HW2 – Solutions

Thomas Marchioro

April 2023

Exercise 1

(a) The inverse transformation that maps x into z for one coupling layer is

$$\begin{bmatrix} z_A \\ z_B \end{bmatrix} = f_\theta^{-1}(x) = \begin{bmatrix} x_A \\ e^{-\alpha_\theta(x_A)} \odot (x_B - m_\theta(x_A)) \end{bmatrix} \tag{1}$$

and the log-likelihood is obtained using the change of variable theorem

$$\log p_X(x) = \log p_Z(f_\theta^{-1}(x)) + \log\det J_x(f_\theta^{-1}(x)) \tag{2}$$

$$= \underbrace{\sum_{j=1}^{d} \log p_{Z_j}(f_{\theta,j}^{-1}(x)) - \alpha_{\theta,j}(x_A)}_{\text{independent components}}. \tag{3}$$

For K coupling layers (i.e., $f_\theta = f_{\theta_1} \circ \cdots \circ f_{\theta_K}$), we can just iteratively apply the change of variable theorem, obtaining an additional log-determinant component for each layer:

$$\log p_X(x) = \sum_{j=1}^{d} \left[\log p_{Z_j}(f_{\theta_K,j}^{-1} \circ \cdots \circ f_{\theta_1,j}^{-1}(x)) - \sum_{k=1}^{K} \alpha_{\theta_k,j}(x_A) \right]. \tag{4}$$

The loss to be minimized is the negative log-likelihood over the dataset for a specific choice of p_{Z_j}. In this specific implementation, we can use a standard normal distribution, i.e. $Z_j \sim \mathcal{N}(0,1)$.

The architecture of each coupling layer is a simple multi-layer perceptron, i.e., a sequential architecture composed of multiple fully-connected layers. The dimensionality of the input is d and depends on the dataset. The output of the coupling layer is of size $2d$, with half of the output being used for the additive term m, and the other half used for the exponent α of the scaling term. The number of intermediate (hidden) layers is a hyperparameter. Additionally, we apply the tanh activation function to the scaling exponent α so that the scaling factor is bounded in the range $[1/e, e]$.

(b) The specific hyperparameters used in the implementation of RealNVP while training it on the MNIST dataset is reported in table 1. Repeating the training process for 5 and 10 coupling layers, we find that using 10 coupling layers yields better results, as shown in figure 1. In particular, when using 5 coupling layers the output still contains traces of noise in the background, which instead completely disappear using 10 layers. However, a higher number of coupling layers makes the model more difficult to train. This problem is specifically due to the introduction of the scaling layer, which may cause the output to either explode or vanish when the number of coupling layer increases.

Hyperparameter	Value
Number of epochs	40
Batch size B	128
Learning rate η	10^{-3}
Input dimension d	784
Hidden dimension	256
Number of fully-connected layers per coupling layer	3
Gradient norm clipping	1

Table 1: Hyperparameters used in the implementation of RealNVP

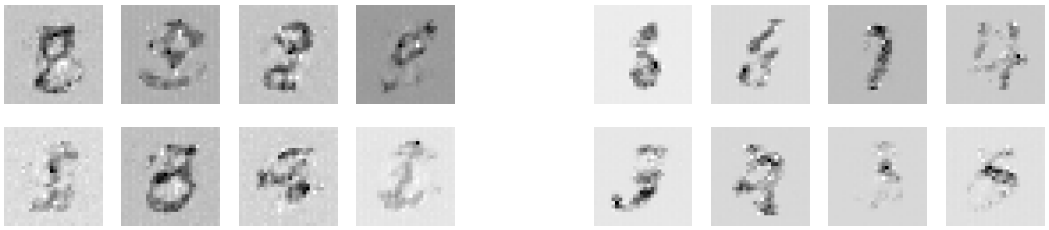

Figure 1: Digits generated by RealNVP with 5 (left) and 10 (right) coupling layers.

