SUPPLEMENTARY MATERIAL FOR "AUTOENCODER BASED IMAGE COMPRESSION: CAN THE LEARNING BE QUANTIZATION INDEPENDENT?"

Thierry Dumas, Aline Roumy, Christine Guillemot

INRIA Rennes Bretagne-Atlantique
thierry.dumas@inria.fr, aline.roumy@inria.fr, christine.guillemot@inria.fr

1. INTRODUCTION

In [1], Section 2.2, it is said that, $\forall i \in [|1, m|]$, $\tilde{p}_i$ can be replaced by a function $\tilde{f}_i$, parametrized by $\boldsymbol{\psi}^{(i)}$, and $\boldsymbol{\psi}^{(i)}$ is learned such that $\tilde{f}_i$ fits $\tilde{p}_i$. This supplementary material provides details on $\tilde{f}_i$.

2. DEFINITION OF THE PARAMETRIZED FUNCTIONS

This section explains why $\tilde{f}_i$ is a piecewise linear function and defines $\tilde{f}_i$.

During the autoencoder training, and especially at the beginning of the training, $\tilde{p}_i$ may have unpredictable shapes. $\tilde{f}_i$ can fit $\tilde{p}_i$ at any time during the training if $\tilde{f}_i$ does not require any assumption regarding $\tilde{p}_i$. That is why, as in [2], $\tilde{f}_i$ is a piecewise linear function.

To define $\tilde{f}_i$, we first sample an axis uniformly and symmetrically with respect to 0. The number of sampling points is $2\rho d + 1$, where $d \in \mathbb{N}^*$ is the number of sampling points per unit interval and $\rho \in \mathbb{N}^*$ is the number of unit intervals in the positive half of the axis. Then, we define the parameters $\boldsymbol{\psi}^{(i)} \in \mathbb{R}^{2\rho d+1}$ of $\tilde{f}_i$ as the values of $\tilde{f}_i$ at the sampling points $\mathbf{u} \in \mathbb{R}^{2\rho d+1}$, see Figure 1.

3. USE OF THE PARAMETRIZED FUNCTIONS

Given $\tilde{y} \in [\mathbf{u}_0, \mathbf{u}_{2\rho d}[$, this section explains how $\tilde{f}_i\left(\tilde{y}; \boldsymbol{\psi}^{(i)}\right)$ is computed.

For $k \in [|0, 2\rho d - 1|]$, we call k the index of the linear piece between $\boldsymbol{\psi}_k^{(i)}$ and $\boldsymbol{\psi}_{k+1}^{(i)}$. To compute $\tilde{f}_i\left(\tilde{y}; \boldsymbol{\psi}^{(i)}\right)$, the 1st step is to find the index $k_{\tilde{y}}$ of the linear piece that contains $\tilde{f}_i\left(\tilde{y}; \boldsymbol{\psi}^{(i)}\right)$, see Figure 1. This is done at low computational cost via (1).

$$k_{\tilde{y}} = \lfloor d\tilde{y} \rfloor + \rho d \tag{1}$$

$\lfloor . \rfloor$ denotes the floor function. The function "index_linear_piece" in the file "kodak_tensorflow_0.11.0/tf_utils/tf_utils.py" implements (1). Then, given $k_{\tilde{y}}$, the 2nd step to compute $\tilde{f}_i\left(\tilde{y}; \boldsymbol{\psi}^{(i)}\right)$ is (2).

$$\tilde{f}_i\left(\tilde{y}; \boldsymbol{\psi}^{(i)}\right) = \left(\boldsymbol{\psi}_{k_{\tilde{y}}+1}^{(i)} - \boldsymbol{\psi}_{k_{\tilde{y}}}^{(i)}\right)\left(\tilde{y} - \boldsymbol{u}_{k_{\tilde{y}}}\right) d + \boldsymbol{\psi}_{k_{\tilde{y}}}^{(i)} \tag{2}$$

The function "approximate_probability" in the file "kodak_tensorflow_0.11.0/tf_utils/tf_utils.py" implements (2).

4. FITTING THE PARAMETRIZED FUNCTIONS

This section describes the learning of $\boldsymbol{\psi}^{(i)}$ for fitting $\tilde{f}_i$ to $\tilde{p}_i$.

