mohamed, 2015). Unsurprisingly, normalizing flows applied to variational inference result in a tighter, modified bound.

It has been shown, however, that tighter bounds are not exclusively a good thing (Alemi et al., 2018). For instance, two models with the same ELBO can have very different qualitative and quantitative properties. Further, Rainforth et al. (2018) show that a techniques for producing tighter ELBOs must include more flexible inference network, without more expressive inference learning will be compromised.

Similarly, Cremer, Morris, and Duvenaud (2017) and Cremer, Li, and Duvenaud (2018) show that even simple normalizing flows are better at reducing inference error than standard VAEs with larger encoder or decoder networks. But, for challenging datasets the majority of the error is not due to the choice in approximation family (referred to as the *approximation gap*). Rather, most model error is associated with how well q is fit to the optimal q^* within that approximation family (referred to as the *amortization gap*). The amortization gap derives it's name because it represents the difference in error caused by amortizing the variational parameters over the entire training set (i. e. AVI) instead of optimizing parameters for each training example individually. Cremer, Li, and Duvenaud (2018) emphasizes that improvements in generative models on more challenging datasets requires reducing the amortization gap. Thus, efforts to build better generative models cannot focus solely on designing more flexible transformations, but instead must have good generalization properties.

4.5 VARIATIONAL INFERENCE WITH GBNF

We integrate a GBNF approximate posterior with a variational autoencoder (VAE, (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014)), replacing the bottleneck Gaussian posterior at the center of the VAE with a more flexible model.

Similar to the density estimation setting, the GBNF approximate posterior is constructed by successively adding new components based on gradient boosting, where each new component $g_K^{(c)}$ is a K-step normalizing flow that is fit to the functional gradient of the loss from the $(c-1)$ previously trained components $G_K^{(c-1)}$.

Gradient boosting assigns a weight ρ_c to the new component $g_K^{(c)}$ and we restrict the weight $\rho_c \in [0, 1]$ to make sure the model stays a valid probability distribution. The resulting variational posterior is a mixture model of the form:

$$G_K^{(c)}(\mathbf{z} \mid \mathbf{x}) = (1 - \rho_c) G_K^{(c-1)}(\mathbf{z} \mid \mathbf{x}) + \rho_c g_K^{(c)}(\mathbf{z} \mid \mathbf{x}) \,, \tag{4.17}$$

where the new variational posterior $G_K^{(c)}(\mathbf{z} \mid \mathbf{x})$ is a convex combination of fixed components (mixture model) $G_K^{(c-1)}$ and the new component $g_K^{(c)}$.

In our formulation of GBNF we consider C components, where C is finite. In our experiments we consider a predefined number of components C and leave C fixed, however in practice the number of components can be increased as necessary until

a sufficiently powerful approximate posterior is achieved. Likewise, in the context of lifelong learning, the number of components C can be increased over the lifetime of the model in order to adapt the model to shifts in the distribution.

We seek a variational posterior that closely matches the true posterior $p(\mathbf{z} \mid \mathbf{x})$. In other words, we wish to minimize $\mathrm{KL}\left(G_K^{(c)}(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z} \mid \mathbf{x})\right)$. Recall, minimizing KL is equivalent to minimizing the negative ELBO, denoted $\mathcal{F}^{(\mathrm{VI})}_{\phi,\theta}(\mathbf{x})$, up to a constant. Thus, a GBNF model has objective:

$$\mathcal{F}^{(\mathrm{VI})}_{\phi,\theta}(\mathbf{x}) = \mathbb{E}_{G_K^{(c)}}\left[\log G_K^{(c)}(\mathbf{z}_K \mid \mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z}_K)\right] . \tag{4.18}$$

4.5.1 *Computing the GBNF Posterior*

The GBNF model takes the form of a mixture, where each component in the mixture is a normalizing flow. To ensure that our objective is computable for (1) expectations that are w.r.t mixtures, as well as (2) expectations that are w.r.t a flow transformation, we establish the following two properties.

(i) EXPECTATIONS W.R.T. A MIXTURE. Let $G^{(c)}(\mathbf{z} \mid \mathbf{x}) = \sum_{c=1}^{C} \rho_c g^{(c)}(\mathbf{z} \mid \mathbf{x})$ be a gradient boosted normalizing flow and consider any function $h(\mathbf{z})$. Then the expectation:

$$\begin{aligned}
\mathbb{E}_{G^{(c)}}[h(\mathbf{z})] &= \int G^{(c)}(\mathbf{z} \mid \mathbf{x}) \cdot h(\mathbf{z}) d\mathbf{z} \\
&= \int \sum_{c=1}^{C} \rho_c g^{(c)}(\mathbf{z} \mid \mathbf{x}) \cdot h(\mathbf{z}) d\mathbf{z} \\
&\overset{(a)}{=} \sum_{c=1}^{C} \rho_c \left[\int g^{(c)}(\mathbf{z} \mid \mathbf{x}) \cdot h(\mathbf{z})\right] \\
&= \sum_{c=1}^{C} \rho_c \mathbb{E}_{g^{(c)}}[h(\mathbf{z})] ,
\end{aligned}$$

where (a) holds because $\sum \rho_c$ is a finite convex combination, and reflects integrating over the choice of mixture component for each sample $\mathbf{z}_K \sim g_K^{(c)}$. Thus, the expectation of a function w.r.t. a mixture model $\mathbb{E}_{G^{(c)}}[h(\mathbf{z})]$ is equivalent to a convex combination of expectations w.r.t. each component distribution $\sum_{c=1}^{C} \rho_c \mathbb{E}_{g^{(c)}}[h(\mathbf{z})]$.

(ii) EXPECTATIONS W.R.T. A FLOW TRANSFORMATION. Recall that expectations w.r.t. a flow transformation can be written as w.r.t. a base distribution. Specifically, let $h(\mathbf{z})$ be a function of $\mathbf{z}$, and g_K some approximate posterior component whose density transformation is a flow of length K. Then the transformed sample $\mathbf{z}_K$ is computed by:

$\mathbf{z}_K = f_K \circ \cdots \circ f_1(\mathbf{z}_0)$. Moreover, by the Law of the Unconscious Statistician (LOTUS), the expectation of $h(\mathbf{z})$ w.r.t. g_K is:

$$\mathbb{E}_{g_K}[h(\mathbf{z})] = \mathbb{E}_{q_0}[h(f_K \circ \cdots \circ f_1(\mathbf{z}_0))] \ .$$

In other words, we can write the expectation w.r.t. an unknown density g_K as an expectation w.r.t. the base distribution $q_0(z)$ if given the flow transformations $f_K, \ldots, f_1$.

Since properties (i) and (ii) hold we can then proceed with deriving gradient boosted updates to components for the loss function in (4.18).

4.5.2 *Updates to New Boosting Components*

Given the objective function in (4.18), we proceed with deriving updates for new components. At stage c, we assume $G_K^{(c-1)}$ to be fixed, and the focus is learning $g_K^{(c)}$ and ρ_c based on functional gradient descent (FGD). Consider the functional gradient w.r.t. $G_K^{(c)}$ at $\rho_c \to 0$:

$$\nabla_{G_K^{(c)}} KF_{\phi,\theta}^{(VI)}(\mathbf{x})\big|_{\rho_c \to 0} = -\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{G_K^{(c-1)}(\mathbf{z} \mid \mathbf{x})} \tag{4.19}$$

Since $G_K^{(c-1)}(\mathbf{z} \mid \mathbf{x})$ are the fixed components, then minimizing the loss $\mathcal{F}_{\phi,\theta}^{(VI)}(\mathbf{x})$ can be achieved by choosing a new component $g_K^{(c)}$ that has the maximum inner product with the negative of the gradient. In other words, we choose a $g_K^{(c)}(\mathbf{z} \mid \mathbf{x})$ such that:

$$\begin{aligned} g_K^{(c)}(\mathbf{z} \mid \mathbf{x}) &= \arg\max_{g_K \in \mathcal{G}_K} \sum_{i=1}^{n} \langle g_K(\mathbf{z} \mid \mathbf{x}_i), -\log \nabla_G \mathcal{F}(\mathbf{x}_i) \rangle \\ &= \arg\min_{g_K \in \mathcal{G}_K} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{g_K(\mathbf{z}|\mathbf{x}_i)} [\log \nabla_G \mathcal{F}(\mathbf{x}_i)] \end{aligned}$$

where $\nabla_G \mathcal{F}$ denotes the functional gradient from (4.19), and $\mathcal{G}_K$ denotes the family of K-step normalizing flows.

To avoid letting g_K degenerate to a point mass at the functional gradient's minimum, we add an entropy regularization term controlled by $\lambda > 0$, hence $g_K^{(c)}(\mathbf{z} \mid \mathbf{x})$ is:

$$\arg\min_{g_K \in \mathcal{G}_K} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{g_K(\mathbf{z}|\mathbf{x}_i)} [\log \nabla_G \mathcal{F}(\mathbf{x}_i) + \lambda \log g_K(\mathbf{z} \mid \mathbf{x}_i)] \, . \tag{4.20}$$

In our experiments that augment the VAE with a GBNF-based posterior, we find good results setting $\lambda = 1.0$. In the density estimation experiments, better results are often achieved with λ near 0.5.

Despite the differences in derivation, optimization of GBNF has a similar structure to other flow-based VAEs. Specifically, with the addition of the entropy regularization term, (4.20) can be rearranged to show the new component $g_K^{(c)}$ minimizes:

$$g_K^{(c)} = \arg\min_{g_K \in \mathcal{G}_K} \mathbb{E}_{g_K(\mathbf{z}|\mathbf{x})}\left[-\log \frac{p_\theta(\mathbf{x} \mid \mathbf{z}_K^{(c)})}{G_K^{(c-1)}(\mathbf{z}_K^{(c)} \mid \mathbf{x})}\right] + \mathrm{KL}\left(\lambda g_K(\mathbf{z}_K^{(c)} \mid \mathbf{x}) \,\|\, p(\mathbf{z}_K^{(c)})\right) . \quad (4.21)$$

where $\mathbf{z}_K^{(c)}$ denotes a sample transformed by component c's flow. If $G_K^{(c-1)}$ is constant, then we recover the VAE objective exactly. Hence, similar to the VAE objective from (2.6), a GBNF has a KL-divergence regularization term between the new component and the prior. The key difference for GBNF, however, is that the negative log-likelihood $-\log p_\theta(\mathbf{x} \mid \mathbf{z}_K^{(c)})$ is down-weighted for samples that are already explained by the fixed portion of the model.

Lastly, we note that during a forward pass the model encodes data to produce $\mathbf{z}_0$. To sample from the posterior $\mathbf{z}_K \sim G_K^{(c)}$, however, we transform $\mathbf{z}_0$ according to $\mathbf{z}_K = f_K^{(j)} \circ \cdots \circ f_1^{(j)}(\mathbf{z}_0)$, where $j \sim \mathtt{Categorical}(\rho)$ randomly chooses a component—similar to sampling from a mixture model. Thus, during training we compute a fast stochastic approximation of the likelihood $G_K^{(c)}$. Likewise, prediction and sampling are as fast as the non-boosted setting, and easily parallelizable across components.

4.5.3 *Updates to Component Weights*

It follows that the weights on each component can be updated by taking the gradient of the loss $\mathcal{F}_{\phi,\theta}^{(\mathrm{VI})}(\mathbf{x})$ with respect to ρ_c. At training iterate t we have:

$$\frac{\partial \mathcal{F}_{\phi,\theta}}{\partial \rho_c} = \sum_{i=1}^{n}\left(\mathop{\mathbb{E}}_{g_K^{(c)}(\mathbf{z}|\mathbf{x}_i)}\left[\gamma_{\rho_c}^{(t-1)}(\mathbf{z} \mid \mathbf{x}_i)\right] - \mathop{\mathbb{E}}_{G_K^{(c-1)}(\mathbf{z}|\mathbf{x}_i)}\left[\gamma_{\rho_c}^{(t-1)}(\mathbf{z} \mid \mathbf{x}_i)\right]\right) ,$$

where we've defined:

$$\gamma_{\rho_c}^{(t-1)}(\mathbf{z} \mid \mathbf{x}_i) \triangleq \log\left(\frac{(1-\rho_c^{(t-1)})G_K^{(c-1)}(\mathbf{z} \mid \mathbf{x}_i) + \rho_c^{(t-1)} g_K^{(c)}(\mathbf{z} \mid \mathbf{x}_i)}{p_\theta(\mathbf{x}_i, \mathbf{z})}\right) .$$

To estimate the gradient with Monte Carlo, we draw samples $\mathbf{z}_K^{(c-1)} \sim G_K^{(c-1)}(\mathbf{z} \mid \mathbf{x}_i)$ and $\mathbf{z}_K^{(c)} \sim g_K^{(c)}(\mathbf{z} \mid \mathbf{x}_i)$ and update ρ_c with stochastic gradient descent in Algorithm 1. To ensure a stable convergence we follow Guo et al., 2016 and implement a decaying learning rate.

Updating a component's weight with Algorithm 1 is only needed once after each component converges. We find, however, that results improve by "fine-tuning" components with additional training after the initial training pass. During the fine-tuning stage, for each c we train $g_K^{(c)}$ and treat all other components $G_K^{(-c)}$ as fixed. Figure 9 demonstrates this phenomenon: when a single flow is not flexible enough to model the target distribution mode-covering behavior arises. Introducing the second component

trained with the boosting objective improves results, and consequently the second component's weight is increased. Fine-tuning the first component leads to a better solution and assigns equal weight to the two components. We also witness improvements in VAE's with GBNF variational posteriors after fine-tuning on real datasets (see Figure 11).

Algorithm 1: Updating Mixture Weight ρ_c.

Let: Tolerance $\epsilon > 0$, and Step-size $\delta > 0$
Initialize weight $\rho_c^{(0)} = 1/C$
Set iteration $t = 0$
while $|\rho_c^{(t)} - \rho_c^{(t-1)}| < \epsilon$ **do**
  Draw mini-batch samples $\mathbf{z}_{K,i}^{(c-1)} \sim G_K^{(c-1)}(\mathbf{z} \mid \mathbf{x}_i)$ and $\mathbf{z}_{K,i}^{(c)} \sim g_K^{(c)}(\mathbf{z} \mid \mathbf{x}_i)$ for $i = 1, \ldots, n$
  Compute Monte Carlo estimate of gradient $\nabla_{\rho_c} \mathcal{F}_{\theta,\phi}^{(VI)}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \gamma_{\rho_c}^{(t-1)}(\mathbf{z}_{K,i}^{(c)} \mid \mathbf{x}_i) - \gamma_{\rho_c}^{(t-1)}(\mathbf{z}_{K,i}^{(c-1)} \mid \mathbf{x}_i)$
  $t = t + 1$
  $\rho_c^{(t)} = \rho_c^{(t-1)} - \delta \nabla_{\rho_c}$
  $\rho_c^{(t)} = \text{clip}(\rho_c^{(t)}, [0, 1])$
return $\rho_c^{(t)}$

After $g_K^{(c)}(\mathbf{z}_K \mid \mathbf{x})$ has been estimated, the mixture model still needs to estimate $\rho_c \in [0, 1]$. Similar to the density estimation setting, the weights on each component can be updated by taking the gradient of the loss $\mathcal{F}_{\phi,\theta}^{(VI)}(\mathbf{x})$ with respect to ρ_c. Recall that $G_K^{(c)}(\mathbf{z}_K \mid \mathbf{x})$ can be written as the convex combination:

$$\begin{aligned} G_K^{(c)}(\mathbf{z}_K \mid \mathbf{x}) =& (1 - \rho_c) G_K^{(c-1)}(\mathbf{z}_K \mid \mathbf{x}) + \rho_c g_K^{(c)}(\mathbf{z}_K \mid \mathbf{x}) \\ &= \rho_c \left(g_K^{(c)}(\mathbf{z}_K \mid \mathbf{x}) - G_K^{(c-1)}(\mathbf{z}_K \mid \mathbf{x}) \right) + G_K^{(c-1)}(\mathbf{z}_K \mid \mathbf{x}) \,, \end{aligned}$$

Then, with $\Delta_K^{(c)}(\mathbf{z}_K \mid \mathbf{x}) \triangleq g_K^{(c)}(\mathbf{z}_K \mid \mathbf{x}_i) - G_K^{(c-1)}(\mathbf{z}_K \mid \mathbf{x}_i)$, the objective function $\mathcal{F}_{\theta,\phi}^{(VI)}(\mathbf{x})$ can be written as a function of ρ_c:

$$\begin{aligned} \mathcal{F}_{\theta,\phi}^{(VI)}(\mathbf{x}) =& \sum_{i=1}^{n} \left\langle \rho_c \Delta_K^{(c)}(\mathbf{z}_K \mid \mathbf{x}_i) + G_K^{(c-1)}(\mathbf{z}_K \mid \mathbf{x}_i), -\log p_\theta(\mathbf{x}_i, \mathbf{z}_K) \right\rangle \\ &+ \sum_{i=1}^{n} \left\langle \rho_c \Delta_K^{(c)}(\mathbf{z}_K \mid \mathbf{x}_i) + G_K^{(c-1)}(\mathbf{z}_K \mid \mathbf{x}_i), \log \left(\rho_c \Delta_K^{(c)}(\mathbf{z}_K \mid \mathbf{x}_i) + G_K^{(c-1)}(\mathbf{z}_K \mid \mathbf{x}_i) \right) \right\rangle . \end{aligned} \tag{4.22}$$

The above expression can be used in a black-box line search method or, as we have done, in a stochastic gradient descent algorithm 1. Toward that end, taking gradient of (4.22) w.r.t. ρ_c yields the component weight updates:

$$\frac{\partial \mathcal{F}_{\phi,\theta}^{(VI)}}{\partial \rho_c} = \sum_{i=1}^{n} \left(\mathop{\mathbb{E}}_{g_K^{(c)}(\mathbf{z}|\mathbf{x}_i)} \left[\gamma_{\rho_c}^{(t-1)}(\mathbf{z} \mid \mathbf{x}_i) \right] - \mathop{\mathbb{E}}_{G_K^{(c-1)}(\mathbf{z}|\mathbf{x}_i)} \left[\gamma_{\rho_c}^{(t-1)}(\mathbf{z} \mid \mathbf{x}_i) \right] \right) , \tag{4.23}$$

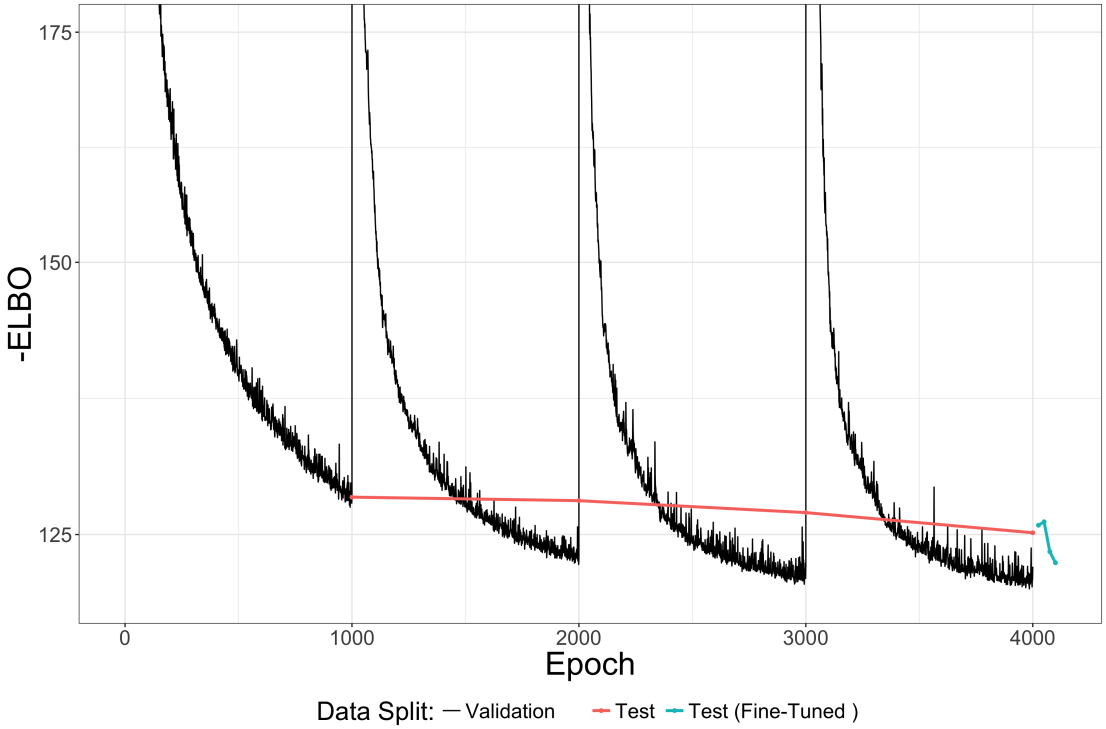


Figure 11: Example of "decoder shock" on Caltech 101 Silhouettes dataset, where new boosting components are introduced every 1000 epochs. While loss on the test set decreases steadily as we add new components, the validation loss jumps dramatically when a new component is introduced due to sudden changes in the distribution of samples passed to the decoder. We also highlight how *fine-tuning* components—by making a second pass with only 25 epochs over each component, improves results at very little computational cost.

where we've defined:

$$\gamma_{\rho_c}^{(t-1)}(\mathbf{z} \mid \mathbf{x}_i) \triangleq \log\left(\frac{(1-\rho_c^{(t-1)})G_K^{(c-1)}(\mathbf{z} \mid \mathbf{x}_i) + \rho_c^{(t-1)} g_K^{(c)}(\mathbf{z} \mid \mathbf{x}_i)}{p_\theta(\mathbf{x}_i, \mathbf{z})}\right).$$

To ensure a stable convergence we follow Guo et al. (2016) and implement an SGD algorithm with a decaying learning rate.

4.5.4 *Decoder Shock*

One challenge in training VAEs using a GBNF variational posterior follows from the sharing of one decoder between all components. During training the decoder naturally acclimates to receiving samples from a particular component (e.g. $g_K^{(old)}$). However, when a new stage begins the decoder begins receiving samples from a different component $g_K^{(new)}$. At this point the loss jumps, a phenomenon we refer to as "decoder shock" (see Figure 11). Reasons for "decoder shock" are as follows.

First, the KL-annealing schedule is reset when $g_K^{(new)}$ begins training. KL-annealing is an important technique for successfully training VAE models Bowman et al., 2016; Sønderby et al., 2016. By reducing the weight of the KL term in (4.21) during the

initial epochs the model is free to discover useful representations of the data before being penalized for complexity. Without KL-annealing, models may choose the "low hanging fruit" and rely purely on a powerful decoder Bowman et al., 2016; Chen et al., 2017c; Cremer, Li, and Duvenaud, 2018; Rainforth et al., 2018; Sønderby et al., 2016. Thus, by resetting the annealing schedule the KL term in the loss increases.

Second, and more importantly, when $g_K^{(new)}$ is introduced a sudden shift occurs in the distribution of samples passed to the decoder. Moreover, because the KL-annealing schedule had reset, $g_K^{(new)}$ is free to create an approximate posterior that is much less constricted than that of the previous component $g_K^{(old)}$. Consequently, this causes a sharp increase in reconstructions errors.

A spike in loss between boosting stages is unique to GBNF. Unlike other boosted models, with GBNF there is a module (the decoder) that depends on the boosted modules—this does not exist when boosting decision trees for regression or classification (for example). To overcome the "decoder shock" problem we propose a simple solution that deviates from a traditional boosting approach. Instead of only drawing samples from $g_K^{(c)}$ during training, we blend in samples from the fixed components $G_K^{(c-1)}$ too. By occasionally sampling from $G_K^{(c-1)}$ we help the decoder *remember* past components and adjust to changes in the full approximate posterior $G_K^{(c)}$. In our experiments we find good results when annealing the $G_K^{(c-1)}$ sampling rate from 0 to 0.5 over the 1000 epochs $g_K^{(c)}$ is trained. We emphasize that despite occasionally sampling from $G_K^{(c-1)}$, the parameter weights of $G_K^{(c-1)}$ remain fixed—the samples from $G_K^{(c-1)}$ are purely for the decoder's benefit.

4.6 RELATED WORK

Below we highlight connections between GBNF and related work, along with unique aspects of GBNF. First, we discuss the catalog of normalizing flows that are compatible with gradient boosting. We then compare GBNF to other boosted generative models and flows with mixture formulations.

FLOWS COMPATIBLE WITH GRADIENT BOOSTING While all normalizing flows can be boosted for density estimation, boosting for variational inference is only practical with *analytically* invertible flows (see Figure 12). The focus of GBNF for variational inference is on training the new component $g_K^{(c)}$, but in order to draw samples $\mathbf{z}_K^{(c)} \sim g_K^{(c)}$ we sample from the base distribution $\mathbf{z}_0 \sim q_0(\mathbf{z} \mid \mathbf{x})$ and transform $\mathbf{z}_0$ according to:

$$\mathbf{z}_K^{(c)} = f_{c,K} \circ \cdots \circ f_{c,2} \circ f_{c,1}(\mathbf{z}_0) \, .$$

However, updating $g_K^{(c)}$ for variational inference requires computing the likelihood $G_K^{(c-1)}(\mathbf{z}_K^{(c)} \mid \mathbf{x})$. Following Figure 12, to compute $G_K^{(c-1)}$ we seek the point $\tilde{\mathbf{z}}_0$ within the base distribution such that:

$$\mathbf{z}_K^{(c)} = f_K^{(j)} \circ \cdots \circ f_2^{(j)} \circ f_1^{(j)}(\tilde{\mathbf{z}}_0) \, ,$$

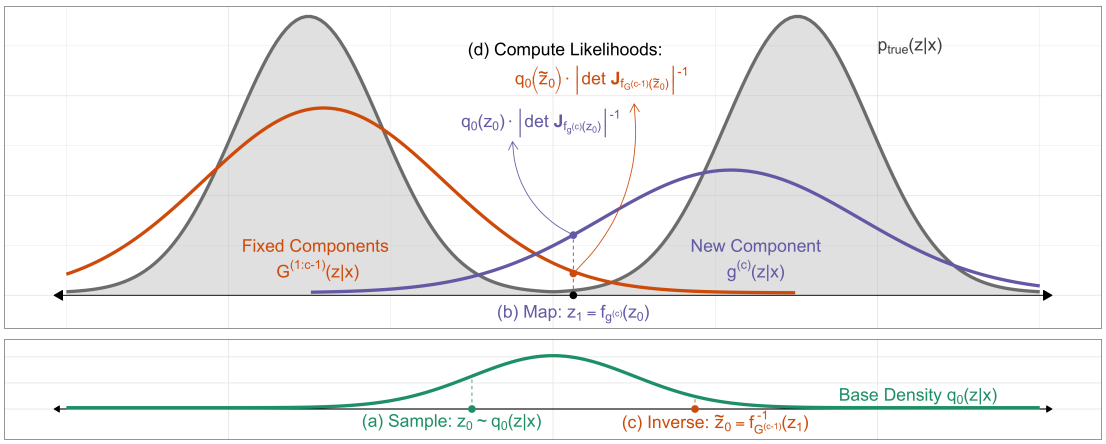


Figure 12: Gradient boosted normalizing flows for variational inference require analytically invertible flows. Similar to a traditional flow-based model: **(a)** samples are drawn from the base density $\mathbf{z}_0 \sim q_0$, and **(b)** transformed by the K-step flow transformation. For GBNF, the sample is transformed by the new component giving $\mathbf{z}_1 = f_{g^{(c)}}(\mathbf{z}_0)$. Gradient boosting fits the new component to the *residuals* of the fixed components, and hence requires computing $G^{(c-1)}(\mathbf{z}_1 \mid \mathbf{x})$. Due to the change of variables formula, $G^{(c-1)}(\mathbf{z}_1 \mid \mathbf{x})$ is computed by **(c)** mapping $\mathbf{z}_1$ back to the base density using the inverse flow transformation $\tilde{\mathbf{z}}_0 = f^{-1}_{G^{(c-1)}}(\mathbf{z}_1)$, and then **(d)** evaluating $q_0(\tilde{\mathbf{z}}_0) \cdot |\det \mathbf{J}_{f_{G^{(c-1)}}}|^{-1}$.

where $\mathbf{z}_K^{(c)}$ is sampled from $g^{(c)}$ and $j \sim \texttt{Categorical}(\rho_{1:c-1})$ randomly chooses one of the fixed components. Then, under the change of variables formula, we approximate $G_K^{(c-1)}(\mathbf{z}_K^{(c)} \mid \mathbf{x})$ by:

$$q_0(\tilde{\mathbf{z}}_0) \prod_{k=1}^{K} \left| \det \frac{\partial f_k^{(j)}}{\partial \tilde{\mathbf{z}}_{k-1}} \right|^{-1}.$$

While planar and radial (Rezende and Mohamed, 2015), Sylvester (van den Berg et al., 2018), and neural autoregressive flows (De Cao, Titov, and Aziz, 2019; Huang et al., 2018b) are provably invertible, we cannot compute the inverse. Inverse and masked autoregressive flows (Kingma et al., 2016; Papamakarios, Pavlakou, and Murray, 2017) are invertible, but D times slower to invert where D is the dimensionality of $\mathbf{z}$.

Analytically invertible flows include those based on coupling layers, such as NICE (Dinh, Krueger, and Bengio, 2015), RealNVP (Dinh, Sohl-Dickstein, and Bengio, 2017), and Glow—which replaced RealNVP's permutation operation with a 1×1 convolution (Kingma and Dhariwal, 2018). Neural spline flows increase the flexibility of both coupling and autoregressive transforms using monotonic rational-quadratic splines (Durkan et al., 2019), and non-linear squared flows (Ziegler and Rush, 2019) are highly multi-modal and can be inverted for boosting. Continuous-time flows (Chen et al., 2017a; Chen et al., 2018; Grathwohl et al., 2018; Salman et al., 2018) use transformations described by ordinary differential equations, with FFJORD being "one-pass" invertible by solving an ODE.

FLOWS WITH MIXTURE FORMULATIONS The main bottleneck in creating more expressive flows lies in the base distribution and the class of transformation function

(Papamakarios et al., 2019). Autoregressive (De Cao, Titov, and Aziz, 2019; Huang et al., 2018b; Jaini, Selby, and Yu, 2019; Kingma et al., 2016; Ma et al., 2019; Papamakarios, Pavlakou, and Murray, 2017), residual (Behrmann et al., 2019; Chen et al., 2019; Rezende and Mohamed, 2015; van den Berg et al., 2018), and coupling-layer flows (Dinh, Krueger, and Bengio, 2015; Dinh, Sohl-Dickstein, and Bengio, 2017; Ho et al., 2019; Kingma and Dhariwal, 2018; Prenger, Valle, and Catanzaro, 2018) are the most common classes of finite transformations, however, discrete (RAD, Dinh et al. (2019)) and continuous (CIF, Cornish et al. (2020)) mixture formulations offer a promising new approach where the base distribution and transformation change according to the mixture component. GBNF also presents a mixture formulation, but trained in a different way, where only the updates to the newest component are needed during training and extending an existing model with additional components is trivial. Moreover, GBNF optimizes a different objective that fits new components to the residuals of previously trained components, which can refine the *mode covering* behavior of VAEs (see Hu et al. (2018)) and maximum likelihood (similar to Dinh et al. (2019)). The continuous mixture approach of CIF, however, cannot be used in the variational inference setting to augment the VAE's approximate posterior (Cornish et al., 2020).

GRADIENT BOOSTED GENERATIVE MODELS By considering convex combinations of distributions G and g, boosting is applicable beyond the traditional supervised setting (Campbell and Li, 2019; Cranko and Nock, 2019; Grover and Ermon, 2018; Guo et al., 2016; Lebanon and Lafferty, 2002; Locatello et al., 2017; Rosset and Segal, 2002; Tolstikhin et al., 2017). In particular, boosting variational inference (BVI, (Cranko and Nock, 2019; Guo et al., 2016; Miller, Foti, and Adams, 2017)) improves a variational posterior, and boosted generative models (BGM, Grover and Ermon (2018)) constructs a density estimator by iteratively combining sum-product networks. Unlike BVI and BGM our approach addresses the unique algorithmic challenges of boosting applied to flow-based models—such as the need for analytically invertible flows and the "decoder shock" phenomena when enhancing the VAE's approximate posterior with GBNF.

4.7 EXPERIMENTS

To evaluate GBNF, we highlight results on two toy problems, density estimation on real data, and boosted flows within a VAE for generative modeling of images. We boost coupling flows Dinh, Sohl-Dickstein, and Bengio, 2017; Kingma and Dhariwal, 2018 parameterized by feed-forward networks with TanH activations and a single hidden layer. While RealNVP (Dinh, Sohl-Dickstein, and Bengio, 2017), in particular, is less flexible and shown to be empirically inferior to planar flows in variational inference (Rezende and Mohamed, 2015), coupling flows are attractive for boosting: sampling and inference require one forward pass, log-likelihoods are computed ex-

actly, and they are trivially invertible. In the toy experiments flows are trained for 25k iterations using the Adam optimizer (Kingma and Ba, 2015).

4.7.1 *Experimental Setup*

4.7.1.1 *Image Modeling Experiments*

We limit the computational complexity of the experiments by reducing the number of convolutional layers in the encoder and decoder of the VAEs from 14 layers to 6. In Table 8 we compare the performance of our GBNF to other normalizing flow architectures. Planar, radial, and Sylvester normalizing flows each use $K = 16$, with Sylvester's bottleneck set to $M = 32$ orthogonal vectors per orthogonal matrix. IAF is trained with $K = 8$ transformations, each of which is a single hidden layer MADE (Germain et al., 2015) with either $h = 256$ or 512 hidden units. RealNVP uses $K = 8$ transformations with either $h = 256$ or $h = 512$ hidden units in the Tanh feed-forward network. For all models, the dimensionality of the flow is fixed at $d = 64$.

Each baseline model is trained for 1000 epochs, annealing the KL term in the objective function over the first 250 epochs as in Bowman et al. (2016) and Sønderby et al. (2016). The gradient boosted models apply the same training schedule to each component. We optimize using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of $1e-3$ (decay of 0.5x with a patience of 250 steps). To evaluate the negative log-likelihood (NLL) we use importance sampling (as proposed in Rezende, Mohamed, and Wierstra (2014)) with 2000 importance samples. To ensure a fair comparison, the reported ELBO for GBNF models is computed by (4.1)—effectively dropping GBNF's fixed components term and setting the entropy regularization to $\lambda = 1.0$.

MODEL ARCHITECTURES In Section 4.7.5, we compute results on real datasets for the VAE and VAEs with a flow-based approximate posterior. In each model we use convolutional layers, where convolutional layers follow the PyTorch convention (Paszke et al., 2017). The encoder of these networks contains the following layers:

$$\begin{aligned}
&\text{Conv}(\text{in} = 1, \text{out} = 16, k = 5, p = 2, s = 2)\\
&\text{Conv}(\text{in} = 16, \text{out} = 32, k = 5, p = 2, s = 2)\\
&\text{Conv}(\text{in} = 32, \text{out} = 256, k = 7, p = 0, s = 1)
\end{aligned}$$

where k is a kernel size, p is a padding size, and s is a stride size. The final convolutional layer is followed by a fully-connected layer that outputs parameters for the diagonal Gaussian distribution and amortized parameters of the flows (depending on model).