Figure 2: Digits interpolated in the latent space with linear (above) and sinusoidal (below) interpolation.

(c) Figure 2 shows some examples of digits interpolated in the latent space using linear and sinusoidal interpolation. The two techniques seem to yield similar results when applied on MNIST digits on this specific trained model.

Exercise 2

(a) The conditional version of RealNVP can be realized by simply concatenating the one-hot encoding of the labels to the input of m_θ and α_θ. Additionally, the sampling method should also take as input a list of labels representing the digits to be generated.

(b) Similarly to the non-conditional RealNVP, the conditional version produces samples that actually resemble digits when using 10 coupling layers, but not with 5. However, the quality of the generated samples also depends on other hyperparameters, such as the size of the hidden layers and batch size. In this case, the model was trained with a relatively small hidden layer size (256) and a batch size of 128. Using a smaller batch size and larger hidden layer size may improve the quality of the generated samples, even with fewer coupling layers. It is worth noting that the digit-like samples

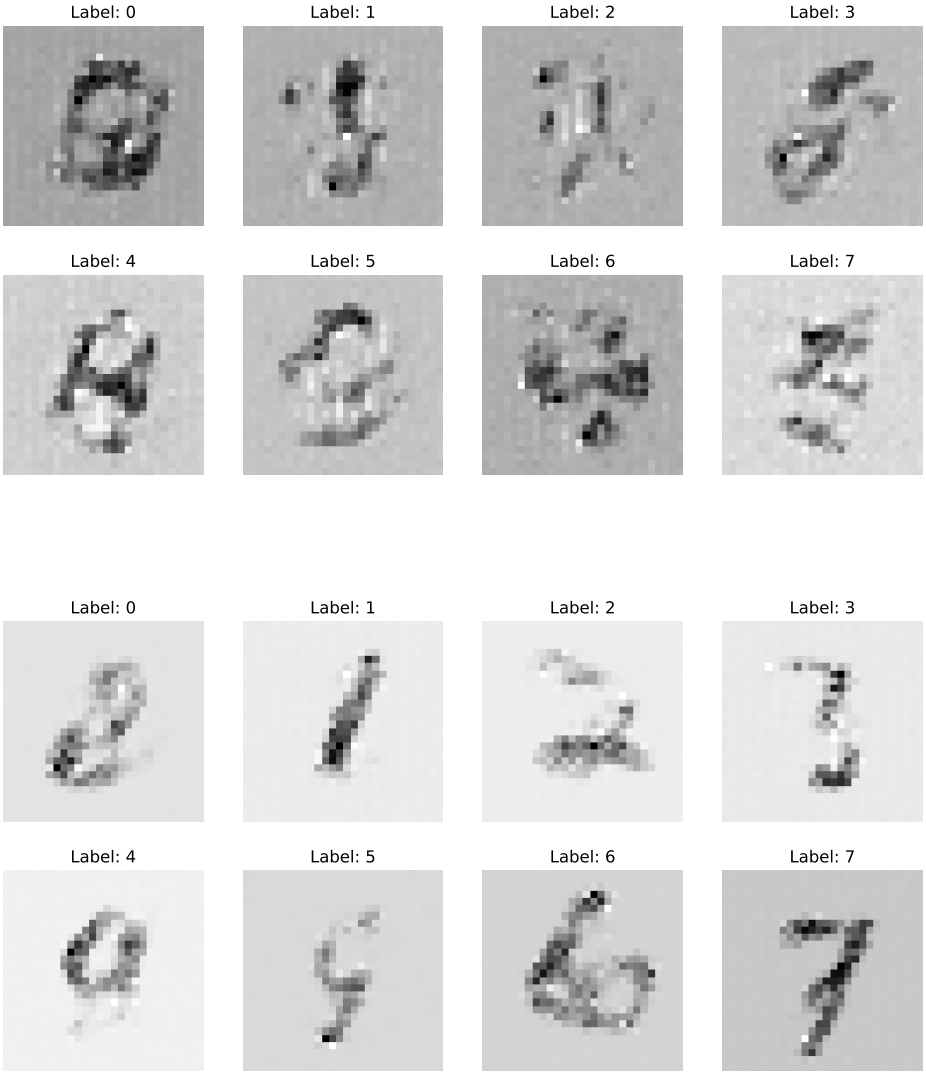

Figure 3: Digits generated by conditional RealNVP with 5 (above) and 10 (below) coupling layers.

