Auxiliary Guided Autoregressive Variational Autoencoders

Thomas Lucas(✉) and Jakob Verbeek

Université. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
{thomas.lucas,jakob.verbeek}@inria.fr

Abstract. Generative modeling of high-dimensional data is a key problem in machine learning. Successful approaches include latent variable models and autoregressive models. The complementary strengths of these approaches, to model global and local image statistics respectively, suggest hybrid models that encode global image structure into latent variables while autoregressively modeling low level detail. Previous approaches to such hybrid models restrict the capacity of the autoregressive decoder to prevent degenerate models that ignore the latent variables and only rely on autoregressive modeling. Our contribution is a training procedure relying on an auxiliary loss function that controls which information is captured by the latent variables and what is left to the autoregressive decoder. Our approach can leverage arbitrarily powerful autoregressive decoders, achieves state-of-the art quantitative performance among models with latent variables, and generates qualitatively convincing samples.

1 Introduction

Unsupervised modeling of complex distributions with unknown structure is a landmark challenge in machine learning. The problem is often studied in the context of learning generative models of the complex high-dimensional distributions of natural image collections. Latent variable approaches can learn disentangled and concise representations of the data [3], which are useful for compression [11] and semi-supervised learning [14,22]. When conditioned on prior information, generative models can be used for a variety of tasks, such as attribute or class-conditional image generation, text and pose-based image generation, image colorization, *etc.* [6,20,23,26]. Recently significant advances in generative (image) modeling have been made along several lines, including adversarial networks [1,10], variational autoencoders [16,24], autoregressive models [21,23], and non-volume preserving variable transformations [8].

In our work we seek to combine the merits of two of these lines of work. Variational autoencoders (VAEs) [16,24] can learn latent variable representations that abstract away from low-level details, but model pixels as conditionally independent given the latent variables. This renders the generative model computationally efficient, but the lack of low-level structure modeling leads to overly smooth and blurry samples. Autoregressive models, such as pixelCNNs [21], on

© Springer Nature Switzerland AG 2019
M. Berlingerio et al. (Eds.): ECML PKDD 2018, LNAI 11051, pp. 443–458, 2019.
https://doi.org/10.1007/978-3-030-10925-7_27

the other hand, estimate complex translation invariant conditional distributions among pixels. They are effective to model low-level image statistics, and yield state-of-the-art likelihoods on test data [25]. This is in line with the observations of [17] that low-level image details account for a large part of the likelihood. These autoregressive models, however, do not learn a latent variable representations to support, *e.g.*, semi-supervised learning.

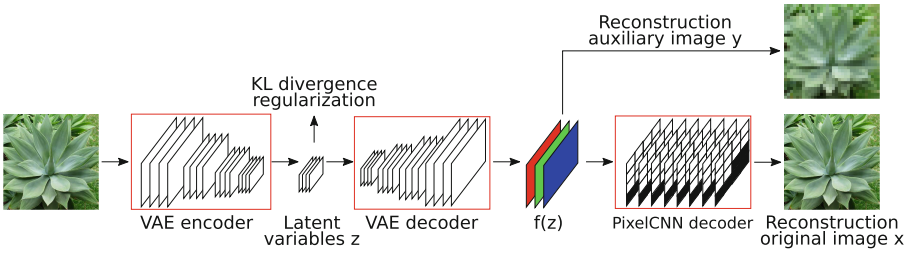


Fig. 1. Schematic illustration of our auxiliary guided autoregressive variational autoencoder (AGAVE). The objective function has three components: KL divergence regularization, per-pixel reconstruction with the VAE decoder, and autoregressive reconstruction with the pixelCNN decoder.

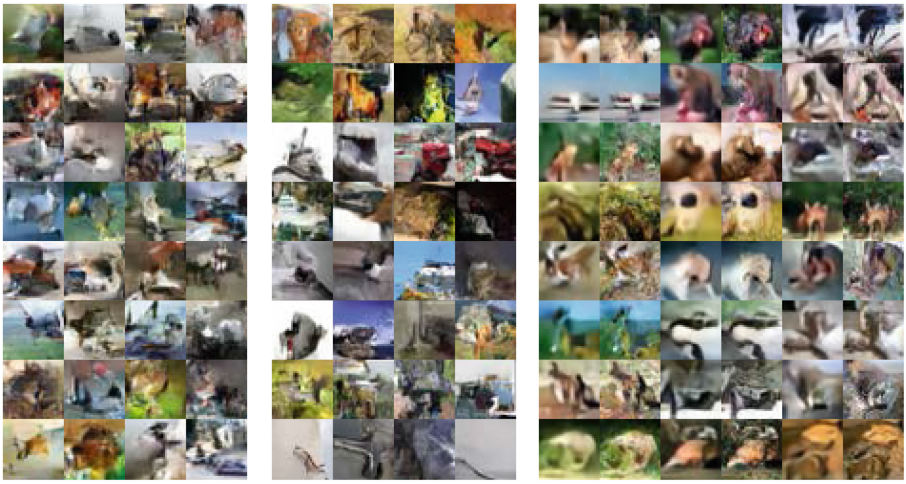
The complementary strengths of VAEs and pixelCNNs, modeling global and local image statistics respectively, suggest hybrid approaches combining the strengths of both. Prior work on such hybrid models needed to limit the capacity of the autoregressive decoder to prevent degenerate models that completely ignore the latent variables and rely on autoregressive modeling only [5,12]. In this paper we describe Auxiliary Guided Autoregressive Variational autoEncoders (AGAVE), an approach to train such hybrid models using an auxiliary loss function that controls which information is captured by the latent variables and what is left to the AR decoder. That removes the need to limit the capacity of the latter. See Fig. 1 for a schematic illustration of our approach.

