Appendix

Quantile regression minimizes the quantile divergence

Proposition 1. *For any distributions P and Q, define the quantile divergence*

$$q(P,Q) := \int_0^1 \left[\int_{F_P^{-1}(\tau)}^{F_Q^{-1}(\tau)} (F_P(x) - \tau)\, dx \right] d\tau.$$

Then the expected quantile loss of a quantile function $\bar{Q}$ implicitly defining the distribution Q satisfies

$$\mathop{\mathbb{E}}_{\tau\sim\mathcal{U}([0,1])} \mathop{\mathbb{E}}_{X\sim P} \left[\rho_\tau(X - \bar{Q}(\tau))\right] = q(P,Q) + h(P),$$

where $h(P)$ does not depend on Q.

Proof. Let P be a distribution with p.d.f. f_P and c.d.f. F_P. Define

$$\rho_\tau(u) = u(\tau - \mathbb{I}\{u \le 0\}),$$
$$g_\tau(q) = \mathop{\mathbb{E}}_{X\sim P}[\rho_\tau(X - q)].$$

We have, for any q and τ,

$$\begin{aligned}
g_\tau(q) &= \int_{-\infty}^{q} (x-q)(\tau-1) f_P(x)\, dx \\
&\quad + \int_q^{\infty} (x-q)\tau f_P(x)\, dx \\
&= \int_{-\infty}^{q} (q-x) f_P(x) dx + \int_{-\infty}^{\infty} (x-q)\tau f_P(x)\, dx \\
&= qF_P(q) + \int_{-\infty}^{q} F_P(x)\, dx - [xF_P(x)]_{-\infty}^{q} \\
&\quad + \tau\left(\mathop{\mathbb{E}}_{X\sim P}[X] - q\right) \\
&= \int_{-\infty}^{q} F_P(x)\, dx + \tau\left(\mathop{\mathbb{E}}_{X\sim P}[X] - q\right),
\end{aligned}$$

where the third equality follows from an integration by parts of $\int_{-\infty}^{q} x f_P(x)\, dx$. Thus the function $q \mapsto g_\tau(q)$ is minimized for $q = F_P^{-1}(\tau)$ and its minimum is

$$g_\tau(F_P^{-1}(\tau)) = \int_{-\infty}^{F_P^{-1}(\tau)} F_P(x)\, dx + \tau\left(\mathop{\mathbb{E}}_{X\sim P}[X] - F_P^{-1}(\tau)\right)$$

We deduce that

$$\begin{aligned}
&g_\tau(q) - g_\tau(F_P^{-1}(\tau)) \\
&= \int_{F_P^{-1}(\tau)}^{q} F_P(x)\, dx + \tau(F_P^{-1}(\tau) - q) \\
&= \int_{F_P^{-1}(\tau)}^{q} (F_P(x) - \tau)\, dx.
\end{aligned}$$

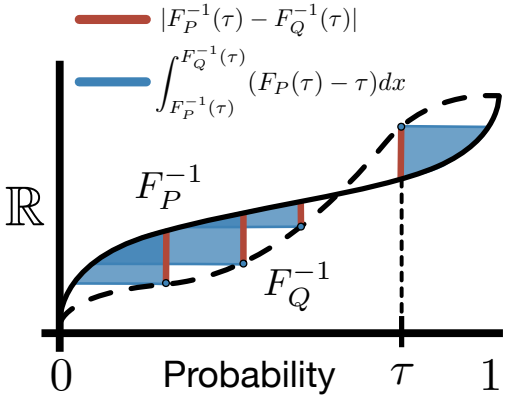

Figure 8. Illustration of the relation between the 1-Wasserstein metric (red) and the quantile divergence (blue).

