Self-Supervised Training of Hierarchical Vector Quantisers

S P Luttrell

DRA, Electronics Division, Malvern, WR14 3PS, UK.

I. INTRODUCTION

In Luttrell [14] we developed a hierarchical vector quantisation (VQ) model, and in Luttrell [5, 6] we successfully applied it to time series and image compression respectively. The goal of this paper is to derive an extension to this model, in which we backpropagate signals from higher to lower layers of the hierarchy to self-supervise the training of the VQ.

We review the basic properties of our VQ model and its relationship to neural network methods. We extend the model to an ensemble of VQs, and we derive its properties in the limit of a large codebook size (i.e. the continuum limit). Finally, we demonstrate how self-supervision emerges naturally in this type of model.

II. HIERARCHICAL VECTOR QUANTISATION MODEL

In this section we present a resumé of the VQ theory of Linde et al. [7] (the LBG algorithm), and its extension to hierarchical VQs [14]. These extensions are related to the VQ theory of Kumazawa et al. [8] for communication over a noisy channel, and to the topographic mapping theory of Kohonen [9] for training self-organising neural networks.

Define \( x \) as the input data, \( y \) as the compressed data, and \( x' \) as the reconstruction of the input data. Define \( y(x) \) as the compression operation \( x \rightarrow y \), and \( x'(y) \) as the reconstruction operation \( y \rightarrow x' \), which yields overall \( x' = x'(y(x)) \). Also define the probability density function (PDF) \( P(x) \) of possible input data \( x \). Note that throughout this paper we use the notation \( P(\cdots) \) to denote a PDF, whose functional form may be deduced from its context.

We may combine these quantities to obtain the average L2 (i.e. Euclidean) distortion \( D_1 \), as follows

\[
D_1 = \int dx P(x) \|x'(y(x)) - x\|^2 \tag{2.1}
\]

The VQ whose (continuum limit) codebook is defined by the pair of functions \( (y(x), x'(y)) \) can be optimised by minimising \( D_1 \) with respect to variations of \( y(x) \) and \( x'(y) \).

A more general form of Equation (2.1) that gives the average L2 distortion for a VQ with a noisy communication channel [3, 8] is

\[
D_2 = \int dx P(x) \int dy' \pi(y' - y(x)) \|x'(y') - x\|^2 \tag{2.2}
\]

In Equation (2.2) we assume that \( y' = y(x) + n \), where \( n \) is an independent random variable with PDF \( \pi(n) \), so \( P(x, n) = P(x) \pi(n) \).

We functionally differentiate \( D_2 \) to calculate the zero(s) of \( \frac{\partial D_2}{\partial y} \) and \( \frac{\partial D_2}{\partial x} \), which yields (see [3] for the details)

\[
y(x) = \arg \min_y \int dy' \pi(y' - y) \|x'(y') - x\|^2 \tag{2.3}
\]

\[
batch: x'(y) = \left( \frac{\partial D_2}{\partial y} \right) \frac{\partial y}{\partial x} x
\]

\[
continuous: \Delta x'(y) = \varepsilon \pi(y - y(x)) (x - x'(y)) \tag{2.4}
\]

In the noiseless case \( \pi(n) = \delta(n) \) Equation (2.3) specifies that the optimum \( y(x) \) is a nearest neighbour prescription, i.e. given \( x \) select as \( y(x) \) the \( y \) that minimises \( \|x'(y') - x\|^2 \). This nearest neighbour property is preserved in lowest order when \( \pi(n) \neq \delta(n) \), provided that the noise is zero mean [6]. Equation (2.4) further specifies two methods of updating \( x'(y) \). Firstly, the batch update prescription in Equation (2.4) is equivalent to one cycle of the LBG algorithm [7]. Secondly, the continuous update prescription in Equation (2.4) is identical to the topographic mapping training algorithm [9], so \( \pi(n) \) can be interpreted as a topographic neighbourhood function.

In Luttrell [10] we derive the asymptotic density of code vectors of topographic mappings trained using the minimum distortion prescription of Equation (2.3) and we find that it is independent of \( \pi(n) \), assuming scalar quantisation and mild monotonicity constraints on \( \pi(n) \). In Luttrell [11] we present an informal derivation of this result for the vector quantisation case. Because \( \pi(n) = \delta(n) \) is a standard VQ, this result leads to some of the asymptotic properties of VQs being the same as those of topographic mappings, provided that we use the minimum distortion rather than the nearest neighbour prescription.