Using high-capacity VAE and autoregressive components allows our models to obtain quantitative results on held-out data that are on par with the state of the art in general, and set a new state of the art among models with latent variables. Our models generate samples with both global coherence and low-level details. See Fig. 2 for representative samples of VAE and pixelCNN models.

2 Related Work

Generative image modeling has recently taken significant strides forward, leveraging deep neural networks to learn complex density models using a variety of approaches. These include the variational autoencoders and autoregressive models that form the basis of our work, but also generative adversarial networks (GANs) [1,10] and variable transformation with invertible functions [8].

While GANs produce visually appealing samples, they suffer from mode dropping and their likelihood-free nature prevents measuring how well they model held-out test data. In particular, GANs can only generate samples on a non-linear manifold in the data space with dimension equal to the number of latent variables. In contrast, probabilistic models such as VAEs and autoregressive models generalize to the entire data space, and likelihoods of held-out data can be used for compression, and to quantitatively compare different models. The non-volume preserving (NVP) transformation approach of [8] chains together invertible transformations to map a basic (*e.g.* unit Gaussian) prior on the latent space to a complex distribution on the data space. This method offers tractable likelihood evaluation and exact inference, but obtains likelihoods on held-out data below the values reported using state-of-the-art VAE and autoregressive models. Moreover, it is restricted to use latent representations with the same dimensionality as the input data, and is thus difficult to scale to model high-resolution images.



(a) (b) (c)

Fig. 2. Randomly selected samples from unsupervised models trained on 32×32 CIFAR10 images: (a) IAF-VAE [15], (b) pixelCNN++ [25], and (c) our hybrid AGAVE model. For our model, we show the intermediate high-level representation based on latent variables (left), that conditions the final sample based on the pixelCNN decoder (right).

Autoregressive density estimation models, such as pixelCNNs [21], admit tractable likelihood evaluation, while for variational autoencoders [16,24] accurate approximations can be obtained using importance sampling [4]. Naively combining powerful pixelCNN decoders in a VAE framework results in a degenerate model which ignores the VAE latent variable structure, as explained through

the lens of bits-back coding by [5]. To address this issue, the capacity of the the autoregressive component can be restricted. This can, for example, be achieved by reducing its depth and/or field of view, or by giving the pixelCNN only access to grayscale values, *i.e.* modeling $p(x_i|\mathbf{x}_{<i}, \mathbf{z}) = p(x_i|\text{gray}(\mathbf{x}_{<i}), \mathbf{z})$ [5,12]. This forces the model to leverage the latent variables $\mathbf{z}$ to model part of the dependencies among the pixels. This approach, however, has two drawbacks. (i) Curbing the capacity of the model is undesirable in unsupervised settings where training data is abundant and overfitting unlikely, and is only a partial solution to the problem. (ii) Balancing what is modeled by the VAE and the pixelCNN by means of architectural design choices requires careful hand-design and tuning of the architectures. This is a tedious process, and a more reliable principle is desirable. To overcome these drawbacks, we propose to instead control what is modeled by the VAE and pixelCNN with an auxiliary loss on the VAE decoder output before it is used to condition the autoregressive decoder. This allows us to "plug in" powerful high-capacity VAE and pixelCNN architectures, and balance what is modeled by each component by means of the auxiliary loss.

In a similar vein, [17] force pixelCNN models to capture more high-level image aspects using an auxiliary representation $\mathbf{y}$ of the original image $\mathbf{x}$, *e.g.* a low-resolution version of the original. They learn a pixelCNN for $\mathbf{y}$, and a conditional pixelCNN to predict $\mathbf{x}$ from $\mathbf{y}$, possibly using several intermediate representations. This approach forces modeling of more high-level aspects in the intermediate representations, and yields visually more compelling samples. [23] similarly learn a series of conditional autoregressive models to upsample coarser intermediate latent images. By introducing partial conditional independencies in the model they scale the model to efficiently sample high-resolution images of up to 512×512 pixels. [11] use a recurrent VAE model to produces a sequence of RGB images with increasing detail derived from latent variables associated with each iteration. Like our work, all these models work with intermediate representations in RGB space to learn accurate generative image models.

3 Auxiliary Guided Autoregressive Variational Autoencoders

We give a brief overview of variational autoencoders and their limitations in Sect. 3.1, before we present our approach to learning variational autoencoders with autoregressive decoders in Sect. 3.2.