Thus for a quantile function $\bar{Q}$, we have the expected quantile loss:

$$\mathop{\mathbb{E}}_{\tau\sim\mathcal{U}([0,1])} \left[g_\tau(Q(\tau))\right] = q(P,Q) + \underbrace{\mathop{\mathbb{E}}_{\tau\sim\mathcal{U}([0,1])} \left[g_\tau(F_P^{-1}(\tau))\right]}_{\text{does not depend on } Q}.$$

This finishes the proof of the proposition. □

We observe that quantile regression is nothing else than a projection under the quantile divergence. Thus for a parametrized quantile function $\bar{Q}_\theta$ with corresponding distribution Q_θ, the sample-based quantile regression gradient $\nabla_\theta \rho_\tau(X - \bar{Q}_\theta(\tau))$ for a sample $\tau \sim \mathcal{U}([0,1])$ and $X \sim P$ is an unbiased estimate of $\nabla_\theta q(P, Q_\theta)$:

$$\begin{aligned}
\mathbb{E}\left[\nabla_\theta \rho_\tau(X - \bar{Q}_\theta(\tau))\right] &= \nabla_\theta \mathop{\mathbb{E}}_{\tau\sim\mathcal{U}([0,1])} \left[g_\tau(\bar{Q}_\theta(\tau))\right], \\
&= \nabla_\theta q(P, Q_\theta).
\end{aligned}$$

We illustrate the relation between the 1-Wasserstein metric and the quantile divergence in Figure 8. Notice that, for each $\tau \in [0,1]$, while the Wasserstein measures the error between the two quantile functions, the quantile divergence measures a subset of the area enclosed between their graphs.

Network and Training Details

All PixelCNN and PixelIQN models in Section 4 are directly based on the small and large conditional Gated PixelCNN models developed in (van den Oord et al., 2016b). For CIFAR-10 (Section 4.1), we are using the smaller variant with 15 layer blocks, convolutional filters of size 5, 128 feature planes in each layer block, and 1024 features planes for the residual connections feeding into the output layer of the network. For small ImageNet (Section 4.2), we use both this model, and a larger 20 layer version with 256 feature planes in each layer block.

For PixelIQN, we rescale the $\tau \in [0, 1]^{3n^2}$ linearly to lie in $[-1, 1]^{3n^2}$, and input it to the network in exactly the same way as the location-dependent conditioning in (van den Oord et al., 2016b), that is, by applying a 1×1 convolution producing the same number of feature planes as in the respective layer block, and adding it to the output of this block prior to the gating activation.

All models on CIFAR-10 were trained for a total of 300K steps, those on ImageNet for 400K steps. We trained the small models with a mini-batch size of 32, running approximately 200K updates per day on a single NVIDIA Tesla P100 GPU, while the larger models were trained with a mini-batch size of 128 with synchronous updates from 16 P100 GPUs, achieving approximately half of this step rate.

Hyperparameter Tuning and Evaluation

All quantitative evaluations of our PixelCNN and PixelIQN models are based on the Fréchet Inception Distance (FID) (Heusel et al., 2017),

$$d(x_1, x_2) = \|\mu_1 - \mu_2\|^2 + Tr(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{1/2}),$$

where (μ_1, Σ_1) are the mean and covariance of $10,000$ samples from the model (PixelCNN or PixelIQN), and (μ_2, Σ_2) are the mean and covariance matrix computed over a set of $10,000$ training data points. We slightly deviate from the usual practice of using the entire training set for FID computation, as this would require an equal number ($50,000$ in the case of CIFAR-10) of samples to be drawn from the model, which is computationally very expensive for autoregressive models like PixelCNN or PixelIQN.

We use Polyak averaging (Polyak & Juditsky, 1992), keeping an exponentially weighted average over past parameters with a weight of 0.9999. This average is being loaded instead of the model parameters before samples are generated, but never used for training.

To tune our small PixelCNN and PixelIQN models, we performed a hyperparameter search over 500 hyperparameter configurations for each model, each configuration evaluated after 100K training steps on CIFAR-10, based on its FID score computed on a small set of 2500 generated samples.

For PixelCNN, the parameter search involved choosing from RMSProp, Adam, and SGD as the optimizer, and tuning the learning rate, involving both constant and decaying learning rate schedules. As a result we settled on the RMSProp optimizer and a set of three possible learning rate regimes, namely a constant learning rate of 10^{-4} or $3 \cdot 10^{-5}$, and a decaying learning rate regime: 10^{-4} in the first 120K, $3 \cdot 10^{-5}$ for the next 60K, and 10^{-5} for the remaining training steps. We found the first of these to work best on ImageNet, and the decaying schedule to work best on CIFAR-10, and only report the best model for each dataset.