Similarly, the decoder mirrors the encoder using the following transposed convolutions:

$$\begin{aligned}
&\text{ConvT}(\text{in} = 64, \text{out} = 32, \text{k} = 7, \text{p} = 0, \text{s} = 2) \\
&\text{ConvT}(\text{in} = 32, \text{out} = 16, \text{k} = 5, \text{p} = 0, \text{s} = 2) \\
&\text{ConvT}(\text{in} = 16, \text{out} = 16, \text{k} = 5, \text{p} = 1, \text{s} = 1, \text{op} = 1)
\end{aligned}$$

where op is an outer padding. The decoders final layer is passed to standard 2-dimensional convolutional layer to reconstruction the output, whereas the other convolutional layers listed above implement a gated action function:

$$\mathbf{h}_l = (\mathbf{W}_l * \mathbf{h}_{l-1} + \mathbf{b}_l) \odot \sigma(\mathbf{V}_l * \mathbf{h}_{l-1} + \mathbf{c}_l),$$

where $\mathbf{h}_{l-1}$ and $\mathbf{h}_l$ are inputs and outputs of the l-th layer, respectively, $\mathbf{W}_l, \mathbf{V}_l$ are weights of the l-th layer, $\mathbf{b}_l, \mathbf{c}_l$ denote biases, $*$ is the convolution operator, $\sigma(\cdot)$ is the sigmoid activation function, and $\odot$ is an element-wise product.

DATASETS In Section 4.7.5, VAEs are modified with GBNF approximate posteriors to model four datasets: Freyfaces[2], Caltech 101 Silhouettes[3] (Marlin et al., 2010), Omniglot[4] (Lake, Salakhutdinov, and Tenenbaum, 2015), and statically binarized MNIST[5] (Larochelle and Murray, 2011). Details of these datasets are given below.

The Freyfaces dataset contains 1965 gray-scale images of size 28×20 portraying one man's face in a variety of emotional expressions. Following van den Berg et al. (2018), we randomly split the dataset into 1565 training, 200 validation, and 200 test set images.

The Caltech 101 Silhouettes dataset contains 4100 training, 2264 validation, and 2307 test set images. Each image portrays the black and white silhouette of one of 101 objects, and is of size 28×28. As van den Berg et al. (2018) note, there is a large variety of objects relative to the training set size, resulting in a particularly difficult modeling challenge.

The Omniglot dataset contains 23000 training, 1345 validation, and 8070 test set images. Each image portrays one of 1623 hand-written characters from 50 different alphabets, and is of size 28×28. Images in Omniglot are dynamically binarized.

Finally, the MNIST dataset contains 50000 training, 10000 validation, and 10000 test set images. Each 28×28 image is a binary, and portrays a hand-written digit.

2 http://www.cs.nyu.edu/~roweis/data/frey_rawface.mat
3 https://people.cs.umass.edu/~marlin/data/caltech101_silhouettes_28_split1.mat
4 https://github.com/yburda/iwae/tree/master/datasets/OMNIGLOT
5 http://yann.lecun.com/exdb/mnist/

4.7.1.2 *Density Estimation Experiments on Real Data*

We compare our results against Glow (Kingma and Dhariwal, 2018), and RealNVP (Dinh, Sohl-Dickstein, and Bengio, 2017). We train models using a small grid search on the depth of the flows $K \in \{5, 10\}$, the number of hidden units in the coupling layers $H \in \{10d, 20d, 40d\}$, where d is the input dimension of the data-points. We trained using a cosine learning rate schedule with the learning rate determined using the learning rate range test Smith, 2017 for each dataset, and similar to Durkan et al. (2019) we use batch sizes of 512 and up to 400,000 training steps, stopping training early after 50 epochs without improvement. The log-likelihood calculation for GBNF follows (4.4), that is we recursively compute and combine log-likelihoods for each component.

DATASET For the unconditional density estimation experiments we follow Papamakarios, Pavlakou, and Murray (2017) and Uria, Murray, and Larochelle (2013), evaluating on four dataset from the UCI machine learning repository (Dua and Taniskidou, 2017) and patches of natural images from natural images (Martin et al., 2001). From the UCI repository the POWER dataset ($d = 6$, N =2,049,280) contains electric power consumption in a household over a period of four years, GAS ($d = 8$, N =1,052,065) contains logs of chemical sensors exposed to a mixture of gases, HEPMASS ($d = 21$, N =525,123) contains Monte Carlo simulations from high energy physics experiments, MINIBOONE ($d = 43$, N =36,488) contains electron neutrino and muon neutrino examples. Lastly we evaluate on BSDS300, a dataset ($d = 63$, N =1,300,000) of patches of images from the homonym dataset. Each dataset is preprocessed following Papamakarios, Pavlakou, and Murray (2017).

4.7.2 *Toy Density Matching*

In the density matching problem the model generates samples from a simple distribution p_Z (such as a standard Normal) and transforms them into a complex distribution $p_{X|\phi}$. The 2-dimensional target's analytical form p^* is given and parameters are learned by minimizing $KL(p_{X|\phi} \| p^*)$ where $p_{X|\phi}$ is formulated using the change of variables formula.

RESULTS Figure 13 highlights results on the density matching problem. For each of the four energy functions we compare our results to a deep 16-flow RealNVP model. The gradient boosted model is configured with two RealNVP components, each of length $K = 4$. In each case the gradient boosted normalizing flows provide an accurate density estimation with half as many total parameters. When the component flows are flexible enough to model most or all of the target density, components can overlap. However, by training the component weights ρ the model down-weights new components that don't provide additional information.

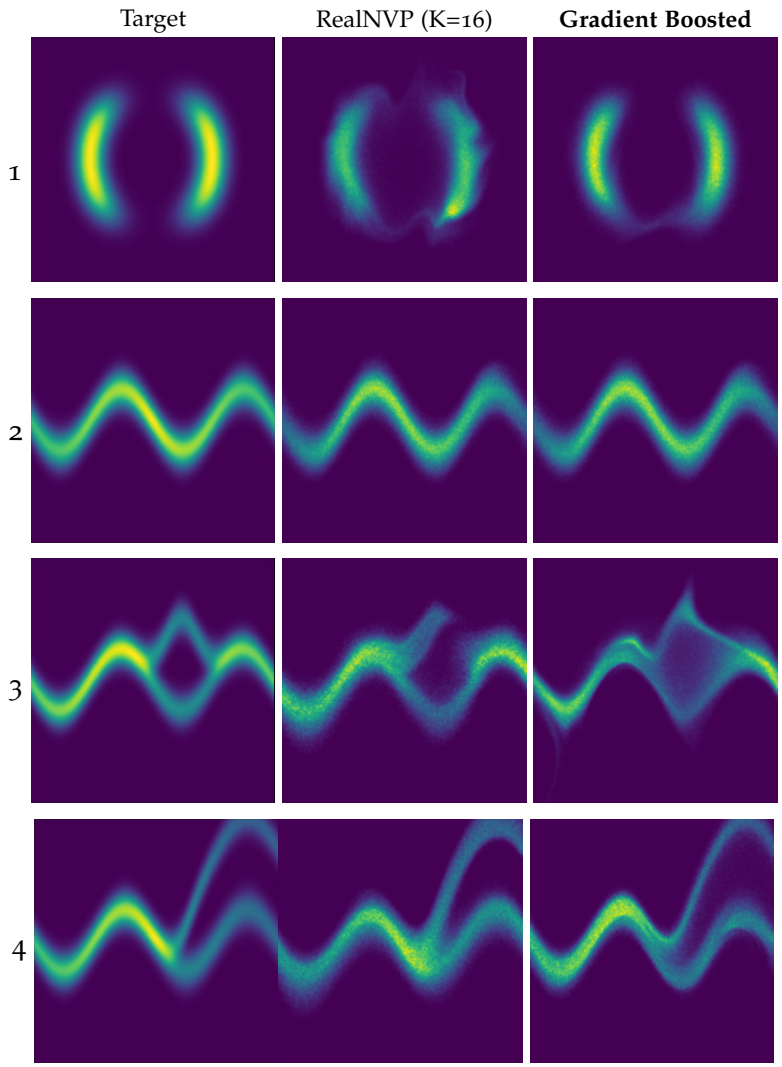


Figure 13: Matching the energy functions from Table 1 of Rezende and Mohamed, 2015. The middle columns show deep RealNVPs with $K = 16$ flows. On the right, we show that GBNF with $C = 2$ components of length $K = 4$ (half as many parameters) can perform as well or better than their deep counterpart.

4.7.3 *Toy Density Estimation*

We apply GBNF to the density estimation problems found in (De Cao, Titov, and Aziz, 2019; Grathwohl et al., 2018; Kingma and Dhariwal, 2018). Here the model receives samples from an unknown 2-dimensional data distribution, and the goal is to learn a density estimator of the data. We consider GBNF with either $c = 4$ or 8 RealNVP components, each of which includes $K = 1, 2, 4$, or 8 coupling layers (Dinh, Sohl-Dickstein, and Bengio, 2017), respectively. Here RealNVP and GBNF use flows of equivalent depth, and we evaluate improvements resulting from GBNF's additional boosted components.

RESULTS As shown in Figure 14, even when individual components are weak the composite model is expressive. For example, the 8-Gaussians figure shows that the

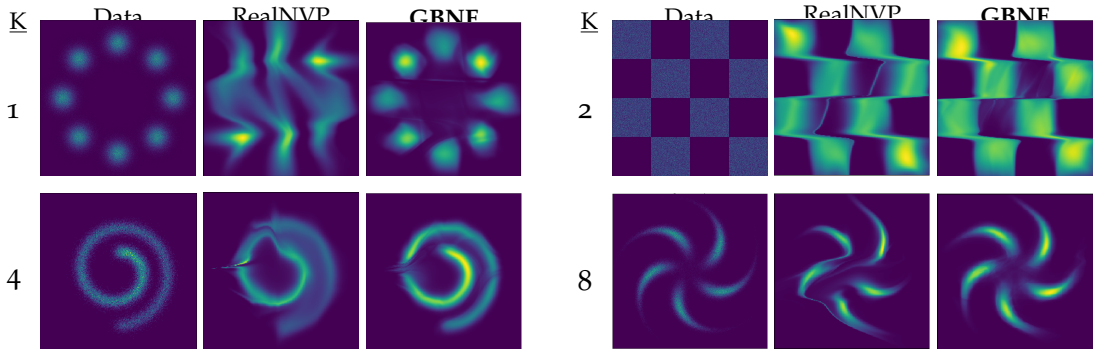


Figure 14: Density estimation for 2D toy data. The **GBNF** columns shows results for a gradient boosted model where each component is a RealNVP flow with $K = 1, 2, 4$ or 8 flow steps, respectively. For comparison the **RealNVP** column shows results for a single RealNVP model, and is equivalent to GBNF's first component. GBNF models train $c = 4$ components, except on the 8-Gaussians data (top left) where results continued to improve up to 8 components. Results show that GBNF produces more accurate density estimates without increasing the complexity of the flow transformations.

first component (RealNVP column) fails to model all modes. With additional 1-step flows, GNBF achieves a multimodal density model. Both the 8-Gaussians and Spiral results show that adding boosted components can drastically improve density estimates without requiring more complex transformations. On the Checkerboard and Pinwheel, where RealNVP matches the data more closely, GBNF sharpens density estimates.

4.7.4 *Density Estimation on Real Data*

Following Grathwohl et al. (2018) we report density estimation results on the POWER, GAS, HEPMASS, and MINIBOONE datasets from the UCI machine learning repository (Dua and Taniskidou, 2017), as well as the BSDS300 dataset (Martin et al., 2001). We compare boosted and non-boosted RealNVP (Dinh, Sohl-Dickstein, and Bengio, 2017) and Glow models (Kingma and Dhariwal, 2018). Glow uses a learned base distribution, whereas our boosted implementation of Glow (and the RealNVPs) use fixed Gaussians. Results for non-boosted models are from (Grathwohl et al., 2018).

RESULTS In Table 7 we find significant improvements by boosting Glow and the more simple RealNVP normalizing flows, even with only $c = 4$ components. Our implementation of Glow was unable to match the results for BSDS300 from (Grathwohl et al., 2018), and only achieves an average log-likelihood of 152.96 without boosting. After boosting Glow with $c = 4$ components, however, the log-likelihood rises significantly to 154.68, which is comparable to the baseline.

4.7.5 *Image Modeling with Variational Autoencoders*

Following Rezende and Mohamed, 2015, we employ NFs for improving VAEs Kingma and Welling, 2014. We compare our model on the same image datasets as those used

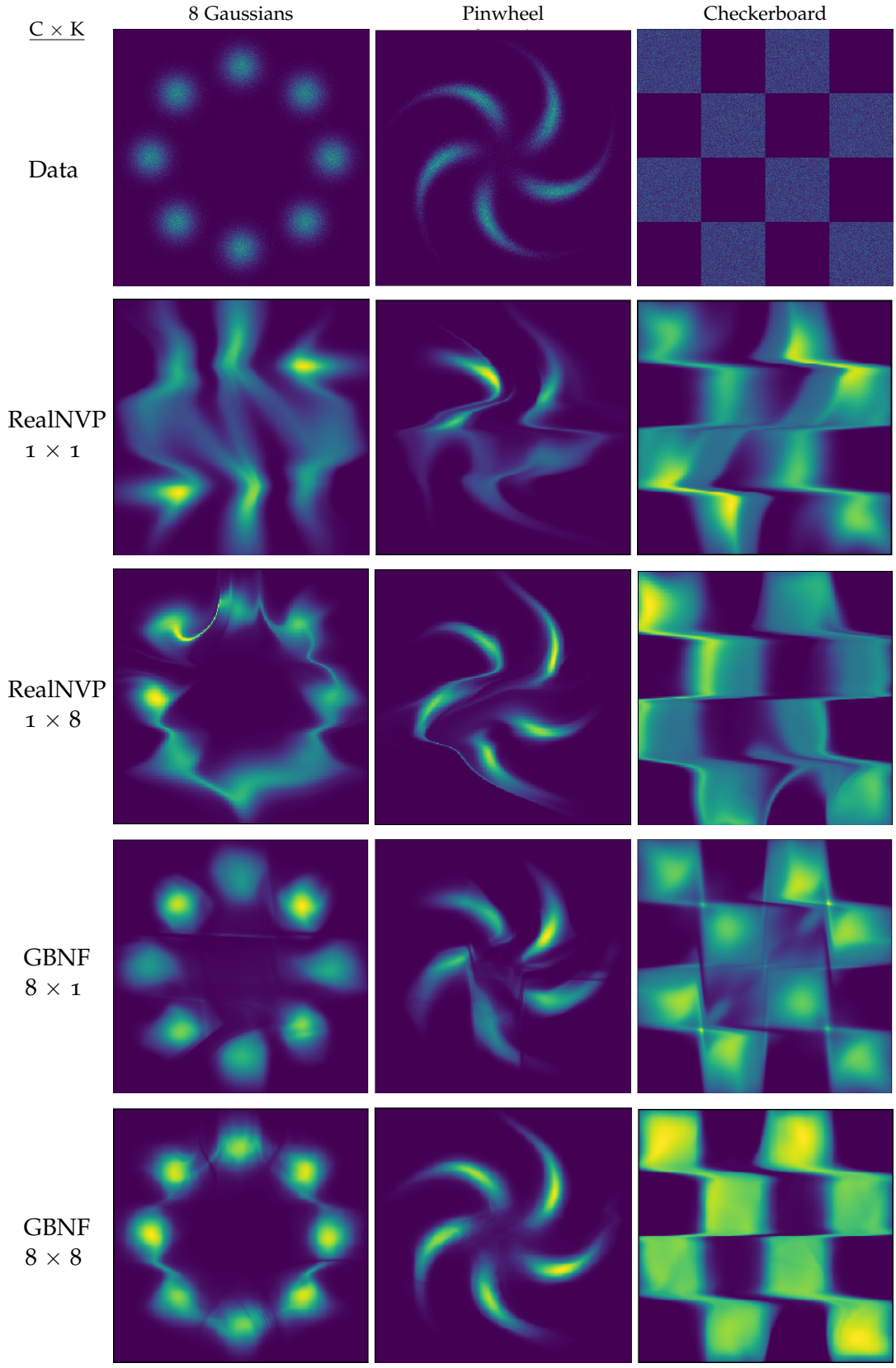


Figure 15: Additional results of density estimation for 2D toy data. For each data distribution we compare the flexibility of a RealNVP with $K = 1$ (second row, 1×1), RealNVP with $K = 8$ (third row, 1×8), as well as shallow-but-wide GBNFs with 8×1 flows (row four) and deep-and-wide GBNFs with 8×8 flows (last row).

Table 7: Log-likelihood on the test set (higher is better) for 4 datasets from UCI machine learning (Dua and Taniskidou, 2017) and BSDS300 (Martin et al., 2001). Here d is the dimensionality of data-points and n the size of the dataset. GBNF models include $c = 4$ components. Mean/stdev are estimated over 3 runs.

Model	POWER↑ d=6;n=2,049,280	GAS↑ d=8;n=1,052,065	HEPMASS↑ d=21;n=525,123	MINIBOONE↑ d=43;n=36,488	BSDS300↑ d=63;n=1,300,000
RealNVP	$0.17_{\pm.01}$	$8.33_{\pm.14}$	$-18.71_{\pm.02}$	$-13.55_{\pm.49}$	$153.28_{\pm1.78}$
Boosted RealNVP	$0.27_{\pm0.01}$	$9.58_{\pm.04}$	$-18.60_{\pm0.06}$	$-10.69_{\pm0.07}$	$154.23_{\pm2.21}$
Glow	$0.17_{\pm.01}$	$8.15_{\pm.40}$	$-18.92_{\pm.08}$	$-11.35_{\pm.07}$	$155.07_{\pm.03}$
Boosted Glow	$0.24_{\pm0.01}$	$9.95_{\pm0.11}$	$-17.81_{\pm0.12}$	$-10.76_{\pm0.02}$	$154.68_{\pm0.34}$

in van den Berg et al., 2018, however we limit the computational complexity of the experiments by reducing the number of convolutional layers in the encoder and decoder of the VAEs from 14 layers to 6. In Table 8 we compare the performance of our gradient boosted normalizing flows to other normalizing flow architectures. Planar, radial, and Sylvester normalizing flows (SNF) each use $K = 16$, with SNF's bottleneck set to $M = 32$ orthogonal vectors per orthogonal matrix. IAF is trained with $K = 8$ transformations, each of which is a single hidden layer MADE Germain et al., 2015 with either $h = 256$ or 512 hidden units. RealNVP uses $K = 8$ transformations with either $h = 256$ or $h = 512$ hidden units in the Tanh feed-forward network. For all models, the dimensionality of the flow is fixed at $d = 64$.

Each baseline model in Table 8 is trained for 1000 epochs, annealing the KL term in the objective function over the first 250 epochs as in Bowman et al., 2016; Sønderby et al., 2016. The gradient boosted models apply the same training schedule to each component. We optimize using the Adam optimizer Kingma and Ba, 2015 with a learning rate of $1e-3$ (decay of 0.5x with a patience of 250 steps). To evaluate the negative log-likelihood (NLL) we use importance sampling (as proposed in Rezende, Mohamed, and Wierstra, 2014) with 2000 importance samples. To ensure a fair comparison, the reported ELBO for GBNF models is computed by (2.9), effectively dropping GBNF's fixed components term and setting the entropy regularization to $\lambda = 1.0$. Since GBNF's variational posterior is a mixture model, we sample components from the mixture and average the ELBO calculation over $3C$ samples drawn from $G_K^{(C)}(\mathbf{z} \mid \mathbf{x})$.

RESULTS In all results RealNVP, which is more ideally suited for parametric density estimation tasks, performs the worst of the flow models. Nonetheless, applying gradient boosting to RealNVP improves the results significantly. On Freyfaces, the smallest dataset consisting of just 1965 images, gradient boosted RealNVP gives the best performance—suggesting that GBNF may help in overfitting. For the larger Omniglot dataset of hand-written characters, Sylvester flows are superior, however, gradient boosting improves the RealNVP baseline considerably and achieves a negative log-likelihood comparable to Sylvester (99.09 versus 98.54). GBNF improves on the baseline RealNVP, however both GBNF and IAF's results are notably higher than tra-

Model	MNIST		Freyfaces		Omniglot		Caltech 101	
	-ELBO↓	NLL↓	-ELBO↓	NLL↓	-ELBO↓	NLL↓	-ELBO↓	NLL↓
VAE	89.32±0.07	84.97±0.01	4.84±0.07	4.78±0.07	109.77±0.06	103.16±0.01	120.98±1.07	108.43±1.81
Planar	86.47±0.09	83.16±0.07	4.64±0.04	4.60±0.04	105.72±0.08	100.18±0.01	116.70±1.70	**104.23** ±1.60
Radial	88.43±0.07	84.32±0.06	4.73±0.08	4.68±0.07	108.74±0.57	102.07±0.50	118.89±1.30	106.88±1.55
Sylvester	84.54±0.01	**81.99**±0.02	4.54±0.03	**4.49**±0.03	101.99±0.23	**98.54**±0.29	112.26±2.01	**100.38**±1.20
IAF	86.46±0.07	83.14±0.06	4.73±0.04	4.70±0.05	106.34±0.14	100.97±0.07	119.62±0.84	108.41±1.31
RealNVP	88.04±0.07	83.36±0.09	4.66±0.17	4.62±0.16	106.22±0.59	100.43±0.19	123.26±2.06	113.00±1.70
GBNF	87.18±0.53	**82.70** ±0.11	4.56±0.07	**4.48**±0.01	105.53±0.21	**99.11** ±0.21	124.15±1.70	107.92±0.81
GBNF+	87.12±0.07	**82.67** ±0.03	4.51±0.03	**4.41** ±0.01	105.60±0.20	**99.09** ±0.17	121.06±0.51	**106.55** ±0.21

Table 8: Negative ELBO (-ELBO, lower is better) and Negative log-likelihood (NLL, lower is better) results on MNIST, Freyfaces, Omniglot, and Caltech 101 Silhouettes datasets. For the Freyfaces dataset the results are reported in bits per dim. Results for the other datasets are reported in nats. For planar, radial and orthogonal Sylvester flows a flow depth of $K = 16$ was used. IAF and RealNVP used $K = 8$ steps with a hidden unit size of 256 on Freyfaces and Caltech 101, and 512 units for MNIST and Omniglot. GBNF models boosted up to $C = 4$ RealNVP components. GBNF+ indicates the GBNF model with an additional pass over each component (50 epochs/component) to "fine-tune" and re-compute the weights ρ. The top 3 NLL results for each dataset are **bolded**.

ditional flows like planar, radial, and Sylvester for the Caltech 101 Silhouettes dataset. Lastly, on MNIST we find the boosting improves the NLL on RealNVP from 83.36 to 82.67, and is on par with Sylvester flows.

In all datasets fine-tuning GBNF components (listed as GBNF+ in Table 8) with an additional 50 epochs per component and re-computing the weights ρ, further improves results. Fine-tuning allows each component in the mixture an opportunity to adjust to the components that were trained after it. As was shown in the toy example show in Figure 9, this adjustment can be crucial to producing a betting fitting approximate posterior. A likely explanation for this phenomenon is that GBNF optimizes a likelihood based objective and hence attempts to explain all of the data (as shown by the mode-covering behavior in 9). Thus, components that over-extended themselves during the initial training pass can focus on producing a tighter approximation on a subset of the posterior during the fine-tuning stage.

4.8 CONCLUSION

In this work we introduce *gradient boosted normalizing flows*, a technique for increasing the flexibility of flow-based models through gradient boosting. GBNF, iteratively adds new NF components, where each new component is fit to the residuals of the previously trained components. We show that GBNF can improve results for existing normalizing flows on density estimation and variational inference tasks. In our experiments we demonstrated that GBNF improves over their baseline single component model, without increasing the depth of the model, and produces image modeling results on par with state-of-the-art flows. Further, we showed GBNF models used for

density estimation create more flexible distributions at the cost of additional training and not more complex transformations.

In the future we wish to further investigate the "decoder shock" phenomenon occurring when GBNF is paired with a VAE. Future work may benefit from exploring other strategies for alleviating "decoder shock", such as multiple decoders or different annealing strategies. In our real data experiments in Section 4.7.5 we fixed the entropy regularization λ at 1.0, however adjusting the regularization on a per-component level may be worth pursuing. Additionally, in our image modeling experiments we used RealNVP as the base component. Future work may consider other flows for boosting, as well as heterogeneous combinations of flows as the different components. We wish to consider GBNF as an approach to lifelong learning since GBNF can easily adapt to shifts in the data distribution.

Lastly, as a first step towards expanding the applications for GBNF we include a formulation for conditional GBNF in Appendix B. Conditional GBNF adapts the formulation for conditional flows presented in Section 6.2 for a gradient boosted meta-learning framework. The applications of conditional GBNF include many conditional density estimation problems, like classification, regression, super-resolution images, image segmentation, and sequence modeling.

4.9 BROADER IMPACTS

As a generative model, gradient boosted normalizing flows (GBNF) are suited for a variety of tasks, including the synthesis of new data-points. A primary motivation for choosing GBNF, in particular, is producing a flexible model that can synthesize new data-points quickly. GBNF's individual components can be less complex and thus faster, yet as a whole the model is powerful. Since the components operate in parallel, prediction and sampling can be done quickly—a valuable characteristic for deployment on mobile devices. One limitation of GBNF is the requirement for additional computing resources to train the added components, which can be costly for deep flow-based models. As such, GBNF advantages research laboratories and businesses with access to scalable computing. Those with limited computing resources may find benchmarking or deploying GBNF too costly.

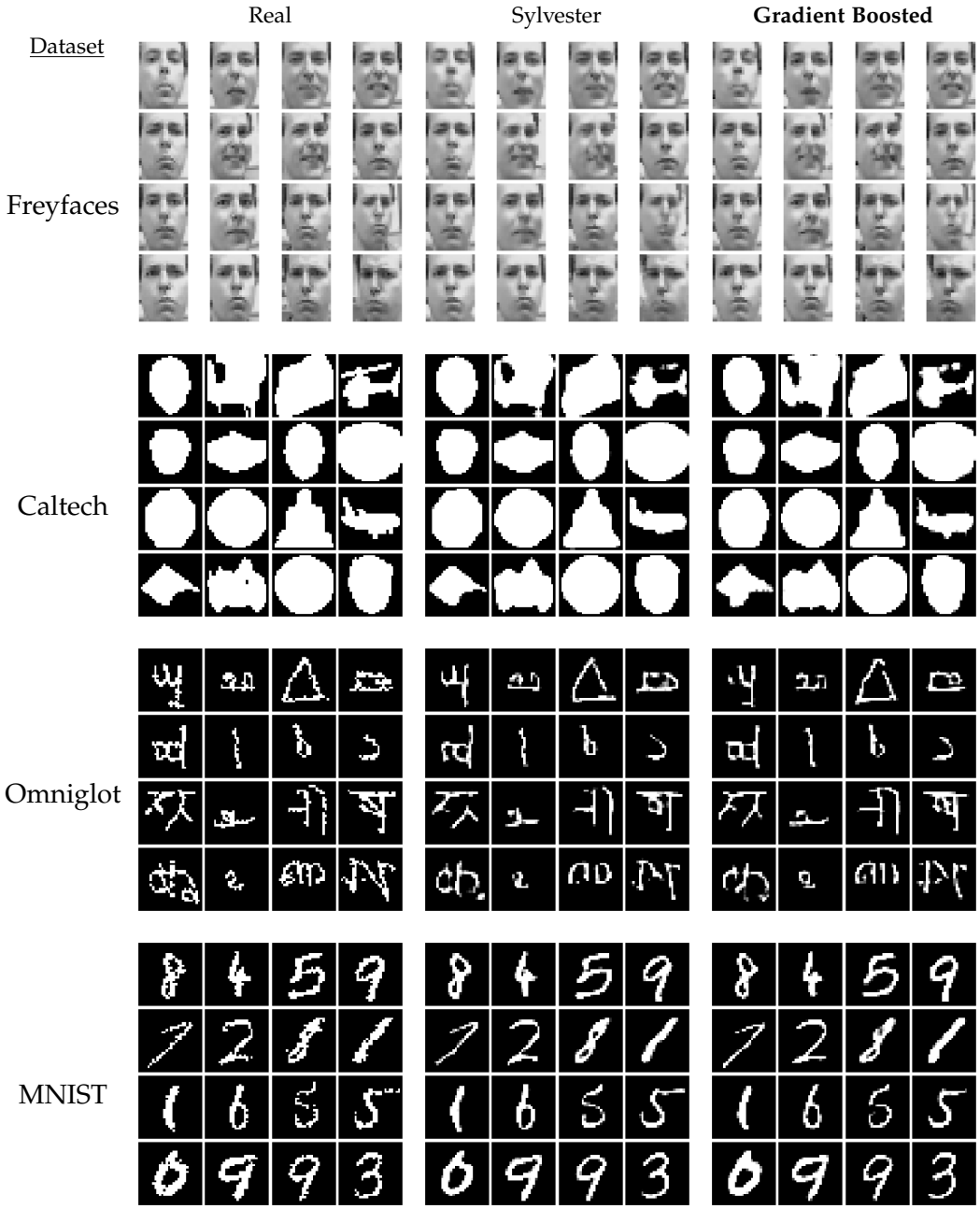


Figure 16: Image reconstructions for the four datasets listed in Table 8.

5

COMPRESSIVE NORMALIZING FLOWS

5.1 INTRODUCTION

Normalizing flows (Dinh, Sohl-Dickstein, and Bengio, 2017; Rezende and Mohamed, 2015; Tabak and Turner, 2013; Tabak and Vanden-Eijnden, 2010) are a powerful class of generative models that express probability distributions through a sequence of diffeomorphic transformations of a simple, easy to compute base distribution. Recent advances in the expressive power of such transformations make normalizing flows an attractive choice for (tractable) density estimation, posterior approximation, and sampling (Papamakarios et al., 2019).

Due to the bijective nature of normalizing flow transformations, the dimension of the base distribution is equal to the data distribution. In high-dimensional settings, such as modeling images or spatio-temporal data, this requirement poses significant architectural, memory, and computational costs. As such, scaling normalizing flows to high-dimensional problems is more challenging than other classes of generative models such as Variational Autoencoders (VAEs, Kingma and Welling, 2014) or Generative Adversarial Networks (Goodfellow, Pouget-Abadie, and Mirza, 2014) where finding lower dimensional representations is straight-forward, and hence flows require specialized solutions (Brehmer and Cranmer, 2020; Cunningham et al., 2020; Nielsen et al., 2020). Further, VAEs that learn latent representations with varying levels of compression are better adept at capturing both high level semantic features as well as low level details (Dhariwal et al., 2020; Razavi, van den Oord, and Vinyals, 2019; Sønderby et al., 2016; Vahdat and Kautz, 2020; van den Oord, Vinyals, and kavukcuoglu, 2017). On the other hand, normalizing flows that use coupling layers can produce quality samples by learning low level features (such as local pixel correlations) but fail to decouple noise from representation well enough for tasks like detecting out-of-distribution samples (Kirichenko, Izmailov, and Wilson, 2020).

For image modeling, where the dimensionality of the data is often large, multi-scale flow architectures are common (Dinh, Sohl-Dickstein, and Bengio, 2017). Multi-scale architectures, however, do not actually learn a compressed representation of the data. Instead, a squeeze operation replaces the spatial dimensions of the image in exchange for more channels, and then some channels may be withheld from deeper steps in the flow. More recently, there has been interest in normalizing flows that relax the bijective

constrains and can change the dimension of the latent space. In this setting we assume there exists a low-dimensional manifold on which the high-dimensional data lie, and our goal is to train a normalizing flow that learns a tractable density estimator on that manifold (Brehmer and Cranmer, 2020; Cunningham et al., 2020; Gemici, Rezende, and Mohamed, 2016; Kumar, Poole, and Murphy, 2020). Alternatively, the surjective flows introduced by Nielsen et al. (2020) include max operations—which creates flow transformations analogous to max-pooling layers, and are capable of directly reducing the dimension of the flow at each step.

In contrast with prior work we introduce compressive normalizing flows that are, in the simplest case, equivalent to probabilistic principal components analysis (PPCA, (Tipping and Bishop, 1999)). The PPCA-based compressive flow relaxes the bijective constraints and learns a compressed latent representation, while offering parameter updates that are available analytically. Drawing on the connection between PPCA and VAEs, we extend our framework to more powerful VAE-based compressive flows that are optimized with amortized variational inference techniques, providing greater flexibility and scalability. Lastly, we consider the architecture of compressive flows: comparing models with bijective flows followed by a single compressive step to models that intermix compressive steps throughout the sequence of flow transformations—leading to varying degrees of compression and drastically fewer parameters in the model. Our experiments show that extending existing flow architectures to find a compressed latent space results in minimal drop in performance for even large compression ratios. Finally, we examine the role of flow depth prior to a compression step, highlighting the importance of combining bijective with compressive steps and suggesting architectural choices that minimize information loss.

5.2 KEY CONCEPTS AND PRELIMINARIES

Normalizing flows (NFs) are an important recent development in density estimation, generative modeling, and variational inference (Dinh, Krueger, and Bengio, 2015; Rezende and Mohamed, 2015; Rippel and Adams, 2013; Tabak and Turner, 2013). Normalizing flows are smooth, invertible transformations with tractable Jacobians, which can map a complex data distribution to a simple easy to evaluate base distribution—such as a standard normal (Papamakarios et al., 2019). More specifically, let g be a bijective transformation between $\mathbf{x}$ and $\mathbf{z}$ where $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$ such that $g^{-1}(\mathbf{x}) = \mathbf{z}$ and $\mathbf{z} \sim \mathcal{N}(0, \mathbb{I})$. Then, by the change of variables formula the log likelihood of $\mathbf{x}$ is:

$$\log p(\mathbf{x}) = \log p_Z(g^{-1}(\mathbf{x})) + \log \left| \det \frac{\partial g^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right| \tag{5.1}$$

where the choice of a smooth invertible function g ensures a tractable log determinant Jacobian.

Tabak and Turner (2013) and Tabak and Vanden-Eijnden (2010) introduce normalizing flows as a composition of simple maps, however parameterizing flows with deep

neural networks (Dinh, Krueger, and Bengio, 2015; Dinh, Sohl-Dickstein, and Bengio, 2017; Rippel and Adams, 2013) has popularized the technique (Papamakarios et al., 2019). Flow-based models provide exact density computation and sampling with only a single neural network pass (in some instances) (Durkan et al., 2019).