Another generalisation of Equation (2.1) is to create a set of nested VQs. For instance, in Figure 1 we show a two-stage VQ. The components of the input pair \( x = (x_1, x_2) \) are separately transformed to yield the corresponding components of the pair \( y = (y_1, y_2) \), which

---

*This appeared in Proceedings of the 2nd IEE International Conference on Artificial Neural Networks, Bournemouth, 5-9, 1991. © British Crown Copyright 1991/MOD. Published with the permission of the Controller of Her Britannic Majesty's Stationery Office.

†Electronic address: lutrell@uk.mod.hermes@relay.mod.uk
In [3, 4] we present a method of training the channels sequentially updated by a training algorithm. Ideally the average approach is appropriate in situations where the channels do not mutually interfere, but this nevertheless leads to useful results. We may generalise incorrectly that the channels do not mutually interfere, it nevertheless leads to useful results. We may generalise Figure 1 by creating a multi-stage structure of nested VQs (as in Equation 2.2). Although this scheme assumes (incorrectly) that the channels do not mutually interfere, it nevertheless leads to useful results. We may generalise Figure 1 by creating a multi-stage structure of nested VQs, which may be used for time series and image compression, as we demonstrate in [5, 6].

![Figure 1: Two-stage vector quantisation. Two channels $x_1 \rightarrow y_1 \ldots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \ldots y_2' \rightarrow x_2'$ are coupled through a common channel $(y_1, y_2) \rightarrow z \rightarrow (y_1', y_2')$.](image1)

![Figure 1](image1)

![Figure 2: Ensemble average two-stage vector quantisation.](image2)

### III. ENSEMBLE AVERAGE TRAINING

In this section we consider the effect that mutual channel coupling has on optimising a nested VQ.

![Figure 2](image2)

In Figure 2 we show an ensemble average version of Figure 1 where we take the average over realisations of the nested VQ $(y_1, y_2) \rightarrow z \rightarrow (y_1', y_2')$. The statistical properties of the ensemble in Figure 2 are expressed using a transition probability $P(y_1', y_2'|y_1, y_2)$.

$$P(y_1', y_2'|y_1, y_2) = \int d[z(y_1, y_2)] d[y_1'(z)] d[y_2'(z)]$$

$$P(y_1'(z), y_2'(z), z(y_1, y_2)) \delta(y_1' - y_1'(z(y_1, y_2))) \delta(y_2' - y_2'(z(y_1, y_2)))$$

(3.1)

where $d[z(y_1, y_2)] d[y_1'(z)] d[y_2'(z)]$ is a volume element in function space, $P(y_1'(z), y_2'(z), z(y_1, y_2))$ is a PDF which specifies a measure in function space, and $\delta(y_1' - y_1'(z(y_1, y_2))) \delta(y_2' - y_2'(z(y_1, y_2)))$ is the contribution to $P(y_1', y_2'|y_1, y_2)$ from each $(y_1, y_2) \rightarrow z \rightarrow (y_1', y_2')$ member of the ensemble. The ensemble average approach is appropriate in situations where the $(y_1, y_2) \rightarrow z \rightarrow (y_1', y_2')$ transformation is continuously updated by a training algorithm. Ideally the $x_1 \rightarrow y_1 \ldots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \ldots y_2' \rightarrow x_2'$ transformations should be optimised to adapt to the changes to the $(y_1, y_2) \rightarrow z \rightarrow (y_1', y_2')$ transformation, but this is time consuming. Rather, it is better to adapt to the properties of the ensemble of $(y_1, y_2) \rightarrow z \rightarrow (y_1', y_2')$ transformations that might occur.

In [3, 4] we did not introduce $P(y_1'(z), y_2'(z), z(y_1, y_2))$ explicitly, but directly modelled $P(y_1', y_2'|y_1, y_2)$ by introducing additive noise independently into the $x_1 \rightarrow y_1 \ldots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \ldots y_2' \rightarrow x_2'$ channels. Because of the independence assumption this approach is crude, but it is demonstrably effective and useful [5, 6]. Using $P(y_1', y_2'|y_1, y_2)$ the expression for the overall distortion in Figure 2 is given by

$$D_3 = \int dx_1 dx_2 P(x_1, x_2) \int dy_1' P(y_1'|y_1(x_1), y_2(x_2)) \|x_1'(y_1') - x_1\|^2 + (1 \longrightarrow 2)$$