For PixelIQN, the parameter search included the above (but with constant learning rates only), and additionally a sweep over a range of values for the Huber loss parameter κ (Equation 2). As a result, we used Adam with a constant learning rate of 10^{-4} for all PixelIQN model variants on both datasets, and set $\kappa = 0.002$. We found that the model is not sensitive to this hyperparameter, but performs somewhat worse if the regular quantile regression loss is used instead of the Huber variant.

AIQN-VAE

One potential drawback to PixelIQN presented above, shared by PixelCNN and more generally autoregressive models, is that due to their autoregressive nature sampling can be extremely time-consuming. This is especially true as the resolution of images increases. Although it is possible to partially reduce this overhead with clever engineering, these models are inherently much slower to sample from than models such as GANs and VAEs. In this section, we demonstrate how PixelIQN, due to the continuous nature of the quantile function, can be used to learn distributions over lower-dimensional, latent spaces, such as those produced by an autoencoder, variational or otherwise. Specifically, we use a standard VAE, but simultaneously train a small AIQN to model the training distribution over latent codes. For sampling, we then generate samples of the latent distribution using AIQN instead of the VAE prior.

This approach works well for two reasons. First, even a thoroughly trained VAE does not produce an encoder that fully matches the Gaussian prior. Generaly, the data distribution exists on a non-Gaussian manifold in the latent space, despite the use of variational training. Second, unlike existing methods, AIQN learns to approximate the full continuous-valued distribution without discretizing values or making prior assumptions about the value range or underlying distribution.

We can see similarities between this approach and two other recent publications. First, the α-GAN proposed by Rosca et al. (2017). In both, there is an attempt to sample from the true latent distribution of a VAE-like latent variable model. In the case of α-GAN this sampling distribution is trained using a GAN, while we propose to learn the distribution using quantile regression. The similarity makes sense considering AIQN shares some of the benefits of GANs. Unlike in this related work, we have not replaced the KL penalty on the latent representation. It would be an interesting direction for future research to explore a similar formulation. Generally, the same trade-offs between GANs and AIQN should be expected to come into play here just as they do when learning image distributions. Second, the VQ-VAE model (van den Oord et al., 2017), learns a PixelCNN model of the (discretized) latent space. Here, especially in the la-

Figure 9. CelebA 64x64: Real example images (left), samples generated by VAE (center), and samples generated by AIQN-VAE (right).

tent space, distribution losses respecting distances between individual points is more applicable than likelihood-based losses.

Let $e\colon \mathbb{R}^n \to \mathbb{R}^m$ and $d\colon \mathbb{R}^m \to \mathbb{R}^n$ be the mean of the encoder and decoder respectively of a VAE, although other forms of autoencoder could be substituted. Then, let Q_τ be an AIQN on the space $\mathbb{R}^m$. During training we propose to minimize

$$\mathcal{L}(x) = \mathcal{L}_{VAE}(x) + \mathop{\mathbb{E}}_{\tau\sim\mathcal{U}([0,1]^m)} \rho_\tau^\kappa(e(x) - Q_\tau),$$

where $\mathcal{L}_{VAE}$ is the standard VAE loss function. Then, for generation, we sample $\tau \sim \mathcal{U}([0,1]^m)$, and reparameterize this sample through the AIQN and the decoder to produce $y = d(Q_\tau)$, a sample from the approximated distribution. We call this simple combination the AIQN-VAE.

CelebA

We demonstrate the AIQN-VAE using the CelebA dataset (Liu et al., 2015), at resolution 64×64. We modified an open source VAE implementation[4] to simultaneously train the AIQN on the output of the VAE encoder, with Polyak averaging (Polyak & Juditsky, 1992) of the AIQN weights. We reduce the latent dimension to 32, as our purpose is to investigate the use of VAEs to learn in lower-dimensional latent spaces. The AIQN used three fully connected layers of width 512 with ReLU activations. For the AIQN-VAE, but not the VAE, we lowered latent dimension variance to 0.1 and the KL-term weight to 0.5. It has been observed that in this setting the VAE prior alone will produce poor samples, thus high-quality samples will only be possible by learning the latent distribution. Figure 9 shows samples from both a VAE and AIQN-VAE after $200K$ training iterations. Both models may be expected to improve with further training, however, we can see that the AIQN-VAE samples are frequently clearer and less blurry than those from the VAE.