generated with 10 coupling layers often match the intended digit label, as shown in figure 3.

Exercise 3

(a) The inverse transformation for each coupling layer is

$$z = f^{-1}(x) = \begin{bmatrix} x_A \\ x_B - m(x_A) \\ x_C - l(x_A, x_B - m(x_A)) \end{bmatrix} \tag{5}$$

It is worth noticing that m receives as input $x_A = z_A$, and l receives as input $x_A = z_A$ and $x_B - m(x_A) = x_B - m(z_A) = z_B$, which implies that the m and l terms are indeed the same in the direct and inverse transformation.

(b) The Jacobian is computed as follows

$$J_z f(z) = \begin{bmatrix} I_{n_A} & 0 & 0 \\ \frac{\partial m}{\partial z} & I_{n_B} & 0 \\ \frac{\partial l}{\partial z} & \frac{\partial l}{\partial z_B} & I_{n_C} \end{bmatrix}, \tag{6}$$

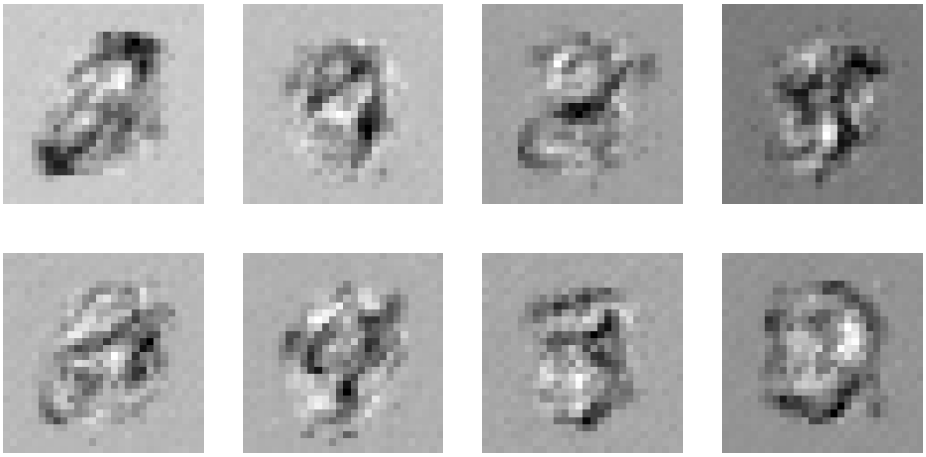

Figure 4: Digits generated by NICE with 10 3-split coupling layers.

with n_A, n_B, n_C are the number of elements of x_A, x_B, and x_C, respectively. Since the matrix is lower triangular, the determinant can be computed by multiplying the elements on the diagonal. Therefore we obtain $\det(J_x f(x)) = 1$ as in the original 2-split version, meaning that the transformation is volume-preserving. Additionally, we also have $\det J_x f^{-1}(x) = 1$. The log-likelihood of x can thus be computed using the change of variable theorem as follows:

$$\log p_x(x) = \log p_z(f^{-1}(x)) + \log \det J_x f^{-1}(x) = \log p_z(f^{-1}(x)) = \log p_z(f^{-1}(x)). \tag{7}$$