(3.2)

where $P(y_1'|y_1(x_1), y_2(x_2))$ is a marginal PDF obtained by integrating $P(y_1', y_2'|y_1(x_1), y_2(x_2))$ over $y_2'$, and $(1 \longrightarrow$
2) indicates that there is an analogous term for channel 2. We optimise the nested VQ by minimising $D_3$ to obtain

$$ (y_1(x_1), y_2(x_2)) = \arg\min_{(y_1, y_2)} \left( \int dy_1' P(y_1'|y_1, y_2) \|x_1'(y_1') - x_1\|^2 + (1 \rightarrow 2) \right) $$

which should be compared with Equation 2.3 and Equation 2.4. Note how Equation 3.3 specifies a minimum distortion prescription in which $y_1(x_1)$ and $y_2(x_2)$ are simultaneously optimised, using $P(y_1'|y_1, y_2)$ and $P(y_2'|y_1, y_2)$ instead of $\pi(y' - y)$. We usually approximate this by using a nearest neighbour prescription, although see [10] for an exception to this.

![Figure 3: The marginal PDFs $P(y_1'|y_1, y_2)$ and $P(y_2'|y_1, y_2)$ of the ensemble average distortion $P(y_1', y_2'|y_1, y_2)$ determine the topographic neighbourhood functions for optimising the $x_1 \rightarrow y_1 \cdots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \cdots y_2' \rightarrow x_2'$ channels.](image)

In Figure 3 we represent diagrammatically in $(y_1, y_2)$-space (and $(y_1', y_2')$-space) the various terms of Equation 3.2. We represent the contours of a typical $P(y_1'(x_1), y_2(x_2))$, a typical $P(y_1', y_2'|y_1(x_1), y_2(x_2))$, and the profiles of its two marginals $P(y_1'(y_1(x_1), y_2(x_2)))$ and $P(y_2'|y_1(x_1), y_2(x_2))$. These marginals have a shape that depends on $(x_1, x_2)$, which therefore mutually couples the contributions to the distortion in Equation 3.2, arising from the two transformations $x_1 \rightarrow y_1 \cdots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \cdots y_2' \rightarrow x_2'$. It is both pleasing and economical that the ensemble average nested VQ in Figure 2 automatically determines the topographic neighbourhood functions for its $x_1 \rightarrow y_1 \cdots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \cdots y_2' \rightarrow x_2'$ transformations, thus eliminating the need to introduce them by hand.

IV. ANALYTICALLY SOLVABLE QUANTISATION MODEL

In this section we present an analytically solvable model of the ensemble distortion $P(y_1', y_2'|y_1, y_2)$ shown in Figure 3. This is the main result that we present in this paper.

A. Code vector density

The definition of the ensemble average in Equation 3.1 is controlled by the $P(y_1'(z), y_2'(z), z|y_1, y_2)$ factor, which specifies these transformations that we consider to be likely outcomes of the optimisation of $y_1, y_2 \rightarrow z \rightarrow y_1', y_2'$. In practice, the global minimum of $D_3$ has many feasible solutions for $(y_1'(z), y_2'(z), z|y_1, y_2)$, and there are many other solutions lying close to this minimum. The problem of deriving $P(y_1'(z), y_2'(z), z|y_1, y_2)$ would seem to be intractable because of the complicated correlations that must exist between the feasible solutions for $y_1'(z), y_2'(z)$ and $z|y_1, y_2$.

If the number of code vectors $(y_1'(z), y_2'(z))$ in the $(y_1, y_2) \rightarrow z \rightarrow (y_1', y_2')$ codebook is very large, we may calculate $P(y_1', y_2'|y_1, y_2)$ directly. Thus we model the ensemble properties of the codebook by defining $\rho(y_1, y_2)$, which specifies the density of code vectors $(y_1'(z), y_2'(z))$ in $(y_1, y_2)$-space.

B. Transition probability: integral equation

Note that we usually write $\rho(y)$ and $P(y'|y)$ to stand for the density of code vectors $\rho(y_1, y_2)$ and the transition probability $P(y_1', y_2'|y_1, y_2)$ respectively.

In Figure 4 we compare the nearest neighbour prescription for a single VQ with that for an ensemble of VQs. In Figure 4a we show an input vector (represented by a cross) and the known positions of the code vectors of a single VQ. The nearest neighbour can be located by expanding a circle centred on the input vector until it grazes the nearest code vector, as shown. In Figure 4b we show the ensemble version of the same diagram, in which the precise code vector positions are unknown, so
there is a distribution $P(y'|y)$ of possible nearest neighbour locations.

For completeness, we discuss here the intermediate case where the positions of the code vectors are partially known. The most important way of acquiring partial knowledge is to note the positions of the nearest neighbour code vectors during training. However, such knowledge must be continuously updated because migration of the code vector positions gradually erases any memory of their earlier positions. Partial knowledge lies between the extremes of Figure 4a and Figure 4b, and its analysis is very complicated. We choose to analyse the extreme case in Figure 4b because it underestimates, rather than overestimates, the knowledge that is available.