For $i \in [|1, m|]$, the target is to minimize the integrated squared error I_i between $\tilde{f}_i$ and $\tilde{p}_i$.

$$\begin{aligned} I_i &= \int_{\mathbf{u}_0}^{\mathbf{u}_{2\rho d}} \left(\tilde{f}_i\left(x; \boldsymbol{\psi}^{(i)}\right) - \tilde{p}_i\left(x\right)\right)^2 dx \\ &= \int_{\mathbf{u}_0}^{\mathbf{u}_{2\rho d}} \tilde{f}_i^2\left(x; \boldsymbol{\psi}^{(i)}\right) dx + \int_{\mathbf{u}_0}^{\mathbf{u}_{2\rho d}} \tilde{p}_i^2\left(x\right) dx \\ &\quad - 2\int_{\mathbf{u}_0}^{\mathbf{u}_{2\rho d}} \tilde{p}_i\left(x\right) \tilde{f}_i\left(x; \boldsymbol{\psi}^{(i)}\right) dx \end{aligned} \tag{3}$$

In (3), the 1st integral can be approximated by the left Riemann sum of $\tilde{f}_i^2$ over $[\mathbf{u}_0, \mathbf{u}_{2\rho d}]$ with partition $\{[\mathbf{u}_0, \mathbf{u}_1], [\mathbf{u}_1, \mathbf{u}_2], ..., [\mathbf{u}_{2\rho d-1}, \mathbf{u}_{2\rho d}]\}$. The 2nd integral does not depend on $\boldsymbol{\psi}^{(i)}$. The 3rd integral can be estimated via samples $\{\tilde{y}_{ij}\}_{j=1...n}$ from $\tilde{p}_i$. Using the previous remarks, the minimization is (4).

$$\begin{aligned} &\mathcal{L}^{(i)}\left(\boldsymbol{\psi}^{(i)}\right) = \frac{1}{d}\sum_{l=0}^{2\rho d-1} {\boldsymbol{\psi}_l^{(i)}}^2 - \frac{2}{n}\sum_{j=1}^{n} \tilde{f}_i\left(\tilde{y}_{ij}; \boldsymbol{\psi}^{(i)}\right) \\ &\mathcal{L}\left(\boldsymbol{\psi}^{(1)}, ..., \boldsymbol{\psi}^{(m)}\right) = \sum_{i=1}^{m} \mathcal{L}^{(i)}\left(\boldsymbol{\psi}^{(i)}\right) \\ &\min_{\boldsymbol{\psi}^{(1)}, ..., \boldsymbol{\psi}^{(m)}} \mathcal{L}\left(\boldsymbol{\psi}^{(1)}, ..., \boldsymbol{\psi}^{(m)}\right) \end{aligned} \tag{4}$$

The use of the left Riemann sum induces a constraint on $\{\boldsymbol{\psi}_l^{(i)}\}_{l=0...2\rho d-1}$, but no constraint on $\boldsymbol{\psi}_{2\rho d}^{(i)}$. This generates fitting errors at the right edge of $\tilde{f}_i$ when running minimiza-