Composing a chain of transformations $g = g_K \circ \cdots \circ g_1$ into a K-step normalizing flow allows for more flexible mappings $\mathbf{x} = g(\mathbf{z})$ of the base density $p_Z(\mathbf{z})$ (Dinh, Krueger, and Bengio, 2015; Dinh, Sohl-Dickstein, and Bengio, 2017; Papamakarios, Pavlakou, and Murray, 2017). Given a set of samples $\{\mathbf{x}_i\}_{i=1}^n$ from a target distribution p^*, our goal is to learn a flow-based model $p_\beta(\mathbf{x})$, which corresponds to minimizing the forward KL-divergence: $\mathrm{KL}(p^*(\mathbf{x}) \,\|\, p_\beta(\mathbf{x}))$, where β denotes the flow parameters. Thus, to estimate the expectation over p^* we take a Monte Carlo approximation of the forward KL, yielding:

$$\mathcal{L} \approx \frac{1}{n} \sum_{i=1}^{n} \left[\log p_Z \left(g^{-1}(\mathbf{x}_i) \right) + \sum_{k=1}^{K} \log \left| \det \frac{\partial g_k^{-1}}{\partial \mathbf{x}_i} \right| \right] , \tag{5.2}$$

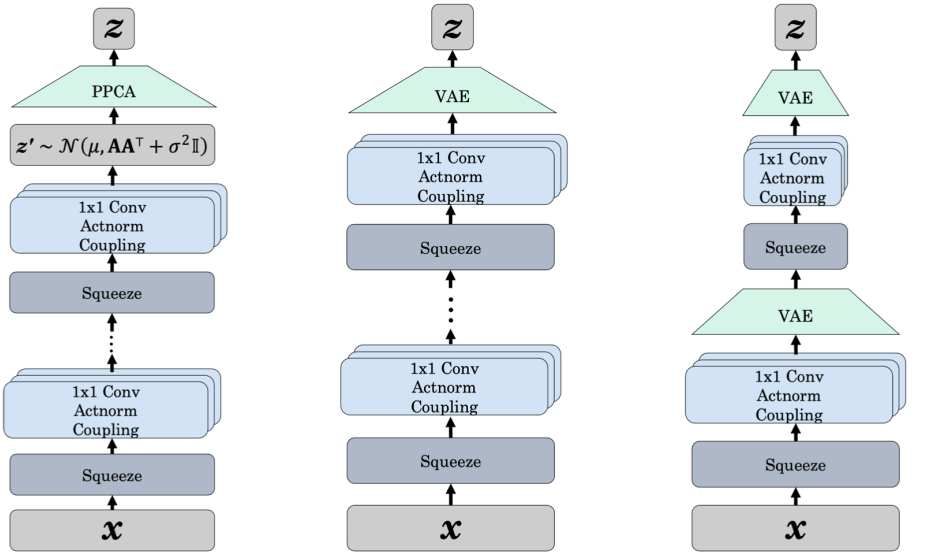
which is equivalent to fitting the model to samples $\{\mathbf{x}_i\}_{i=1}^n$ by maximum likelihood estimation (Papamakarios et al., 2019).

The bijective constraints of normalizing flows require that $\mathbf{x}$ and $\mathbf{z}$ match dimensionality, that is $\mathbf{x}, \mathbf{z} \in \mathbb{R}^M$. We denote bijective flow transformations by $g(\cdot)$. Next we review recent work examining the necessity of the bijective constraints, and introduce flow transformations $f(\cdot)$ which can alter the dimensionality of the latent space.

5.3 RELATED WORK

We are interested in mappings between data $\mathbf{x} \in \mathbb{R}^D$ and latent $\mathbf{z} \in \mathbb{R}^d$ where $D > d$. Kumar, Poole, and Murphy (2020) consider normalizing flows that relax the bijective constraint, and consider *injective* mappings. The change in volume under an injective mapping f is given by $[\det(J_f(\mathbf{z})^\top J_f(\mathbf{z}))]^{1/2} d\mathbf{z}$ (Boothby, 1986), replacing the determinant Jacobian term in (5.1). Optimizing the determinant Jacobian term is computationally challenging for large models as it requires computing a $D \times d$ Jacobian matrix for every data point.

To overcome the challenge of computing the change in volume under an injective mapping, Kumar, Poole, and Murphy (2020) stochastically approximate the Jacobian term to derive a lower bound on the objective and use a regularized autoencoder to perform the forward (encoding) and inverse (generative) mapping. The Relaxed Injective Probability Flow (Kumar, Poole, and Murphy, 2020), however, doesn't have a tractable inverse or likelihood computation. Brehmer and Cranmer (2020) make similar efforts to approximate the difficult Jacobian term for injective flows that learn a manifold. The $\mathcal{M}$-flows of Brehmer and Cranmer (2020) project latents directly to a manifold, thereby learning a low-dimensional manifold using a deterministic treatment of the data.



(a) Dual-Base Compression (b) Single Compression (c) Intermix Compressions

Figure 17: Architectures for compressive flows. Bijective flows are indicated by sequences of squeeze operations followed by coupling blocks (Kingma and Dhariwal, 2018). Squeeze operations (Dinh, Sohl-Dickstein, and Bengio, 2017) are used in multiscale architectures to exchange spatial dimensions for more channels, whereas our compressive PPCA, linear VAE, and Gaussian VAE-based flow steps reduce the overall size of the latent space. In all architectures we learn a low dimensional latent space $\mathbf{z} \sim \mathcal{N}(0, \mathbb{I}_d)$.

Both $\mathcal{M}$-flows and Relaxed Injective Flows model data that lies exactly on a manifold, an assumption that may not always hold in practice. Noisy Injective Flows (NIF) add noise to the model—that is, $\mathbf{x} = \mathbf{x}' + \epsilon$ where $\mathbf{x}' = f(\mathbf{z})$, which makes the mappings non-deterministic and introduces a lower bound to the objective, but allows for modeling data around the manifold (Cunningham et al., 2020). The simplest case of NIF, which includes linear mappings $\mathbf{x} = \mathbf{A}\mathbf{z} + \mu$, overlaps our work where both yield analytical solutions. NIF defines the encoding distribution $q(\mathbf{z} \mid \mathbf{x})$ as a normalized likelihood using Bayes' rule with an improper prior, which may however lead to unwanted behavior—whereas, defining q using Bayes' Rule with transformations mimicking PPCA may be more stable and more easily extendable.

More broadly, explicit and implicit density estimators represent the two major paradigms in generative modeling. Implicit models, like GANs, are a stochastic procedure that directly generates the data Mohamed and Lakshminarayanan, 2017. Our focus, however, is on explicit models like flows and VAEs, which effectively construct a distribution based on observed samples and offer likelihood evaluation as well as sampling.

5.4 CHANGE OF DIMENSION FLOWS

We consider three approaches for changing the dimension of a flow, with progression from simple to more complex: first, by refining the analysis of linear change of dimension transformations with closed-form analytical solutions equivalent to PPCA; sec-

ond, by drawing on the connection between PPCA and linear VAEs, which broadens the model definition in exchange for optimizing a lower bound with amortized variational inference (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014); and third, going from linear to deep VAEs, which can be considered as a stochastic normalizing flow (Nielsen et al., 2020; Wu, Köhler, and Noé, 2020), to create normalizing flows with flexible mappings for altering the dimension of the latent space. Finally, we discuss more complex compressive flow architectures by combining and interlacing the basic ingredients.

5.4.1 *PPCA as an Injective Flow*

We begin by revisiting the classical technique of probabilistic principal components analysis (PPCA) (Tipping and Bishop, 1999) as a simple probabilistic model for change of dimension: $\mathbf{x} = \mathbf{A}\mathbf{z} + \boldsymbol{\mu} + \epsilon$, for data $\mathbf{x} \in \mathbb{R}^D$ and latent $\mathbf{z} \in \mathbb{R}^d$ where $D > d$ with parameters $\mathbf{A} \in \mathbb{R}^{D\times d}$ and a non-zero mean given by $\boldsymbol{\mu} \in \mathbb{R}^D$, and isotropic Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_D)$. Then, the $\mathbf{z}$-conditional probability over $\mathbf{x}$ is:

$$\mathbf{x} \mid \mathbf{z} \sim \mathcal{N}(\mathbf{A}\mathbf{z} + \mu, \sigma^2 \mathbb{I}_D) \ .$$

As per convention, the marginal distribution over latents is a factorial Gaussian, denoted $\mathbf{z} \sim \mathcal{N}(0, \mathbb{I}_d)$. The marginal distribution over the observed $\mathbf{x}$ is found after integrating out the latents and is simply Gaussian $\mathcal{N}(\mu, \mathbf{C})$, where the $D \times D$ covariance matrix is $\mathbf{C} = \mathbf{A}\mathbf{A}^\top + \sigma^2 \mathbb{I}_D$ and $\mathbf{A}\mathbf{A}^\top$ has rank (at most) d.

Let $\theta = \{\mathbf{A}, \mu, \sigma\}$ be parameters we seek to optimize. For a set of n i.i.d. samples $\{\mathbf{x}_i\}_{i=1}^n$, the log-likelihood of the model is:

$$\begin{aligned} \mathcal{L}_\theta &= \log p(\mathbf{x}; \theta) \\ &= \frac{n}{2} \log |\mathbf{C}| - \frac{1}{2} \sum_i (\mathbf{x}_i - \mu)\mathbf{C}^{-1}(\mathbf{x}_i - \mu)^\top \\ &= -\frac{n}{2} \log |\mathbf{C}| - \frac{1}{2} \operatorname{Tr}\left(\mathbf{C}^{-1}\mathbf{S}\right) \ , \end{aligned} \tag{5.3}$$

where $\mathbf{S}$ is the sample covariance matrix $\mathbf{S} = \frac{1}{n}\sum_{i=1}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^\top$ of the observations $\{\mathbf{x}_i\}_{i=1}^n$, and the maximum likelihood estimator μ is the mean of the data. The parameters $\theta = \{\mathbf{A}, \mu, \sigma\}$ can be estimated by maximizing $\mathcal{L}$ via MLE (Lucas et al., 2019; Tipping and Bishop, 1999). Moreover, since the model is Gaussian, the posterior can be obtained analytically:

$$p(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mathbf{M}^{-1}\mathbf{A}^\top(\mathbf{x} - \mu), \sigma^2 \mathbf{M}^{-1}) \ ,$$

where $\mathbf{M} = \mathbf{A}^\top\mathbf{A} + \sigma^2 \mathbb{I}_d$ is a $d \times d$ matrix.

We propose combining normalizing flows, which keep the dimensionality unchanged, with PPCA (or its nonlinear generalizations), which reduces the dimensionality. Thus, like Noisy Injective Flows (NIFs) (Cunningham et al., 2020), we begin with a simple

probabilistic model which maps high-dimensional data to a low-dimensional subspace which, for the simplest case of a linear dimensionality reduction, has a closed form solution. However, unlike NIFs which define the posterior q as the normalized reconstruction $\frac{p(\mathbf{x}|\mathbf{z})}{\int p(\mathbf{x}|\mathbf{z}')d\mathbf{z}'}$ Cunningham et al., 2020 (i.e., Bayes' rule with an improper prior), we instead treat the model as an application of PPCA where the true posterior is available analytically.

PPCA FLOW FORMULATION When used within a normalizing flow framework, the PPCA module introduces an intermediate high-dimensional latent (Gaussian) distribution $p_{Z'}$ before being mapped to the low-dimensional base distribution p_Z. Specifically, for a bijective flow g_β in which β denotes the flow parameters, we maximize:

$$\log p(\mathbf{x}) = \log p_{Z'}(g_\beta^{-1}(\mathbf{x})) + \log|\det \mathbf{J}_g| \,, \tag{5.4}$$

where $\log|\det \mathbf{J}_{g_\beta}| = \log|\det \nabla_{\mathbf{x}} g_\beta^{-1}(\mathbf{x})|$ is the absolute value of the log determinant Jacobian of the bijective transformations. With the addition of a PPCA-based transformation f_θ we then replace our high-dimensional Gaussian base distribution $p_{Z'}$ with a low-dimensional distribution. Hence, the full (forward) generative model is:

$$\begin{aligned} \mathbf{z} &\sim \mathcal{N}(0, \mathbb{I}_d) \,, \\ \mathbf{z}'|\mathbf{z} &\sim \mathcal{N}(\mathbf{A}\mathbf{z} + \mu, \sigma^2 \mathbb{I}_D) \,, \\ \mathbf{x} &= g_\beta(\mathbf{z}') \,, \end{aligned}$$

where latent samples $\mathbf{z}$ are sampled from the low-dimension isotropic Gaussian. Analogous to the PPCA marginal distribution over observations, we marginalize over $\mathbf{z}$ to write:

$$\mathbf{z}' \sim \mathcal{N}(\mu, \mathbf{A}\mathbf{A}^T + \sigma^2 \mathbb{I}_D) \,.$$

Then, for any given $\mathbf{x} = g_\beta(\mathbf{z}')$, by (5.3) we have the likelihood of $\mathbf{x}$ under a bijective flow g_β and the change of dimension parameters f_θ:

$$\begin{aligned} \log p(\mathbf{x}) &= \log p_{Z'}(g_\beta^{-1}(\mathbf{x})) + \log|\det \mathbf{J}_{g_\beta}| \\ &= -\log|\mathbf{C}| - \frac{1}{2}\operatorname{Tr}(\mathbf{C}^{-1}(g_\beta^{-1}(\mathbf{x}) - \mu)(g_\beta^{-1}(\mathbf{x}) - \mu)^T) \\ &\quad + \log|\det \mathbf{J}_{g_\beta}| \,. \end{aligned} \tag{5.5}$$

where $\mathbf{C} = \mathbf{A}\mathbf{A}^T + \sigma^2 \mathbb{I}_D$ is as before. Here the objective optimizes parameters $\Gamma = \{\beta, \theta\}$, including the bijective flow parameters β and the PPCA parameters $\theta = \{\mathbf{A}, \boldsymbol{\mu}, \sigma\}$. Since estimates for θ for a fixed β can be obtained explicitly, optimization can be performed by alternating optimization of bijective flow parameters β with stochastic mini-batches and closed-form updates for θ. Alternatively, optimization may proceed in an iterative fashion, where each parameter $\{\beta, \theta\}$ is updated from stochastic mini-batches.

One challenge in iterative maximization, however, is that parameters θ for the change of dimension flow may initially struggle if the high-dimensional distribution $p_{Z'}$, which is learned by the bijective flows g_β, is continuously changing. We consider modifying the objective (5.5) to penalize $\mathbf{z}' \in \mathbb{R}^D$ to also be Gaussian (see Figure 17 (a)). In doing so, we Gaussianize the input to the compression step. Similar to the β-VAE which anneals the VAE's regularization term (Higgins et al., 2017), we can stabilize training by annealing the compressive step's likelihood contribution:

$$\begin{aligned}\log p(\mathbf{x}) = \log \mathcal{N}(g_\beta^{-1}(\mathbf{x}) \mid \mu, \mathbf{A}\mathbf{A}^\top + \sigma^2 \mathbb{I}_D) + \log|\det \mathbf{J}_{g_\beta}| + \\ \alpha \left[-\log|\mathbf{C}| - \frac{1}{2}\operatorname{Tr}(\mathbf{C}^{-1}(g_\beta^{-1}(\mathbf{x}) - \mu)(g_\beta^{-1}(\mathbf{x}) - \mu)^\top) \right]\end{aligned} \tag{5.6}$$

where the first term $\mathcal{N}(g_\beta^{-1}(\mathbf{x}) \mid \mu, \mathbf{A}\mathbf{A}^\top + \sigma^2 \mathbb{I}_D)$ is new and represents a high-dimensional base distribution. The scalar $\alpha \in [0, 1]$ is annealed during training[1] to encourage the model to find a good representation in high dimensions before penalizing the model's ability to learn a useful low-dimensional representation. We refer to the objective in (5.6) as the Dual-Base Compressive Flow since it includes all the terms found in a bijective normalizing flow as well as a term for the change of dimensions.

5.4.2 *Extending PPCA to Linear VAE Flows*

A linear VAE with a Gaussian encoder and decoder recovers the maximum likelihood estimate of PPCA (Bao et al., 2020; Lucas et al., 2019; Rolinek, Zietlow, and Martius, 2019). To interpret encoding and decoding as flow transformations we specify the encoder model $q(\mathbf{z} \mid \mathbf{x})$ as the inverse transformation of the decoder. For the decoder we use a linear transformation $f(\mathbf{z}) = \mathbf{A}\mathbf{z} + \mu$ that maps the latent variables $\mathbf{z}$ to a manifold defined along a hyperplane. The decoder model is then $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x} \mid \mathbf{A}\mathbf{z} + \mu, \sigma^2 \mathbb{I}_D)$.

By formulating the decoder as a Gaussian, the encoder $q(\mathbf{z} \mid \mathbf{x})$ has a closed form solution. Specifically, $q(\mathbf{z} \mid \mathbf{x})$ is the posterior and found through Bayes' rule:

$$q(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})}{\int_{z'} p(\mathbf{x} \mid \mathbf{z}')p(\mathbf{z}')} = \mathcal{N}(\mathbf{z} \mid \mathbf{B}(\mathbf{x} - \mu), \mathbf{D}) \,, \tag{5.7}$$

for suitable $\mathbf{B} \in \mathbb{R}^{d \times D}$ and $\mathbf{D} \in \mathbb{R}^{d \times d}$, where $\mathbf{D}$ is a covariance matrix over all data points. By Lemma 1 of (Lucas et al., 2019) the linear VAE decoder $\mathbf{A}$ recovers weights $\mathbf{A}_{MLE}$ for the global maximum of the marginal log-likelihood for PPCA, and by setting the encoder parameters:

$$\begin{aligned}\mathbf{B} &= \mathbf{M}^{-1}\mathbf{A}_{MLE}^\top \\ \mathbf{D} &= \sigma_{MLE}^2 \mathbf{M}^{-1} \,,\end{aligned} \tag{5.8}$$

1 In experiments we find that annealing α from 0 to 1 during the first 20,000 iterations can stabilize performance.

we recover the true posterior and covariance at the global optimum with the maximum likelihood estimates (Rolinek, Zietlow, and Martius, 2019), where we have $\mathbf{M} = \mathbf{A}^\top \mathbf{A} + \sigma^2 \mathbb{I}_d$, just as in PPCA.

To derive the objective for the linear VAE first let $\hat{\mathbf{x}}$ be the reconstruction corresponding to $\mathbf{x}$. Under the Gaussian formulation with a linear transformation, we have:

$$\hat{\mathbf{x}} \mid \mathbf{x} \sim \mathcal{N}(\mathbf{AB}(\mathbf{x} - \mu) + \mu, \mathbf{ADA}^\top) \,. \tag{5.9}$$

Following standard techniques (Blei, Kucukelbir, and McAuliffe, 2017; Jordan et al., 1999b; Kingma and Welling, 2019; Wainwright and Jordan, 2007) log-likelihood for the VAE is:

$$\begin{aligned} \log p(\mathbf{x}) = & \underbrace{\mathrm{KL}\left(q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})\right)}_{\text{Bound Looseness}} \\ & - \underbrace{\mathrm{KL}\left(q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z})\right)}_{\text{Regularization}} + \underbrace{\mathbb{E}_q\left[\log p(\mathbf{x} \mid \mathbf{z})\right]}_{\text{Reconstruction}} \,. \end{aligned} \tag{5.10}$$

In the case of the linear VAE, however, the approximate posterior q may match the posterior p and parameters are available in closed form, and hence the first term in (5.10) denoting the tightness of the bound on the log-likelihood of the data is zero. Moreover, the remaining objective terms from (5.10) can be optimized in closed form (Lucas et al., 2019) by taking the gradient of the loss with respect to the model parameters $\theta = \{\mathbf{A}, \mathbf{B}, \mathbf{D}, \mu, \sigma\}$.

LINEAR VAE FLOW FORMULATION Thus, when used within a normalizing flow framework, the linear VAE module maps the intermediate latent Gaussian distribution $p_{Z'}$ to the low dimensional base distribution p_Z and presents the following objective:

$$\begin{aligned} \log p(\mathbf{x}) &= \log p_{Z'}(g_\beta^{-1}(\mathbf{x})) + \log \left|\det \mathbf{J}_{g_\beta}\right| \\ &= -\mathrm{KL}(q(\mathbf{z}|\mathbf{z}')\|p(\mathbf{z})) + \mathbb{E}_q\left[\log p(\mathbf{z}' \mid \mathbf{z})\right] + \log \left|\det \mathbf{J}_{g_\beta}\right| \end{aligned} \tag{5.11}$$

where latent variable $\mathbf{z}'$ is computed $\mathbf{z}' = g_\beta^{-1}(\mathbf{x})$ by the bijective flow transformation with parameters β, and the linear VAE's KL-Divergence and reconstruction terms have the closed form solutions:

$$\mathrm{KL}(q(\mathbf{z}|\mathbf{z}')\|p(\mathbf{z})) = \frac{1}{2}\left(\mathrm{Tr}(\mathbf{D}) - \log\det(\mathbf{D}) + \bar{\mathbf{z}}\mathbf{B}^\top\mathbf{B}\bar{\mathbf{z}}) - d\right) \,,$$

and,

$$\begin{aligned} \mathbb{E}_q\left[\log p(\mathbf{z}' \mid \mathbf{z})\right] &= \mathbb{E}_q\left[\frac{-1}{2\sigma^2}\left(\mathbf{Az} - \bar{\mathbf{z}})^\top \mathbf{Az} - \bar{\mathbf{z}})\right) - c\right] \\ &= \frac{1}{2\sigma^2}\left[-\mathrm{Tr}(\mathbf{ADA}^\top) - \bar{\mathbf{z}}^\top\mathbf{B}^\top\mathbf{A}^\top\mathbf{AB}\bar{\mathbf{z}} + 2\bar{\mathbf{z}}^\top\mathbf{AB}\bar{\mathbf{z}} - \bar{\mathbf{z}}^\top\bar{\mathbf{z}}\right] - c \,, \end{aligned}$$

where $\bar{\mathbf{z}} = (\mathbf{z}' - \mu) = (g_\beta^{-1}(\mathbf{x}) - \mu)$ and $c = -\frac{M}{2}\log 2\pi\sigma^2$ are denoted for compact notation.

Note that the generative process for a change of dimension with the linear VAE flow is equivalent to the PPCA-base flow. That is, for a given data-point $\mathbf{x}$ we can project $\mathbf{x}$ to a subspace by $\mathbf{z} = f_\theta^{-1}(g_\beta^{-1}(\mathbf{x})) = \mathbf{M}^{-1}\mathbf{A}^\top(g_\beta^{-1}(\mathbf{x}) - \mu)$. Likewise, generating a new data-point $\mathbf{x}^{new}$ can be done by sampling $\mathbf{z} \sim \mathcal{N}(0, \mathbb{I}_d)$ and computing $\mathbf{x}^{new} = g_\beta(\mathbf{A}\mathbf{z} + \mu)$. The key difference with linear VAE-based flows is that they are an extendable version of the PPCA-based flow that is optimized with amortized variational inference (Kingma and Welling, 2014) techniques—leading to greater flexibility and scalability.

5.4.3 *Nonlinear VAEs for Compressive Flows*

The objective function for the VAE (5.10) includes a KL-Divergence term between the approximate and true posterior, which corresponds to the looseness of the lower bound looseness. When the "bound looseness" term is omitted, we arrive at the evidence lower bound (ELBO) (Blei, Kucukelbir, and McAuliffe, 2017; Jordan et al., 1999b; Wainwright and Jordan, 2007) on the marginal log-likelihood of the data. However, writing this same lower bound as:

$$\log p(\mathbf{x}) \geqslant \underbrace{\mathbb{E}_q\left[\log p(\mathbf{z})\right]}_{\text{Reparameterized Prob}} + \underbrace{\mathbb{E}_q\left[\log\left(\frac{p(\mathbf{x} \mid \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\right)\right]}_{\text{Likelihood Contribution}} \tag{5.12}$$

where the first term can be interpreted as the reparameterization over the random variables $\mathbf{x}$ to $\mathbf{z}$ (i.e., encoding) and the second term acts as the volume correction factor found in a bijective normalizing flow framework (Nielsen et al., 2020) (i.e., the determinant Jacobian). The likelihood contribution is derived from the lower bound (5.10), where the tightness of the bound follows from by how well the variational approximation (encoder) $q(\mathbf{z} \mid \mathbf{x})$ matches the true posterior. In the case of PPCA and Linear VAEs the variational approximation can match the simple posterior exactly.

The bound in (5.12) mimics the normalizing flow structure but for the stochastic class of flow transformations (Nielsen et al., 2020; Wu, Köhler, and Noé, 2020). Thus, in our VAE-based change of dimension flow framework we compute the stochastic flow's likelihood contribution $\mathcal{V}$ by:

$$\mathcal{V} = \log \frac{p(\mathbf{z}' \mid \mathbf{z})}{q(\mathbf{z} \mid \mathbf{z}')} \text{ with } \mathbf{z} \sim q(\mathbf{z} \mid \mathbf{z}') \,, \tag{5.13}$$

where $\mathbf{z}' = g_\beta^{-1}(\mathbf{x})$, and we can compute empirical estimates of the population expectation based on Monte Carlo samples $\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{z}')$. While bijective normalizing flows are known for exact likelihood computations, the practice of intermixing lower bounds from stochastic flows like (5.13) is commonly practiced in the form of variational dequantization (Ho, Lohn, and Abbeel, 2019; Ho et al., 2019; Nielsen and Winther, 2020; Nielsen et al., 2020; Uria, Murray, and Larochelle, 2013), where discrete inputs (e.g.,

pixels) are dequantized for continuous distributions as a common first step in flow models.

Thus, when using Gaussian nonlinear VAEs as a change of dimension flow, we write the marginal log likelihood objective in (5.12) in terms of Gaussian distributions for the encoder $q_\phi(\mathbf{z}|\mathbf{z}') = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{z}'), \sigma_\phi(\mathbf{z}'))$ and the decoder $p_\theta(\mathbf{z}'|\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mu_\theta(\mathbf{z}'), \sigma_\theta(\mathbf{z}'))$, where ϕ, θ are respectively the parameters for the encoder and decoder networks which model the respective means and variances. The encoder and decoder (parameters) learn by maximizing an empirical estimate of the likelihood lower bound:

$$\begin{aligned} \log p(\mathbf{x}) &\geqslant \log p_{Z'}(g_\beta^{-1}(\mathbf{x})) + \log|\det \mathbf{J}_{g_\beta}| \\ &= \hat{\mathbb{E}}_{\mathbf{z}\sim q(\mathbf{z}|g_\beta^{-1}(\mathbf{x}))}\left[\log \mathcal{N}(\mathbf{z} \mid 0, \mathbb{I}_d) + \log \frac{p_\theta(\mathbf{z}'|\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{z}')}\right] \\ &\quad + \log|\det \mathbf{J}_{g_\beta}|\,, \end{aligned} \tag{5.14}$$

where $\mathbf{z}' = g_\beta^{-1}(\mathbf{x})$ and $\hat{\mathbb{E}}$ denotes an estimated expectation based on samples. The (stochastic) optimization now updates three sets of parameters: β, θ, ϕ respectively corresponding to the bijective flow, encoder, and decoder.

CONTRAST WITH VARIATIONAL AUTOENCODERS. Given recent progress in building more powerful VAEs (Dhariwal et al., 2020; Kingma et al., 2016; Razavi, van den Oord, and Vinyals, 2019; Rezende and Mohamed, 2015; Vahdat and Kautz, 2020; van den Oord, Vinyals, and kavukcuoglu, 2017), the benefits of using VAEs as compressive steps in normalizing flows may not be immediately clear. While VAEs learn latent representations of data and have straight-forward sampling (Kingma and Welling, 2019), there are unique advantages to the normalizing flow framework. Normalizing flows are composable, modular models with exact likelihood calculation (Kobyzev, Prince, and Brubaker, 2020; Papamakarios et al., 2019). Simple linear approaches like the PPCA and linear VAE-based flow steps intermix into the normalizing flow framework while maintaining exact likelihood computation. More flexible change of dimension transformations like the Gaussian VAEs introduce lower bounded objective function terms only for those specific flow steps.

Lastly, there is an efficiency to flow-based models which only need to learn a single invertible mapping to perform both the encoding and decoding. Even VAEs with flow-based posteriors (Hoogeboom et al., 2020; Rezende and Mohamed, 2015; van den Berg et al., 2018) (typically) train very deep encoders and decoders separately.

5.4.4 *Designing Compressive Flow Architectures*

When composing normalizing flows that intermix both bijective and stochastic compression transformations the order of flow steps is an important consideration. More specifically, we advocate for architectures that apply bijective flows first followed by one or more compressive steps.

BIJECTIONS FIRST. While compressing first can reduce the overall number of parameters in the model, there are many advantages to first applying bijective transformations. In particular, bijective transformations are more adept at mapping inputs to a Gaussian latent space (with exact likelihood computation), where simpler linear VAEs or PPCA based steps then reduce the dimensionality in closed form (see Figure 17 (b), for example). Deep Gaussian VAEs may perform better at encoding inputs that are not perfectly Gaussian, but at the cost of optimizing a lower bound on the log-likelihood.

Finishing a bijective flow with a compressive step may not improve performance but a compact latent representation is more attractive for potential down-stream tasks. Similarly, we can extend large pre-trained flow models with a compressive step and then fine-tune all the weights in the network, as shown in Figure 17 (b).

ALTERNATING BIJECTIVE AND COMPRESSIVE STEPS. Because Gaussian VAEs are amenable to flow frameworks, we also consider compositions of flow steps that include bijective and stochastic compression steps (see Figure 17 (c)). It is possible to significantly reduce the overall size of the model by alternating between dimension preserving bijective flows and change of dimension flows. We explore flow architectures that compress the latent space before applying the multiscale architecture (Dinh, Sohl-Dickstein, and Bengio, 2017) technique of squeezing[2]. In other words, we Gaussianize the latent space with bijective flows, squeeze to shrink the spatial dimension of the image while increasing channels, and then reduce the number of channels with a VAE.

5.5 EXPERIMENTS

The focus in evaluating our methods is, first, to understand the trade-off between performance and compression for various types of compressive flows (Section 5.5.2). A desirable result will be minimal and graceful degradation in model performance as we do more compression. Second, in Section 5.5.3 we examine how performance varies as a result of the architectural configuration. For example, compressing as a first step reduces the number of parameters more than compressing as a final step—but is the difference in performance significant? Lastly, we show that very large pretrained bijective flows can be extended with a compression step without significantly harming performance. To be clear, our goal with compressive flows is not improve the state-of-the-art, rather we study how the benefits of a compressed latent space can be realized without a significant drop in performance. Such compressed models can then be used for downstream tasks with considerably less memory and communication foot-print.

2 The squeeze operation does not change the dimension, but merely reshapes the data to increase the number of channels.

5.5.1 *Experimental Setup*

In all our experiments we use a common architectural pattern of alternating squeeze operations, which exchange spatial dimensions for more channels, followed by bijective flow steps consisting of 1×1 convolutions and ActNorm (Kingma and Dhariwal, 2018) with coupling layers (Dinh, Krueger, and Bengio, 2015) (see 17). We follow the Flow++ technique and use logistic mixture CDF coupling layers whose parameters are learned by self-attention (Ho et al., 2019; Vaswani et al., 2017). Compression with PPCA-based flows are trained analytically, and where indicated can include the Dual-Base objective from (5.6) which Gaussianizes the input to the PPCA step. Linear VAE steps are designed with a single hidden dense layer in the encoder and decoder and no activation functions. Nonlinear Gaussian VAEs include two dense layers in the encoder and decoder with ReLU activations. Lastly, flow models that intermix compressive VAE steps throughout use convolutional VAEs to maintain the spatial structure of the image data while reducing the latent space channel-wise. We evaluate our models on `MNIST`, `CIFAR-10`, and `ImageNet` 32×32 with the preset training and testing splits. Model performance is reported in bits/dim $= -\mathcal{L}/(\dim(\mathbf{x}) \cdot \log 2)$, where $\mathcal{L}$ is the log-likelihood for a test sample and $\dim(\mathbf{x})$ is the number of pixels in the image Papamakarios, Pavlakou, and Murray, 2017, which we seek to minimize since it is proportional to negative log-likelihood.

DATASETS For image modeling results we experiment on three datasets. `MNIST`[3] contains 60,000 and 10,000 images pre-split into training and test sets, each of which is a 28×28 grayscale image and portrays a hand-written digit (LeCun et al., 1998). `CIFAR-10` consists of 60,000 32×32 color images featuring 10 classes of things, with 50,000 and 10,000 images pre-split into the training and test sets, respectively(Krizhevsky, 2009). The `ImageNet` 32×32 dataset also comes pre-split with 1,281,149 training images and a validation set of 49,999 images which we use as a test set (Deng et al., 2009; Oord, Kalchbrenner, and Kavukcuoglu, 2016).

5.5.1.1 *Setup: Experiments on Latent Dimensionality and Flow Depth*

TRAINING AND HYPERPARAMETERS For experiments in which we compare performance for varying latent space sizes (Section 5.5.2), and the experiments where we evaluate performance with varying depths of bijective flows preceding the compression step (Section 5.5.3) we train our baseline flow with a single VAE compression step and a common training experimental setup. We follow a similar training procedure as (Nielsen et al., 2020) and extend their codebase that uses the Pytorch framework (Paszke et al., 2019). Specifically, we use the Adamax optimizer (Kingma and Ba, 2015) with a batch size of 32 and an initial learning rate of 10^{-3}. For the first 5000 training iterations the learning rate is warmed up from 0.0. We train `MNIST` and `CIFAR-10` models for 300 epochs, during which the learning rate decayed by 0.995 every epoch.

3 http://yann.lecun.com/exdb/mnist/

Finally, for `CIFAR-10` data augmentation was applied during training, including random flipping and rotations.

BASELINE BIJECTIVE FLOWS In all `MNIST` models we use multiscale architectures with 2 scales and 8 steps/scale, whereas for `CIFAR-10` and 3 scales and 8 steps/scale. Each step in the flows is an affine coupling bijection (Dinh, Sohl-Dickstein, and Bengio, 2017) parameterized by Logistic Mixture CDF with attention (Ho et al., 2019), a 1×1 invertible convolution, and Actnorm layer (Kingma and Dhariwal, 2018). All models are trained with variational dequantization (Ho et al., 2019). We use an initial squeezing layer, as well as squeezing between each scale that reduces each of the spatial dimensions by half and quadruples the number of channels.

COMPRESSIVE FLOW STEPS In experiments we compressive steps based on PPCA, linear VAEs, and nonlinear VAEs. The PPCA-based compressive steps is a simple linear mapping with a trainable mean and variance, which are trained iteratively with analytically derived mini-batch updates. For linear VAEs we use only simple Gaussian VAEs without activation functions and a single dense hidden layer of size 384 in the encoder and decoder. In experiments with a single compressive step we implement nonlinear Gaussian VAEs with ReLU activations and two dense hidden layers of size $[512, 256]$ in the encoder and a mirrored architecture for the decoder.

In experiments with multiple compressive steps intermixed between bijective flows, we use convolutions to preserve the spatial structure of the latent space. More specifically, use convolutional layers, where convolutional layers follow the PyTorch convention (Paszke et al., 2017). The encoder of these networks contains the following layers:

$$
\begin{aligned}
&\text{Conv}(\text{in} = C, \text{out} = 64, k = 3, p = 2, s = 1)\\
&\text{MaxPool}(k = 2)\\
&\text{Conv}(\text{in} = 64, \text{out} = 128, k = 3, p = 2, s = 1)\\
&\text{MaxPool}(k = 2)\\
&\text{Conv}(\text{in} = 128, \text{out} = 256, k = 3, p = 2, s = 1)\\
&\text{AdaptiveAvgPool}(\text{outsize} = (1, 1))\\
&\text{Conv}(\text{in} = 256, \text{out} = 2L, k = 1, s = 1)
\end{aligned}
$$

where C is the number of input channels directly following the squeezing layer at that scale, 2L denotes number of parameters for the mean and variance of approximate posterior latent space, k is a kernel size, p is a padding size, and s is a stride size. The $\text{MaxPool}()$ operation reduces the spatial dimension by half, whereas $\text{AdaptiveAvgPool}()$ is an adaptive average pooling layer that reduces the remaining spatial dimensions to $(1, 1)$ so that the final 1×1 convolution layer (used in place of a dense layer) outputs a single vector corresponding to parameters to the latent dimension.