(c) Figure 4 shows examples of digits generated by NICE with 10 3-split coupling layers. The results are not much different from those obtained in the tutorial. In the implementation of the model, we made use of three masks:

$$\text{mask}_1 = [1, 0, 0, 1, 0, 0, \dots], \tag{8}$$
$$\text{mask}_2 = [0, 1, 0, 0, 1, 0, \dots], \tag{9}$$
$$\text{mask}_3 = [0, 0, 1, 0, 0, 1, \dots] \tag{10}$$

The masks are used to split x into x_A, x_B, x_C and z into z_A, z_B, z_C, rotating them at each coupling layer. This means that the first coupling layer uses masks 1,2,3 for z_A, z_B, z_C respectively. The second coupling layer uses masks 2,3,1, and the third uses 3,1,2. After the fourth coupling layer, the masks repeat. Both x and z are 1D vectors with 784 elements, and the masks are applied through element-wise multiplication.

Exercise 4

(a) The Markov chain in figure 5 is described by the following transition probability matrix

$$P = \begin{bmatrix} 1-p_{12} & p_{12} & 0 & 0 & 0 & 0 \\ 1-p_{23} & 0 & p_{23} & 0 & 0 & 0 \\ 0 & 1-p_{34} & 0 & p_{34} & 0 & 0 \\ 0 & 0 & 1-p_{45} & 0 & p_{45} & 0 \\ 0 & 0 & 0 & 1-p_{56} & 0 & p_{56} \\ 0 & 0 & 0 & 0 & 1-p_{66} & p_{66} \end{bmatrix} \tag{11}$$

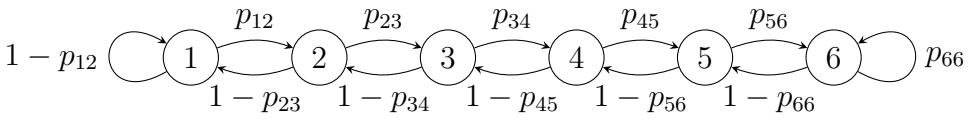

Figure 5: Markov chain studied in exercise 4.

where the element P_{ij} is the probability of going from state i to state j. Additionally, the structure of P can be further simplified based on the values assigned in the next questions, i.e.

$$P = \begin{bmatrix} 1-p & p & 0 & 0 & 0 & 0 \\ 1-q & 0 & q & 0 & 0 & 0 \\ 0 & 1-p & 0 & p & 0 & 0 \\ 0 & 0 & 1-p & 0 & p & 0 \\ 0 & 0 & 0 & 1-p & 0 & p \\ 0 & 0 & 0 & 0 & 1-p & p \end{bmatrix} \tag{12}$$

(b) In the case $p_{12} = \cdots = p_{66} = \frac{1}{2}$ (i.e., $p = q = \frac{1}{2}$), we need to simulate two random walks of $T = 10^3$ and $T = 10^5$ steps, respectively. The procedure to simulate a random walk is described in algorithm 1. We start from the initial state $x_0 = 1$ and then at each iteration we jump from the previous state x_{t-1} to the next one x_t according to the corresponding row $P_{x_{t-1}}$ of the transition probability matrix. At each step, we update the counter n_{x_t} of the state that we reached. At the end of the random walk, we estimate the stationary probabilities as $\hat{\pi}_i = n_i/T$ for $i = 1, \ldots, 6$.

Algorithm 1 Random walk

Require: Transition matrix P, number of steps T
  Set $x_0 = 1$
  Initialize counters $n_1 = \cdots = n_6 = 0$
  for $t = 1, \ldots, T$ **do**
    Choose $x_t \sim \text{Categorical}(P_{x_{t-1}})$ ▷ $P_{x_{t-1}}$ is the x_{t-1}th row of P
    Update counter $n_{x_t} \leftarrow n_{x_t} + 1$
  end for
  Estimate stationary probabilities $\hat{\pi} = \frac{n}{T}$ ▷ Normalize counters w.r.t. the number of steps
  return $\hat{\pi}$

The estimated stationary probabilities are plotted in figure 6 (blue bars) and compared with the true values of π obtained via eigenvalue decomposition (pink bars). Unsurprisingly, due to the symmetries in the Markov chain, for a large number of steps (i.e., $T = 10^5$) all states are reached with equal probability. For $T = 10^3$, instead, the Markov chain has not reached the steady state regime, and the frequency of the states are still influenced by the initial state $x_0 = 1$. Specifically, states that are "closer" (i.e., reached in fewer steps) to $x_0 = 1$ are more frequent for a short random walk.