3.1 Variational Autoencoders

Variational autoencoders [16,24] learn deep generative latent variable models using two neural networks. The "decoder" network implements a conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ over observations $\mathbf{x}$ given a latent variable $\mathbf{z}$, with parameters $\boldsymbol{\theta}$. Together with a basic prior on the latent variable $\mathbf{z}$, *e.g.* a unit Gaussian, the generative model on $\mathbf{x}$ is obtained by marginalizing out the latent variable:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \, d\mathbf{z}. \tag{1}$$

The marginal likelihood can, however, not be optimized directly since the non-linear dependencies in $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ render the integral intractable. To overcome this problem, an "encoder" network is used to compute an approximate posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, with parameters $\boldsymbol{\phi}$. The approximate posterior is used to define a variational bound on the data log-likelihood, by subtracting the Kullback-Leibler divergence between the true and approximate posterior:

$$\ln p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = \ln(p_{\boldsymbol{\theta}}(\mathbf{x})) - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \tag{2}$$

$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\ln(p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]}_{\text{Reconstruction}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{\text{Regularization}}. \tag{3}$$

The decomposition in (3) interprets the bound as the sum of a reconstruction term and a regularization term. The first aims to maximize the expected data log-likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ given the posterior estimate $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. The second term prevents $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ from collapsing to a single point, which would be optimal for the first term.

Variational autoencoders typically model the dimensions of $\mathbf{x}$ as conditionally independent,

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^{D} p_{\boldsymbol{\theta}}(x_i|\mathbf{z}), \tag{4}$$

for instance using a factored Gaussian or Bernoulli model, see *e.g.* [15,16,26]. The conditional independence assumption makes sampling from the VAE efficient: since the decoder network is evaluated only once for a sample $\mathbf{z} \sim p(\mathbf{z})$ to compute all the conditional distributions $p_{\boldsymbol{\theta}}(x_i|\mathbf{z})$, the x_i can then be sampled in parallel.

A result of relying on the latent variables to account for all pixel dependencies, however, is that all low-level variability must also be modeled by the latent variables. Consider, for instance, a picture of a dog, and variants of that image shifted by one or a few pixels, or in a slightly different pose, with a slightly lighter background, or with less saturated colors, *etc.* If these factors of variability are modeled using latent variables, then these low-level aspects are confounded with latent variables relating to the high-level image content. If the corresponding image variability is not modeled using latent variables, it will be modeled as independent pixel noise. In the latter case, using the mean of $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ as the synthetic image for a given $\mathbf{z}$ results in blurry samples, since the mean is averaged over the low-level variants of the image. Sampling from $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ to obtain synthetic images, on the other hand, results in images with unrealistic independent pixel noise.

3.2 Autoregressive Decoders in Variational Autoencoders

Autoregressive density models, see *e.g.* [9,19], rely on the basic factorization of multi-variate distributions,

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \prod_{i=1}^{D} p_{\boldsymbol{\theta}}(x_i|\mathbf{x}_{<i}) \tag{5}$$

with $\mathbf{x}_{<i} = x_1, \ldots, x_{i-1}$, and model the conditional distributions using a (deep) neural network. For image data, PixelCNNs [20,21] use a scanline pixel ordering, and model the conditional distributions using a convolution neural network. The convolutional filters are masked so as to ensure that the receptive fields only extend to pixels $\mathbf{x}_{<i}$ when computing the conditional distribution of x_i.

PixelCNNs can be used as a decoder in a VAE by conditioning on the latent variable $\mathbf{z}$ in addition to the preceding pixels, leading to a variational bound with a modified reconstruction term:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\sum_{i=1}^{D} \ln p_{\boldsymbol{\theta}}(x_i|\mathbf{x}_{<i}, \mathbf{z})\right] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})). \tag{6}$$

The regularization term can be interpreted as a "cost" of using the latent variables. To effectively use the latent variables, the approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ must differ from the prior $p(\mathbf{z})$, which increases the KL divergence.

[5] showed that for loss (6) and a decoder with enough capacity, it is optimal to encode no information about x in z by setting $q(z|x) = p(z)$. To ensure meaningful latent representation learning [5,12] restrict the capacity of the pixelCNN decoder. In our approach, in contrast, it is always optimal for the autoregressive decoder, regardless of its capacity, to exploit the information on $\mathbf{x}$ carried by z. We rely on two decoders in parallel: the first one reconstructs an auxiliary image $\mathbf{y}$ from an intermediate representation $f_{\boldsymbol{\theta}}(\mathbf{z})$ in a non-autoregressive manner. The auxiliary image can be either simply taken to be the original image ($\mathbf{y} = \mathbf{x}$), or a compressed version of it, *e.g.* with lower resolution or with a coarser color quantization. The second decoder is a conditional autoregressive model that predicts $\mathbf{x}$ conditioned on $f_{\boldsymbol{\theta}}(\mathbf{z})$. Modeling $\mathbf{y}$ in a non-autoregressive manner ensures a meaningful representation $\mathbf{z}$ and renders $\mathbf{x}$ and $\mathbf{z}$ dependent, inducing a certain non-zero KL "cost" in (6). The uncertainty on $\mathbf{x}$ is thus reduced when conditioning on $\mathbf{z}$, and there is no longer an advantage in ignoring the latent variable for the autoregressive decoder. We provide a more detailed explanation of why our auxiliary loss ensures a meaningful use of latent variables in powerful decoders in Sect. 3.3. To train the model we combine both decoders in a single objective function with a shared encoder network:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{y}) = \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\sum_{i=1}^{D} \ln p_{\boldsymbol{\theta}}(x_i|\mathbf{x}_{<i}, \mathbf{z})\right]}_{\text{Primary Reconstruction}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\sum_{j=1}^{E} \ln p_{\boldsymbol{\theta}}(y_j|\mathbf{z})\right]}_{\text{Auxiliary Reconstruction}} \\ - \underbrace{\lambda\, D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right)}_{\text{Regularization}}.\end{aligned} \tag{7}$$