Model	CIFAR-10	ImageNet32
RealNVP (Dinh, Sohl-Dickstein, and Bengio, 2017)	3.49	4.28
Glow (Kingma and Dhariwal, 2018)	3.35	4.09
Flow++ (Ho et al., 2019)	3.08	3.86
MaxPoolFlow (Nielsen et al., 2020)	3.09	4.01
Pretrained (Nielsen et al., 2020)	3.08	4.00
Compressed (ours)	3.10	4.00

Table 9: Average bits/dim for flow models. The pretrained model from Nielsen et al., 2020 is extended with a VAE compression step reducing the latent space by 88%.

Similarly, the decoder mirrors the encoder using the following transposed convolutions:

$$\begin{aligned}
&\text{ConvT}(\text{in} = L, \text{out} = 256, k = 3, s = 2)\\
&\text{ConvT}(\text{in} = 256, \text{out} = 128, k = 3, s = 2)\\
&\text{ConvT}(\text{in} = 128, \text{out} = 64, k = 3, s = 2)\\
&\text{AdaptiveAvgPool}(\text{outsize} = (h, w))\\
&\text{Conv}(\text{in} = 64, \text{out} = D, k = 1, s = 1)
\end{aligned}$$

The decoders AdaptiveAvgPool() reduces spatial dimension to the height h and w of the latent space to the size of the original input, and the 1×1 convolution acts as a dense layer to match the number of channels of the original input. Between all convolutional layers we apply batch normalization and then a ReLU activation (not shown).

5.5.1.2 *Setup: Experiments on Compressing Pretrained Models*

TRAINING AND HYPERPARAMETERS In Section 5.5.4 we extend the nonpooled pretrained models[4] from Nielsen et al. (2020). Pretrained models with an additional compression step are trained by first freezing all layers in the pretrained model and only training the compressive step with a learning rate of 0.001 for 100 epochs on `CIFAR-10` and 5 epochs on `ImageNet`. Lastly we fine-tune all layers in the model with a learning rate of $2 \cdot 10^{-5}$.

PRETRAINED MODELS Unlike the baseline bijective flows used in our other experiments, the pretrained models uses no ActNorm (Kingma and Dhariwal, 2018) layers and coupling layers are based on DenseNets (Gao Huang et al., 2017) that are much more highly parameterized than the networks in our experiments—with approximately 105M parameters in the pretrained model.

4 https://github.com/didriknielsen/survae_flows/releases/tag/v1.0.0

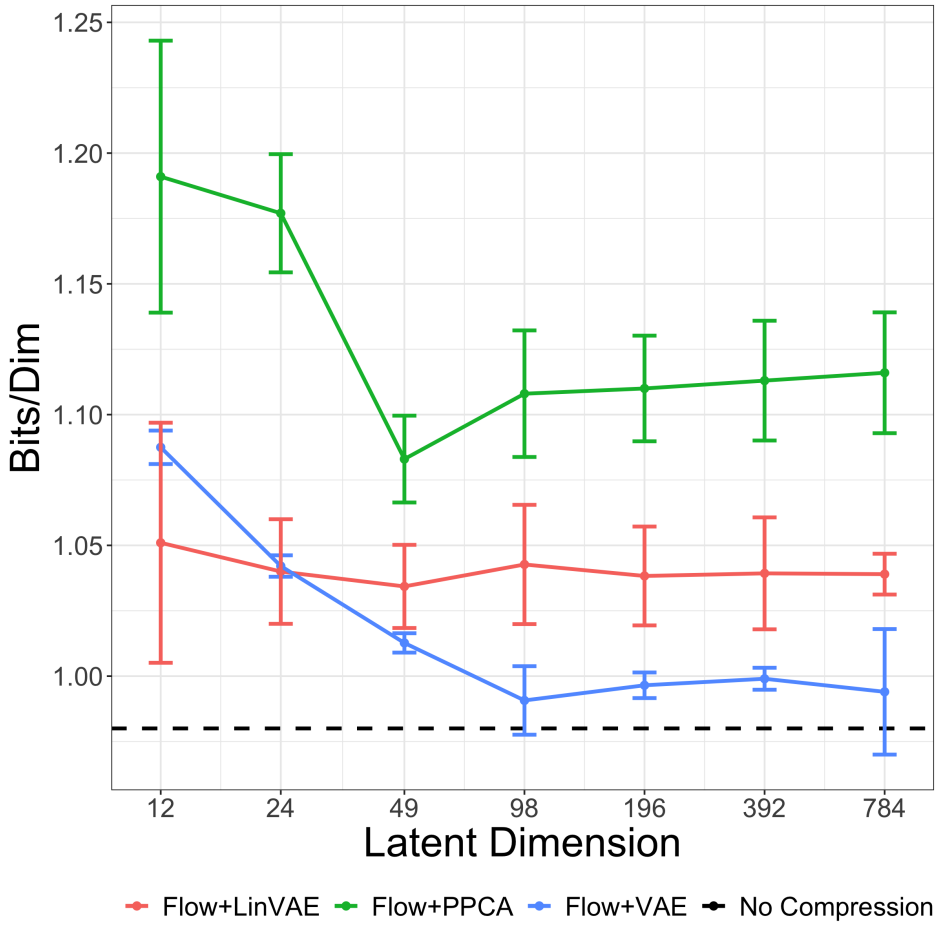


Figure 18: Bits/dim (lower is better) on `MNIST` for varying latent sizes. The dashed line indicates performance for a purely bijective flow with a latent size of 784 (no compression performed). The more flexible VAE-based compressive steps allow for the most compression with the least drop in performance.

Table 10: Average bits/dim (lower is better) for compression models. Models substantially compress the latent space by ≈94%, but we only observe a a small drop in performance compared to a purely bijective flow (No compression).

Model	**MNIST**	**CIFAR-10**
No compression	0.98	3.27
MaxPoolFlow	1.10	3.41
PPCA Compression	1.08	3.37
Linear VAE Compression	1.07	3.35
Intermixed Compression	1.04	3.35
VAE Compression	1.01	3.30

Figure 19: Samples from a large model pretrained on `CIFAR-10` (top), and the same model extended with a compressive step reducing the latent space by 88% (bottom).

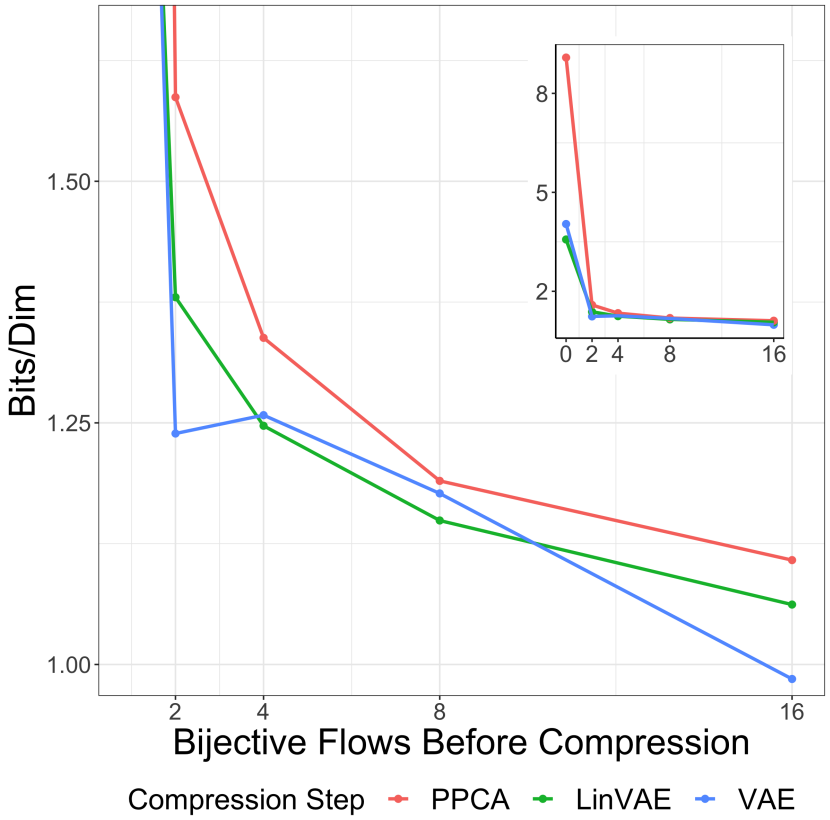


Figure 20: Bits/dim for models with varying bijective flow depths before compressing. Without bijective flows our compressive flows cannot effectively model the data, but combining bijective and compressive flows leads to a powerful generative model with a compressed latent space.

5.5.2 *Compression Performance Trade-off*

Compression can cause a drop in performance when the model is forced to discard too much information. To better understand the trade-off between learning a compressed latent space and performance, we evaluate flow models with PPCA-based and Gaussian VAE-based compressive steps for varying latent dimensions. Overall we find that the performance of compressive flows is resilient over a range of compression levels.

In Figure 18 we find, unsurprisingly, that all models perform worse when dramatically compressing the latent space for `MNIST` from 784 to 12 dimensions (a 98% reduction). Models that use nonlinear VAEs, which are more flexible than PPCA, almost always perform the best. Surprisingly, despite reducing the latent dimension by as much as 88% (latent size of 98) the VAE-based compressive flow performs nearly as well as a purely bijective flow, suggesting that a VAE-based compressive step helps map the data to the desired base distribution while simultaneously finding good low-dimensional latent representations.

In Table 10 we compare the performance of a variety of compressive flows that reduce the latent space by 94%. Unlike the models in Table 9, here the models use far fewer parameters (≈5-10x smaller). Despite reducing the latent space greatly, we see a minimal drop in performance relative to a purely bijective flow that learns a latent space that is the same size as the data. Nonlinear VAEs perform the best, whereas PPCA-based compression flows and the MaxPoolFlow Nielsen et al., 2020, which also optimizes a lower bound, perform worse. Normalizing flows that intermix VAE compressions between bijective steps reduce the latent space at each scale of our multi-scale architecture (two scales for `MNIST`, three for `CIFAR-10`). Intermixing compressive steps throughout the chain of flow steps results in a model with 38% fewer parameters (for `CIFAR-10`) but still performs comparably.

5.5.3 *Flow Depth Before Compression*

To evaluate the interaction between bijective and compressive flows we compare normalizing flows with varying bijective flow depths followed by a single compression step that compresses the latent space by 75% on `MNIST` (Figure 20). Because our compressive flows are relatively weak—specifically VAEs with just two dense hidden layers, we find that performance depends highly on first applying bijective flows. For all bijective flow depths, compressing with the more simple PPCA-based flows leads to worse performance, indicating that the VAE flow steps are a powerful compressive model, while also helping to fit the latent space to the Gaussian base distribution.

5.5.4 *Compressing Pretrained Flows*

In Table 9 we report the performance on `CIFAR-10` and `ImageNet32` of state-of-the-art normalizing flows along with the model from Nielsen et al., 2020 that we extend with

a single nonlinear VAE-based compressive flow. The compressive VAE contains only two dense hidden layers in the encoder and decoder, and reduces the size of the latent space by 88%. We train the compressive flow by first freezing the pretrained model's layers and training the VAE, then fine-tuning all the model weights at a low learning rate. We find the performance degrades minimally, suggesting that the latent space for bijective flows may contain redundant information and that information loss due to compression can be mitigated by the powerful bijective flow steps.

5.6 CONCLUSION

We introduce a PPCA-based normalizing flow, capable of relaxing the bijective constraints on flows and learning a low dimensional latent space. Building on recent work connecting PPCA and Linear VAEs (Lucas et al., 2019) and the perspective of VAEs as stochastic flows (Nielsen et al., 2020), we extend our PPCA-based flow to more flexible compressive flows using linear and nonlinear VAEs. While PPCA-based flows can be optimized in closed form, VAE-based flows are more flexible and scalable. We derive objectives for models that combine bijective and compressive flow steps, and introduce architectures for compressive flows. Our experiments show that reducing the latent space significantly results in only minor model degradation. Similarly, extending large pretrained bijective flows with a compressive flow produces more compact latent representations. Our work shows that preserving dimensionality is a mathematical constraint on bijective transformations, but not a requirement for producing powerful normalizing flows.

6

PROBABILISTIC SUPER-RESOLUTION WITH NORMALIZING FLOWS

6.1 INTRODUCTION

Normalizing flows are a powerful class of generative models that express probability distributions through a sequence of diffeomorphic transformations of a simple base distribution (Dinh, Sohl-Dickstein, and Bengio, 2017; Rezende and Mohamed, 2015; Tabak and Turner, 2013; Tabak and Vanden-Eijnden, 2010). Recent advances in the expressive power of such transformations make normalizing flows an attractive choice for (tractable) density estimation, posterior approximation, and sampling (Papamakarios et al., 2019). Much of normalizing flow research has focused on unconditional density estimation and variational inference (Papamakarios et al., 2019). However flows can be extended to condition on side information y, such as labels, to guide generated samples $\mathbf{x} \sim p(\mathbf{x}|y)$ (Kingma and Dhariwal, 2018). Alternatively, Chen et al. (2019) show that flows can be combined with a hybrid modeling approach to predict class labels y; similarly Lu and Huang (2020) and Winkler et al. (2019) propose using conditional flows to model higher-dimensional outputs for image segmentation.

In this paper, we focus on using normalizing flows to model joint distributions $p(\mathbf{x}, \mathbf{y})$ and conditional distributions $p(\mathbf{y}|\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^D$, where $d < D$. We specifically consider the setting where $\mathbf{x} = \pi(\mathbf{y})$ for some $\pi : \mathbb{R}^D \mapsto \mathbb{R}^d$, so that modeling $p(\mathbf{y}|\mathbf{x})$ over $\mathbf{y} \in \mathbb{R}^D$ given a "low resolution" version $\mathbf{x} \in \mathbb{R}^d$ is *probabilistic super-resolution* (PSR). We present models for joint PSR to model $p(\mathbf{x}, \mathbf{y})$ and conditional PSR to model $p(\mathbf{y}|\mathbf{x})$. Such models use two normalizing flows, one for the low-resolution (LR) marginal $p(\mathbf{x})$ starting from a base distribution $p(\boldsymbol{\epsilon})$ with $\boldsymbol{\epsilon} \in \mathbb{R}^d$, and the other for the super-resolution (SR) conditional $p(\mathbf{y}|\mathbf{x})$ given $\mathbf{x}$ and a base distribution $p(\mathbf{z}), \mathbf{z} \in \mathbb{R}^D$. The approach can be extended to multi-resolution generative models, with more than two levels.

Conditional density estimation with normalizing flows (Ardizzone et al., 2019, 2021; Lu and Huang, 2020; Winkler et al., 2019) is a unique problem in the PSR setting because the conditional $\mathbf{x}$ is not new side information, but rather a lossy representation of $\mathbf{y}$, and the model must fill in the missing information. Interestingly, the flows we construct for PSR overcome some of the unique challenges facing unconditional normalizing flows. First, the PSR flow addresses the information bottleneck problem in flows

(Ardizzone et al., 2020; Chen et al., 2020; Huang, Dinh, and Courville, 2020), wherein a flow's dimension preserving constraint limits the amount of information that can be passed from one flow step to the next. Second, by utilizing the low-resolution $\mathbf{x}$, which can contain global information, the PSR flow incorporates higher-level structure in the latent space that overcomes the inductive bias of flows to focus on modeling low-level pixel correlations (Kirichenko, Izmailov, and Wilson, 2020), as opposed to modeling high-level semantic information.

In order to better understand the distribution learned by PSR flows, we create a highly controlled student-teacher evaluation setup. We select a teacher normalizing flow that is specifically designed to model a particular multi-modal distribution. Since the teacher is a normalizing flow it can generate samples and evaluate the likelihoods of samples produced by students exactly. We compare a variety of probabilistic models as students, and show that PSR flows are highly adept at learning complex multi-modal output distributions—even at the tails of the distributions.

In the context of image modeling (Ardizzone et al., 2021; Lugmayr et al., 2020), PSR has many potential applications ranging from surveillance, medical imaging, and efficient video content delivery. Training a PSR flow is possible on any image dataset, and produces sharp image generation while retaining all the same advantages afforded by unconditional flows. The super-resolution flows introduced in Ardizzone et al. (2021) and Lugmayr et al. (2020) are conditional models, however we show that modeling the full joint distribution $p(\mathbf{x}, \mathbf{y})$ over the LR and HR images allows us to easily sample novel HR images $\mathbf{y}$, just as in the unconditional setup. Further, to improve on PSR flows we combine state-of-the-art techniques for training flows with our flow steps that are conditional on a learned upsampling of LR features. To evaluate our approach we present qualitative and quantitative results on a variety of image datasets.

6.2 BACKGROUND AND RELATED WORK

Normalizing flows (NF) play an important role in the recent developments of both density estimation and variational inference (Rezende and Mohamed, 2015). Normalizing flows apply a differentiable, invertible transformation $f(\cdot; \boldsymbol{\phi})$ with tractable Jacobians, which defines a mapping between a complex distribution p_X and a simple distribution p_Z, such as a standard normal where X and Z are random variables with values $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$ (Tabak and Turner, 2013; Tabak and Vanden-Eijnden, 2010). Through the change of variables formula the density function of X is:

$$p_X(\mathbf{x}) = p_Z(f^{-1}(\mathbf{x}; \boldsymbol{\phi})) \left| \det \frac{\partial f}{\partial \mathbf{x}} \right| \tag{6.1}$$

Parameterizing normalizing flows with deep neural networks (Dinh, Krueger, and Bengio, 2015; Dinh, Sohl-Dickstein, and Bengio, 2017; Rippel and Adams, 2013) has popularized the technique for density estimation and variational inference Papamakarios et al., 2019. A key challenge in constructing flows capable of powerful transforma-

tions is retaining the invertibility property of f. While autoregressive flows (Huang et al., 2018b; Kingma et al., 2016) are powerful, they are D times slower to invert. A popular approach we employ throughout our experiments is the use of coupling layers (Dinh, Krueger, and Bengio, 2015; Dinh, Sohl-Dickstein, and Bengio, 2017) which remain invertible by applying affine or additive transformations to half of the inputs and use the remaining inputs to learn the transformation. Despite only learning a transformation from half the feature space, advances in coupling layers (Behrmann et al., 2019; Chen et al., 2020; Chen et al., 2019; Cornish et al., 2020; Dinh et al., 2019; Durkan et al., 2019; Giaquinto and Banerjee, 2020; Ho et al., 2019; Huang, Dinh, and Courville, 2020; Kingma and Dhariwal, 2018) create flexible transformations.

CONDITIONAL NORMALIZING FLOWS Recently, interest has grown in adapting the normalizing flow framework for *conditional density estimation* (CDE) (Lu and Huang, 2020; Trippe and Turner, 2017; Winkler et al., 2019), where the goal is to model the distribution over labels or regression outputs, denoted $p_{Y|X}(\mathbf{y}|\mathbf{x})$. Flow-based CDE models are attractive due to their ability compute likelihoods of outputs exactly—providing a probabilistic framework that is straight-forward to optimize.

Similar to other generative approaches, normalizing flows cannot always match the predictive performance of their discriminative counterparts on regression and classification tasks (Ardizzone et al., 2020; Mackowiak et al., 2020). However, the ability of flows to compute likelihoods exactly, generate samples, interpolate, and simulate a distribution of outcomes make them an important development in modeling conditional distributions. In particular, flow-based CDEs have successfully been applied to tasks like image segmentation (Lu and Huang, 2020; Sorkhei, Henter, and Kjellström, 2020; Winkler et al., 2019), time-series and sequence modeling (Bhattacharyya et al., 2019; Both and Kusters, 2019; Deng et al., 2020; Rasul et al., 2020; Ziegler and Rush, 2019), speech and video synthesis (Kumar et al., 2019; Prenger, Valle, and Catanzaro, 2018), and image super-resolution (Ardizzone et al., 2019, 2021; Lugmayr et al., 2020; Yu, Derpanis, and Brubaker, 2020).

A flow-based CDE considers input output pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, and the goal is to learn $p_{Y|X}(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\phi})$ with parameters $\boldsymbol{\phi}$ which corresponds to minimizing the forward KL-divergence:

$$\mathrm{KL}\left(p^*(\mathbf{y} \mid \mathbf{x}) \,||\, p_{Y|X}(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\phi})\right) \tag{6.2}$$

where p^* denotes true distribution. Optimization of the flow parameters $\boldsymbol{\phi}$ proceeds in a similar fashion as the unconditional density estimation problem, except we consider the expected KL-divergence with respect to the inputs $\mathbf{x} \sim p(X)$ for the objective:

$$\begin{aligned}\boldsymbol{\phi}^* &= \arg\min_{\boldsymbol{\phi}} \mathop{\mathbb{E}}_{\mathbf{x}\sim p(X)} \left[\mathrm{KL}\left(p^*(\mathbf{y} \mid \mathbf{x}) \,||\, p_{Y|X}(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\phi})\right)\right] \\ &= \arg\min_{\boldsymbol{\phi}} \underbrace{\mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim p(X,Y)} \left[\log p^*(\mathbf{y} \mid \mathbf{x})\right]}_{\text{Constant}} - \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim p(X,Y)} \left[\log p_{Y|X}(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\phi})\right] \\ &= \arg\min_{\boldsymbol{\phi}} -\mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim p(X,Y)} \left[\log p_{Y|X}(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\phi})\right] \ . \end{aligned} \tag{6.3}$$

For the conditional normalizing flow we define $p_{Y|X}(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\phi})$ using a base distribution p_Z and an invertible mapping f that is a bijective between outputs Y and latent Z, and is conditioned on X.

With invertible flow transformation f, the generative process for the model is $\mathbf{z} \sim \mathcal{N}(0, 1)$ and $\mathbf{y} = f(\mathbf{z}; \mathbf{x}, \boldsymbol{\phi})$. To allow for flexible transformations, we henceforth denote $f = f_K \circ \cdots \circ f_1$ as the K-step normalizing flow which retains the invertible property (Papamakarios et al., 2019) to give $\mathbf{z} = f^{-1}(\mathbf{y}; \mathbf{x}, \boldsymbol{\phi})$.

The change of variables formula then extends to the conditional setting, and thus to estimate the expectation over the true data distribution we take a Monte Carlo approximation of (6.3), yielding the log-likelihood objective:

$$\mathcal{L} \approx \frac{1}{n} \sum_{i=1}^{n} \left[\log p_Z\left(f^{-1}(\mathbf{y}_i; \mathbf{x}_i, \boldsymbol{\phi})\right) + \sum_{k=1}^{K} \log \left|\frac{\partial f_k^{-1}}{\partial \mathbf{y}_i}\right|\right] \ , \tag{6.4}$$

where $\mathbf{y} = \mathbf{z}_K$ and the latent $\mathbf{z}_k = f_k(\mathbf{z}_{k-1}; \mathbf{x}, \phi_k)$ is computed by passing the true label $\mathbf{y}$ through the flow transformations, which takes the inputs $\mathbf{x}$ as conditioning information.

SINGLE IMAGE SUPER-RESOLUTION Single image super-resolution (SR) is the task of enhancing the resolution of an image by filling in the missing information. Transforming a low-resolution (LR) image into a high-resolution (HR) has been studied from many angles in computer vision. The simplest approaches use interpolation rather than learning, examples include Lanczos sampling (Duchon, 1979) and bicubic interpolation (Keys, 1981). Interpolation based methods, while fast and easy to implement, often perform poorly. Reconstruction-based methods, which rely on prior knowledge to limit the possible solution space, can be more flexible and generate sharper details (Dai et al., 2009; Marquina and Osher, 2008; Sun, Xu, and Shum, 2008; Yan et al., 2015). However, many reconstruction-based methods are time-consuming and can perform poorly for larger differences in scale between the LR and HR image.

Early efforts using end-to-end deep learning improved results by approaching SR as purely supervised learning problem with a feed forward convolutional neural networks and L_2 or L_1-based reconstruction loss functions (Dong et al., 2014, 2016; Kim, Lee, and Lee, 2016). The choice of loss in the traditional supervised approach seeks

a model that produces an optimal high-resolution output on *average*. By favoring the best average prediction these early efforts often produced blurry images. Changing the loss function to account for perceptual quality improves the results (Haris, Shakhnarovich, and Ukita, 2018; Yu and Porikli, 2016). Alternatively, models with adversarial losses, such as GAN (Goodfellow, Pouget-Abadie, and Mirza, 2014) based approaches produce sharper output (Ledig et al., 2017; Wang et al., 2018). Models trained with adversarial loses, however, are difficult to train, cannot estimate the likelihood of samples, and have loss functions that tend towards mode-collapsing as opposed to mode covering (Hu et al., 2018; Isola et al., 2017; Mathieu, Couprie, and LeCun, 2016) which can lead to limited diversity in outputs. The single-image super-resolution problem is ill-posed, however, since one LR image may correspond to multiple HR images (Yang et al., 2019)

6.3 PROBABILISTIC SUPER-RESOLUTION (PSR)

We start with a general formulation for the probabilistic super-resolution problem. Consider n samples $S = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, n\}$ drawn i.i.d. from an unknown joint distribution P, where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d, \mathbf{y}_i \in \mathcal{Y} \subseteq \mathbb{R}^D$. The goal is to build models for (a) the conditional distribution $Y|X$ as well as the (b) the joint distribution (X, Y). The goodness of these models will be assessed based on performance on test samples $(\mathbf{x}_{\text{test}}, \mathbf{y}_{\text{test}}) \sim P$, e.g., low conditional log-loss: $-\log p_{Y|X}(\mathbf{y}_{\text{test}}|\mathbf{x}_{\text{test}})$, or, low joint log-loss: $-\log p_{(X,Y)}(\mathbf{x}_{\text{test}}, \mathbf{y}_{\text{test}})$. In this paper, we specifically focus on the setting where $d < D$, and $X = \pi(Y)$ for some (unknown) function $\pi : \mathbb{R}^D \mapsto \mathbb{R}^d$. In this setting, modeling $p(\mathbf{y}|\mathbf{x}_{\text{test}})$ corresponds to conditional *probabilistic super-resolution* (PSR), and modeling the joint distribution $p(\mathbf{x}, \mathbf{y})$ corresponds to joint PSR.[1]

As a warm-up, we first consider a parametric and a non-parametric approach to the PSR problem. A simple parametric approach would be to consider the joint distribution (X, Y) to be a multi-variate Gaussian, i.e.,

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim N(\mu, \Sigma), \mu = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{xy}^T & \Sigma_{yy} \end{bmatrix},$$

so that the conditional distribution $p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\hat{\mu}_y, \hat{\Sigma}_y)$ with $\hat{\mu}_y = \mu_y + \Sigma_{xy}^T \Sigma_{xx}^{-1}(\mathbf{x} - \mu_x), \hat{\Sigma}_y = \Sigma_{yy} - \Sigma_{xy}^T \Sigma_{xx}^{-1} \Sigma_{xy}$. An arguably more sophisticated non-parametric model will model the joint distribution as a Gaussian Processes (GP) (Rasmussen and Williams, 2006), with the conditional distributions characterized similar to the parametric Gaussian case but using suitable mean functions and covariance functions. The parametric approach is too simple and will be insufficient for most realistic problems. The GP-based non-parametric approach is powerful, but one is left with the task of choosing the right kernel with suitable hyper-parameters.

1 The models we introduce are valid beyond this setting, but we only evaluate the models in the conditional and joint PSR setting in this paper.

6.3.1 *PSR with Normalizing Flows*

We propose modeling the joint distribution $p_{(X,Y)}(\mathbf{x},\mathbf{y})$ with Normalizing Flows (NFs) essentially using the chain rule: $p_{(X,Y)}(\mathbf{x},\mathbf{y}) = p_{Y|X}(\mathbf{y}|\mathbf{x})p_X(\mathbf{x})$. From a NF perspective, the chain rule implies that we need to construct two NFs, one corresponding to the marginal distribution over $\mathcal{X} \in \mathbb{R}^d$, and the other corresponding to the conditional distribution over $\mathcal{Y} \in \mathbb{R}^D$. For $p_X(\mathbf{x})$, we construct a NF over $\mathcal{X} \in \mathbb{R}^d$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{I}_d)$, and $p_X(\mathbf{x}) = p_\epsilon\left(g_\theta^{-1}(\mathbf{x})\right)\left|\det \frac{\partial g_\theta}{\partial \mathbf{x}}\right|$ for a suitable flow $g_\theta(\boldsymbol{\epsilon})$. For $\mathbf{x} \in \mathbb{R}^d$, we construct another NF over $\mathcal{Y} \in \mathbb{R}^D$ with $\mathbf{z} \sim \mathcal{N}(0, \mathbb{I}_D)$, and $p_{Y|X}(\mathbf{y}|\mathbf{x}) = p_Z\left(f_\phi^{-1}(\mathbf{y};\mathbf{x})\right)\left|\det \frac{\partial f_\phi}{\partial \mathbf{y}}\right|$ for a suitable flow $f_\phi(\mathbf{z};\mathbf{x})$. Then, the joint distribution is

$$\begin{aligned} p_{(X,Y)}(\mathbf{y},\mathbf{x};\boldsymbol{\phi},\theta) &= p_{Y|X}(\mathbf{y} \mid \mathbf{x},\boldsymbol{\phi}) \cdot p_X(\mathbf{x};\theta) \\ &= p_Z\left(f_\phi^{-1}(\mathbf{y};\mathbf{x})\right)\left|\det \frac{\partial f_\phi}{\partial \mathbf{y}}\right| \cdot p_\epsilon\left(g_\theta^{-1}(\mathbf{x})\right)\left|\det \frac{\partial g_\theta}{\partial \mathbf{x}}\right| . \end{aligned} \tag{6.5}$$

Here $\boldsymbol{\phi}$ and θ are respectively the parameters for the bijective flow transformations of the high-dimensional conditional $p_{Y|X}$ and the low-dimensional marginal p_X distributions. Further, p_Z is the high-dimensional (Gaussian) base distribution for $p_{Y|X}$ and p_ϵ is the low-dimensional (Gaussian) base distribution for p_X. Given samples $S = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, n\}$, the parameters $\{\boldsymbol{\phi}, \theta\}$ can be estimated by maximizing likelihood of the observations. In the conditional density estimation setting, i.e., when we only want to model $p_{Y|X}(\mathbf{y}|\mathbf{x})$, we can just focus on the conditional part of the model outlined above and estimate $\boldsymbol{\phi}$.

One advantage of specifying the joint distribution $p_{(X,Y)}$ based on two normalizing flows is that once the model is learned, we can draw samples from the model. The generative model is as follows:

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \mathcal{N}(0, \mathbb{I}_d) , & \mathbf{z} &\sim \mathcal{N}(0, \mathbb{I}_D) , \\ \mathbf{x} &= g_\theta(\boldsymbol{\epsilon}) , & \mathbf{y} &= f_\phi\left(\mathbf{z}; g_\theta(\boldsymbol{\epsilon})\right) . \end{aligned} \tag{6.6}$$

where here we define the the high and low-dimensional base distributions p_Z and p_ϵ as Gaussians $\mathcal{N}(0, \mathbb{I}_d)$ and $\mathcal{N}(0, \mathbb{I}_D)$, respectively. In general, instead of using Gaussians, one can use other suitable base distributions (Deng et al., 2020; Gemici, Rezende, and Mohamed, 2016; Kobyzev, Prince, and Brubaker, 2020; Papamakarios et al., 2019; Rasul et al., 2020; Rezende and Mohamed, 2015). Further, the modeling strategy can be seamlessly extended to model a joint distribution of more than two random variables by using the chain rule, and even consider known conditional independence structures.

With $\Gamma = \{\phi, \theta\}$, assuming each NF has K-steps, the parameters can be estimated by maximizing the likelihood, or equivalently minimizing the log-loss:

$$\mathcal{L} = \frac{1}{2n} \sum_{i=1}^{n} \left[\|f_\phi(\mathbf{y}_i; \mathbf{x}_i)\|_2^2 + \|g_\theta(\mathbf{x}_i)\|_2^2 + \sum_{k=1}^{K} \left(\log \left| \det \frac{\partial f_{\phi_k}^{-1}}{\partial \mathbf{y}_i} \right| + \log \left| \det \frac{\partial g_{\theta_k}^{-1}}{\partial \mathbf{x}_i} \right| \right) \right] . \tag{6.7}$$

For our experiments, we estimate the parameters with a ridge regularization $\alpha(\|\phi\|_2^2 + \|\theta\|_2^2$. Note that number of steps in the two NFs can be different, e.g. K_1 and K_2.

While training $p_X(\mathbf{x})$ and $p_{Y|X}(\mathbf{y}|\mathbf{x})$ separately still yields the ability to sample a novel $\mathbf{x} \sim p_X$ and super-resolve $\mathbf{x}$ into $\mathbf{y}$, there are advantages to training $p_X(\mathbf{x})$ and $p_{Y|X}(\mathbf{y}|\mathbf{x})$ jointly. Empirically, we find that performance improves as a result of conditioning f_ϕ on a learned embedding of $\mathbf{x}$, as opposed to the raw LR image $\mathbf{x}$, thus training $p_X(\mathbf{x})$ jointly with $p_{Y|X}(\mathbf{y}|\mathbf{x})$ provides an opportunity for the model to learn features that are more useful for super-resolution.

Lastly, since $\mathbf{x}$ is a downsampling of $\mathbf{y}$—or in some cases a subset of the available features, modeling the joint density $p_{(X,Y)}$ supplies the model with virtually no new information compared to an unconditional model p_Y. Thus a jointly trained PSR flow is similar to an unconditional flow from the perspective of a data model, but the architecture lends itself to certain advantages which we discuss next.

6.3.2 *Probabilistic Super-Resolution Architecture*

The architecture of our PSR flow, shown in Figure 21, incorporates techniques present in many state-of-the-art normalizing flows as well as our own modifications to improve the power of PSR flow. The general structure involves a multi-scale architecture in order to increase the receptive field available to the flow transformations and create a hierarchy within the latent space.

At each of the L scales of the model we apply a K-step sequence of transformations consisting of conditional coupling layers, 1×1 invertible convolutions to mix channels, and activation normalization (Actnorm) (Kingma and Dhariwal, 2018). We implement a conditional variant of the coupling layer flows from (Ho et al., 2019; Huang, Dinh, and Courville, 2020). We follow Ho et al. (2019) and apply variational dequantization as a transition between the discrete pixel input and the continuous latent space. Below we present unique modifications in our approach, followed by additional details on all flow components.