(c) After estimating the stationary distribution with a random walk, we are required to compute its true value using eigenvalue decomposition. We know that the stationary probabilities π of a Markov chain P should satisfy the following equation:

$$\pi = P^\top \pi. \tag{13}$$

This should remind us of the definition of eigenvector. We recall that x is an eigenvector for a square matrix $A \in \mathbb{R}^{d \times d}$, if for some value $\lambda \in \mathbb{R}$ (called eigenvalue) we have

$$\lambda x = Ax. \tag{14}$$

Comparing equations 13 and 14, we can conclude that π must be the eigenvector of $P^\top$ relative to the eigenvalue $\lambda = 1$. To be more precise, π must be a multiple of such eigenvector, since all multiples $ax, a \in \mathbb{R}$ of an eigenvector x satisfy the same property. The additional requirement is that π, being a probability distribution, should satisfy the normalization condition $\sum_{i=1}^{6} \pi_i = 1$. Calculating the eigenvector of $P^\top$ for $\lambda = 1$ using NumPy's eigenvalue decomposition function and

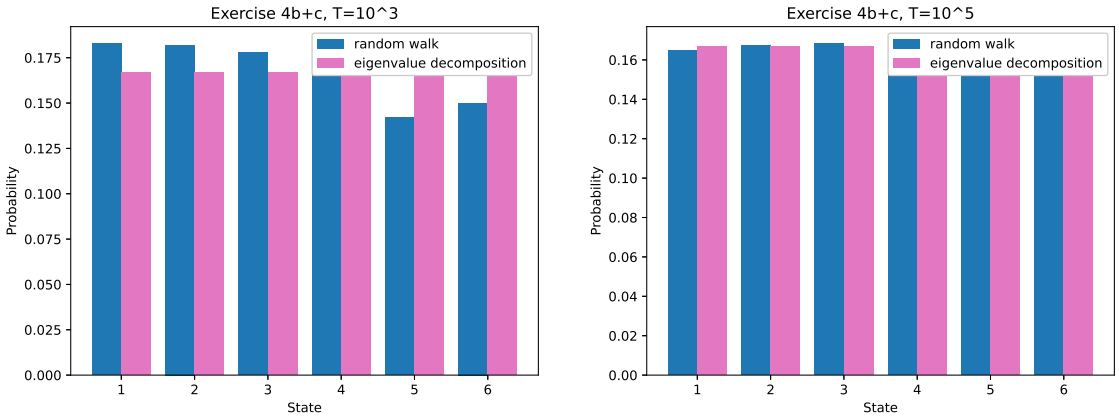

Figure 6: Stationary distribution of the states, estimated with random walk and calculated according to eigenvalue decomposition.