Treating $\mathbf{x}$ and $\mathbf{y}$ as two variables that are conditionally independent given a shared underlying latent variable $\mathbf{z}$ leads to $\lambda = 1$. Summing the lower bounds in Eqs. (3) and (6) of the marginal log-likelihoods of $\mathbf{y}$ and $\mathbf{x}$, and sharing the encoder network, leads to $\lambda = 2$. Larger values of λ result in valid but less tight lower bounds of the log-likelihoods. Encouraging the variational posterior to be closer to the prior, this leads to less informative latent variable representations.

Sharing the encoder across the two decoders is the key of our approach. The factored auxiliary VAE decoder can only model pixel dependencies by means of the latent variables, which ensures that a meaningful representation is learned. Now, given that the VAE encoder output is informative on the image content, there is no incentive for the autoregressive decoder to ignore the intermediate representation $f(\mathbf{z})$ on which it is conditioned. The choice of the regularization parameter λ and auxiliary image $\mathbf{y}$ provide two levers to control *how much* and *what type* of information should be encoded in the latent variables.

3.3 It Is Optimal for the Autoregressive Decoder to Use z

Combining a VAE with a flexible decoder (for instance an autoregressive one) leads to the latent code being ignored. This problem could be attributed to optimization challenges: at the start of training $q(\mathbf{z}|\mathbf{x})$ carries little information about $\mathbf{x}$, the KL term pushes the model to set it to the prior to avoid any penalty, and training never recovers from falling into that local minimum. [5] have proposed extensive explanations showing that the problem goes deeper: if a sufficiently expressive decoder is used, ignoring the latents actually is the optimal behavior. The gist of the argument is based on bits-back coding as follows: given an encoder $q(\mathbf{z}|\mathbf{x})$, a decoder $p(\mathbf{x}|\mathbf{z})$ and a prior $p(\mathbf{z})$, $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$ can be encoded in a lossless manner using $p(\mathbf{z})$, and $\mathbf{x}$ can be encoded, also losslessly, using $p(\mathbf{x}|\mathbf{z})$. Once the receiver has decoded $\mathbf{x}$, $q(\mathbf{z}|\mathbf{x})$ becomes available and a secondary message can be decoded from it. This yields and average code length of:

$$C_{BitsBack} = \mathbb{E}_{\mathbf{x}\sim D, \mathbf{z}\sim q(.|\mathbf{x})}[\log(q(\mathbf{z}|\mathbf{x})) - \log(p(\mathbf{z})) - \log(p(\mathbf{x}|\mathbf{z}))].$$

$C_{BitsBack}$ corresponds to the standard VAE objective. A lower-bound on the expected code length for the data being encoded is given by the Shannon entropy: $\mathcal{H}(D) = \mathbb{E}_{\mathbf{x}\sim D}[-\log p_D(\mathbf{x})]$, which yields:

$$\begin{aligned} C_{BitsBack} &= \mathbb{E}_{\mathbf{x}\sim D}[-\log(p(\mathbf{x})) + D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))] \\ &\geq \mathcal{H}(D) + \mathbb{E}_{\mathbf{x}\sim D}[D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))]. \end{aligned}$$

If $p(.|\mathbf{x}_{j<i})$ is expressive enough, or if $q(.|\mathbf{x})$ is poor enough, the following inequality can be verified:

$$\mathcal{H}(D) \leq \mathbb{E}_{\mathbf{x}\sim D}[-\log p(\mathbf{x}|\mathbf{x}_{j<i})] < \mathcal{H}(D) + \mathbb{E}_{\mathbf{x}\sim D}[D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))]$$

This is always true in the limit of infinitely expressive autoregressive decoders. In that case, any use of the latents that p might decrease performance. The optimal

behavior is to set $q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z})$ to avoid the extra KL cost. Then $\mathbf{z}$ becomes independent from $\mathbf{x}$ and no information about $\mathbf{x}$ is encoded in $\mathbf{z}$. Therefore, given an encoder, the latent variables will only be used if the capacity of the autoregressive decoder is sufficiently restricted. This is the approach taken by [5,12]. This approach works: it has obtained competitive quantitative and qualitative performance. However, it is not satisfactory in the sense that autoregressive models cannot be used to the full extent of their potential, while learning a meaningful latent variable representation.

In our setting, both (Y, X) have to be sent to and decoded by the receiver. Let us denote C_{VAE} the expected code length required to send the auxiliary message, $\mathbf{y}$. Once $\mathbf{y}$ has been sent, sending $\mathbf{x}$ costs: $\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}[-\sum_i \log(p(x_i|\mathbf{z}, \mathbf{x}_{j<i}))]$, and we have:

$$C_{VAE} = \mathbb{E}_{\mathbf{x}\sim D, \mathbf{z}\sim q(.|\mathbf{x})}[\log(q(\mathbf{z}|\mathbf{x})) - \log(p(\mathbf{z})) - \log(p(\mathbf{y}|\mathbf{z}))] \tag{8}$$

$$C_{AGAVE} = C_{VAE} + \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}[-\sum_i \log(p(x_i|\mathbf{z}, \mathbf{x}_{j<i}))]. \tag{9}$$

Using the fact that the Shannon entropy is the optimal expected code length for transmitting $X|Z$, we obtain $C_{AGAVE} \geq C_{VAE} + \mathcal{H}(X|Z)$.