ENCODING THE LOW-RESOLUTION IMAGE Concatenating or adding the conditional $\mathbf{x}$ to the latent $\mathbf{z}_k$ for flow transformation f_k is a simple approach that works well for many applications of conditional normalizing flows (Lu and Huang, 2020; Winkler et al., 2019). Since the conditional $\mathbf{x}$ transfers high-level semantic information about the desired high-resolution output $\mathbf{y}$, we use a conditioning network $h(\cdot)$ to encode $\mathbf{x}$ which provides rich features to the conditional flows similar to (Ardizzone

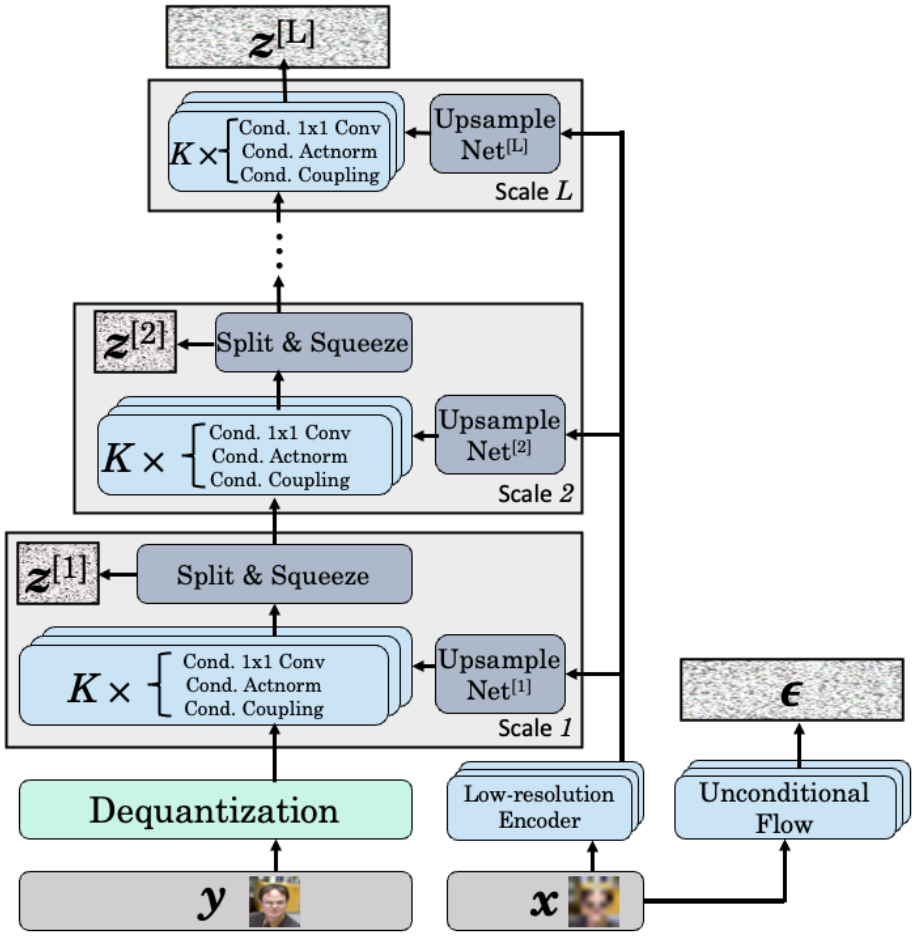


Figure 21: Joint PSR Flow architecture for learning joint distribution over both high-resolution and low-resolution images $p_{(X,Y)}(\mathbf{x}, \mathbf{y})$.

et al., 2019, 2021; Lugmayr et al., 2020). For PSR, however, the conditional encoding $h(\mathbf{x})$ is not the same shape as the high-resolution $\mathbf{y}$ and must be reshaped. While simple upsampling techniques suffice—such as bicubic interpolation as done in Lugmayr et al. (2020), we instead simultaneously train upsampling networks $u_1(\cdot), \ldots, u_L(\cdot)$ at each of the L scales of our multiscale architecture. The upsampling networks use deconvolutional layers in order to reshape the learned features $h(\mathbf{x})$ to match the dimensionality of the latent space for $\mathbf{z}_l$ at each scale $l \in 1, \ldots, L$ in the flow, which are then concatenated and passed to the coupling layers. In addition to reshaping $h(\mathbf{x})$, upsampling networks allow each scale to refine the features as needed for that particular scale. Note, the conditioning network $h(\cdot)$ does not need to be invertible, and thus we encode the low-resolution image with a powerful DenseNet architecture (Gao Huang et al., 2017).

CONDITIONAL COUPLING LAYERS The conditional flow steps at scale $l \in [1, L]$ are computed by:

$$
\begin{aligned}
\text{Encode Conditional:} \quad & \mathbf{v} = u_l(h(\mathbf{x})) \\
\text{Cond. Actnorm:} \quad & s, b = c(\mathbf{v}), \qquad \mathbf{y}_{i,j} = s \odot \mathbf{z}_{i,j} + b \\
\text{Cond. } 1 \times 1 \text{ Conv:} \quad & \mathbf{W} = c(\mathbf{v}), \qquad \mathbf{y}_{i,j} = \mathbf{W}\mathbf{z}_{i,j} \\
\text{Partition:} \quad & [\mathbf{z}^{[a]}, \mathbf{z}^{[b]}] = \mathbf{z} \\
\text{Cond. Coupling:} \quad & \mathbf{y}^{[a]} = \exp\left(2 \cdot \tanh\left(s(\mathbf{z}^{[b]}, \mathbf{v})/2\right)\right) \odot \mathbf{z}^{[a]} + t(\mathbf{z}^{[b]}, \mathbf{v}), \\
& \mathbf{y}^{[b]} = \mathbf{z}^{[b]}
\end{aligned} \tag{6.8}
$$

where h is the low-resolution feature extractor shared at all scales, and u_l upsamples the features to match the latent space at scale l using deconvolutional layers. The Partition function divides the latent space into two for the coupling layer. We use a checkerboard partitioning (Dinh, Sohl-Dickstein, and Bengio, 2017) in the first scale and divide the latent space channel-wise for all remaining scales. In the coupling layer the non-linearity $\exp(2 \cdot \tanh(\frac{s(\cdot)}{2}))$ bounds the scale $s(\cdot)$ output, which can potentially explode in longer training runs. Despite augmenting the coupling network with conditional information, the computation of the Jacobian of the transformations remains lower triangular and thus computing the change in value is computed efficiently. The Conditional Actnorm and Conditional 1×1 Convolutions in (6.8) learn a normalization and permutation of the channels based on learned features from the LR input, however it is not necessary that the same conditioning network c is applied to each operation.

MULTI-SCALE ARCHITECTURES The typical architecture for normalizing flows for modeling high-dimensional data, such as images, consists K flow steps repeated at each scale L in a multi-scale architecture. In additional to flow steps, multi-scale architectures use *squeeze* and (optionally) *slice* operations between scales in order to reshape the latent space and create a hierarchy in latent variables, respectively. The squeeze operation reshapes the inputs from $C \times H \times W$ to $4C \times \frac{H}{2} \times \frac{W}{2}$ thereby increasing the number of channels and receptive field, where C is the number of channels, H the height and W the width of the image. The *slice* operation allows only a subset of the latent variables to be passed on to deeper layers in the model in order to create a hierarchy within the latent space and reducing the total number of parameters in the model.

COUPLING LAYERS For each of the K flow steps we consider coupling layer flows, which define an easily invertible affine transformation f that has a strictly lower (or upper) triangular Jacobian matrix—and hence easy to compute to the log determinant Jacobian term in the objective. Flows like NICE (Dinh, Krueger, and Bengio, 2015), RealNVP (Dinh, Sohl-Dickstein, and Bengio, 2017), and Glow (Kingma and Dhariwal, 2018) have popularized the use of coupling layers which, despite their simplicity, are powerful transformations with many computational benefits. An affine coupling layer

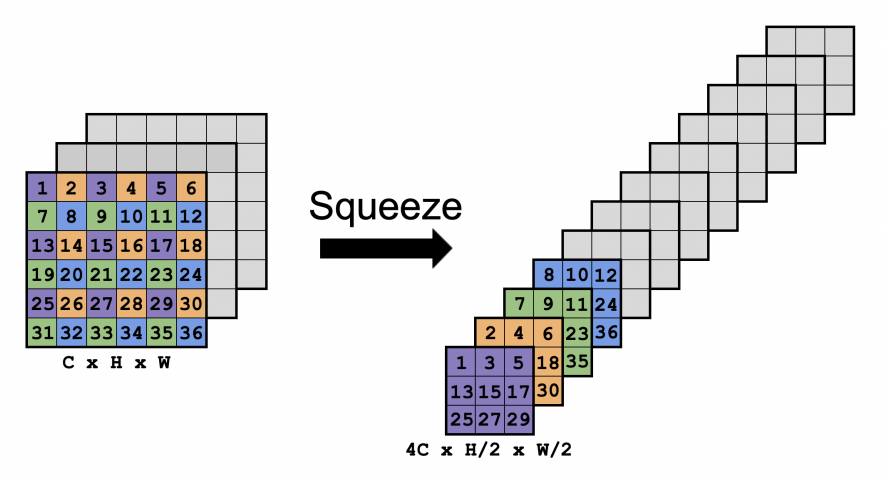


Figure 22: Squeeze operation for multi-scale normalizing flow architectures. The image is reshaped using a checkerboard pattern, similar to that of a dilated convolution, to replace spatial size for additional channels.

splits the input $\mathbf{z} = [\mathbf{z}^{[a]}, \mathbf{z}^{[b]}]$ either channel-wise or in a checkerboard fashion (Dinh, Sohl-Dickstein, and Bengio, 2017) and applies a scale $s(\cdot)$ and shift $t(\cdot)$ transformation to half the inputs, and the other half are used to learn the scale and shift function. Specifically, the forward $\mathbf{z}_1 = f(\mathbf{z}_0)$ and inverse $\mathbf{z}_0 = f^{-1}(\mathbf{z}_1)$ are shown on the left and right, respectively:

$$\begin{aligned} \mathbf{z}_1^{[a]} &= \mathbf{z}_0^{[a]} \odot s(\mathbf{z}_0^{[b]}) + t(\mathbf{z}_0^{[b]}) & \mathbf{z}_0^{[a]} &= \left(\mathbf{z}_1^{[a]} - t(\mathbf{z}_1^{[b]})\right) / s(\mathbf{z}_0^{[b]}) \\ \mathbf{z}_1^{[b]} &= \mathbf{z}_0^{[b]} & \mathbf{z}_0^{[b]} &= \mathbf{z}_1^{[b]} \\ \mathbf{z}_1 &= [\mathbf{z}_1^{[a]}, \mathbf{z}_1^{[b]}] & \mathbf{z}_0 &= [\mathbf{z}_0^{[a]}, \mathbf{z}_0^{[b]}] \end{aligned}$$

where in the final step the split latent variables are concatenated back together. As Dinh, Sohl-Dickstein, and Bengio (2017) show, the log determinant Jacobian for such coupling layers is simply the sum of s over the image dimensions, even though the $s(\cdot)$ and $t(\cdot)$ functions can be arbitrarily complex deep neural networks. In order to ensure that one half of the latent variables are not left unaltered, each flow step is preceded by an invertible 1×1 convolution which acts to permute the channels in a learnable way (Kingma and Dhariwal, 2018). Lastly, we include an activation normalization (Actnorm) layer before each step, which similar is to batch normalization, but applies a channel-wise normalization of the latent space via a learned affine transformation.

PRIOR NORMALIZING FLOW IN JOINT PSR FLOW To learn the joint distribution $p_{(X,Y)}(\mathbf{x}, \mathbf{y})$ we train an unconditional normalizing flow on $\mathbf{x}$. We use a similar coupling layers as the super-resolution flow, but do not condition on any additional information. We train unconditional flows that have many fewer parameters than the super-resolution model, using only 2 scales, each with 6 flow steps. The first scale uses checkerboard splits, whereas the second scale applies a squeeze operation and

then splits channel-wise. Unlike the super-resolution flow we do not slice off any of the latent space, and instead keep the dimensionality of each flow step the same.

6.3.3 *Inductive Biases and The Bottleneck Problem*

One challenge in designing flexible normalizing flows is the bottleneck problem (Ardizzone et al., 2020; Chen et al., 2020), which is rooted in the bijective nature of flows, requiring them to preserve dimensionality. A flow transformation may be implemented by a neural network with δ hidden units where $\delta \gg D$, but the transformation's output must have dimension D and cannot pass all the learned information to subsequent flow steps. Augmented (Huang, Dinh, and Courville, 2020) and VFlows (Chen et al., 2020) propose augmenting the data $\mathbf{y}$ with with extra dimensions $\mathbf{z}$ and learns the joint distribution over the data and latent variables. For discriminative models augmenting the input with zero-padding has a similar effect (Dupont, Doucet, and Teh, 2019; Jacobsen, Smeulders, and Oyallon, 2018).

The bottleneck problem is exasperated in coupling layer flows, which learn their transformation from only half of the inputs in order to modify the remaining inputs. Kirichenko, Izmailov, and Wilson (2020) shows that transformations produced by coupling layer flows are biased towards learning local pixel correlations, as opposed to encoding high-level semantic information.

The PSR architecture offers an alternate approach to overcoming the bottleneck problem as well as the inductive bias towards learning local pixel correlations. In order for PSR flows to generate faithful representations of the ground-truth $\mathbf{y}$, the LR encoding $h(\mathbf{x})$ must capture the appropriate structure—such as color patterns or the shapes and locations of objects. The conditional flow transformations, on the other hand, can fill in the missing information which is requires more expertise in learning low-level information.

Our architecture is designed with this hypothesis in mind, and we view the low-resolution encoding and upsampling networks $u_l(h(\mathbf{x}))$ as mimicking a residual connection (He et al., 2015), which ensures that each flow transformation does not need to propagate high-level information. Thus, a PSR architecture naturally separates the job of capturing the high-level structure from the task of filling in the missing information. In our experiments we evaluate these hypotheses by examining how the generated samples vary due to changes in the low-resolution encoding $h(\mathbf{x})$ versus the variations caused by sampling different points within the base distribution $\mathbf{z} \sim p_Z$.

6.4 LEARNING A DISTRIBUTION OVER OUTPUTS

Despite producing output images with a high perceptual quality, prior work using feed-forward networks treat the SR problem as an inherently one-to-one—that is, that for each low-resolution image there is a single best high-resolution output that the model should seek to recover. A probabilistic treatment of the problem based on nor-

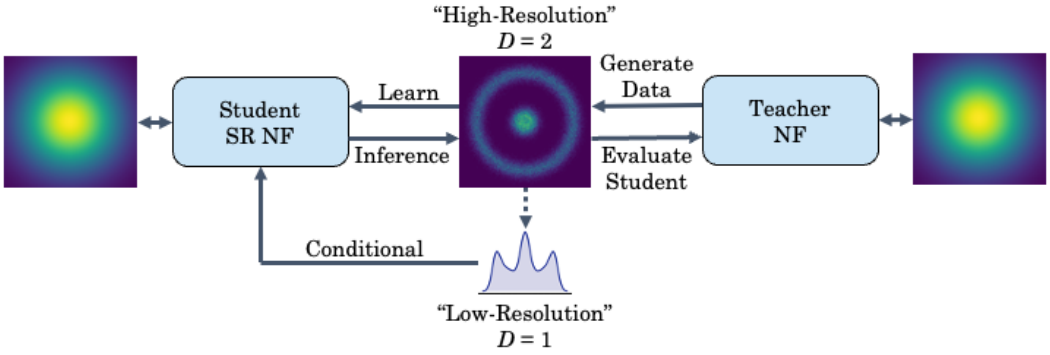


Figure 23: Student teacher setup with a super-resolution flow shown as the student. By using a NF teacher, which uses absolute-value surjective flows (Nielsen et al., 2020) to precisely model specific densities, we generate training data and evaluate probabilistic student models in a highly controlled setting.

malizing flows, such as in Ardizzone et al. (2021) and Lugmayr et al. (2020), posits that there is a *distribution* of high-resolution outputs which are all valid. As we consider greater disparities in resolution between the LR and HR image, there is a wider opportunity for producing HR images that are consistent with the LR input.

To better study the distribution of HR images learned by normalizing flows we propose a student-teacher evaluation setup. We construct a teacher normalizing flow that is specifically designed to perfectly represent a challenging toy problem. For example, a teacher model equipped with absolute-value surjective flows (Nielsen et al., 2020) can perfectly represent densities with a symmetrical structure (see Figure 23). Because the teacher $p_{\mathcal{T}}(\mathbf{y})$ is a normalizing flow it can generate samples $\mathbf{y} \in \mathbb{R}_D$ and give their exact likelihood. The student normalizing flow $p_{\mathcal{S}}(\mathbf{y} \mid \mathbf{x})$ has a super-resolution architecture but is not hand-crafted for the particular problem and instead relies on commonly used bijective coupling layers which are conditioned on a coarsened subset of the available features $\mathbf{x} \in \mathbb{R}_d$. The student PSR normalizing flow then trains on pairs $\{\mathbf{y}_i, \mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x} = \pi(\mathbf{y})$ deterministically coarsens and reduces the dimensionality of $\mathbf{y}$. The objective of the student is then to minimize $\mathrm{KL}(p_{\mathcal{T}}(\mathbf{y}) \| p_{\mathcal{S}}(\mathbf{y} \mid \mathbf{x}))$, which yields a maximum likelihood objective.

Our goal is to first compare the student flow to other conditional probabilistic models, such as a concrete dropout network (Gal, Hron, and Kendall, 2017) and Gaussian Process. In Table 11 we report the earth mover's distance (Rubner, Tomasi, and Guibas, 2000), also known as Wasserstein distance, as a measure of the distance between the true and generated distributions. In Table 11 we also report the traditional super-resolution metric RMSE. As we show, however, metrics based on mean-squared-error can be misleading when the output distribution is multi-model—as is the case in the toy distributions we experiment on. For Table 11 we also show results for both high and low temperature samples from the PSR Flow, showing that at the lowest temperature PSR Flow performs more closely to the baseline models on RMSE.

In Figure 24 we use the student-teacher setup to examine the learned distribution of the student flow for every possible conditional input $\mathbf{x}$ as well as the noise $\mathbf{z} \sim \mathcal{N}(0, \mathbb{I}_D)$ drawn from the student flow's base distribution. When the output distribution is multi-

Table 11: Root mean-squared error (RMSE, the lower the better) and Earth Mover's Distance (EMD, the lower the better) of student normalizing flow, concrete dropout (Gal, Hron, and Kendall, 2017), and Gaussian process (GP) with a Matern kernel on toy datasets. Samples $\{\mathbf{x}_i, \mathbf{y}_i\}_i^n$ are first generated by the Teacher to compute RMSE, whereas EMD computes transport distance between the Teacher and distributions learned by the Students. The output $\mathbf{y}$ is 2-dimensional and input $\mathbf{x}$ is a quantized version of $\mathbf{y}_1$. Despite being probabilistic, the Gaussian process and concrete dropout do not generate multi-modal output distributions. Flow models easily model the output distribution when sampling noise from the full latent space ($\tau = 1.0$), and perform comparably on RMSE when only sampling from the highest density region of the latent space ($\tau = 0.0$).

Model	**Circle & Cluster**		**Two Moons**		**Four Circles**		**Abs**	
	RMSE↓	EMD↓	RMSE↓	EMD↓	RMSE↓	EMD↓	RMSE↓	EMD↓
Dropout	1.38	1.59	0.58	0.46	**1.23**	1.40	0.21	0.05
GP	**1.37**	1.58	**0.56**	0.41	**1.23**	1.38	**0.16**	0.05
PSR NF τ=0	1.41	1.38	0.65	0.25	**1.23**	1.37	**0.16**	0.05
PSR NF τ=0.8	1.96	**0.05**	0.80	**0.02**	1.72	**0.05**	0.23	**0.01**

modal the flexibility of the PSR Flow show and it is generates an output distribution that closely matches the ground truth. In the final row, which whose the absolute-value function and is not multi-modal, all model perform similarly.

We compare the likelihoods learned by the student PSR Flow and the true likelihoods (as given by the teacher normalizing flow). Figure 25 plots the likelihoods for the student against the teacher likelihoods for samples generated by the teacher (left) as well as for samples generated by the student (right). Not only are likelihoods for the conditional student model highly correlated with the ground truth likelihoods given by the teacher, but we see that this correlation occurs for both "real" data generated by the teacher (left) as well for samples generated by the student (right). To generate samples from the student we super-resolve all possible "low-resolution" inputs, which we can then compute likelihood for according to either the student or teacher.

6.5 EXPERIMENTS

In our experiments we examine the performance of our PSR Flow quantitatively with traditional super-resolution metrics and qualitatively with samples and measures of visual fidelity. We report Peak Signal to Noise Ratio (PSNR) but, as our Student Teacher evaluation shows, MSE-based metrics do not measure fitting a distribution (nor visual quality). Since our main motivation is to learn a distribution over the outputs we further assess our models using measures of perceptual quality—such as Structural Similarity Index (SSIM), Learned Perceptual Image Patch Similarity (LPIPS, (Zhang et al., 2018)), pixel diversity, and Fréchet Inception Distance (FID, (Heusel et al., 2018)). Further, we qualitatively evaluate samples generated by PSR Flows and survey the variation in SR images resulting from sampling from various points in the latent spaces.

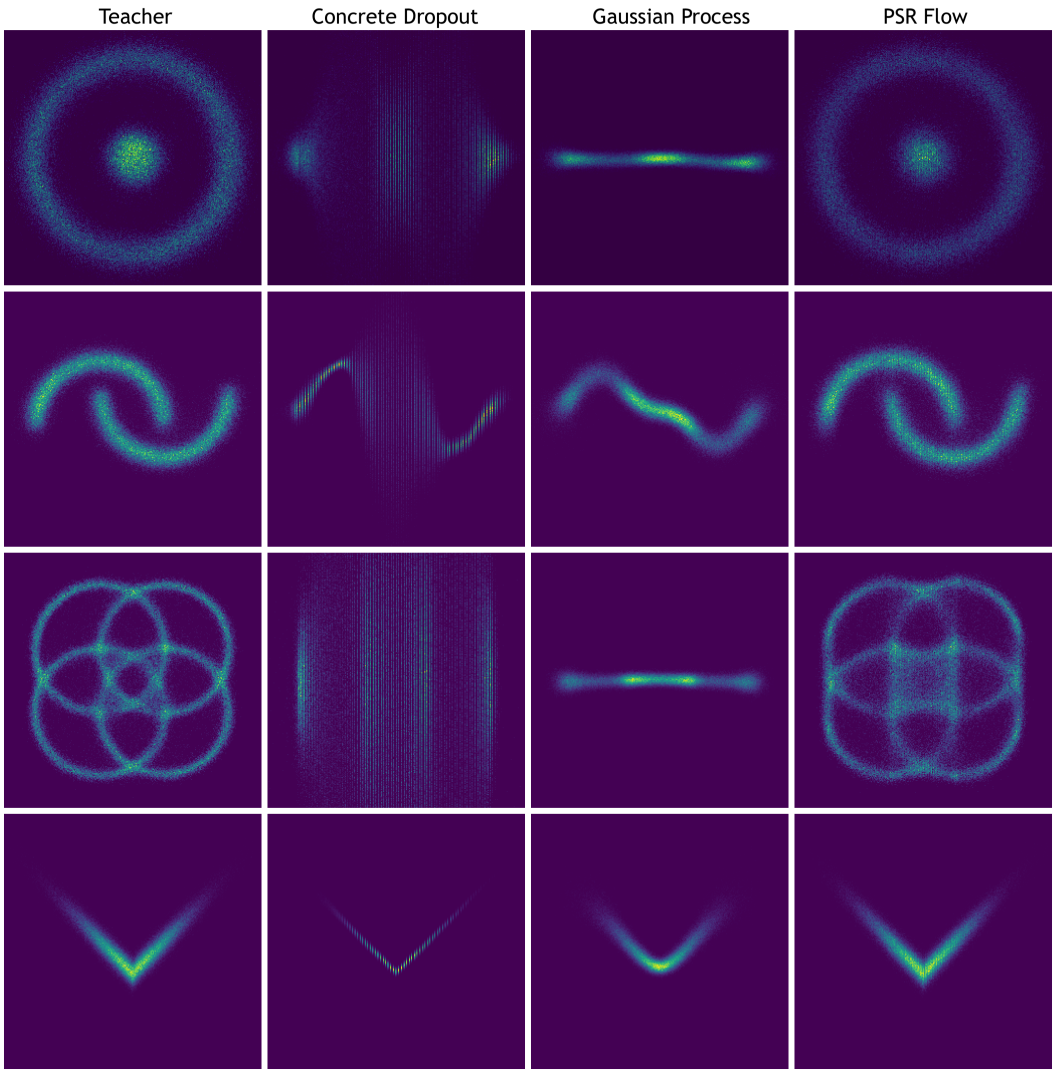


Figure 24: Samples from learned distribution of Concrete Dropout (Gal, Hron, and Kendall, 2017), Gaussian Process with Matern Kernel, and a PSR normalizing flow. The output samples $\mathbf{y}$ are 2-dimensional and predicted given input $\mathbf{x}$ which is a quantized version of $\mathbf{y}_1$. Flow models can learn complex, multi-modal conditional distributions, while still retaining their probabilistic advantages.

6.5.1 *Experimental Setup*

DATASETS For image modeling results we experiment on seven datasets. `MNIST`[2] contains 60,000 and 10,000 images pre-split into training and test sets, each of which is a 28×28 grayscale image and portrays a hand-written digit (LeCun et al., 1998). `CIFAR-10` consists of 60,000 32×32 color images featuring 10 classes of things, with 50,000 and 10,000 images pre-split into the training and test sets (Krizhevsky, 2009), respectively. The Street View House Numbers dataset `SVHN` contains 73,257 training and 26,032 images of house numbers of size 32×32. The `ImageNet` 32×32 and 64×64 datasets also come pre-split with 1,281,149 training images and a validation set of 50,000 images which we use as a test set (Deng et al., 2009; Oord, Kalchbrenner, and

2 http://yann.lecun.com/exdb/mnist/

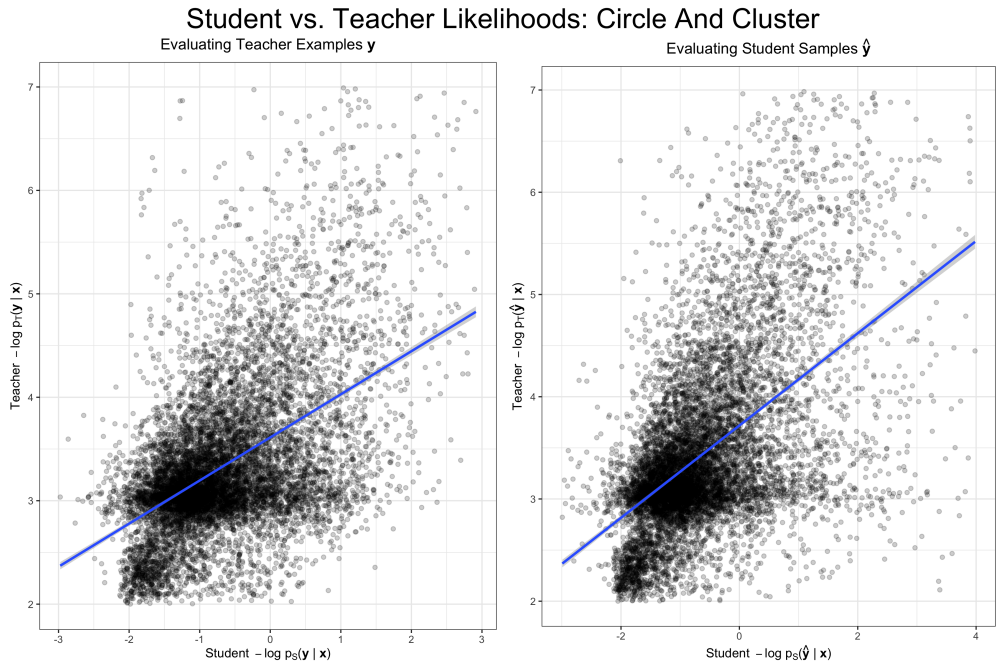


Figure 25: Using student teacher evaluation we compare the learned likelihoods of the student PSR flow to the true likelihoods. Likelihoods for student and teacher are compared using (1) ground truth samples generated by the teacher (*left*), and (2) samples generated by the student (*right*). For both teacher and student generated samples the conditional likelihoods learned by the student correlate with the true likelihoods given by the teacher.

Kavukcuoglu, 2016). We train on the Celebrity Face Attributes texttttCelebA[3] dataset consisting of 162,770 training and 19,962 test images at the resolution 64×64 and 128×128 (Liu et al., 2015). To evaluate super-resolution results we also test our models on standard benchmark datasets: Set5 (Bevilacqua et al., 2012) and Set14 (Zeyde, Elad, and Protter, 2010) by taking random crops of size 64×64.

IMPLEMENTATION DETAILS For image modeling we implement our PSR flows using $L = 3$ scales each with $K = 4$ flow steps. We slice off a portion of the latent space after each scale, leaving only half of the original dimensions in the final scale. The low-resolution encoder uses 2 DenseNet blocks, and each upsampling networks uses 2 or 3 deconvolutional layers depending on the size of HR image. During training models minimizing negative log-likelihood using data pairs $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ is a coarse nearest neighbors downsampling of $\mathbf{y}$. For joint PSR models, the prior p_X uses $L = 2$ and $K = 6$ but each coupling layer has significantly fewer parameters than the conditional PSR flow. As an optimizer we use Adam (Kingma and Ba, 2015) with a linear warmup and then exponentially decay the learning rate starting at $1 \cdot 10^{-3}$. Depending on the dataset, our conditional PSR flow takes 3-4 days to train on a single NVIDIA V100 GPU.

In Table 12 we list architecture details for models trained on 32×32, 64×64, and 128×128 image datasets. In general we follow the architecture and hyperparameter

3 http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

settings[4] used in Chen et al. (2020) and Ho et al. (2019), except we simplify the architecture by only using checkerboard splits in the first scale and channel-wise splits in the remainder, all conditional PSR Flow modules are conditioned on the LR input that has been passed through the LR Encoder and Upsample Net at that scale. Following Chen et al. (2020) the augmented flow layer increases the number of channels by 3. After each scale we squeeze the latent space to increase the number of channels and then slice off a portion of the latent such that the final, deepest latent representation is half the size of the original latent space.

The flow modules listed in Table 12 include conditional flow steps f_{checker} and f_{checker}, as well as unconditional flow steps g_{checker} and g_{checker}. For ease of notation, we denote conditional inverse flows, which are used for variational dequantization and augmentation layers by f'_{checker}, which contain the same flow steps as the stand conditional flow but without an attention mechanism.

Each flow step follows Ho et al. (2019) with three types of invertible transformations: activation normalization **ActNorm** (Kingma and Dhariwal, 2018), invertible 1×1 convolution **Pointwise** (Kingma and Dhariwal, 2018), and then a mixture-of-logistic attention coupling layer **MixLogisticAttnCoupling** (Ho et al., 2019). For the conditional PSR Flow we include conditional variants **CondActNorm**, **CondPointwise**, and **CondMixLogisticAttnCoupling**. The coupling layers are defined by the number of hidden layers B, number of filters D_H, and number of logistic mixture components K. Each coupling layer can either split the input based on a **CheckerboardSplit** in the spatial dimensions of the images, or a **ChannelSplit** partition performed on the channel dimension. The squeezing operation Dinh, Sohl-Dickstein, and Bengio, 2017, denoted **SpaceToDepth** and described in Figure 22, exchanges spatial dimensions for additional channels. Conditional distributions, including augmentation and dequantization, are implemented[5] following Chen et al. (2020). Let **TupleFlip** denote flipping the two split inputs to encourage mixing, and **Inverse(·)** as the inverse transformation of the channel or checkerboard split. Finally, **DenseNet**(B, D_H) denotes the DenseNet low-resolution encoder, and **Deconv** shows the number of filters in each layer of a deconvolutional upsampling network used at each scale.

4 Our hyperparameter choices are taken from the official releases available at: https://github.com/aravindsrinivas/flowpp.

5 We use the parameter settings for official results provided in https://github.com/thu-ml/vflow.

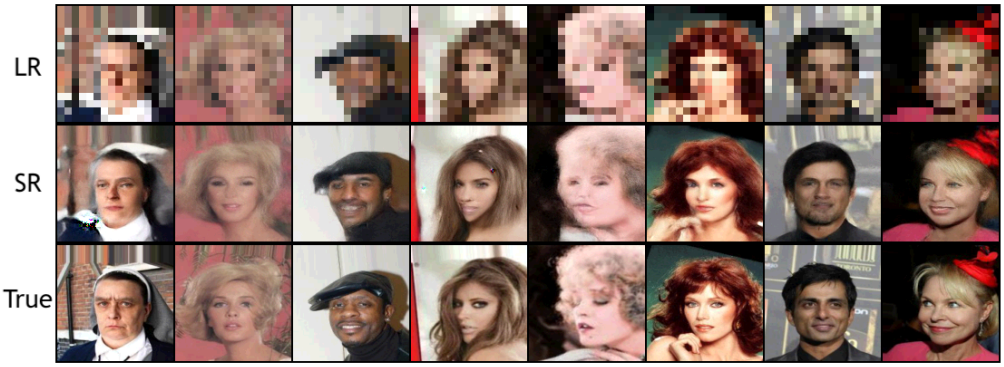


Figure 26: Super-resolution on random sample of CelebA 128×128 images. *Top row:* Low-resolution images are 16×16 pixels (compressed $8\times$). *Middle row:* Super-resolved images using temperature $\tau = 0.8$. *Bottom row:* Ground truth images from test set.

$$\begin{aligned} f_{\text{checker}}(B, D_H, K) = \text{CondActNorm} &\longrightarrow \text{CondPointwise} \longrightarrow \text{CheckerboardSplit} \\ &\longrightarrow \text{CondMixLogisticAttnCoupling}(B, D_H, K) \longrightarrow \text{TupleFlip} \\ &\longrightarrow \text{Inverse(CheckerboardSplit)} \end{aligned}$$

$$\begin{aligned} f_{\text{channel}}(B, D_H, K) = \text{CondActNorm} &\longrightarrow \text{CondPointwise} \longrightarrow \text{ChannelSplit} \\ &\longrightarrow \text{CondMixLogisticAttnCoupling}(B, D_H, K) \longrightarrow \text{TupleFlip} \\ &\longrightarrow \text{Inverse(ChannelSplit)} \end{aligned}$$

$$\begin{aligned} g_{\text{checker}}(B, D_H, K) = \text{ActNorm} &\longrightarrow \text{Pointwise} \longrightarrow \text{CheckerboardSplit} \\ &\longrightarrow \text{MixLogisticAttnCoupling}(B, D_H, K) \longrightarrow \text{TupleFlip} \\ &\longrightarrow \text{Inverse(CheckerboardSplit)} \end{aligned}$$

$$\begin{aligned} g_{\text{channel}}(B, D_H, K) = \text{ActNorm} &\longrightarrow \text{Pointwise} \longrightarrow \text{ChannelSplit} \\ &\longrightarrow \text{MixLogisticAttnCoupling}(B, D_H, K) \longrightarrow \text{TupleFlip} \\ &\longrightarrow \text{Inverse(ChannelSplit)} \end{aligned}$$

6.5.2 *Generating High-Fidelity Super Resolution Images*

In Figure 26 we show qualitative results for our PSR flow on 128×128 images from the test set of CelebA. The low-resolution conditional input is $8\times$ smaller than the original high-resolution image. Super-resolution images are sampled at a temperature of $\tau = 0.8$ and show high-fidelity results for a variety of faces and angles. Our results show super-resolution images that are faithful representations of the ground truth, but

Table 12: Conditional PSR Flow architectures for $D \times D$ image datasets, where $D = 32, 64$, and 128. Prior PSR Flow architectures shown for training on the corresponding $d \times d$ low-resolution images, where $d = D/scale$ where $scale$ is either the $4\times$ or $8\times$ super-resolution scaling. The number of channels C in the latent space is not shown, however each SpaceToDepth operation results in 4C channels in the output.