[4]https://github.com/LynnHo/VAE-Tensorflow

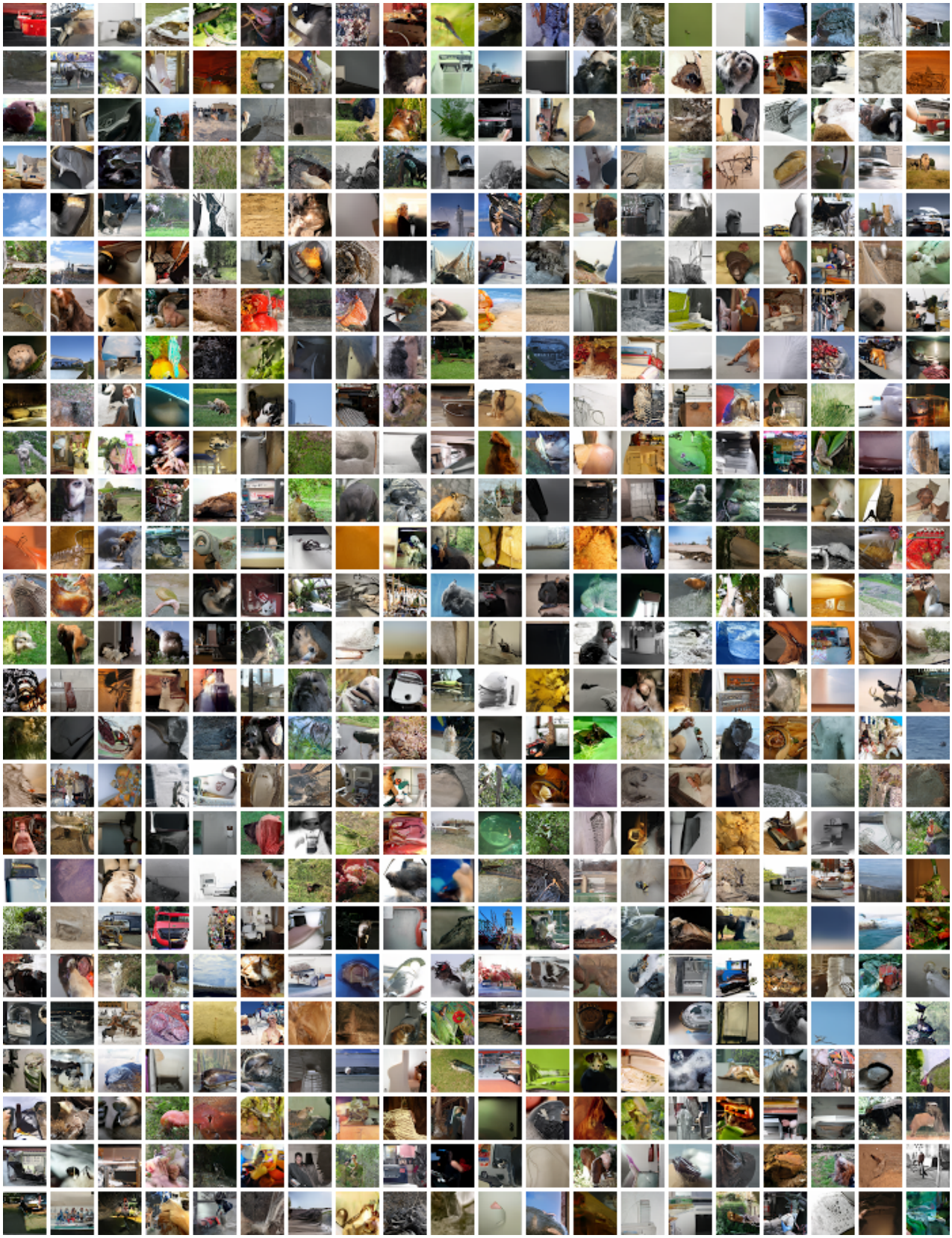

Figure 10. Samples from PixelIQN trained on small ImageNet.