The entropy of a random variable decreases when it is conditioned on another, i.e. $\mathcal{H}(X|Z) \leq \mathcal{H}(X)$. Therefore, the theoretical lower-bound on the expected code length in our setup is always better when the autoregressive component takes Z into account, no matter its expressivity. In the limit case of an infinitely expressive autoregressive decoder, denoted by $*$, the lower bound is attained and $C^*_{AGAVE} = C_{VAE} + \mathcal{H}(X|Z) \leq C_{VAE} + \mathcal{H}(X)$. In non-degenerate cases, the VAE is optimized to encode information about X into a meaningful Z, with potentially near perfect reconstructions, and there exists $\epsilon > 0$ such that $\mathcal{H}(X|Z) < \mathcal{H}(X) - \epsilon$, making the lower bound strictly better by a possibly big margin.

This analysis shows that in our setup it is theoretically always better for the autoregressive model to make use of the latent and auxiliary representation it is conditioned on. That is true no matter how expressive the model is. It also shows that in theory our model should learn meaningful latent structure.

4 Experimental Evaluation

In this section we describe our experimental setup, and present results on CIFAR10.

4.1 Dataset and Implementation

The CIFAR10 dataset [18] contains 6,000 images of 32×32 pixels for each of the 10 object categories *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck*. The images are split into 50,000 training images and 10,000 test images. We train all our models in a completely unsupervised manner, ignoring the class information.

We implemented our model based on existing architectures. In particular we use the VAE architecture of [15], and use logistic distributions over the RGB color values. We let the intermediate representation $f(\mathbf{z})$ output by the VAE decoder be the per-pixel and per-channel mean values of the logistics, and learn per-channel scale parameters that are used across all pixels. The cumulative density function (CDF), given by the sigmoid function, is used to compute probabilities across the 256 discrete color levels, or fewer if a lower quantization level is chosen in $\mathbf{y}$. Using RGB values $y_i \in [0, 255]$, we let b denote the number of discrete color levels and define $c = 256/b$. The probabilities over the b discrete color levels are computed from the logistic mean and variance μ_i and s_i as

$$p(y_i|\mu_i, s_i) = \sigma\left(c + c\lfloor y_i/c \rfloor|\mu_i, s_i\right) - \sigma\left(c\lfloor y_i/c \rfloor|\mu_i, s_i\right). \tag{10}$$

Table 1. Bits per dimension (lower is better) of models on the CIFAR10 test data.

Model	BPD	$.\|z$	$.\|x_{j<i}$
NICE [7]	4.48	✓	
Conv. DRAW [11]	≤3.58	✓	
Real NVP [8]	3.49	✓	
MatNet [2]	≤3.24	✓	
PixelCNN [21]	3.14		✓
VAE-IAF [15]	≤3.11	✓	
Gated pixelCNN [20]	3.03		✓
Pixel-RNN [21]	3.00		✓
Aux. pixelCNN [17]	2.98		✓
Lossy VAE [5]	≤2.95	✓	✓
AGAVE, $\lambda = 12$ (this paper)	≤2.92	✓	✓
pixCNN++ [25]	2.92		✓

For the pixelCNN we use the architecture of [25], and modify it to be conditioned on the VAE decoder output $f(\mathbf{z})$, or possibly an upsampled version if $\mathbf{y}$ has a lower resolution than $\mathbf{x}$. In particular, we apply standard non-masked convolutional layers to the VAE output, as many as there are pixelCNN layers. We allow each layer of the pixel-CNN to take additional input using non-masked convolutions from the feature stream based on the VAE output. This ensures that the conditional pixelCNN remains autoregressive.

To speed up training, we independently pretrain the VAE and pixelCNN in parallel, and then continue training the full model with both decoders. We use the Adamax optimizer [13] with a learning rate of 0.002 without learning rate decay. We will release our TensorFlow-based code to replicate our experiments upon publication.

4.2 Quantitative Performance Evaluation

Following previous work, we evaluate models on the test images using the bits-per-dimension (BPD) metric: the negative log-likelihood divided by the number of pixels values ($3 \times 32 \times 32$). It can be interpreted as the average number of bits per RGB value in a lossless compression scheme derived from the model.

Fig. 3. Effect of the regularization parameter λ. Reconstructions (a) and samples (b) of the VAE decoder (VR and VS, respectively) and corresponding conditional samples from the pixelCNN (PS).

The comparison in Table 1 shows that our model performs on par with the state-of-the-art results of the pixelCNN++ model [25]. Here we used the importance sampling-based bound of [4] with 150 samples to compute the BPD metric for our model.[1] We refer to Fig. 2 for qualitative comparison of samples from our model and pixelCNN++, the latter generated using the publicly available code.