Flow Module	**32×32 Images**	**64×64 Images**	**128×128 Images**
	Conditional PSR Flow Architecture $p_{Y\|X}(\mathbf{y} \mid \mathbf{x})$		
Dequantize	$f'_{\text{checker}}(2, 96, 4)$ $\times 4$ Sigmoid	$f'_{\text{checker}}(4, 96, 4)$ $\times 4$ Sigmoid	$f'_{\text{checker}}(4, 96, 4)$ $\times 4$ Sigmoid
		SpaceToDepth	SpaceToDepth $\times 2$
Augment	$f'_{\text{checker}}(2, 96, 4)$ $\times 4$ Sigmoid	$f'_{\text{checker}}(4, 96, 4)$ $\times 4$ Sigmoid	$f'_{\text{checker}}(4, 96, 4)$ $\times 4$ Sigmoid
L1: 32×32	$f_{\text{checker}}(12, 96, 4)$ $\times 4$	$f_{\text{checker}}(12, 96, 4)$ $\times 4$	$f_{\text{checker}}(12, 96, 4)$ $\times 4$
	SpaceToDepth Slice 25%	SpaceToDepth Slice 25%	SpaceToDepth Slice 25%
L2: 16×16	$f_{\text{channel}}(12, 96, 4)$ $\times 4$	$f_{\text{channel}}(12, 96, 4)$ $\times 4$	$f_{\text{channel}}(12, 96, 4)$ $\times 4$
	SpaceToDepth Slice 33%	SpaceToDepth Slice 33%	SpaceToDepth Slice 33%
L3: 8×8	$f_{\text{channel}}(12, 96, 4)$ $\times 4$	$f_{\text{channel}}(12, 96, 4)$ $\times 4$	$f_{\text{channel}}(12, 96, 4)$ $\times 4$
LR Encoder	DenseNet(2, 32)	DenseNet(2, 64)	DenseNet(2, 64)
Upsample Net_l	Deconv 32, 64	Deconv 64, 128	Deconv $64^{(2)}, 128^{(2)}$
Parameters	44M	54M	66M
	Prior PSR Flow Architecture $p_X(\mathbf{x})$		
Dequantize	$f'_{\text{checker}}(2, 64, 4)$ $\times 4$ Sigmoid	$f'_{\text{checker}}(2, 64, 4)$ $\times 4$ Sigmoid	$f'_{\text{checker}}(2, 64, 4)$ $\times 4$ Sigmoid
L1: $d \times d$	$g_{\text{checker}}(8, 64, 4)$ $\times 4$	$g_{\text{checker}}(8, 96, 4)$ $\times 4$	$g_{\text{checker}}(8, 96, 4)$ $\times 4$
	SpaceToDepth	SpaceToDepth	SpaceToDepth
L2: $\frac{d}{2} \times \frac{d}{2}$	$g_{\text{channel}}(8, 64, 4)$ $\times 4$	$g_{\text{channel}}(8, 96, 4)$ $\times 4$	$g_{\text{channel}}(8, 96, 4)$ $\times 4$
Parameters	15M ($4\times$ SR)	34M ($8\times$ SR)	34M ($8\times$ SR)

Figure 27: Interpolating across the latent space for super-resolved images from CelebA 64 × 64. *Left:* Low-resolution conditional input image which has been down sampled by a factor of 4× or 8× (16 × 16 or 8 × 8 pixels, respectively). *Right:* Generated samples of super-resolved image by interpolating across the latent space. The first and final generated samples are taken from a high temperature of 1.25, and begins to show artifacts in the image. The center-most generated samples are taken with a temperature of 0.0, resulting in some blurriness. We sample from two tails across the latent space to show variety in the super resolved image as a result of sampling from various points within the latent space. *Left:* True high-resolution image. The 8× super-resolution model shows a higher variety in samples as a result of needing to fill in more missing information.

with facial features that differ subtly—such as different hairstyle, eyes, smiles, shape, and expressions. Given that these specific facial features are indiscernible in the low-resolution input, these results indicate that the model is able to learn such features and blends them in during super-resolution.

PSR flows trained on ImageNet64 performs better at both 4× and 8× scales than bicubic and nearest neighbors interpolation baselines. The fidelity oriented metrics reported in Table 13 on the test set of ImageNet64 as well as images from the Set5 and Set14 show that PSR flow generalizes well to new data. Samples generated at low temperatures tend to perform better on PSNR, but sampling from the full distribution yields better scores on visual metrics with only a minor decrease in PSNR. Compared to unconditional generative models, PSR flow generates images with a high visual fidelity, as shown by the FID scores in Table 14 for MNIST, CIFAR-10, and CelebA.

6.5.3 *Diversity in Super-Resolution Faces*

One of our primary goals is to better understand the factors that control diversity of super-resolved images. To that end, in Figure 27 we show SR images at a range of sampling temperatures and scaling factors. Low-resolution inputs are super-resolved from 16 × 16 and 8 × 8 pixels up to 64 × 64. Samples are generated by drawing noise $\mathbf{z}$ interpolated along a randomly chosen vector between two opposite tails of the latent space ($\tau = 1.25$) passing through the origin ($\tau = 0.0$). Sampling from the outer edges of the latent space ($\tau = 1.25$) produces sharper features but also corruption in the super-resolved image. Unsurprisingly, models that produce 8× super-resolution show more

Table 13: Results for SR models trained on ImageNet 64×64. The LR input for each model is computed using a nearest neighbors coarsening. Evaluation on the Set5 and Set14 datasets is averaged over 100 random crops of 64×64 of each image. At low temperatures ($\tau = 0$) the PSR flow performs better as traditional MSE-based metrics like PSNR, however better perceptual quality and variation in outputs occurs when sampling from the full latent space.

Model	Scale	ImageNet64 PSNR↑/SSIM↑/LPIPS↓	Set5 PSNR↑/SSIM↑/LPIPS↓	Set14 PSNR↑/SSIM↑/LPIPS↓
Nearest	4×	16.8 / 0.50 / 0.18	21.9 / 0.68 / 0.16	20.2 / 0.54 / 0.16
Bicubic	4×	17.8 / 0.55 / 0.26	22.8 / 0.72 / 0.16	22.9 / 0.60 / 0.19
PSR Flow τ=0	4×	**20.4** / **0.69** / 0.12	**28.0** / **0.84** / 0.07	**25.0** / **0.71** / 0.10
PSR Flow τ=0.8	4×	20.0 / 0.67 / **0.11**	27.5 / 0.83 / **0.06**	24.8 / 0.69 / **0.09**
Nearest	8×	13.9 / 0.26 / 0.31	18.5 / 0.46 / 0.27	18.0 / 0.34 / 0.27
Bicubic	8×	14.8 / 0.30 / 0.42	19.0 / 0.49 / 0.29	20.6 / 0.41 / 0.31
PSR Flow τ=0	8×	**17.2** / **0.44** / 0.25	**23.5** / **0.66** / 0.15	**22.4** / **0.51** / 0.21
PSR Flow τ=0.8	8×	16.8 / 0.42 / **0.22**	22.7 / 0.64 / **0.14**	21.5 / 0.49 / **0.18**

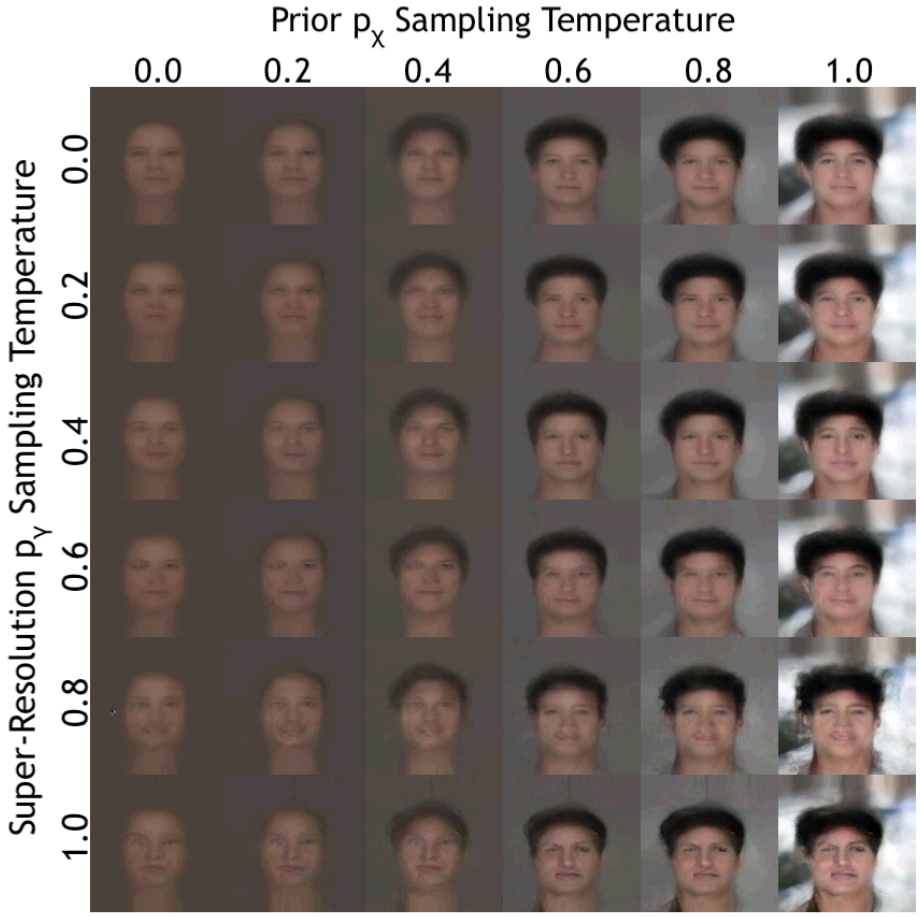


Figure 28: Samples from joint 8× PSR flow. High-level structure in the image is determined by the temperature of prior flow, whereas the temperature of the conditional flow controls fine details.

Table 14: FID score (lower is better) comparisons of GANs and VAE-based models. Top two results in bold. All results except PSR Flow taken from Xiao, Yan, and Amit (2019). The PSR flow performs 4× super-resolution for MNIST and CIFAR-10, and 8× super-resolution for CelebA.

Model	**MNIST**	**CIFAR-10**	**CelebA**
MM GAN	9.8	72.7	65.6
NS GAN	6.8	58.5	55.0
LSGAN	7.8	87.1	53.9
DRAGAN	7.6	69.8	42.3
WGAN	**6.7**	**55.2**	41.3
BEGAN	13.1	71.4	**38.9**
VAE	28.2	142.5	71.0
WAE-GAN	12.4	93.1	66.5
RAE + GMM	10.8	91.6	57.8
VAE+flow prior	28.3	110.4	54.3
GLF	8.2	88.3	53.2
PSR Flow $\tau = 0.8$	**2.2**	**55.6**	**32.8**

variety, and even showing a transition from more masculine to feminine features in some instances. However, the 8× super-resolution results tend to have softer features without as many precise details.

The conditional LR input plays an important role in determining the high-level structure of a SR image. To demonstrate, Figure 28 shows an image generated by a jointly trained PSR flow. Stochasticity in the super-resolution samples depends on the temperature of noise sampled from both the prior p_X and the conditional flow $p_{Y|X}$. The prior p_X maps noise $\boldsymbol{\epsilon}$ to a low-resolution image $\mathbf{x}$, and the PSR flow combines noise $\mathbf{z}$ and $\mathbf{x}$ to generate the samples $\mathbf{y}$ shown. The temperature of the prior p_X and conditional $p_{Y|X}$ are steadily increased from zero to one. By moving along a vector to higher temperatures in the prior's latent space (moving left to right) the main features of the image become more clear—facial structure and color patterns emerge. On the other hand, moving along a vector to higher temperatures in the conditional flow's latent space (moving top to bottom) results in subtle variations of the facial features—such as smiling versus frowning. For both the prior p_X and conditional flows $p_{Y|X}$ high sampling temperatures result in more crisp details in the generated images, but too high of temperatures can cause poor results.

In Table 15 we report bits-per-dimension for various unconditional normalizing flow, models with auto-regressive components, and our conditional PSR Flows. Bits-per-dimension is defined as $= -\mathcal{L}/(\dim(\mathbf{x}) \cdot \log 2)$, where $\dim(\mathbf{x})$ is the number of pixels in the image (Papamakarios, Pavlakou, and Murray, 2017). We emphasize that the bits-per-dimension of unconditional and conditional models is not directly comparable, however probabilistic approaches towards super-resolution is relatively new and like-

Table 15: Bits-per-dim estimates of standard benchmarks (the lower the better). Results of Flow++, MaCow, and ANF are models that employ variational dequantization instead of uniform noise injection. The Super-Resolution models are optimizing a conditional likelihood and hence not directly comparable to the unconditional models.

Model	MNIST	CIFAR 10	ImageNet 32	ImageNet 64
Autoregressive Components				
VAE + IAF (Kingma et al., 2016)	–	3.11	–	–
PixelCNN (Oord, Kalchbrenner, and Kavukcuoglu, 2016)	–	3.14	–	–
PixelCNN (multi-scale) (Reed et al., 2017)	–	–	3.95	3.70
PixelSNAIL (Chen et al., 2017b)	–	**2.85**	3.80	–
SPN (Menick and Kalchbrenner, 2018)	–	–	**3.79**	**3.52**
Flow-based				
Real NVP (Dinh, Sohl-Dickstein, and Bengio, 2017)	1.06	3.49	4.28	3.98
Glow (Kingma and Dhariwal, 2018)	1.05	3.35	4.09	3.81
FFJORD (Grathwohl et al., 2018)	0.99	3.40	–	–
Residual (Chen et al., 2019)	0.97	3.28	4.01	3.76
Flow++ (Ho et al., 2019)	–	3.09	3.86	3.69
MaCow (Ma et al., 2019)	–	3.16	–	3.69
ANF (Huang, Dinh, and Courville, 2020)	**0.93**	3.05	3.92	**3.66**
VFlow (Huang, Dinh, and Courville, 2020)	–	**2.98**	**3.83**	**3.66**
Wavelet Flow (Yu, Derpanis, and Brubaker, 2020)	–	–	4.08	3.78
Super-Resolution (Conditional)				
PSR Flow 4× (ours)	**0.78**	**2.82**	**3.79**	**3.26**
PSR Flow 8× (ours)	–	–	–	3.34

lihoods for conditional super-resolution models is not widely reported—we therefore include results from unconditional models as a rough reference point.

In Table 16 we report common super-resolution metrics for the CelebA dataset. Results for models using a bicubic LR images taken from Lugmayr et al. (2020). We train models on nearest neighbors low-resolution images, which are much more coarse than images down-sampled with bicubic interpolation, because taking only a subset of the available pixels (or features) more closely aligns with our approach of applying super-resolution architectures to general conditional density estimation tasks. Using nearest neighbors low-resolution conditionals results in poorer performance for equivalent baselines models. Our results show that PSR Flows perform worse on traditional SR metrics like PSNR, which is in-line with the drop in performance of the baseline bicubic interpolation when trained on nearest neighbors LR images, but we achieve better LPIPS (measuring visual quality) and greater diversity of outputs.

For simple datasets, such as the MNIST hand-written digits in Figure 29, we see that even LR images that are difficult to decipher can be super-resolved into sharp, plausible HR images. The examples shown in Figure 29 are specifically selected because the LR is vague, and accordingly we see that the distribution over possible SR includes multiple digits. For example, in the forth row the digit "3" is not clearly visible in the LR image, and hence the PSR Flow distribution over SR images includes both the digits "5" and "3" with a smooth interpolation between them.

Table 16: Super-resolution performance on CelebA benchmark dataset. For CelebA 128 we achieved a bits-per-dim for 5-bit color images of 1.63, and for 8-bit colors the bits-per-dim is 1.87, with both using 8× super-resolution.

LR	Model	↑ PSNR	↑ SSIM	↓ LPIPS	↑ LR-PSNR	↑ Diversity σ
Bicubic	Bicubic	23.2	0.63	0.52	35.2	0
	RRDB (Wang et al., 2018)	26.6	0.77	0.23	48.2	0
	ESRGAN (Wang et al., 2018)	22.9	0.63	0.12	34.0	0
	SRFlow (Lugmayr et al., 2020) $\tau = 0.8$	25.2	0.71	0.11	50.9	5.2
Nearest	Nearest Neighbor	15.6	0.40	0.36	100.0	0
	Bicubic	16.7	0.46	0.32	19.1	0
	PSR Flow τ=0.8 (ours)	21.1	0.69	0.10	53.5	12.5

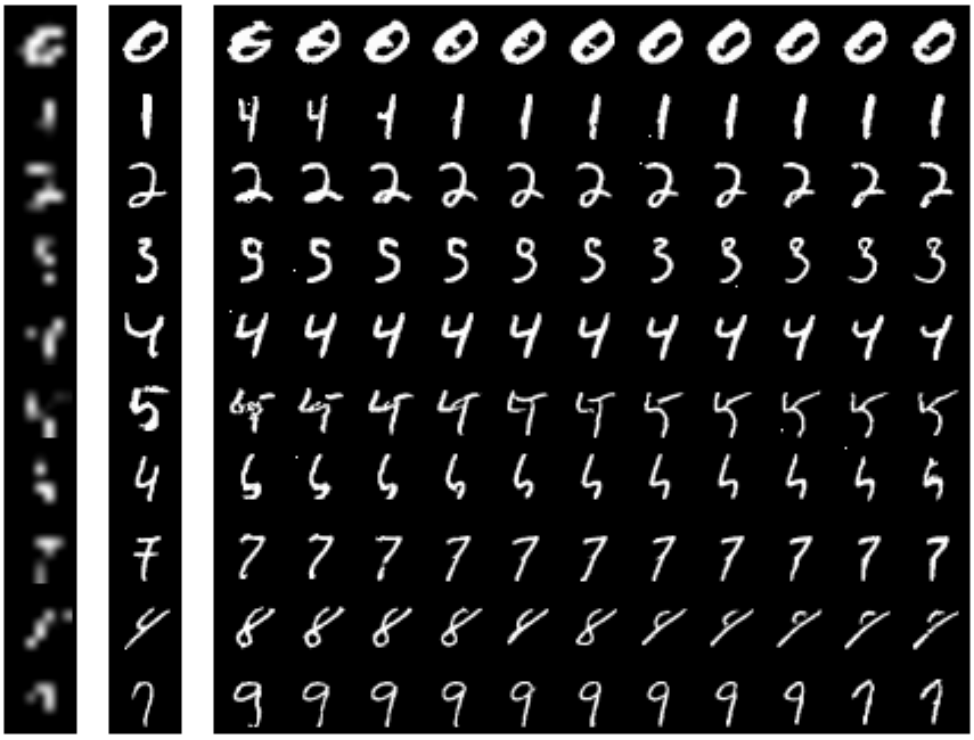


Figure 29: Interpolating across the latent space for super-resolved images from MNIST. *Left:* Low-resolution conditional input image which has been down sampled to 7×7 pixels. *Middle:* True high-resolution image is 28×28 pixels. *Right:* Generated samples of super-resolved image by interpolating along a vector between two tails of the latent space and passing through the origin. The first and final generated samples are taken from a high temperature of 1.4. The center-most generated samples are taken with a temperature of 0.0. We sample from two tails across the latent space to show variety in the super resolved image as a result of sampling from various points within the latent space. In most instances the super-resolution flow produces a faithful representation of the true high-resolution image, however these examples have been selected to show how the super-resolution model behaves when the low-resolution image is particularly uninformative.

Figure 30: Interpolating across the latent space for super-resolved images from SVHN. *Left:* Low-resolution conditional input image which has been down sampled to 8×8 pixels. *Middle:* True high-resolution image is 32×32 pixels. *Right:* Generated samples of super-resolved image by interpolating across the latent space. The first and final generated samples are taken from a high temperature of 1.4. The center-most generated samples are taken with a temperature of 0.0. We sample from two tails across the latent space to show variety in the super resolved image as a result of sampling from various points within the latent space. In most instances the super-resolution flow produces a faithful representation of the true high-resolution image, however these examples have been selected to show how the super-resolution model behaves when the low-resolution image is particularly uninformative.

In Figure 30 we perform the same analysis on the Street View House Numbers (SVHN) dataset. Here the LR images 8×8 pixels and the ground truth HR image is 32×32 pixels. Despite the additional information in the images relative to MNIST, the PSR Flow shows super-resolved outputs that are faithful to the ground truth. Similar to the MNIST examples, in instances where the LR input is vague the SR output correctly shows multiple possible digits. For example in the final row the SR output shows SR outputs that include "0", "6", and "5".

In Figure 31 we show super-resolution for samples from the CIFAR-10 dataset. We begin with very low-resolution inputs of 8×8 pixels and super-resolve to generate 32×32 images. Because the low-resolution input is extremely course, there is a lot of missing information to fill in. At the highest temperatures the super-resolved images begin to show some finer details not shown in the low-resolution image—such as

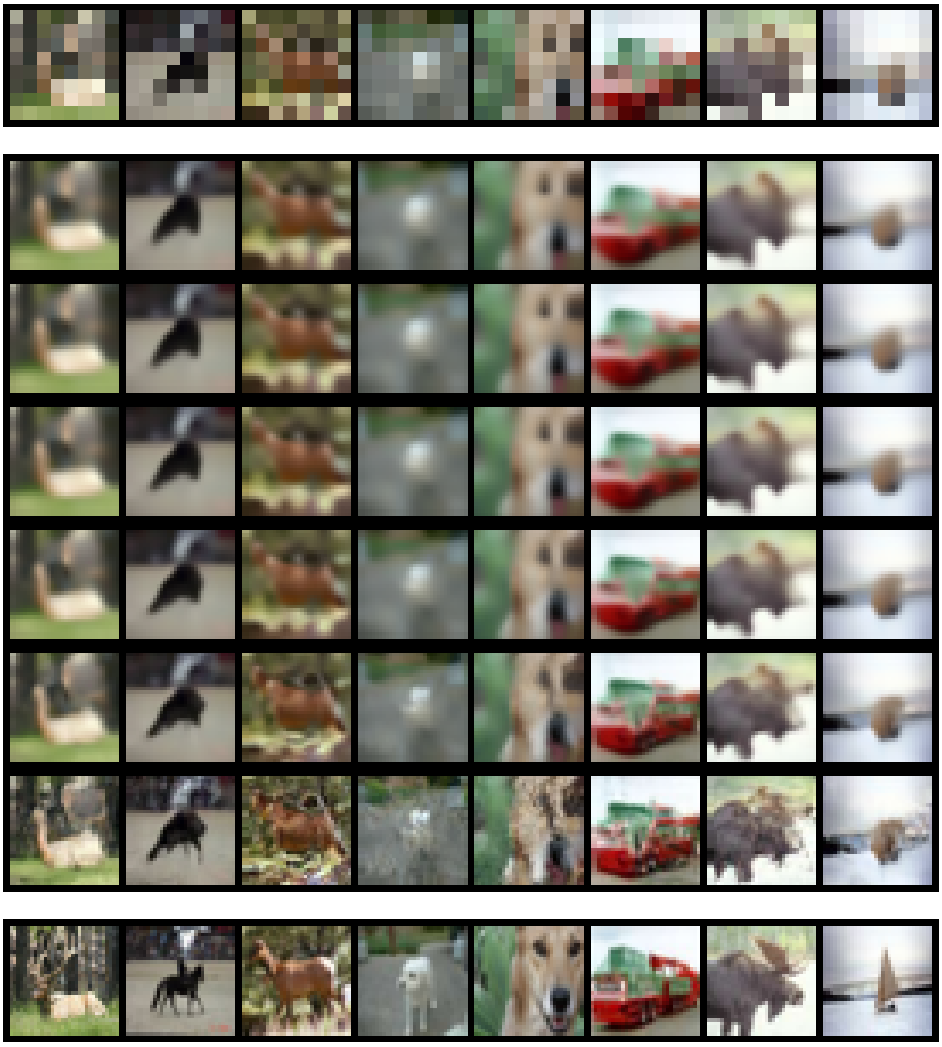


Figure 31: Random sample of super-resolution samples on CIFAR-10 dataset. *Top:* Low-resolution images that are 8×8 pixels. *Middle:* Super-resolved images, generated with temperatures $[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]$. Too low of a temperature of results in a more blurred super-resolution image, however sampling from the tails of the base distribution with a high temperature can result in poor reconstruction. *Bottom:* True high-resolution images of size 32×32.

antlers on the deer (column 1, although the deer's head and tail have been flipped), and legs on the horse (column 2).

6.5.4 *Comparing 4x and 8x PSR Flows*

In Figures 32 and 33 we show low-resolution, true high-resolution as well as super-resolved images by interpolating along a vector between two tails of the latent space and through the origin. We sample from two tails across the latent space to show variety in the super resolved image as a result of sampling from various points within the latent space, however the latent space is high-dimensional and contains uncountably more variations on the super-resolved image. The first and final generated samples are taken from a high temperature of 1.4, and begins to show artifacts in the images.

Figure 32: Interpolating across the latent space for a $4\times$ PSR flow trained on CelebA 64×64. *Left:* Low-resolution conditional input image which has been down sampled to 16×16 pixels ($\frac{1}{4}$ of the original size). *Middle:* True high-resolution image. *Right:* Generated samples of the super-resolved image by interpolating along a vector in the latent space between two tails (first and final images taken at temperature $\tau = 1.4$) and through the origin (center image taken at temperature $\tau = 0.0$).

The center-most generated samples are taken with a temperature of 0.0, resulting in softened features.

Figures 32 shows results for a model performing $4\times$ super-resolution, whereas Figures 33 produces an $8\times$ super-resolution result. Both models output images of size 64×64. In the $8\times$ PSR Flow shown in Figure 33, we see more variety in the super-resolved samples because the model must fill in more of the missing information. However, features for the $8\times$ PSR Flow tend to be slightly more blurry, whereas the $4\times$ PSR Flow shown in Figure 32 has more fine-grained detail but less variety.

In Figure 34 we show super-resolution results for $4\times$ and $8\times$ PSR Flows on a random sample of 8 images. The top two rows show the LR and SR images for the $4\times$ PSR Flow, respectively, whereas the bottom two rows show results for the $8\times$ PSR Flow. While the $8\times$ PSR Flow largely matches the structure of the ground truth, many of the smaller details are lost compared to the $4\times$ PSR Flow.

Figure 33: Interpolating across the latent space for a 8× PSR flow trained on CelebA 64×64. *Left:* Low-resolution conditional input image which has been down sampled to 8×8 pixels ($\frac{1}{8}$ of the original size). *Middle:* True high-resolution image. *Right:* Generated samples of the super-resolved image by interpolating along a vector in the latent space between two tails (first and final images taken at temperature $\tau = 1.4$) and through the origin (center image taken at temperature $\tau = 0.0$).

Figure 34: Super-resolution on random sample of ImageNet64. *Top rows:* PSR Flow for 4× super-resolution of images that are 16×16 pixels. *Middle row:* Ground truth 64×64 image. *Bottom rows:* PSR Flow for 8× super-resolution of images that are 8×8 pixels.

6.6 CONCLUSION

We present probabilistic super-resolution as a general framework for modeling joint and conditional distributions with normalizing flows. PSR architectures retain the advantages of flow-based models. And, by propagating high-level information learned from the LR input throughout the model, PSR flows address the information bottleneck and inductive bias problems often associated with flow models. We introduce a student teacher evaluation of PSR flows and show that they can learn complex conditional distributions. When applied to image modeling, PSR flows generate diverse high-fidelity images that are faithful to the ground truth. We find that image modeling with very low-resolution conditionals (up to $8\times$ super-resolution) still produces high quality samples exhibiting fine details.

6.7 BROADER IMPACTS

POTENTIAL NEGATIVE SOCIETAL IMPACTS Advancing probabilistic super-resolution is an important problem with applications to medical imaging, surveillance, agriculture, image and video content delivery, and improving the quality of cameras. There are however potential negative impacts to society as a result of better super-resolution algorithms. For one, as our work on *probabilistic* SR explores, there are many possible super-resolved outputs which are all reasonable. In some domains there may be serious consequences for producing incorrect outputs, no matter how plausible the outputs are given the low-resolution image.

Similarly, our work is focused on generative models, which can sample and manipulation new high resolution images that appear realistic. As such, there is a risk that generative models like these can aid in the spread of misinformation. As generative models become more powerful, it becomes harder to identify these generated data used for misinformation. If detecting fake data generated by models becomes increasingly more difficult and generating the data is accessible by nefarious parties, then preventing the spread misinformation may become impossible to prevent.

LIMITATIONS Our experiments largely focused on examining the diversity of PSR outputs, and showed the importance of either having a real conditional input or generating a quality low-resolution input (in the case of the joint PSR flow). Learning a quality generator of the low-resolution is challenging because for higher levels of super-resolution scaling there a very limited amount of information in the image that the model can learn from. For example, we experiment on 64×64 images with $8\times$ scaling, which results in low-resolution images that are difficult to discern even for humans. As such, generating low-resolution images that capture high-level semantic information, such as people or objects in the image, may not be possible. Similarly, given the importance of the low-resolution image in guiding the high-resolution output, future work may wish to perform an ablation study to better understand the trade-

off in model performance as a result of increasing the capacity of the low-resolution encoder and upsampling networks versus the capacity of the conditional normalizing flow components.

On a computational level, training generative models like normalizing flows that have likelihood-based objective functions is a challenge. The maximum likelihood objective function we use requires that the model explain all of the data (mode-covering behavior). In order for the model to explain all the data and produce sharp, realistic looking outputs it must have a relatively high capacity—on the order of 10-100 million parameters for models trained on the image datasets used in this work. Consequently, training these large models is computationally demanding, often requiring multiple days on high-end GPUs to train. Moreover, models like these can use a large portion of the GPU's available memory, which leaves less memory for data and hence training on bigger image sizes may be prohibitive.

Part IV

CONCLUDING REMARKS

7

EPILOGUE

The core merits of probabilistic modeling include quantifying uncertainty in models and predictions, estimating latent variables given data, and constructing models that understand the data well enough to be able to generate novel and plausible samples. In this thesis we focused on designing probabilistic models with likelihood-based objectives, with contributions spanning foundational techniques like probabilistic graphical models to deep generative models.

Generative models with likelihood-based objectives, like the models we proposed, require the model to explain all of the data. For small data problems graphical models are a prudent choice, since they are robust to overfitting and explicitly account for uncertainty. In cases where there is a lot of high-dimensional data (such as texts, images, video, or audio), asking the model to explain all the data is a daunting challenge. Early VAEs, for instance, lacked capacity and were overly entropic which generated blurry looking images while GANs—their implicit counterparts, generated sharp images (Higgins et al., 2017; Hu et al., 2018). However, as many recent works (Kingma and Dhariwal, 2018; Kumar et al., 2019; Lugmayr et al., 2020; Prenger, Valle, and Catanzaro, 2018; Vahdat and Kautz, 2020) and the results in this thesis showed, designing models powerful enough to accurately explain all the data is an achievable challenge—even when the data is complex and high-dimensional.

We began in Part (ii) with contributions to probabilistic graphical models, introducing a new topic model and efficient inference algorithms. Graphical models require an explicit parametric specification of the observed and latent random variables, but the model can be tailored to the unique structure of a dataset. We used the customizable nature of graphical models to represent the unique structure of the CaringBridge dataset consisting of multiple authors writing over time, yielding a powerful tool to zoom in and out on a massive collection of texts.

We also proposed a fast variational inference algorithm for optimizing the ELBO, a lower bound on the model evidence (i. e. marginal likelihood of the data), in order to scale training to corpora with billions of words. Lastly, with the invention of *regularized* variational inference we were able to guide the model to seek out specific solutions, such as diverse topics and latent personas describing the texts.

The main contribution we presented is in Part (iii) with a set normalizing flows that advance the performance and capabilities of this compelling new framework for deep generative modeling. In Chapter 4 we presented gradient boosted normalizing flows,

where we use gradient boosting to iteratively train an ensemble model made up of normalizing flows components. Despite gradient boosting being commonly associated with simple base learners like decision trees, we showed that boosting normalizing flows creates powerful probabilistic models that resemble a mixture model. Instead of needing deeper and more complex architectures, gradient boosted training allows the practitioner to exchange additional training time for a more flexible aggregate model. The aggregate GBNF model retains all the same strengths and capabilities of a normalizing flow, but has greater flexibility and allows for fast, highly-parallelizable inference. We showed that augmenting a VAE with GBNF improves posterior approximation, and GBNF is powerful enough to model high-dimensional data directly as a density estimator. Many opportunities exist for extending and adapting GBNF to new problems, among the most promising are conditional GNBFs and applying GBNF for life-long learning.

In Chapter 5 we reconsidered the bijective constraint on flow transformations, which requires that the base and data distribution have the same dimensionality. By relaxing the bijective constraint we constructed normalizing flows that can change the dimension of the latent space to learn low-dimensional representations. In high-dimensional settings, such as modeling images or spatio-temporal data, reducing dimensionality alleviates significant architectural, memory, and computational costs. Moreover, learning a compressed representation is a core idea in machine learning that forces the model to identify only the most important features and discard the noise.

We designed compressive normalizing flows in the simplest case based on PPCA—a core technique in dimensionality reduction. We then leveraged the connection between PPCA and VAEs to extend our idea to a more powerful class of compressive transformations. Compressive flow steps using VAEs optimize a lower bound on the marginal log-likelihood, which is common in probabilistic modeling. However, compressive flow models are unique in that they combine bijective transformations with exact likelihood calculations alongside the lower-bounded compressive transformations. Further work is still needed to better understand how tight heterogeneous objectives, which mix exact and lower-bound terms, are to the true log-likelihood.

Chapter 6 introduces probabilistic super-resolution. Building on conditional normalizing flows for image super-resolution, we presented PSR flows as a general modeling framework for normalizing flows which helps alleviate a few major challenges facing flow models. Conditional PSR flows use a residual-like connections to guide the high-dimensional output based on the low-dimensional conditional. We also presented joint PSR flows that model the distribution over both high- and low-dimensional samples. Joint PSR flows combine the structural advantages of conditional PSR flows, while also generating novel high-dimensional data similar to an unconditional flow. Our Student-Teacher evaluation helped to better understand the learned distribution over high-dimensional outputs, and showed that normalizing flows are highly adept at learning multi-modal distributions. As a next step, we are interested in evaluating conditional PSR flows on datasets where quantifying uncertainty in predictions is a

more central focus, such as in the domains of climate or finance, and to better understand out-of-distribution detection in PSR flows.

This thesis presented empirical and theoretical work to establish connections between core machine learning methods and the new probabilistic deep learning paradigm. From new inference algorithms to new models that fully explain the data, this thesis advances probabilistic modeling of complex high-dimensional data. We hope that the work presented in this thesis leads to continued progress on *complete* models of the data, with generalization and quantifying uncertainty at the forefront of model design.

REFERENCES

Alemi, Alexander A. et al. (2018). "Fixing a Broken ELBO." In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. PMLR, pp. 159–168.

Amari, Shun-ichi (Feb. 1998). "Natural Gradient Works Efficiently in Learning." In: *Neural Computation* 10.2, pp. 251–276. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/089976698300017746.

Ardizzone, Lynton et al. (July 2019). "Guided Image Generation with Conditional Invertible Neural Networks." In: *arXiv:1907.02392 [cs]*. arXiv: 1907.02392.