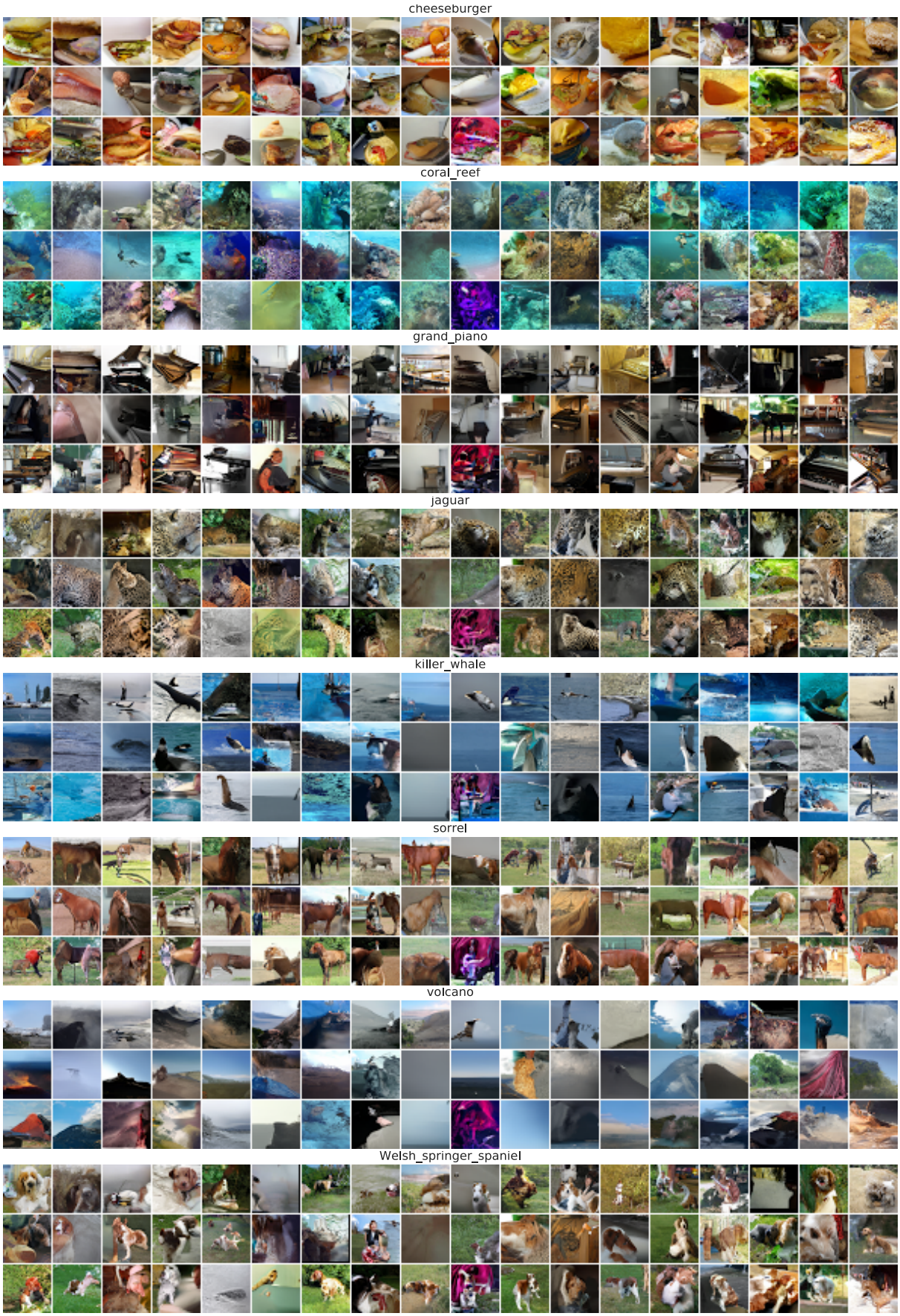

Figure 11. Class-conditional samples from PixelIQN trained on small ImageNet.