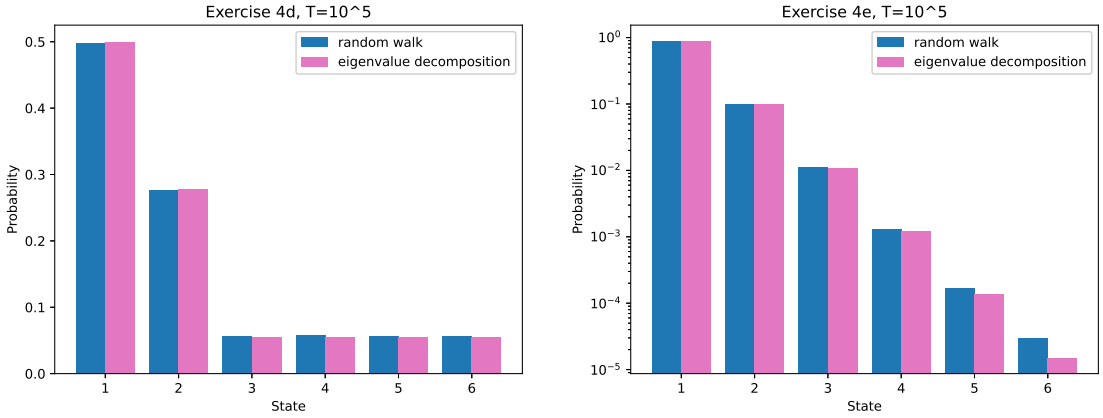

Figure 7: Stationary distribution of the states for Markov chains of points (4d) and (4e), estimated with random walk and calculated according to eigenvalue decomposition. The y-axis of the right figure is in log-scale for better visualization.

normalizing it w.r.t. its sum, we find that $\pi = \left[\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right]^\top$, confirming the results observed from the random walk.

(d) Re-estimating the stationary probabilities for $p_{23} = 0.1$ (i.e., $p = \frac{1}{2}$ and $q = 0.1$), we obtain that state 2 divides the Markov chain in two components. The first component (consisting of state 1 alone) is reached with higher probability, while the other (consisting of states 3, 4, 5, and 6) is reached with lower probability. All the states in the second component have equal stationary probability. This is expected, since 2 is the only state in which "breaks" the symmetry in the Markov chain.

(e) Re-estimating the stationary probabilities for $p_{12} = \cdots = p_{66} = 0.1$ (i.e., $p = 0.1$ and $q = 0.1$), we obtain that the probability of reaching each state is exponentially decreasing starting from state 1. This is expected, since, from each state i, a random walk moves the previous one with 90% probability and to the next one with only 10%. Thus, it becomes more and more difficult to reach farther states in the chain. Histograms are reported in fig. 7.

Exercise 5

Intuition:

$$D_{\mathrm{KL}}(p(X_0,\dots,X_n)\|q(X_0,\dots,X_n)) = \mathbb{E}\left[\log\left(\frac{p(X_0,\dots,X_n)}{q(X_0,\dots,X_n)}\right)\right] \tag{15}$$

$$= \mathbb{E}\left[\log\left(\frac{\prod_{i=1}^n p(X_i|X_{i-1})}{\prod_{i=1}^n q(X_i|X_{i-1})}\right)\right] \tag{16}$$

$$= \mathbb{E}\left[\sum_{i=1}^n \left(\log p(X_i|X_{i-1}) - \log q(X_i|X_{i-1})\right)\right] \tag{17}$$

$$= \sum_{i=1}^n \mathbb{E}\left[\log\left(\frac{p(X_i|X_{i-1})}{q(X_i|X_{i-1})}\right)\right] \tag{18}$$

$$= nD_{\mathrm{KL}}(p(X_i|X_{i-1})\|q(X_i|X_{i-1})) \tag{19}$$

We can start by noticing that, once the prior $\pi(x_i)$ is given, we have the following result for the KL divergence between $p(X_i|X_{i-1})$ and $q(X_i|X_{i-1})$:

$$D_{\mathrm{KL}}(p(X_i|X_{i-1})\|q(X_i|X_{i-1})) = \tag{20}$$

$$= \int_{\mathcal{X}^{n+1}} p(x_0,\dots,x_n)\log\left(\frac{p(x_i|x_{i-1})}{q(x_i|x_{i-1})}\right)dx_0\cdots dx_n \tag{21}$$

$$= \int_{\mathcal{X}^{i+1}} p(x_0,\dots,x_i)\log\left(\frac{p(x_i|x_{i-1})}{q(x_i|x_{i-1})}\right)dx_0\cdots dx_i \tag{22}$$