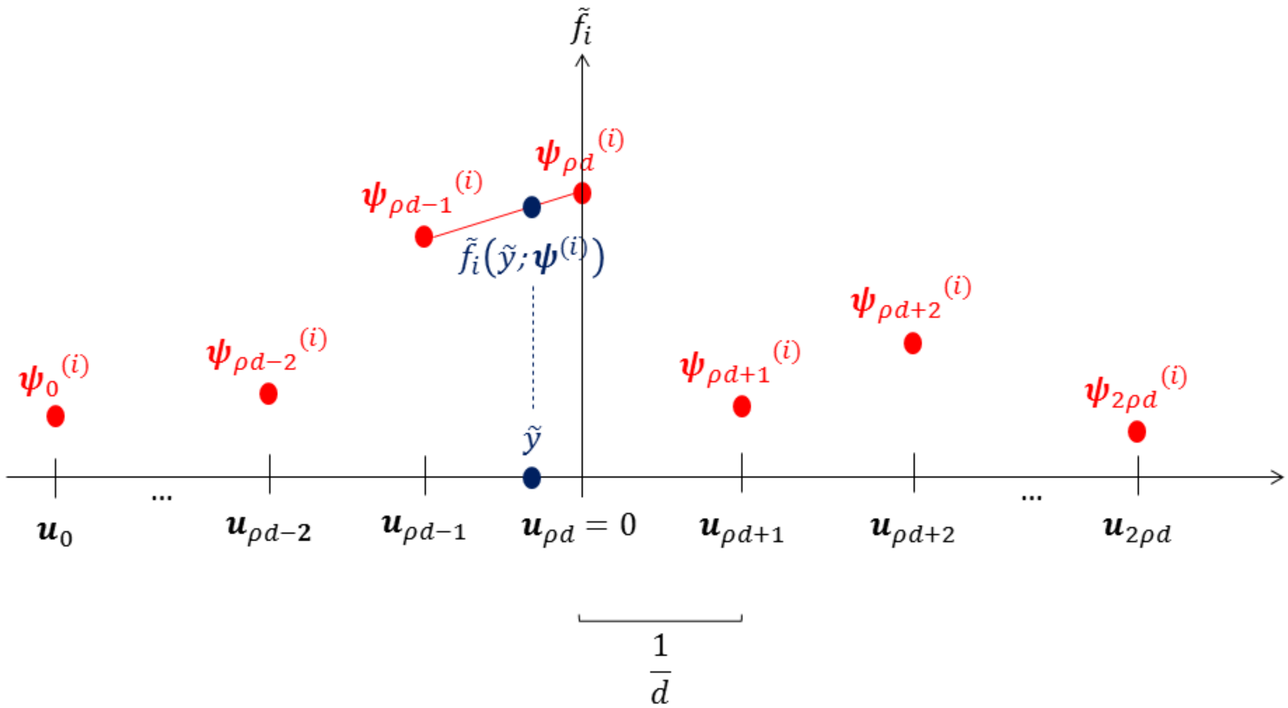

Fig. 1. Illustration of $\tilde{f}_i$. Here, $k_{\tilde{y}} = \rho d - 1$.

tion (4) [1]. To correct this, ${\psi^{(i)}_{2\rho d}}^2/d$ is added to $\mathcal{L}^{(i)}\left(\boldsymbol{\psi}^{(i)}\right)$, slightly overestimating the integral of $\tilde{f}_i^2$. Note that ${\psi^{(i)}_{2\rho d}}^2/d$ is very small as the two tails of probability density functions are near 0. Including the correction, minimization (4) becomes (5).

$$\begin{aligned}
&\overline{\mathcal{L}}^{(i)}\left(\boldsymbol{\psi}^{(i)}\right) = \frac{1}{d}\sum_{l=0}^{2\rho d} {\psi_l^{(i)}}^2 - \frac{2}{n}\sum_{j=1}^{n} \tilde{f}_i\left(\tilde{y}_{ij};\boldsymbol{\psi}^{(i)}\right) \\
&\overline{\mathcal{L}}\left(\boldsymbol{\psi}^{(1)}, ..., \boldsymbol{\psi}^{(m)}\right) = \sum_{i=1}^{m} \overline{\mathcal{L}}^{(i)}\left(\boldsymbol{\psi}^{(i)}\right) \\
&\min_{\boldsymbol{\psi}^{(1)}, ..., \boldsymbol{\psi}^{(m)}} \overline{\mathcal{L}}\left(\boldsymbol{\psi}^{(1)}, ..., \boldsymbol{\psi}^{(m)}\right) \qquad (5)
\end{aligned}$$

The function "loss_density_approximation" in the file "kodak_tensorflow_0.11.0/tf_utils.py" computes $\overline{\mathcal{L}}\left(\boldsymbol{\psi}^{(1)}, ..., \boldsymbol{\psi}^{(m)}\right)$.

Stochastic gradient descent solves minimization (5) and each parameter in the set $\{\psi_l^{(i)}\}_{l=0...2\rho d}^{i=1...m}$ is projected onto $[10^{-6}, +\infty[$ after each gradient step.

[1] Changing the Riemann sum does not help solve this kind of issue. For instance, if the middle Riemann sum is used, $\psi_0^{(i)}$ and $\psi_{2\rho d}^{(i)}$ are less constrained than $\{\psi_l^{(i)}\}_{l=1...2\rho d-1}$. This causes fitting errors at the two edges of $\tilde{f}_i$ when running minimization (4).

5. REFERENCES

[1] Thierry Dumas, Aline Roumy, and Christine Guillemot, "Autoencoder based image compression: can the learning be quantization independent?," in *ICASSP*, 2018.

[2] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli, "End-to-end optimized image compression," in *ICLR*, 2017.