Ardizzone, Lynton et al. (2020). "Training Normalizing Flows with the Information Bottleneck for Competitive Generative Classification." In: *Advances in Neural Information Processing Systems*. arXiv: 2001.06448.

Ardizzone, Lynton et al. (2021). "Conditional Invertible Neural Networks for Diverse Image-to-Image Translation." In: *Pattern Recognition*. Ed. by Zeynep Akata et al. Vol. 12544. Cham: Springer International Publishing, pp. 373–387. ISBN: 978-3-030-71277-8 978-3-030-71278-5. DOI: 10.1007/978-3-030-71278-5_27.

Arjovsky, Martin and Léon Bottou (2017). "Towards Principled Methods for Training Generative Adversarial Networks." In: *arXiv preprint arXiv:1701.04862*. arXiv: 1701.04862.

Arjovsky, Martin et al. (Jan. 2017). "Wasserstein GAN." In: *ICML*.

Banerjee, Arindam (2006). "On Bayesian Bounds." In: *Proceedings of the 23rd International Conference on Machine Learning*. ACM, pp. 81–88.

Banerjee, Arindam (Apr. 2007). "An Analysis of Logistic Models: Exponential Family Connections and Online Performance." In: *Proceedings of the 2007 SIAM International Conference on Data Mining*. Proceedings. Society for Industrial and Applied Mathematics, pp. 204–215. ISBN: 978-0-89871-630-6. DOI: 10.1137/1.9781611972771.19.

Banerjee, Arindam (2010). *Introduction to Graphical Models for Data Mining*. Seattle, WA: 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

Bao, Xuchan et al. (July 2020). "Regularized Linear Autoencoders Recover the Principal Components, Eventually." In: *arXiv:2007.06731 [cs, stat]*. arXiv: 2007.06731.

Bartlett, Peter L et al. (Mar. 2006). "Convexity, Classification, and Risk Bounds." In: *Journal of the American Statistical Association* 101.473, pp. 138–156. ISSN: 0162-1459, 1537-274X. DOI: 10.1198/016214505000000907.

Beaudoin, Christopher E. and Chen-Chao Tao (2007). "Benefiting from Social Capital in Online Support Groups: An Empirical Study of Cancer Patients." In: *CyberPsychology & Behavior* 10.4, pp. 587–590.

Behrmann, Jens et al. (2019). "Invertible Residual Networks." In: *International Conference on Machine Learning*, p. 10.

Bevilacqua, Marco et al. (2012). "Low-Complexity Single-Image Super-Resolution Based on Nonnegative Neighbor Embedding." In:

Beygelzimer, Alina et al. (2015). "Online Gradient Boosting." In: *Advances in Neural Information Processing Systems*, p. 9.

Bhattacharyya, Apratim et al. (Oct. 2019). "Conditional Flow Variational Autoencoders for Structured Sequence Prediction." In: *arXiv:1908.09008 [cs, stat]*. arXiv: 1908.09008.

Bishop, Christopher M and Julia Lasserre (2007). "Generative or Discriminative? Getting the Best of Both Worlds." In: *Bayesian Statistics* 8, pp. 3–24.

Blei, David M. (2002). "Variational Inference." In: pp. 1–12.

Blei, David M. and Michael I. Jordan (2006). "Variational Inference for Dirichlet Process Mixtures." In: *Bayesian Analysis* 1.1 A, pp. 121–144.

Blei, David M. and John D. Lafferty (2006a). "Correlated Topic Models." In: *Advances in Neural Information Processing Systems 18*, pp. 147–154.

Blei, David M. and John D. Lafferty (2006b). "Dynamic Topic Models." In: *International Conference on Machine Learning*, pp. 113–120.

Blei, David M. and John D. Lafferty (June 2007). "A Correlated Topic Model of Science." In: *The Annals of Applied Statistics* 1.1, pp. 17–35. arXiv: 0708.3601.

Blei, David M et al. (2003). "Latent Dirichlet Allocation." In: *Journal of Machine Learning Research* 3.4-5, pp. 993–1022.

Blei, David M. et al. (2017). "Variational Inference: A Review for Statisticians." In: *Journal of the American Statistical Association* 112.518, pp. 859–877.

Bonnet, Georges (1964). "Transformations des signaux aléatoires a travers les systemes non linéaires sans mémoire." In: *Annales des Télécommunications* 19, pp. 203–220.

Boothby, W. M. (1986). *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Vol. 120. Academic press.

Both, Gert-Jan and Remy Kusters (Dec. 2019). "Temporal Normalizing Flows." In: *arXiv:1912.09092 [physics, stat]*. arXiv: 1912.09092.

Bowman, Samuel R. et al. (2016). "Generating Sentences from a Continuous Space." In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, pp. 10–21. DOI: 10.18653/v1/K16-1002.

Brehmer, Johann and Kyle Cranmer (June 2020). "Flows for Simultaneous Manifold Learning and Density Estimation." In: *arXiv:2003.13913 [cs, stat]*. arXiv: 2003.13913.

Bühlmann, Peter and Torsten Hothorn (Nov. 2007). "Boosting Algorithms: Regularization, Prediction and Model Fitting." In: *Statistical Science* 22.4, pp. 477–505. ISSN: 0883-4237. DOI: 10.1214/07-STS242. arXiv: 0804.2752.

Campbell, Trevor and Xinglong Li (Oct. 2019). "Universal Boosting Variational Inference." In: *Advances in Neural Information Processing Systems*. arXiv: 1906.01235.

Casale, Francesco Paolo et al. (2018). "Gaussian Process Prior Variational Autoencoders." In: *NIPS*, p. 11.

Chang, Jonathan et al. (2009). "Reading Tea Leaves: How Humans Interpret Topic Models." In: *Advances in Neural Information Processing Systems 22*, pp. 288–296.

Che, Tong et al. (Mar. 2020). "Your GAN Is Secretly an Energy-Based Model and You Should Use Discriminator Driven Latent Sampling." In: *arXiv:2003.06060 [cs, stat]*. arXiv: 2003.06060.

Chen, Changyou et al. (Sept. 2017a). "Continuous-Time Flows for Efficient Inference and Density Estimation." In: *arXiv:1709.01179 [stat]*. arXiv: 1709.01179.

Chen, Jianfei et al. (Feb. 2020). "VFlow: More Expressive Generative Flows with Variational Data Augmentation." In: *arXiv:2002.09741 [cs, stat]*. arXiv: 2002.09741.

Chen, Ricky T. Q. et al. (2018). "Neural Ordinary Differential Equations." In: *Advances in Neural Information Processing Systems*. arXiv: 1806.07366.

Chen, Tian Qi et al. (2019). "Residual Flows for Invertible Generative Modeling." In: *Advances in Neural Information Processing Systems*, p. 11.

Chen, Xi et al. (Dec. 2017b). "PixelSNAIL: An Improved Autoregressive Generative Model." In: *arXiv:1712.09763 [cs, stat]*. arXiv: 1712.09763.

Chen, Xi et al. (2017c). "Variational Lossy Autoencoder." In: *ICLR*. arXiv: 1611.02731.

Corney, David et al. (2016). "What Do a Million News Articles Look Like?" In: *Proceedings of the First International Workshop on Recent Trends in News Information Retrieval co-located with 38th European Conference on Information Retrieval {(ECIR} 2016), Padua, Italy, March 20, 2016.*, pp. 42–47.

Cornish, Rob et al. (Feb. 2020). "Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows." In: *arXiv:1909.13833 [cs, stat]*. arXiv: 1909.13833.

Cranko, Zac and Richard Nock (2019). "Boosting Density Estimation Remastered." In: *Proceedings of the 36th International Conference on Machine Learning*. Proceedings of Machine Learning Research 97, pp. 1416–1425.

Cremer, Chris et al. (2017). "Reinterpreting Importance-Weighted Autoencoders." In: *International Conference on Learning Representations*, p. 6.

Cremer, Chris et al. (2018). "Inference Suboptimality in Variational Autoencoders." In: *International Conference on Machine Learning*. Stockholm, Sweden. arXiv: 1801.03558.

Creswell, Antonia et al. (2018). "Generative Adversarial Networks: An Overview." In: *IEEE Signal Processing Magazine* 35.1, pp. 53–65.

Cunningham, Edmond et al. (June 2020). "Normalizing Flows Across Dimensions." In: *arXiv:2006.13070 [cs, stat]*. arXiv: 2006.13070.

Dai, Shengyang et al. (2009). "Softcuts: A Soft Edge Smoothness Prior for Color Image Super-Resolution." In: *IEEE Transactions on Image Processing* 18.5, pp. 969–981.

De Cao, Nicola et al. (Apr. 2019). "Block Neural Autoregressive Flow." In: *35th Conference on Uncertainty in Artificial Intelligence (UAI19)*. arXiv: 1904.04676.

Deng, Jia et al. (2009). "ImageNet: A Large-Scale Hierarchical Image Database." In: *IEEE Computer Vision and Pattern Recognition (CVPR)*, p. 8.

Deng, Ruizhi et al. (Oct. 2020). "Modeling Continuous Stochastic Processes with Dynamic Normalizing Flows." In: *34th Conference on Neural Information Processing Systems (NeurIPS)*. arXiv: 2002.10516.

Dhariwal, Prafulla et al. (Apr. 2020). "Jukebox: A Generative Model for Music." In: *arXiv:2005.00341 [cs, eess, stat]*. arXiv: 2005.00341.

Dibya Ghosh (Aug. 2018). *KL Divergence for Machine Learning*.

Dinh, Laurent et al. (2015). "NICE: Non-Linear Independent Components Estimation." In: *International Conference on Learning Representations*. arXiv: 1410.8516.

Dinh, Laurent et al. (2017). "Density Estimation Using Real NVP." In: *International Conference on Learning Representations*.

Dinh, Laurent et al. (Mar. 2019). "A RAD Approach to Deep Mixture Models." In: *arXiv:1903.07714 [cs, stat]*. arXiv: 1903.07714.

Dong, Chao et al. (2014). "Learning a Deep Convolutional Network for Image Super-Resolution." In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Vol. 8692. Cham: Springer International Publishing, pp. 184–199. ISBN: 978-3-319-10592-5 978-3-319-10593-2. DOI: 10.1007/978-3-319-10593-2_13.

Dong, Chao et al. (2016). "Image Super-Resolution Using Deep Convolutional Networks." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2, pp. 295–307. arXiv: 1501.00092.

Donsker, M. D. and S. R. S. Varadhan (Nov. 1976). "On the Principal Eigenvalue of Second-Order Elliptic Differential Operators." In: *Communications on Pure and Applied Mathematics* 29.6, pp. 595–621. ISSN: 00103640, 10970312. DOI: 10.1002/cpa.3160290606.

Dua, D. and E. Karra Taniskidou (2017). *UCI Machine Learning Repository*. archive.ics.uci.edu/ml/index.php. [Online; accessed 1-April-2020].

Duchon, Claude E (1979). "Lanczos Filtering in One and Two Dimensions." In: *Journal of Applied Meteorology and Climatology* 18.8, pp. 1016–1022.

Dupont, Emilien et al. (Oct. 2019). "Augmented Neural ODEs." In: *arXiv:1904.01681 [cs, stat]*. arXiv: 1904.01681.

Durkan, Conor et al. (2019). "Neural Spline Flows." In: *Advances in Neural Information Processing Systems*. arXiv: 1906.04032.

Frank, Marguerite and Philip Wolfe (1956). "An Algorithm for Quadratic Programming." In: *Naval Research Logistics Quarterly* 3, pp. 95–110.

Freund, Yoav and Robert E Schapire (1996). "Experiments with a New Boosting Algorithm." In: *International Conference on Machine Learning*, p. 9.

Freund, Yoav and Robert E Schapire (Aug. 1997). "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting." In: *Journal of Computer and System Sciences* 55.1, pp. 119–139. ISSN: 00220000. DOI: 10.1006/jcss.1997.1504.

Friedman, Jerome H. (2001). "Greedy Function Approximation: A Gradient Boosting Machine." In: *Annals of statistics*, pp. 1189–1232.

Friedman, Jerome H. (2002). "Stochastic Gradient Boosting." In: *Computational Statistics & Data Analysis* 38.4, pp. 367–378.

Friedman, Jerome et al. (2000). "Additive Logistic Regression: A Statistical View of Boosting." In: *The annals of statistics* 28.2, pp. 337–407.

Gal, Yarin et al. (2017). "Concrete Dropout." In: *Advances in Neural Information Processing Systems*. arXiv: 1705.07832.

Gao Huang et al. (2017). "Densely Connected Convolutional Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2261–2269.

Gelman, Andrew et al. (2014). "Expectation Propagation as a Way of Life: A Framework for Bayesian Inference on Partitioned Data." In:

Gemici, Mevlana C. et al. (Nov. 2016). "Normalizing Flows on Riemannian Manifolds." In: *Workshop on Bayesian Deep Learning at NIPS 2016*. arXiv: 1611.02304.

Germain, Mathieu et al. (Feb. 2015). "MADE: Masked Autoencoder for Distribution Estimation." In: *arXiv:1502.03509 [cs, stat]*. arXiv: 1502.03509.

Ghahramani, Zoubin (2015). "Probabilistic Machine Learning and Artificial Intelligence." In: *Nature* 521.7553, pp. 452–459.

Giaquinto, Robert and Arindam Banerjee (2018a). "DAPPER: Scaling Dynamic Author Persona Topic Model to Billion Word Corpora." In: *2018 IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20*. arXiv: 1811.01931.

Giaquinto, Robert and Arindam Banerjee (2018b). "Topic Modeling on Health Journals with Regularized Variational Inference." In: *AAAI*, pp. 3021–3028.

Giaquinto, Robert and Arindam Banerjee (June 2020). "Gradient Boosted Normalizing Flows." In: *Advances in Neural Information Processing Systems*. arXiv: 2002.11896.

Glynn, Peter W. (Oct. 1990). "Likelihood Ratio Gradient Estimation for Stochastic Systems." In: *Communications of the ACM* 33.10, pp. 75–84. ISSN: 00010782. DOI: 10.1145/84537.84552.

Goodfellow, Ian (2016). "Nips 2016 Tutorial: Generative Adversarial Networks." In: *arXiv preprint arXiv:1701.00160*. arXiv: 1701.00160.

Goodfellow, Ij et al. (2014). "Generative Adversarial Networks." In: *arXiv preprint arXiv: \ldots*, pp. 1–9.

Grathwohl, Will et al. (Oct. 2018). "FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models." In: *arXiv:1810.01367 [cs, stat]*. arXiv: 1810.01367.

Grover, Aditya and Stefano Ermon (2018). "Boosted Generative Models." In: *AAAI*.

Grover, Aditya et al. (Jan. 2018). "Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models." In: *arXiv:1705.08868 [cs, stat]*. arXiv: 1705.08868.

Guo, Fangjian et al. (Nov. 2016). "Boosting Variational Inference." In: *arXiv:1611.05559 [cs, stat]*. arXiv: 1611.05559.

Haris, Muhammad et al. (Mar. 2018). "Deep Back-Projection Networks For Super-Resolution." In: *arXiv:1803.02735 [cs]*. arXiv: 1803.02735.

Hasenclever, Leonard et al. (2017). "Variational Inference with Orthogonal Normalizing Flows." In: *NIPS Workshop on Bayesian Deep Learning*, p. 4.

Hastie, Trevor et al. (2001). *The Elements of Statistical Learning*. Second. Vol. 1.

He, Kaiming et al. (Dec. 2015). "Deep Residual Learning for Image Recognition." In: *arXiv:1512.03385 [cs]*. arXiv: 1512.03385.

Heusel, Martin et al. (Jan. 2018). "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." In: *arXiv:1706.08500 [cs, stat]*. arXiv: 1706.08500.

Higgins, Irina et al. (2017). "β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." In: *ICLR*, p. 22.

Hinton, G. et al. (May 1995). "The "Wake-Sleep" Algorithm for Unsupervised Neural Networks." In: *Science* 268.5214, pp. 1158–1161. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.7761831.

Ho, Jonathan et al. (2019). "Compression with Flows via Local Bits-Back Coding." In: *Advances in Neural Information Processing Systems*. arXiv: 1905.08500.

Ho, Jonathan et al. (2019). "Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design." In: *International Conference on Machine Learning*. arXiv: 1902.00275.

Hoffman, Matt et al. (2013). "Stochastic Variational Inference." In: *Journal of Machine Learning Research* 14, pp. 1303–1347.

Hoffman, Matthew D et al. (2010). "Online Learning for Latent Dirichlet Allocation." In: *Advances in Neural Information Processing Systems*. Vol. 23, pp. 1–9.

Hoogeboom, Emiel et al. (Oct. 2020). "The Convolution Exponential and Generalized Sylvester Flows." In: *Advances in Neural Information Processing Systems*. arXiv: 2006.01910.

Hu, Zhiting et al. (2018). "On Unifying Deep Generative Models." In: *ICLR*, pp. 1–19.

Huang, Chin-Wei et al. (2018a). "Improving Explorability in Variational Inference with Annealed Variational Objectives." In: *Advances in Neural Information Processing Systems*. Montréal, Canada, p. 11.

Huang, Chin-Wei et al. (2018b). "Neural Autoregressive Flows." In: *ICML*, p. 10.

Huang, Chin-Wei et al. (Feb. 2020). "Augmented Normalizing Flows: Bridging the Gap between Generative Flows and Latent Variable Models." In: *arXiv:2002.07101 [cs, stat]*. arXiv: 2002.07101.

Isola, Phillip et al. (July 2017). "Image-to-Image Translation with Conditional Adversarial Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, pp. 5967–5976. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.632.

Jacobsen, Jörn-Henrik et al. (Feb. 2018). "I-RevNet: Deep Invertible Networks." In: *arXiv:1802.07088 [cs, stat]*. arXiv: 1802.07088.

Jaggi, Martin (2013). "Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization." In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28(1). Proceedings of Machine Learning Research. PMLR, p. 9.

Jaini, Priyank et al. (2019). "Sum-of-Squares Polynomial Flow." In: *International Conference on Machine Learning*. Vol. 97. Long Beach, California: PMLR. arXiv: 1905.02325.

Jing, Longlong and Yingli Tian (Feb. 2019). "Self-Supervised Visual Feature Learning with Deep Neural Networks: A Survey." In: *arXiv:1902.06162 [cs]*. arXiv: 1902.06162.

Jordan, Michael I. et al. (1999a). "An Introduction to Variational Methods for Graphical Models." In: *Learning in Graphical Models*. Ed. by Michael I. Jordan. Dordrecht: Springer Netherlands, pp. 105–161.

Jordan, Michael I. et al. (1999b). "Introduction to Variational Methods for Graphical Models." In: *Machine Learning* 37.2, pp. 183–233.

Keys, R. (Dec. 1981). "Cubic Convolution Interpolation for Digital Image Processing." In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.6, pp. 1153–1160. ISSN: 0096-3518. DOI: 10.1109/TASSP.1981.1163711.

Khan, Mohammad Emtiyaz et al. (June 2016). "Faster Stochastic Variational Inference Using Proximal-Gradient Methods with General Divergence Functions." In: *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*. Arlington, Virginia, United States: AUAI Press, pp. 319–328.

Khan, Mohammad and Wu Lin (2017). "Conjugate-Computation Variational Inference: Converting Variational Inference in Non-Conjugate Models to Inferences in Conjugate Models." In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* 54, pp. 878–887.

Kim, Jiwon et al. (Nov. 2016). "Accurate Image Super-Resolution Using Very Deep Convolutional Networks." In: *arXiv:1511.04587 [cs]*. arXiv: 1511.04587.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization." In: *ICLR*.

Kingma, Diederik P. and Prafulla Dhariwal (July 2018). "Glow: Generative Flow with Invertible 1x1 Convolutions." In: *Advances in Neural Information Processing Systems*. Montréal, Canada. arXiv: 1807.03039.

Kingma, Diederik P and Max Welling (Dec. 2014). "Auto-Encoding Variational Bayes." In: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, pp. 1–14.

Kingma, Diederik P. and Max Welling (June 2019). "An Introduction to Variational Autoencoders." In: *Foundations and Trends® in Machine Learning* 12.4, pp. 307–392. arXiv: 1906.02691.

Kingma, Diederik P. et al. (2016). "Improving Variational Inference with Inverse Autoregressive Flow." In: *NIPS*. arXiv: 1606.04934.

Kingma, Diederik (2017). "Variational Inference and Deep Learning: A New Synthesis." PhD thesis.

Kirichenko, Polina et al. (June 2020). "Why Normalizing Flows Fail to Detect Out-of-Distribution Data." In: *arXiv:2006.08545 [cs, stat]*. arXiv: 2006.08545.

Kobyzev, Ivan et al. (2020). "Normalizing Flows: An Introduction and Review of Current Methods." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2020.2992934. arXiv: 1908.09257.

Kolesnikov, Alexander et al. (2019). "Revisiting Self-Supervised Visual Representation Learning." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1920–1929.

Koller, Daphne and Nir Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT press.

Krizhevsky, Alex (2009). *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. Vol. 1. No. 4. University of Toronto, p. 60.

Kumar, Abhishek et al. (2020). "Regularized Autoencoders via Relaxed Injective Probability Flow." In: *AISTATS*. arXiv: 2002.08927.

Kumar, Manoj et al. (Mar. 2019). "VideoFlow: A Flow-Based Generative Model for Video." In: *arXiv:1903.01434 [cs]*. arXiv: 1903.01434.

Lake, Brenden M. et al. (Dec. 2015). "Human-Level Concept Learning through Probabilistic Program Induction." In: *Science* 350.6266, pp. 1332–1338. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aab3050.

Larochelle, Hugo and Iain Murray (2011). "The Neural Autoregressive Distribution Estimator." In: *International Conference on Artificial Intelligence and Statistics (AISTATS)* 15, p. 9.

LeCun, Yann et al. (1998). "Gradient-Based Learning Applied to Document Recognition." In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Lebanon, Guy and John D Lafferty (2002). "Boosting and Maximum Likelihood for Exponential Models." In: *NIPS*, p. 8.

Ledig, Christian et al. (May 2017). "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network." In: *arXiv:1609.04802 [cs, stat]*. arXiv: 1609.04802.

Li, Yingzhen et al. (2019). "Are Generative Classifiers More Robust to Adversarial Attacks?" In: *International Conference on Machine Learning*, p. 11.

Lim, Yew Jin and Yee Whye Teh (2007). "Variational Bayesian Approach to Movie Rating Prediction." In: *KDD*, p. 7.

Liu, Ziwei et al. (2015). "Deep Learning Face Attributes in the Wild." In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730–3738.

Locatello, Francesco et al. (Aug. 2017). "Boosting Variational Inference: An Optimization Perspective." In: *arXiv:1708.01733 [cs, stat]*. arXiv: 1708.01733.

Lu, You and Bert Huang (Feb. 2020). "Structured Output Learning with Conditional Generative Flows." In: *arXiv:1905.13288 [cs, stat]*. arXiv: 1905.13288.

Lucas, James et al. (Nov. 2019). "Don't Blame the ELBO! A Linear VAE Perspective on Posterior Collapse." In: *arXiv:1911.02469 [cs, stat]*. arXiv: 1911.02469.

Lugmayr, Andreas et al. (2020). "SRFlow: Learning the Super-Resolution Space with Normalizing Flow." In: *European Conference on Computer Vision*. arXiv: 2006.14200.

Ma, Xuezhe et al. (2019). "MaCow: Masked Convolutional Generative Flow." In: *Advances in Neural Information Processing Systems*, p. 10.

Mackowiak, Radek et al. (Dec. 2020). "Generative Classifiers as a Basis for Trustworthy Image Classification." In: *Advances in Neural Information Processing Systems*. arXiv: 2007.15036.

Mandt, Stephan and David Blei (2014). "Smoothed Gradients for Stochastic Variational Inference." In: *Advances in Neural Information Processing Systems*, pp. 2438–2446.

Marlin, Benjamin M et al. (2010). "Inductive Principles for Restricted Boltzmann Machine Learning." In: *13thInternational Conference on Artificial Intelligence and Statistics (AISTATS)* 9, p. 8.

Marquina, Antonio and Stanley J Osher (2008). "Image Super-Resolution by TV-Regularization and Bregman Iteration." In: *Journal of Scientific Computing* 37.3, pp. 367–382.

Martin, D. et al. (2001). "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*. Vol. 2. Vancouver, BC, Canada: IEEE Comput. Soc, pp. 416–423. ISBN: 978-0-7695-1143-6. DOI: 10.1109/ICCV.2001.937655.

Mason, Llew et al. (1999). "Boosting Algorithms as Gradient Descent." In: *Advances in Neural Information Processing Systems*, pp. 512–518.

Mathieu, Michael et al. (Feb. 2016). "Deep Multi-Scale Video Prediction beyond Mean Square Error." In: *arXiv:1511.05440 [cs, stat]*. arXiv: 1511.05440.

McCallum, a. et al. (2005). "The Author-Recipient-Topic Model for Topic and Role Discovery in Social Networks: Experiments with Enron and Academic Email." In: *NIPS'04 Workshop on'Structured Data and Representations in Probabilistic Models for Categorization*, pp. 1–16.

Menick, Jacob and Nal Kalchbrenner (Dec. 2018). "Generating High Fidelity Images with Subscale Pixel Networks and Multidimensional Upscaling." In: *arXiv:1812.01608 [cs, stat]*. arXiv: 1812.01608.

Miller, Andrew C. et al. (2017). "Variational Boosting: Iteratively Refining Posterior Approximations." In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, pp. 2420–2429.

Mimno, David and A McCallum (2007). "Expertise Modeling for Matching Papers with Reviewers." In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 500–509.

Mimno, David and Andrew McCallum (2008). "Topic Models Conditioned on Arbitrary Features with Dirichlet-Multinomial Regression." In: *Proceedings of the 24th Annual Conference on Uncertainty in Artificial Intelligence (UAI-08)*, pp. 411–418.

Minka, Thomas P (2001a). "A Family of Algorithms for Approximate Bayesian Inference." In: *Ph.D. Thesis*, pp. 1–482.

Minka, Tom (2001b). "Expecatation Propagation for Approximate Bayesian Inference." In: *Conference on Uncertainty in Artificial Intelligence*, pp. 362–369.

Mnih, Andriy and Karol Gregor (2014). "Neural Variational Inference and Learning in Belief Networks." In: 32.

Mohamed, Shakir and Balaji Lakshminarayanan (Feb. 2017). "Learning in Implicit Generative Models." In: *arXiv:1610.03483 [cs, stat]*. arXiv: 1610.03483.

Narayanan, Arvind and Vitaly Shmatikov (June 2010). "Myths and Fallacies of "Personally Identifiable Information"." In: *Communications of the ACM* 53.6, p. 24. ISSN: 00010782. DOI: 10.1145/1743546.1743558.

Ng, Andrew Y and Michael I Jordan (2001). "On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes." In: *Advances in Neural Information Processing Systems*, p. 8.

Nguyen, XuanLong et al. (2005). "On Divergences, Surrogate Loss Functions, and Decentralized Detection." In: *CoRR* abs/math/0510521, p. 35.

Nguyen, XuanLong et al. (Apr. 2009). "On Surrogate Loss Functions and f-Divergences." In: *The Annals of Statistics* 37.2, pp. 876–904. ISSN: 0090-5364. DOI: 10.1214/08-AOS595. arXiv: math/0510521.

Nielsen, Didrik and Ole Winther (Oct. 2020). "Closing the Dequantization Gap: PixelCNN as a Single-Layer Flow." In: *arXiv:2002.02547 [cs, stat]*. arXiv: 2002.02547.

Nielsen, Didrik et al. (July 2020). "SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows." In: *arXiv:2007.02731 [cs, stat]*. arXiv: 2007.02731.

Nielsen, Frank and Vincent Garcia (2009). "Statistical Exponential Families: A Digest with Flash Cards." In: 2011.

Oord, Aaron van den et al. (Jan. 2016). "Pixel Recurrent Neural Networks." In: *arXiv:1601.06759 [cs]*. arXiv: 1601.06759.

Paisley, John et al. (2012). "Variational Bayesian Inference with Stochastic Search." In: *International Conference on Machine Learning* 2000, pp. 1367–1374.

Papamakarios, George et al. (2017). "Masked Autoregressive Flow for Density Estimation." In: *Advances in Neural Information Processing Systems*. arXiv: 1705.07057.

Papamakarios, George et al. (Dec. 2019). "Normalizing Flows for Probabilistic Modeling and Inference." In: *arXiv:1912.02762 [cs, stat]*. arXiv: 1912.02762.

Paszke, Adam et al. (2017). "Automatic Differentiation in PyTorch." In: *Advances in Neural Information Processing Systems*, p. 4.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems*. arXiv: 1912.01703.

Pathak, Nishith et al. (2008). "Social Topic Models for Community Extraction." In: *The 2nd SNA-KDD Workshop '08 (SNA-KDD'08)*.

Prenger, Ryan et al. (Oct. 2018). "WaveGlow: A Flow-Based Generative Network for Speech Synthesis." In: *arXiv:1811.00002 [cs, eess, stat]*. arXiv: 1811.00002.

Price, Robert (1958). "A Useful Theorem for Nonlinear Devices Having Gaussian Inputs." In: *IRE Transactions on Information Theory* 4, pp. 69–72.

Rainforth, Tom et al. (Feb. 2018). "Tighter Variational Bounds Are Not Necessarily Better." In: *arXiv:1802.04537 [cs, stat]*. arXiv: 1802.04537.

Ranganath, Rajesh et al. (2013). "Black Box Variational Inference." In: *Aistats* 33.

Ranganath, Rajesh et al. (2014). "Black Box Variational Inference (Extra Materials)." In: *Aistats*.

Rasmussen, Carl Edward and Christopher K. I. Williams (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press. ISBN: 978-0-262-18253-9.

Rasul, Kashif et al. (May 2020). "Multi-Variate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows." In: *arXiv:2002.06103 [cs, stat]*. arXiv: 2002.06103.

Razavi, Ali et al. (June 2019). "Generating Diverse High-Fidelity Images with VQ-VAE-2." In: *arXiv:1906.00446 [cs, stat]*. arXiv: 1906.00446.

Reed, Scott et al. (Mar. 2017). "Parallel Multiscale Autoregressive Density Estimation." In: *arXiv:1703.03664 [cs]*. arXiv: 1703.03664.

Rezende, Danilo Jimenez and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows." In: *ICML*. arXiv: 1505.05770.

Rezende, Danilo Jimenez et al. (Jan. 2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models." In: *arXiv:1401.4082 [cs, stat]*. arXiv: 1401.4082.

Rippel, Oren and Ryan Prescott Adams (Feb. 2013). "High-Dimensional Probability Estimation with Deep Density Models." In: *arXiv:1302.5125 [cs, stat]*. arXiv: 1302.5125.

Robbins, Herbert and Sutton Monro (1951). "A Stochastic Approximation Method." In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407.

Rodgers, S and Q Chen (2005). "Internet Community Group Participation: Psychosocial Benefits for Women with Breast Cancer." In: *Journal of Computer-Mediated Communication* 10.4, pp. 1–27.

Rolinek, Michal et al. (2019). "Variational Autoencoders Pursue PCA Directions (by Accident)." In: *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12406–12415. arXiv: 1812.06775.

Rosen-Zvi, M. et al. (2004). "The Author-Topic Model for Authors and Documents." In: *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 487–494.

Rosset, Saharon and Eran Segal (2002). "Boosting Density Estimation." In: *Advances in Neural Information Processing Systems*, p. 8.

Rubner, Yossi et al. (2000). "The Earth Mover's Distance as a Metric for Image Retrieval." In: *International journal of computer vision* 40.2, pp. 99–121.

Salman, Hadi et al. (Oct. 2018). "Deep Diffeomorphic Normalizing Flows." In: *arXiv:1810.03256 [cs, stat]*. arXiv: 1810.03256.

Schapire, Robert E. et al. (Oct. 1998). "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods." In: *The Annals of Statistics* 26.5, pp. 1651–1686. DOI: 10.1214/aos/1024691352.

Seeger, Matthias (2003). "Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations." PhD thesis. Institute for Adaptive and Neural Computation, Division of Informatics: University of Edinburgh.

Smith, Leslie N. (Mar. 2017). "Cyclical Learning Rates for Training Neural Networks." In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Santa Rosa, CA, USA: IEEE, pp. 464–472. ISBN: 978-1-5090-4822-9. DOI: 10.1109/WACV.2017.58.

Sønderby, Casper Kaae et al. (2016). "Ladder Variational Autoencoders." In: *NIPS*. arXiv: 1602.02282.

Sorkhei, Moein et al. (Dec. 2020). "Full-Glow: Fully Conditional Glow for More Realistic Image Generation." In: *arXiv:2012.05846 [cs]*. arXiv: 2012.05846.

Steyvers, M. et al. (2004). "Probabilistic Author-Topic Models for Information Discovery." In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 315.

Sun, Jian et al. (2008). "Image Super-Resolution Using Gradient Profile Prior." In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Tabak, E. G. and Cristina V. Turner (Feb. 2013). "A Family of Nonparametric Density Estimation Algorithms." In: *Communications on Pure and Applied Mathematics* 66.2, pp. 145–164. ISSN: 00103640. DOI: 10.1002/cpa.21423.

Tabak, Esteban G. and Eric Vanden-Eijnden (2010). "Density Estimation by Dual Ascent of the Log-Likelihood." In: *Communications in Mathematical Sciences* 8.1, pp. 217–233. ISSN: 15396746, 19450796. DOI: 10.4310/CMS.2010.v8.n1.a11.

Tipping, Michael E. and Christopher M. Bishop (Aug. 1999). "Probabilistic Principal Component Analysis." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3, pp. 611–622. ISSN: 1369-7412, 1467-9868. DOI: 10.1111/1467-9868.00196.

Tolstikhin, Ilya et al. (May 2017). "AdaGAN: Boosting Generative Models." In: *arXiv:1701.02386 [cs, stat]*. arXiv: 1701.02386.

Tomczak, Jakub M. and Max Welling (2016). "Improving Variational Auto-Encoders Using Householder Flow." In: *Bayesian Deep Learning Workshop (NIPS 2016)*. arXiv: 1611.09630.

Tomczak, Jakub M. and Max Welling (June 2017). "Improving Variational Auto-Encoders Using Convex Combination Linear Inverse Autoregressive Flow." In: *arXiv:1706.02326 [stat]*. arXiv: 1706.02326.

Tomczak, Jakub and Max Welling (2018). "VAE with a VampPrior." In: *International Conference on Artifi- Cial Intelligence and Statistics (AISTATS)*. Vol. 84. Lanzarote, Spain.

Trippe, Brian L and Richard E Turner (2017). "Conditional Density Estimation with Bayesian Normalizing Flows." In: *Workshop on Bayesian Deep Learning (NIPS)*, p. 12.

Uria, Benigno et al. (2013). "RNADE: The Real-Valued Neural Autoregressive Density-Estimator." In: *Advances in Neural Information Processing Systems*. arXiv: 1306.0186.

Vahdat, Arash and Jan Kautz (July 2020). "NVAE: A Deep Hierarchical Variational Autoencoder." In: *arXiv:2007.03898 [cs, stat]*. arXiv: 2007.03898.

Vaswani, Ashish et al. (June 2017). "Attention Is All You Need." In: *arXiv:1706.03762 [cs]*. arXiv: 1706.03762.