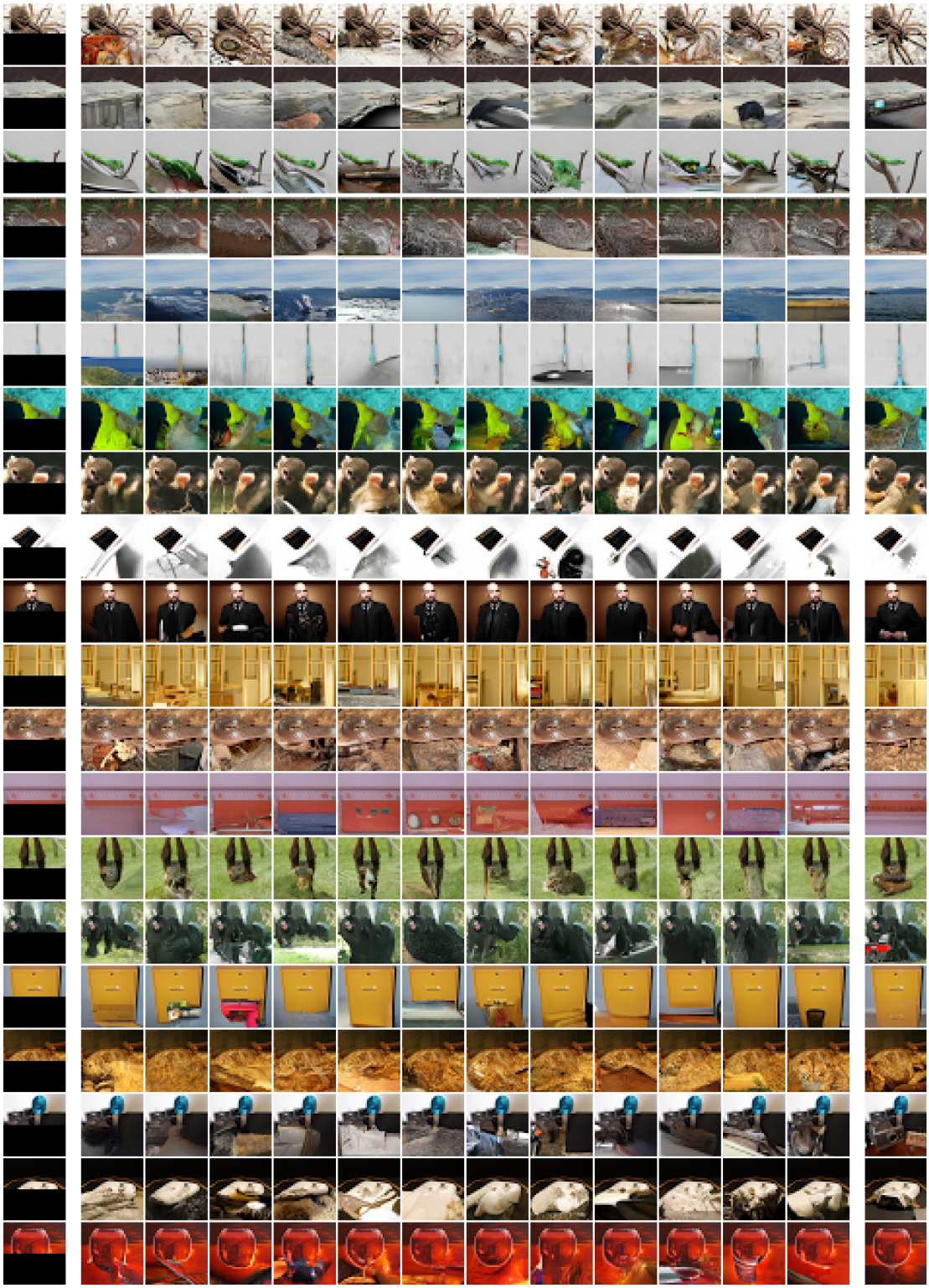

Figure 12. Inpainting. Left column: Masked image given to the network. Middle columns: alternative image completions by the PixelIQN network for different values of τ. Right column: Original image.