4.3 Effect of KL Regularization Strength

In Fig. 3 we show reconstructions of test images and samples generated by the VAE decoder, together with their corresponding conditional pixelCNN samples for different values of λ. As expected, the VAE reconstructions become less accurate for larger values of λ, mainly by lacking details while preserving the global shape of the input. At the same time, the samples become more appealing for larger λ, suppressing the unrealistic high-frequency detail in the VAE samples

[1] The graphs in Figs. 4 and 8 are based on the bound in Eq. (7) to reduce the computational effort.

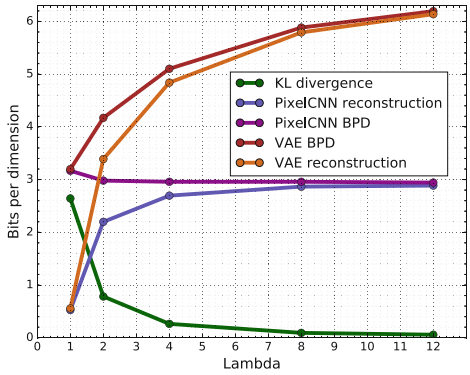


Fig. 4. Bits per dimension of the VAE decoder and pixelCNN decoder, as well as decomposition in KL regularization and reconstruction terms.

obtained at lower values of λ. Note that the VAE samples and reconstructions become more similar as λ increases, which makes the input to the pixelCNN during training and sampling more consistent.

For both reconstructions and samples, the pixelCNN clearly takes into account the output of the VAE decoder, demonstrating the effectiveness of our auxiliary loss to condition high-capacity pixelCNN decoders on latent variable representations. Samples from the pixelCNN faithfully reproduce the global structure of the VAE output, leading to more realistic samples, in particular for higher values of λ.

For $\lambda = 2$ the VAE reconstructions are near perfect during training, and the pixelCNN decoder does not significantly modify the appearance of the VAE output. For larger values of λ, the pixelCNN clearly adds significant detail to the VAE outputs.

Figure 4 traces the BPD metrics of both the VAE and pixelCNN decoder as a function of λ. We also show the decomposition in regularization and reconstruction terms. By increasing λ, the KL divergence can be pushed closer to zero. As the KL divergence term drops, the reconstruction term for the VAE rapidly increases and the VAE model obtains worse BPD values, stemming from the inability of the VAE to model pixel dependencies other than via the latent variables. The reconstruction term of the pixelCNN decoder also increases with λ, as the amount of information it receives drops. However, in terms of BPD which sums KL divergence and pixelCNN reconstruction, a substantial gain of 0.2 is observed increasing λ from 1 to 2, after which smaller but consistent gains are observed.

4.4 Role of the Auxilliary Representation

The Auxilliary Variables are Taken into Account: Sect. 3.3 shows that in theory it is always optimal for the autoregressive decoder to take the latent variables

into account. Figure 5 demonstrates this empirically by displaying auxiliary representations $f(\mathbf{z})$ with z sampled from the prior $f(z)$ as well as nine different samples from the autoregressive decoder conditioned on $f(\mathbf{z})$. This qualitatively shows that the low level detail added by the pixelCNN, which is crucial for log-likelihood performance, always respects the global structure of the image being conditioned on. The VAE decoder is trained with $\lambda = 8$ and weights very little in terms of KL divergence. Yet it controls the global structure of the samples, which shows that our setup can be used to get the best of both worlds. Figure 6 demonstrates that the latent variables z of the encoder have learned meaningfull structure with latent variable interpolations. Samples are obtained by encoding ground truth images, then interpolating the latent variables obtained, decoding them with the decoder of the VAE and adding low level detail with the pixelCNN.

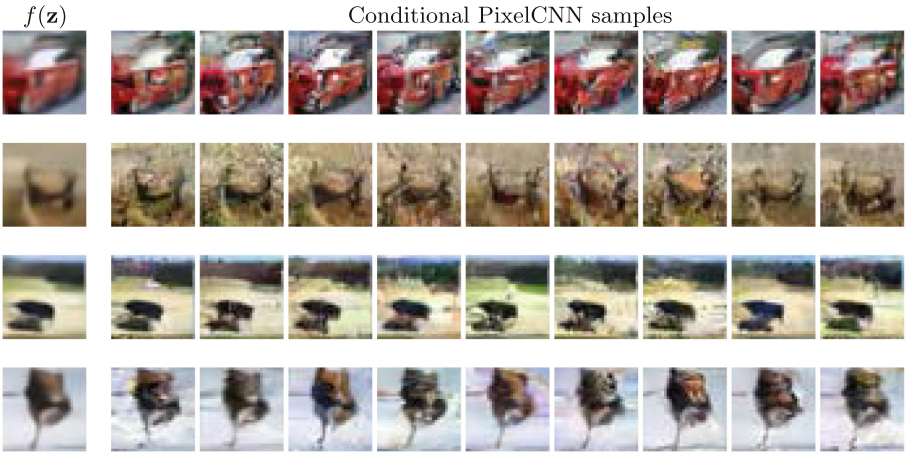


Fig. 5. The column labeled $f(\mathbf{z})$ displays auxiliary representations, with $\mathbf{z}$ sampled from the unit Gaussian prior $p(\mathbf{z})$, accompanied by ten samples of the conditional pixelCNN.