Wainwright, Martin J. and Michael I. Jordan (2007). "Graphical Models, Exponential Families, and Variational Inference." In: *Foundations and Trends® in Machine Learning* 1.1–2, pp. 1–305.

Wang, Chong et al. (2008). "Continuous Time Dynamic Topic Models." In: *Proc of UAI*, pp. 579–586.

Wang, Xintao et al. (Sept. 2018). "ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks." In: *arXiv:1809.00219 [cs]*. arXiv: 1809.00219.

Wang, Xuerui and Andrew McCallum (2006). "Topics over Time: A Non-Markov Continuous-Time Model of Topical Trends." In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 424–433.

Wei, Xing et al. (2007). "Dynamic Mixture Models for Multiple Time Series." In: *Ijcai* Dmm, pp. 2909–2914.

Wen, Miaomiao and Carolyn Penstein Rose (2012). "Understanding Participant Behavior Trajectories in Online Health Support Groups Using Automatic Extraction Methods." In: *Proceedings of the 17th ACM international conference on Supporting group work - GROUP '12*, p. 179.

Williams, Ronald J (1992). "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." In: *Machine Learning* 8, p. 28.

Winkler, Christina et al. (Nov. 2019). "Learning Likelihoods with Conditional Normalizing Flows." In: *arXiv:1912.00042 [cs, stat]*. arXiv: 1912.00042.

Wu, Hao et al. (Feb. 2020). "Stochastic Normalizing Flows." In: *arXiv:2002.06707 [physics, stat]*. arXiv: 2002.06707.

Xiao, Zhisheng et al. (Sept. 2019). "Generative Latent Flow." In: *arXiv:1905.10485 [cs]*. arXiv: 1905.10485.

Yan, Qing et al. (2015). "Single Image Superresolution Based on Gradient Profile Sharpness." In: *IEEE Transactions on Image Processing* 24.10, pp. 3187–3202.

Yang, Wenming et al. (Dec. 2019). "Deep Learning for Single Image Super-Resolution: A Brief Review." In: *IEEE Transactions on Multimedia* 21.12, pp. 3106–3121. ISSN: 1520-9210, 1941-0077. DOI: 10.1109/TMM.2019.2919431. arXiv: 1808.03344.

Yu, Jason J. et al. (Oct. 2020). "Wavelet Flow: Fast Training of High Resolution Normalizing Flows." In: *Advances in Neural Information Processing Systems*. arXiv: 2010.13821.

Yu, Xin and Fatih Porikli (2016). "Ultra-Resolving Face Images by Discriminative Generative Networks." In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Vol. 9909. Cham: Springer International Publishing, pp. 318–333. ISBN: 978-3-319-46453-4 978-3-319-46454-1. DOI: 10.1007/978-3-319-46454-1_20.

Zeyde, Roman et al. (2010). "On Single Image Scale-up Using Sparse-Representations." In: *International Conference on Curves and Surfaces*, pp. 711–730.

Zhang, Richard et al. (Apr. 2018). "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric." In: *arXiv:1801.03924 [cs]*. arXiv: 1801.03924.

Ziegler, Zachary M. and Alexander M. Rush (2019). "Latent Normalizing Flows for Discrete Sequences." In: *Advances in Neural Information Processing Systems*.

van den Berg, Rianne et al. (2018). "Sylvester Normalizing Flows for Variational Inference." In: *Uncertainty in Artificial Intelligence (UAI)*.

van den Oord, Aaron et al. (2017). "Neural Discrete Representation Learning." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 6306–6315.

Part V

APPENDIX

A

COMMON NOTATIONS

Notation	Description
$\mathbf{x}, \mathbf{y}\, \mathbf{z}$	Bold characters denote random vectors.
x, y, z	Italic characters denote random scalars, i. e. single real-valued numbers.
$\mathbf{x} \odot \mathbf{y}$	Hadamard product (or element-wise multiplication) of two vectors. The resulting vector is $(x_1 y_1, \ldots, x_K y_K)^\top$.
$\mathbf{X}, \mathbf{Y}, \mathbf{Z}$	Bold capitalized letters denote random matrices.
$\det(\mathbf{X})$	Determinant of the matrix $\mathbf{X}$.
$\Gamma = \{\phi, \theta\}$	Parameters of models are typically denoted with lowercase Greek letters ϕ and θ, and sets of parameters by Γ.
$p(\mathbf{x}), p(\mathbf{z})$	Probability density functions (PDFs) and probability mass functions (PMFs), also simply called *distributions*, are denoted by $p(\cdot)$ or $q(\cdot)$.
$p(\mathbf{x}, \mathbf{y}, \mathbf{z})$ or $p_{(X,Y,Z)}(\mathbf{x}, \mathbf{y}, \mathbf{z})$	Joint distributions are denoted by $p(\cdot, .)$
$p(\mathbf{x} \mid \mathbf{z})$ or $p_{X\mid Z}(\mathbf{x} \mid \mathbf{z})$	Conditional distributions are denoted by $p(\cdot \mid \cdot)$
$p(\cdot\,; \theta), p_\theta(\mathbf{x})$	The parameters of a distribution are denoted with $p(\cdot\,; \theta)$ or equivalently with subscript $p_\theta(\cdot)$.
$p(\cdot), q(\cdot)$	Different letters to refer to different probabilistic models, such as $p(\cdot)$ or $q(\cdot)$. Conversely, the same letter across different marginals/conditionals indicates they relate to the same probabilistic model.

B

CONDITIONAL GRADIENT BOOSTED NORMALIZING FLOWS

Learning complex conditional distributions is a common but critical problem, and subsumes general tasks like regression and classification. While generative cannot always match the predictive performance of their discriminative counterparts (Ng and Jordan, 2001), they can offer many advantages as probabilistic models—such likelihood calculation, as well as generate novel and plausible high-dimensional data like images.

Modern generative models, like Generative Adversarial Networks (GAN, (Goodfellow, Pouget-Abadie, and Mirza, 2014)) and Variational Autoencoders (VAE, (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014)), avoid simple parametric forms and instead learn to transform noise into the target distribution. Conditional variants of GANs and VAEs can sample from a conditional distribution but do not provide tractable likelihoods.

The majority of normalizing flow research focuses on the application of density estimation $p(\mathbf{x})$, and variational inference (i. e. construction an approximation $q(\mathbf{z} \mid \mathbf{x})$ of the true posterior) (Papamakarios et al., 2019). In many instances, however, we have additional data $\mathbf{y}$ such as labels that we wish to treat as conditionals in our model $p(\mathbf{x} \mid \mathbf{y})$, effectively acting as side information to the model (Germain et al., 2015; Kingma and Dhariwal, 2018). Conditioning the model of high-dimensional data $\mathbf{x}$ on some label y introduces structure in the model and allows for control of generated samples based on a specific label.

There are challenges in modeling a high-dimensional $\mathbf{x}$ with labels or a regression target y jointly. In particular, when $\mathbf{x}$ is high-dimensional it contains much more information and hence a model of the joint distribution $p(\mathbf{x}, y)$ is not strongly incentivized to accurately model y. Techniques to balance the weight of information between $\mathbf{x}$ and y exist for particular applications (Ardizzone et al., 2020).

In Chapter 4 we introduced gradient boosted normalizing flows, showing that additional model flexibility is possible by incorporating additional flow component trained with gradient boosting. Gradient boosted normalizing flows require no fundamental change to the underlying flow model—we train each new flow component to optimize an objective that over-weights residuals from the previously trained components. As such, it is natural to consider a gradient boosted training regime for conditional normalizing flows.

In addition to providing probabilistic estimates of outputs (e. g. of image labels, or interpolated and extrapolated regression), conditional density estimation models have

successfully been applied to tasks like image segmentation (Lu and Huang, 2020), and super-resolution image modeling (Lugmayr et al., 2020; Winkler et al., 2019) where $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ are the pixels of the low and high-resolution images, respectively.

Similar to (6.3), the objective function that we seek to minimize is:

$$\mathcal{F} \coloneqq \underset{(\mathbf{x},y)\sim p(X,Y)}{-\mathbb{E}} \left[\log G_K^{(c)}(y \mid \mathbf{x}; \boldsymbol{\phi}_{1:C})\right] , \tag{B.1}$$

where $\boldsymbol{\phi}_{1:C}$ are the parameters to each of the component flows.

Here we minimize the forward KL-divergence between the true distribution p^* and a conditional gradient boosted normalizing flow. In the conditional density estimation setting, the gradient boosted normalizing flow takes the form:

$$G_K^{(c)}(y \mid \mathbf{x}) = \psi\left((1-\rho_c)\psi^{-1}(G_K^{(c-1)}(y \mid \mathbf{x})) + \rho_c \psi^{-1}(g_K^{(c)}(y \mid \mathbf{x}))\right)/\Gamma_{(c)} , \tag{B.2}$$

where we have omitted notation for parameters $\boldsymbol{\phi}_{1:C}$ for compactness. The full model $G_K^{(c)}(y \mid \mathbf{x})$ is a monotonic function ψ of a convex combination of fixed components $G_K^{(c-1)}$ and new component $g_K^{(c)}$, and $\Gamma_{(c)}$ is the partition function. For the sake of brevity we only consider additive boosting where $\psi(a) = a$ and the partition function $\Gamma_c = 1$, however the formulation can be extended to additive boosting just as in the model presented for unconditional density estimation.

Thus, we seek a new component $g_K^{(c)}$ that minimizes the negative log-likelihood over the samples:

$$-\mathcal{L} \approx -\frac{1}{n}\sum_{i=1}^{n}\left[\log\left((1-\rho_c)G_K^{(c-1)}(y_i \mid \mathbf{x}_i; \boldsymbol{\phi}_{1:C-1}) + \rho_c g_K^{(c)}(y_i \mid \mathbf{x}_i; \phi_c)\right)\right] . \tag{B.3}$$

B.1 NEW COMPONENT UPDATES

Our goal is to derive an update to the new component $g_K^{(c)}$ that minimizes (B.3) by using functional gradient descent. Note, for compactness we have used more compact notation to write $G_K^{(c)}(y \mid \mathbf{x})$ as $G^{(c)}$. Consider, then the gradient of our objective in expectation, as given by (B.1), with respect to the parameters $\boldsymbol{\phi}$ of $G^{(c)}$ at $\rho_c \to 0$:

$$\nabla_{\boldsymbol{\phi}_{1:C}} \mathcal{F}\big|_{\rho_c \to 0} = \nabla_{\boldsymbol{\phi}_{1:C-1}, \phi_c} - \mathbb{E}_{p(X,Y)}\left[\log\left((1-\rho_c)G^{(c-1)} + \rho_c g^{(c)}\right)\right] .$$

Then, taking the gradient of $\mathcal{F}$ with respect to the fixed parameters gives:

$$\begin{aligned}\nabla_{\boldsymbol{\phi}_{1:C-1}} \mathcal{F}\big|_{\rho_c \to 0} &= -\mathbb{E}_{p(X,Y)}\left[\frac{1-\rho_c}{(1-\rho_c)G^{(c-1)} + \rho_c g^{(c)}}\right]\Bigg|_{\rho_c \to 0} \\ &= -\mathbb{E}_{p(X,Y)}\left[\frac{1}{G^{(c-1)}}\right] , \end{aligned} \tag{B.4}$$

$$\tag{B.5}$$

whereas taking the gradient of $\mathcal{F}$ with respect to ϕ_c is simply:

$$\nabla_{\phi_c} \mathcal{F}\big|_{\rho_c \to 0} = -\mathbb{E}_{p(X,Y)} \left[\frac{\rho_c}{(1-\rho_c) G^{(c-1)} + \rho_c g^{(c)}} \right] \Bigg|_{\rho_c \to 0} = 0$$

Thus, since $G^{(c-1)}$ is fixed, then minimizing $\mathcal{F}$ can be achieved by choosing a new component $g^{(c)}$ with a maximum inner product with the negative of the gradient (B.4) over the n data-points. Hence, we choose $g^{(c)}$ as

$$\begin{aligned} g_K^{(c)} &= \arg\max_{g_K \in \mathcal{G}} \sum_{i=1}^{n} \left\langle g_K^{(c)}(y_i \mid \mathbf{x}_i; \phi_c), \frac{1}{G_K^{(c-1)}(y_i \mid \mathbf{x}_i; \boldsymbol{\phi}_{1:c-1})} \right\rangle \\ &= \arg\min_{g_K \in \mathcal{G}} - \sum_{i=1}^{n} \log \frac{g_K^{(c)}(y_i \mid \mathbf{x}_i; \phi_c)}{G_K^{(c-1)}(y_i \mid \mathbf{x}_i; \boldsymbol{\phi}_{1:c-1})} \,, \end{aligned} \tag{B.6}$$

where in the last step we take the log, which follows from directly from the optimization conditions for a convex function in a compact domain, and then flip the sign to show that the new $g^{(c)}$ is chosen by minimizing the negative log-likelihood in which the observations are *weighted* by the fixed distribution $G_K^{(c-1)}(y_i \mid \mathbf{x}_i; \boldsymbol{\phi}_{1:c-1})$.

Similar to the unconditional additive boosting case, the update in (B.6) yields a degenerate probability distribution where the entire mass is placed at the minimum point of the constant term $G_K^{(c)}(y \mid \mathbf{x})$. We find that entropy regularization avoids such degenerate solutions, or alternatively the analogous conditional version the multiplicative boosting model can be trained stably without augmenting the objective.

B.2 GRADIENT BOOSTED TRAINING

To improve the performance of our conditional density estimators we train conditional normalizing flows with gradient boosting. Gradient boosted normalizing flows allow for more flexible aggregate models with needing to make individual flow components deeper or more complex. One of the primary applications of a conditional density estimation model is to generate samples given the conditional input, and we are primarily interested in flow-based approaches as a method of learning the full distribution over possible outputs.

Gradient boosting allows us to easily model more complex conditional distributions, where each flow component in the GBNF covers a subset of the full conditional distribution. Each new component trained in a GBNF model seeks regions that are not well modeled by the previously trained regions—however, components are not restricted from overlapping one another. Additionally, each component in GBNF acts independently during prediction and sampling. Hence, each component must have sufficient capacity to accurately model all or a portion of the output distribution. As a result, when modeling images of faces, for example, a flow component must learn all features

making up a face, and cannot focus on solely on outputting accurate eyes or ears. Because GBNF has a mixture-like formulation, sampling proceeds by first sampling a component, and then applying the forward flow transformation:

$$\begin{aligned} \text{Choose Component:} \quad & c \sim \mathtt{Categorical}(\rho_{1:C}) \\ \text{Sample Noise:} \quad & \mathbf{z} \sim p_Z(\mathbf{z}) \\ \text{Forward Flow:} \quad & \hat{\mathbf{y}} \sim g_K^{(c)}(\mathbf{z} \mid \mathbf{x}) \end{aligned}$$

where $c \in [1, C]$ is the component index drawn based on model weights $\rho_{1:C}$ corresponding to component $g_K^{(c)}$, and $\mathbf{z} \sim p_Z(\mathbf{z})$ is noise sample from the base distribution.

C

ORIGINAL VARIATIONAL EM ALGORITHM FOR DAP TOPIC MODEL

Given the DAP model structure from Section 3.3, we provide the original inference algorithm used to estimate the model's latent parameters using standard mean-field variational inference techniques—as opposed to the CVI based updates provided in Section 3.5.

Much like LDA and its extensions, the DAP model's posterior:

$$p(\kappa, \mathbf{x}, \alpha, \beta, \theta, \mathbf{z} \mid \mathbf{w}, \omega, \eta) = \frac{p(\kappa, \mathbf{x}, \alpha, \beta, \theta, \mathbf{z}, \mathbf{w} \mid \omega, \eta)}{p(\mathbf{w} \mid \mu_0, \sigma_0, \eta, \omega)} ,$$

is intractable due to the normalization term. In order learn optimal values to the model's parameters we use a form of variational inference (VI), which approximates the difficult to compute posterior distribution p with a simpler distribution q (see Blei, Kucukelbir, and McAuliffe (2017) for a review). Variational inference casts an inference problem as an optimization problem with the goal of finding parameters to the variational distribution such that $q = q(\kappa, \mathbf{x}, \alpha, \theta, \mathbf{z}, \beta)$ closely approximates $p = p(\kappa, \mathbf{x}, \alpha, \theta, \mathbf{z}, \beta \mid \mathbf{w})$. Our regularized variational inference (RVI) algorithm seeks a distribution $q \in \mathcal{Q}$ such that

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q \parallel p) + \rho r(q) , \tag{C.1}$$

where $\mathrm{KL}(\cdot)$ is KL-Divergence. The added term $r(q)$ is a regularization function we've introduced to discourage similar personas (further detail given in Section C.2), and ρ the corresponding hyperparameter.

To make q easy to compute, we apply *mean field* variational inference which assumes that the parameters are *posteriori* independent. Under the mean field assumption the variational distribution factorizes as:

$$\begin{aligned} &\prod_{k=1}^{K} q(\beta_k \mid \lambda_k) \prod_{a=1}^{A} q(\kappa_a \mid \delta_a) \prod_{p=1}^{P} q(\alpha_{1:T,k,p} \mid \hat{\alpha}_{1:T,k,p}) \times \\ &\quad \prod_{t=1}^{T} \prod_{d=1}^{D_t} q(x_{t,d,p} \mid \tau_{t,d,p}) q(\theta_{t,d} \mid \gamma_{t,d}) \prod_{n=1}^{N_{d_t}} q(z_n \mid \phi_n) \end{aligned} \tag{C.2}$$

where we have introduced the following variational parameters: the persona for each author κ_a is endowed with a free Dirichlet parameter δ_a; each assignment of a persona

to an author $\mathbf{x}_{t,d}$ is endowed with a free Multinomial parameter $\tau_{t,d}$; in the variational distribution of $\alpha_{1:T,k,p}$ the sequential structure is kept intact with variational observations $\hat{\alpha}_{1:T,k,p}$; each document-topic proportion vector $\theta_{t,d}$ is endowed with a free γ_d. The variance for the document-topic parameters are $\nu_{t,d}$ and $\hat{\nu}_{t,d}$, for the model and variational parameter, respectively; each word-topic indicator is endowed with a free multinomial parameter $\phi_{t,d,n}$.

An optimal q cannot be computed directly, but following Jordan et al. (1999b) an optimization of the variational parameters proceeds by maximizing a bound on the log-likelihood of the data. In the DAP model, data is observed in words $\mathbf{w}$ for each document d at time step t, hence we re-write the log-likelihood of the data $\log p(\mathbf{w}_{t,d})$ by:

$$\begin{aligned}
\log p(\mathbf{w}_{t,d}) &= \log \iiiint \sum_{\mathbf{z}} \sum_{\mathbf{x}} p \;\; \partial\theta_{t,d}, \partial\alpha_t, \partial\beta, \partial\kappa \\
&= \log \iiiint \sum_{\mathbf{z}} \sum_{\mathbf{x}} \frac{pq}{q} \\
&\geqslant \iiiint \sum_{\mathbf{z}} \sum_{\mathbf{x}} q \log p - \iiiint \sum_{\mathbf{z}} \sum_{\mathbf{x}} q \log q \\
&= \mathbb{E}_q[\log p] - \mathbb{E}_q[\log q]
\end{aligned} \tag{C.3}$$

The inequality in (C.3) follows from Jensen's inequality. Moreover, it can be shown that the difference between $\log p(\mathbf{w}_{t,d})$ and $\mathbb{E}_q[\log p] - \mathbb{E}_q[\log q]$ is $\mathrm{KL}(q \parallel p)$. Hence maximizing the bound in (C.3) is equivalent to minimizing the KL divergence between the variational and true posteriors. We denote the Evidence Lower BOund (ELBO) by $\mathcal{L}(\delta_a, \tau_{t,d}, \gamma_{t,d}, \phi_n, \lambda_k) = \mathbb{E}_q[\log p] - \mathbb{E}_q[\log q]$. Since our objective defined in (C.1) includes a regularization term, we therefore maximize a surrogate likelihood consisting of the ELBO minus the regularization term (see Wainwright and Jordan (2007) for a review of penalized surrogate likelihoods). Hence our objective function $\mathcal{L}_\rho$ for some regularization ρ is defined as:

$$\mathcal{L}_\rho(\delta_a, \tau_{t,d}, \gamma_{t,d}, \phi_n, \lambda_k) \triangleq \mathbb{E}_q[\log p] - \mathbb{E}_q[\log q] - \rho r(q)\,, \tag{C.4}$$

where $\mathbb{E}_q[\log p]$ expands for each term in the model, that is:

$$
\begin{aligned}
\mathbb{E}_q[\log p] = &\sum_{k=1}^{K}\sum_{v=1}^{V}\mathbb{E}_q[\log p(\beta_{k,v} \mid \eta)] + \\
&\sum_{a=1}^{A}\sum_{p=1}^{P}\mathbb{E}_q[\log p(\kappa_{a,p} \mid \omega)] + \\
&\sum_{t=1}^{T}\Bigg[\sum_{p=1}^{P}\sum_{k=1}^{K}\mathbb{E}_q[\log p(\alpha_{t,k,p} \mid \alpha_{t-1,k,p})] + \\
&\quad\sum_{d=1}^{D_t}\Bigg[\mathbb{E}_q[\log p(\mathbf{x}_{t,d} \mid \kappa_a)] + \\
&\quad\mathbb{E}_q[\log p(\theta_{t,d} \mid \alpha_t \mathbf{x}_{t,d}, \Sigma_t)] + \\
&\quad\sum_{n=1}^{N_{d_t}}\Big[\mathbb{E}_q[\log p(\mathbf{z}_n \mid \pi(\theta_{t,d}))] + \\
&\quad\mathbb{E}_q[\log p(\mathbf{w}_{t,d,n} \mid \mathbf{z}_n, \beta)]\Big]\Bigg]\Bigg]
\end{aligned}
\tag{C.5}
$$

Similarly, $-\mathbb{E}_q[\log q]$ is the entropy term associated with each of the parameters. Some terms in (C.5) are simple, and well-known from foundational topic models like LDA and CTM Blei and Lafferty, 2006a; Blei, Ng, and Jordan, 2003. For example, the topic distributions over words $\mathbb{E}_q[\log p(\beta_{k,v} \mid \eta)]$ term is found in LDA, and in the DAP model the distributions over personas for each author $\mathbb{E}_q[\log p(\kappa_{a,p} \mid \omega)]$ follows a similar structure. Similarly, the non-conjugate pairs for word-topic assignment $\mathbb{E}_q[\log p(\mathbf{z}_n \mid \pi(\theta_{t,d}))]$ has been studied in the CTM.

Expanding the objective function $\mathcal{L}_\rho$ according to the distribution associated with each parameter allows updates to be derived for each parameter. The parameters are optimized using a variational expectation-maximization algorithm, the details of the algorithm are given below.

C.1 VARIATIONAL E-STEP

During the E-step the model estimates variational parameters for each document and saves the sufficient statistics required to compute global parameters. The structure of the DAP model, while unique, has some components that mimic previous topic models. Specifically, the word-topic assignment parameter ϕ has the same update found in the CTM due to the Logistic-Normal γ parameter. Hence ϕ has a closed form update: $\phi_{n,k} \propto \exp(\gamma_k)\beta_{k,v}$ Blei and Lafferty, 2006a.

Each author's persona is parameterized by a τ. To find an update for τ we select ELBO terms featuring τ, and then take the derivative with respect to each document and persona. The terms in the ELBO containing τ are:

$$\mathcal{L}_{[\tau]} = \sum_{t=1}^{T}\sum_{d=1}^{D_t}\sum_{p=1}^{P}\tau_{t,d,p}\Big(\Psi(\delta_{a,p}) - \Psi(\sum_{i=1}^{P}\delta_{a,i})\Big) + \\ -\frac{1}{2}\Big((\gamma_{t,d} - \hat{\alpha}_t\tau_{t,d})^\top \Sigma_t^{-1}(\gamma_{t,d} - \hat{\alpha}_t\tau_{t,d}) + \\ \operatorname{Tr}(\Sigma_t^{-1}\operatorname{diag}(\tau_{t,d,p}(\hat{\alpha}^2_{t,p,k} + \hat{\Sigma}_{t,k,k})))\Big) + \\ -\tau_{t,d,p}\log\tau_{t,d,p} + \lambda_{t,d}(\sum_{i=1}^{P}\tau_{t,d,i} - 1),$$

where we've included the Lagrange constraint $\sum_{t=1}^{T}\sum_{d=1}^{D_t}\lambda_{t,d}(\sum_{i=1}^{P}\tau_{t,d,i} - 1)$ since each vector $\tau_{t,d}$ must sum to one. Takings the derivative with respect to one specific document and persona we find that:

$$\frac{\partial\mathcal{L}}{\partial\tau_{t,d,p}} = \Psi(\delta_{a,p}) - \Psi(\sum_{i=1}^{P}\delta_{a,i}) - \log\tau_{a,p} - 1 + \lambda + \\ \hat{\alpha}_{t,p}\Sigma_t^{-1}(\gamma_{t,d} - \hat{\alpha}_{t,p}\tau_{t,d,p}) - \frac{1}{2}\operatorname{Tr}(\Sigma_t^{-1}\operatorname{diag}(\hat{\alpha}^2_{t,p} + \hat{\Sigma}_t))$$

A closed form solution for $\tau_{t,d}$ doesn't exist. We therefore estimate $\tau_{t,d}$ using exponential gradient descent.

Since the model includes non-conjugate terms (much like DTM, CDTM, and CTM), an additional variational parameter ζ is introduced to preserve the lower-bound during the expansion of the term containing a non-conjugate pair: $\mathbb{E}_q[\log p(\mathbf{z}_n \mid \pi(\theta_{t,d}))]$. Taking the derivative of all terms containing ζ and setting it to zero yields an analogous closed form update to the one found in the CTM: $\hat{\zeta}_t = \sum_{k=1}^{K}\exp(\gamma_{t,d,k} + \hat{v}^2_{t,k}/2)$

Finally, the DAP model estimates a topic distribution for each document via the $\gamma_{t,d}$ parameter. To update $\gamma_{t,d}$ the terms in the ELBO featuring $\gamma_{t,d}$ are selected:

$$\mathcal{L}(\gamma_{t,d}) = \sum_{t=1}^{T}\sum_{d=1}^{D_t} -\frac{1}{2}(\gamma_{t,d} - \hat{\alpha}_t\tau_{t,d})^\top\Sigma_t^{-1}(\gamma_{t,d} - \hat{\alpha}_t\tau_{t,d}) + \\ \sum_{n=1}^{N_{d_t}}\gamma_{t,d}\phi_n - \zeta^{-1}(\sum_{k=1}^{K}\exp(\gamma_{t,d,k} + \hat{v}^2_t/2))$$

Taking a derivative of these terms with respect to $\gamma_{t,d,k}$ yields:

$$\frac{\partial\mathcal{L}}{\partial\gamma_{t,d,k}} = -\Sigma_t^{-1}(\gamma_{t,d,k} - \hat{\alpha}_{t,1:P,k}\tau_{t,d,k}) + \\ \sum_{n=1}^{N_{d_t}}\phi_{n,k} - \frac{N_{d_t}}{\zeta}\exp(\gamma_{t,d,k} + \hat{v}^2_{t,k}/2)) \tag{C.6}$$

Since a closed form solution isn't available, a conjugate gradient algorithm is run using the gradient in (C.6).

Whereas $\gamma_{t,d}$ represents the mean of the Logistic-Normal for a document's topic distribution, the parameter $\hat{v}_{t,d}$ is the variance. The ELBO terms featuring $\hat{v}_{t,d}$ are:

$$\mathcal{L}(\hat{v}_t) = \mathrm{Tr}(\Sigma_t^{-1}\hat{v}) \sum_{k=1}^{K} \frac{1}{2}(\log \hat{v}_k^2 + \log 2\pi) - N_{d_t}\zeta^{-1} \sum_{k=1}^{K} \exp(\gamma_{t,d,k} + \hat{v}_{t,k}^2/2)$$

Therefore, setting the derivative of $\mathcal{L}(\hat{v}_{t,d})$ with respect to $\hat{v}_{t,d}$ to zero and solving yields:

$$\frac{\partial \mathcal{L}}{\partial v_{t,d,k}^2} = \Sigma_{t,k,k}^{-1} + \frac{1}{2\hat{v}_{t,d,k}^2} - \frac{N_{d_t}}{2\zeta} \exp(\gamma_{t,d,k} + \hat{v}_{t,k}^2/2),$$

which requires Newton's method for each coordinate, constrained such that $\hat{v}_{t,k} > 0, \forall k$.

The parameter $\hat{\alpha}_t$ represents the noisy estimate of α_t. After calculating $\hat{\alpha}_t$, the forward and backward equations will be applied in the M-step to give a final posterior estimate α_t. The terms in the ELBO containing $\hat{\alpha}_t$ are found by expanding $\mathbb{E}_q[\log p(\alpha_{t,p}) \mid \alpha_{t-1,p})]$ for (C.7a) and $\mathbb{E}_q[\log p(\theta_{t,d} \mid \alpha_t \mathbf{x}_{t,d}, \Sigma_t)]$ for (C.7b) and (C.7c):

$$\mathcal{L}(\hat{\alpha}) = \sum_{t=1}^{T}\sum_{p=1}^{P} -\frac{1}{2}(\hat{\alpha}_{t,p} - \hat{\alpha}_{t-1,p})^\top \Sigma_t^{-1}(\hat{\alpha}_{t,p} - \hat{\alpha}_{t-1,p}) + \tag{C.7a}$$

$$\sum_{t=1}^{T}\sum_{d=1}^{D_t} -\frac{1}{2}\Big((\gamma_{t,d} - \hat{\alpha}_t\tau_{t,d})^\top \Sigma_t^{-1}(\gamma_{t,d} - \hat{\alpha}_t\tau_{t,d}) + \tag{C.7b}$$

$$\sum_{t=1}^{T}\sum_{d=1}^{D_t}\sum_{p=1}^{P} \mathrm{Tr}\left[\Sigma_t^{-1}\,\mathrm{diag}\left(\tau_{t,d,p}(\hat{\alpha}_{t,p}\hat{\alpha}_{t,p}^\top + \hat{\Sigma}_t)\right)\right]\Big) \tag{C.7c}$$

Taking the derivative with respect to the mean term for each persona gives the closed form update:

$$\hat{\alpha}_{t,p} = \frac{\hat{\alpha}_{t-1,p} + \sum_{d=1}^{D_t}(\gamma_{t,d} - 1)\tau_{t,d,p}}{1 + \sum_{d=1}^{D_t}\tau_{t,d,p}^2} \tag{C.8}$$

We solve for $\hat{\alpha}_{t,p}$ sequentially over time steps. For the initial time step $t = 1$, we use the prior μ_0 in place of $\hat{\alpha}_{t-1,p}$. Note that the summations in (C.8) are collected during the E-step and $\hat{\alpha}_{t,p}$ need only be computed once after performing inference on all documents.

C.2 REGULARIZED VARIATION INFERENCE

Our RVI algorithm nudges α_t to find topic distributions that are different for each persona. A natural choice for capturing this idea is an inner product between each of the personas (excluding a persona with itself). Hence, we define the regularization function by:

$$r(q) = \sum_{p=1}^{P} \sum_{1 \leqslant q \leqslant P, q \neq p} \frac{D_t}{2} \rho \hat{\alpha}_{t,p}^{\top} \Sigma_t^{-1} \hat{\alpha}_{t,q} \ , \tag{C.9}$$

The parameter Σ_t^{-1} in included in the regularization for two reasons. First, it simplifies the update to $\hat{\alpha}_{t,p}$. In (C.7) the term Σ^{-1} appears in every term, which allows it to be factored out and canceled. By including Σ^{-1} in the regularization the same cancellation can occur. Second, since $\Sigma_t^{-1} \propto I$ then its inclusion has the effect of encouraging personas to be orthogonal to one another. We include the number of documents D_t at time t in $r(q)$ so that the regularization is applied evenly, regardless of dataset size or a skewed distribution of documents over time. After including the regularization term in (C.9) with the ELBO terms in (C.7), the regularized $\hat{\alpha}_{t,p}$ update is:

$$(1 + \sum_{d=1}^{D_t} \tau_{d,p}^2)\hat{\alpha}_{t,p} + \rho D_t \sum_{q \neq p} \hat{\alpha}_{t,q} = \hat{\alpha}_{t-1,p} + \sum_{d=1}^{D_t} (\gamma_d - 1)\tau_{d,p} \tag{C.10}$$

Since the vector $\sum_{d=1}^{D_t}(\gamma_d - 1)\tau_{d,p}$ (of length K) is computed during the E-step, then the RHS is known. Similarly, the term $(1 + \sum_{d=1}^{D_t} \tau_{d,p}^2)$ is known, and in combination with ρD_t form the weights over the unknown vector $\hat{\alpha}_{t,p}$, also of length K. Therefore, (C.10) can be solved as a system of linear equations. Through experiments we've found an optimal value of $\rho \in [0, 0.5]$. The model exhibits sensitivity to the hyperparameter ρ, if ρ is large (e.g. > 1.0) then model quality drops due to personas overfitting to a single topic. Since $\hat{\alpha}$ is only used to estimate the global parameter α during the M-step, computing $\hat{\alpha}$ isn't necessary for inference on holdout datasets.

C.3 M-STEP

In the M-step the global parameters α, κ, and β are updated such that the lower bound of the log likelihood of the data is maximized. Note, the update for β is exactly the same as derived for the LDA model, and hence omitted.

The parameter δ represents the distribution over personas for each author. The closed form update for $\delta_{a,p}$ is:

$$\delta_{a,p} \propto \omega_p + \sum_{t=1}^{T} \sum_{d=1}^{D_t} \tau_{t,d,p}$$

shows that δ's closed form update is an average of the persona assignments, smoothed by the author-persona prior ω.

Once the variational observations $\hat{\alpha}_{t,p}$ are computed, our approach follows the variational Kalman filtering method from Wang's Continuous Time Dynamic Topic Model, see Appendix for further details. Specifically, we employ the Brownian motion to model time dynamics. However, because the DAP model's time-varying parameter is a distribution over latent topics, it performs best on data discretized in time (resulting in a smaller T). The forward equations mimic a Kalman filter:

$$m_{t,p} = \frac{\hat{\alpha}_{t,p} P_{t,p} + m_{t-1,p} \hat{w}_t}{P_{t,p} + \hat{w}_t}$$
$$V_{t,p} = \hat{w}_t \frac{P_{t,p}}{P_{t,p} + \hat{w}_t}$$

where $\hat{w}_t$ is the known process noise, and $P_{t,p} = V_{t-1,p} + \sigma\Delta_{s_t}$ captures the increase in variance as time between data points grows. Finally, the backward equations:

$$\alpha_{t-1,p} = m_{t-1,p} \frac{\sigma\Delta_{s_t}}{P_{t,p}} + \alpha_{t,p} \frac{V_{t-1,p}}{P_{t,p}}$$
$$\Sigma_{t-1,p} = V_{t-1,p} + \frac{(V_{t-1,p})^2}{(P_{t,p})^2} (\Sigma_{t,p} - P_{t,p}) \,,$$

give the updates to the remaining global parameters.

DECLARATION

I hereby declare that this thesis represents my own work which has been done after registration for the degree of Doctor of Philosophy at the University of Minnesota, and has not been previously included in a thesis or dissertation submitted to this or any other institution for a degree, diploma or other qualifications.

Minneapolis, Minnesota, July 2021

Robert Giaquinto