$$\overset{\text{Markov}}{=} \int_{\mathcal{X}^{i+1}} \pi(x_0)\prod_{j=1}^i p(x_j|x_{j-1})\log\left(\frac{p(x_i|x_{i-1})}{q(x_i|x_{i-1})}\right)dx_0\cdots dx_i \tag{23}$$

$$= \int_{\mathcal{X}} p(x_i|x_{i-1})\log\left(\frac{p(x_i|x_{i-1})}{q(x_i|x_{i-1})}\right)\int_{\mathcal{X}^i}\pi(x_0)\prod_{j=1}^i p(x_j|x_{j-1})dx_0\dots dx_{i-1} \tag{24}$$

$$= \int_{\mathcal{X}}\int_{\mathcal{X}} \pi(x_{i-1})p(x_i|x_{i-1})\log\left(\frac{p(x_i|x_{i-1})}{q(x_i|x_{i-1})}\right)dx_{i-1}dx_i \tag{25}$$

$$= \delta > 0 \tag{26}$$

where at step 25 we use iteratively the fact that $\int_{\mathcal{X}}\pi(x)p(y|x)dx = \pi(y)$. With this step we have essentially proven that $D_{\mathrm{KL}}(p(X_i|X_{i-1})\|q(X_i|X_{i-1}))$ is some constant quantity $\delta > 0$, which does not depend on the state index. This is due to the joint distribution of X_{i-1}, X_i, X_{i+1} being stationary w.r.t. i.

Our goal is to find an expression for

$$D_{\mathrm{KL}}(p(X_0,\dots,X_n)\|q(X_0,\dots,X_n)) = \int_{\mathcal{X}^n} p(x_0,\dots,x_n)\log\left(\frac{p(x_0,\dots,x_n)}{q(x_0,\dots,x_n)}\right)dx_0\cdots dx_n. \tag{27}$$

that depends on n and on δ.

$$D_{\mathrm{KL}}(p(X_0,\dots,X_n)\|q(X_0,\dots,X_n)) = \tag{28}$$

$$= \int_{\mathcal{X}^{n+1}} p(x_0,\dots,x_n)\log\left(\frac{p(x_0,\dots,x_n)}{q(x_0,\dots,x_n)}\right)dx_0\cdots dx_n \tag{29}$$

$$= \int_{\mathcal{X}^{n+1}} \pi(x_0)\prod_{i=1}^n\left[p(x_i|x_{i-1})\log\left(\frac{p(x_i|x_{i-1})}{q(x_i|x_{i-1})}\right)\right]dx_0\cdots dx_n \tag{30}$$

$$= \int_{\mathcal{X}^{n+1}} \pi(x_0)\prod_{j=1}^n p(x_j|x_{j-1})\sum_{i=1}^n\log\left(\frac{p(x_i|x_{i-1})}{q(x_i|x_{i-1})}\right)dx_0\cdots dx_n \tag{31}$$

$$= \sum_{i=1}^n\int_{\mathcal{X}^{n+1}} \pi(x_0)\prod_{j=1}^n p(x_j|x_{j-1})\log\left(\frac{p(x_i|x_{i-1})}{q(x_i|x_{i-1})}\right)dx_0\cdots dx_n \tag{32}$$

$$\overset{\text{eq. 23–25}}{=} \sum_{i=1}^n \delta = n\delta. \tag{33}$$

At step 32 we have a finite sum of non-negative elements, so we can always swap the sum with the integral without verifying the conditions for Fubini-Tonelli's theorem.

It follows that

$$\lim_{n\to\infty} \frac{1}{n} D_{\mathrm{KL}}(p(X_0,\ldots,X_n)\|q(X_0,\ldots,X_n)) = \lim_{n\to\infty} \frac{1}{n} n\delta = \delta \tag{34}$$

where $\delta = D_{\mathrm{KL}}(p(X_i|X_{i-1})\|q(X_i|X_{i-1}))$.