The Auxilliary Loss is Necessary: The fact that the autoregressive decoder ignores the latent variables could be attributed to optimization challenges, as explained in Sect. 3.3. In that case, the auxilliary loss could be used as an initialization scheme only, to guide the model towards a good use of the latent variables. To evaluate this we perform a control experiment where during training we first optimize our objective function in Eq. (7), *i.e.* including the auxiliary reconstruction term, and then switch to optimize the standard objective function of Eq. (6) without the auxiliary term. We proceed by training the full model to convergence then removing the auxiliary loss and fine-tuning from there. Figure 7 displays ground-truth images, with corresponding auxiliary reconstructions and

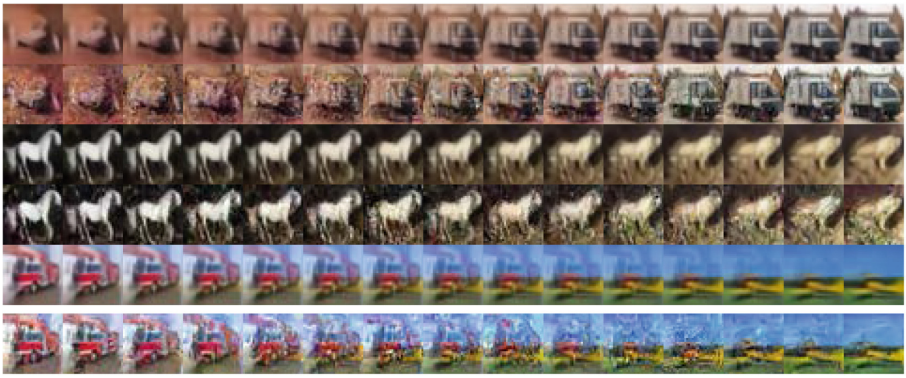


Fig. 6. The first and last columns contain auxilliary reconstructions, images in between are obtained from interpolation of the corresponding latent variables. Odd rows contain auxilliary reconstructions, and even rows contain outputs of the full model.

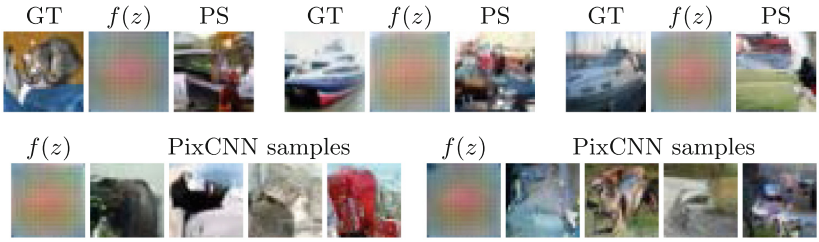


Fig. 7. Auxiliary reconstructions obtained after dropping the auxilliary loss. (GT) denotes ground truth images unseen during training, $f(z)$ is the corresponding intermediate reconstruction, (PS) denotes pixelCNN samples, conditioned on $f(z)$.

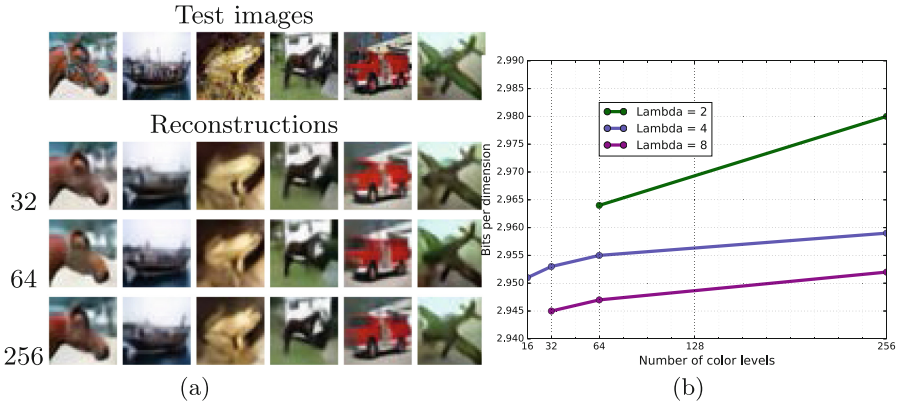


Fig. 8. Impact of the color quantization in the auxiliary image. (a) Reconstructions of the VAE decoder for different quantization levels ($\lambda = 8$). (b) BPD as a function of the quantization level. (Color figure online)

Fig. 9. Samples from models trained with grayscale auxiliary images with 16 color levels (a), 32×32 auxiliary images with 32 color levels (b), and at reduced resolutions of 16×16 (c) and 8×8 pixels (d) with 256 color levels. For each model the auxilliary representation $f(\mathbf{z})$, with $\mathbf{z}$ sampled from the prior, is displayed above the corresponding conditional pixelCNN sample. (Color figure online)

conditional samples, as well as pure samples. The reconstructions have become meaningless and independent from the ground truth images. The samples display the same behavior: for each auxiliary representation four samples from the autoregressive component are displayed and they are independent from one another. Quantitatively, the KL cost immediately drops to zero when removing the auxiliary loss, in approximately two thousand steps of gradient descent. The approximate posterior immediately collapses to the prior and the pixel CNN samples become independent of the latent variables. This is the behavior predicted by the analysis of [5]: the autoregressive decoder is sufficiently expressive that it suffers from using the latent variables.

4.5 Effect of Different Auxiliary Images

We assess the effect of using coarser RGB quantizations, lower spatial resolutions, and grayscale in the auxiliary image. All three make the VAE reconstruction task easier, and transfer the task of modeling color nuances and/or spatial detail to the pixelCNN.

The VAE reconstructions in Fig. 8(a) obtained using coarser color quantization carry less detail than reconstructions based on the original images using 256 color values, as expected. To understand the relatively small impact of the

quantization level on the reconstruction, recall that the VAE decoder outputs the continuous means of the logistic distributions regardless of the quantization level. Only the reconstruction loss is impacted by the quantization level via the computation of the probabilities over the discrete color levels in Eq. (10). In Fig. 8(b) we observe small but consistent gains in the BPD metric as the number of color bins is reduced, showing that it is more effective to model color nuances using the pixelCNN, rather than the latent variables. We trained models with auxiliary images down-sampled to 16×16 and 8×8 pixels, which yield 2.94 and 2.93 BPD, respectively. This is comparable to the 2.92 BPD obtained using our best model at scale 32×32. We also trained models with 4-bit per pixel grayscale auxiliary images, as in [17]. While the grayscale auxilliary images are subjectively the ones that have the best global structure, the results are still qualitatively inferior to those obtained by [17] with a pixelCNN modelling grayscale images. Our model does, however, achieve better quantitative performance at 2.93 BPD. In Fig. 9(a) we show samples obtained using models trained with 4-bit per pixel grayscale auxiliary images, in Fig. 9(b) with 32 color levels in the auxiliary image, and in Fig. 9(c) and (d) with auxiliary images of size 16×16 and 8×8. The samples are qualitatively comparable, showing that in all cases the pixelCNN is able to compensate the less detailed outputs of the VAE decoder and that our framework can be used with a variety of intermediate reconstruction losses.

5 Conclusion

We presented a new approach to training generative image models that combine a latent variable structure with an autoregressive model component. Unlike prior approaches, it does not require careful architecture design to trade-off how much is modeled by latent variables and the autoregressive decoder. Instead, this trade-off can be controlled using a regularization parameter and choice of auxiliary target images. We obtain quantitative performance on par with the state of the art on CIFAR10, and samples from our model exhibit globally coherent structure as well as fine details.

Acknowledgments. This work has been partially supported by the grant ANR-16-CE23-0006 “Deep in France” and LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01).

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: ICML (2017)
2. Bachman, P.: An architecture for deep, hierarchical generative models. In: NIPS (2016)
3. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. PAMI **35**(8), 1798–1828 (2013)
4. Burda, Y., Salakhutdinov, R., Grosse, R.: Importance weighted autoencoders. In: ICLR (2016)

5. Chen, X., et al.: Variational lossy autoencoder. In: ICLR (2017)
6. Deshpande, A., Lu, J., Yeh, M.C., Chong, M., Forsyth, D.: Learning diverse image colorization. In: CVPR (2017)
7. Dinh, L., Krueger, D., Bengio, Y.: NICE: non-linear independent components estimation. In: ICLR (2015)
8. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real NVP. In: ICLR (2017)
9. Germain, M., Gregor, K., Murray, I., Larochelle, H.: MADE: masked autoencoder for distribution estimation. In: ICML (2015)
10. Goodfellow, I., et al.: Generative adversarial nets. In: NIPS (2014)
11. Gregor, K., Besse, F., Rezende, D., Danihelka, I., Wierstra, D.: Towards conceptual compression. In: NIPS (2016)
12. Gulrajani, I., et al.: PixelVAE: a latent variable model for natural images. In: ICLR (2017)
13. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
14. Kingma, D., Rezende, D., Mohamed, S., Welling, M.: Semi-supervised learning with deep generative models. In: NIPS (2014)
15. Kingma, D., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., Welling, M.: Improved variational inference with inverse autoregressive flow. In: NIPS (2016)
16. Kingma, D., Welling, M.: Auto-encoding variational Bayes. In: ICLR (2014)
17. Kolesnikov, A., Lampert, C.: PixelCNN models with auxiliary variables for natural image modeling. In: ICML (2017)
18. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, University of Toronto (2009)
19. Larochelle, H., Murray, I.: The neural autoregressive distribution estimator (2011)
20. van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., Kavukcuoglu, K.: Conditional image generation with PixelCNN decoders. In: NIPS (2016)
21. Oord, A.v.d., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. In: ICML (2016)
22. Rasmus, A., Berglund, M., Honkala, M., Valpola, H., Raiko, T.: Semi-supervised learning with ladder networks. In: NIPS (2015)
23. Reed, S., et al.: Parallel multiscale autoregressive density estimation. In: ICML (2017)
24. Rezende, D., Mohamed, S., Wierstra, D.: Stochastic back propagation and approximate inference in deep generative models. In: ICML (2014)
25. Salimans, T., Karpathy, A., Chen, X., Kingma, D.: Pixelcnn++: improving the pixel CNN with discretized logistic mixture likelihood and other modifications. In: ICLR (2017)
26. Yan, X., Yang, J., Sohn, K., Lee, H.: Attribute2image: conditional image generation from visual attributes. In: ECCV (2016)