We may develop an integral equation that relates $P(y'|y)$ to $\rho(y)$ as follows.

$$P(y'|y)\delta y' = \left(1 - \int_{\|\xi - y\|\leq\|y' - y\|} d\xi P(\xi|y)\right)\rho(y') \delta y'$$

(4.1)

where the first term on the right hand side is the probability that there is no nearest neighbour code vector in the sphere of radius $\|y' - y\|$ centred on $y$, and the second term is the probability of finding a code vector in the volume $\delta y'$ at $y'$. The product of these two terms gives the probability of finding the nearest neighbour code vector in the volume $\delta y'$ at $y$.

C. Transition probability: constant code vector density case

We now solve Equation 4.1 for the case $\rho(y) = \rho_0 = \text{constant}$. The nearest neighbour code vector is then equally likely to lie in any direction from $y$, so $P(y'|y)$ must be a function only of the radial distance $\|y' - y\|$, which gives

$$P(\|y' - y\|) = \left(1 - \int_{\|\xi - y\|\leq\|y' - y\|} d\xi P(\|\xi - y\|)\right)\rho_0$$

(4.2)

where $\|y' - y\| = (y' - y)^T(y' - y)$. The integrand is spherically symmetric so we may use the transformation

$$\int_{\|\xi\|\leq\|y' - y\|} d\|\xi\|\|\xi\|^{N-2} P(\|\xi\|)$$

(4.3)

where $P_0$ should be adjusted to ensure that $P(\|y' - y\|)$ is normalised correctly. The $N = 2$ case reduces to a Gaussian distribution with $P_0 = \rho_0$.

D. Transition probability: variable code vector density case

We now extend the previous results to the case

$$\rho(y') = \rho(y) + (y' - y)^T \nabla \rho(y)$$

(4.6)

which is a first order Taylor expansion of $\rho(y')$ about
the point \( y' = y \). We anticipate that the first order expansion of \( P(y'|y) \) has the form of Equation 4.5 with an extra factor to account for the angular dependence in Equation 4.6

\[
P(y'|y) \approx P_0(y) \left( 1 + (y' - y)^T \cdot a(y) \right) \exp(-\frac{\alpha_N \rho(y) \|y' - y\|^N}{N})
\]

(4.7)

where \( a(y) \) has to be determined. Insert Equation 4.7 into Equation 4.1 and differentiate with respect to \( y' \) to obtain

\[
P(y'|y) = P_0(y) \left( 1 + (y' - y)^T \frac{\nabla \rho(y)}{\rho(y)} \right) \exp(-\frac{\alpha_N \rho(y) \|y' - y\|^N}{N})
\]

(4.8)

In practical applications we must estimate \( \rho(y) \) and \( \nabla \rho(y) \) from limited information. Clearly, it is not correct to approximate \( \rho(y) \) as a mixture of Dirac deltas located at each of the current code vector positions, because this would provide knowledge of the exact VQ, rather than its ensemble properties. We suggest that a local average of this mixture should be made in order to eradicate precise knowledge of the code vector positions.

Strictly speaking, Equation 4.8 does not specify a valid probability distribution because it yields a negative probability when \( (y' - y)^T \frac{\nabla \rho(y)}{\rho(y)} < -1 \). However, this result is the leading order term in a Taylor expansion about \( y' = y \) (see Equation 4.6), therefore we implicitly assume \( \|(y' - y)^T \frac{\nabla \rho(y)}{\rho(y)}\| \ll 1 \). The effect of the

\[
1 + (y' - y)^T \frac{\nabla \rho(y)}{\rho(y)}
\]

term is to relocate the maximum of \( P(y'|y) \) from its original position at \( y' = y \) (see Equation 4.5) to a new position given by

\[
y' \approx y + \left( \frac{\|\nabla \rho(y)\|}{\alpha_N \rho(y)^2} \right) \frac{\nabla \rho(y)}{\|\nabla \rho(y)\|}
\]

(4.9)

The direction of shift is consistent with the bias in \( P(y'|y) \) that we show in Figure 3 and Figure 4.

Finally, we marginalise the joint distribution \( P(y_1', y_2'|y_1, y_2) \) in Equation 4.8 in order to calculate \( P(y_1'|y_1, y_2) \) and \( P(y_2'|y_1, y_2) \) (which are needed in Equation 3.3 and Equation 3.4). For the 2-dimensional case \((N = 2, \alpha_2 = 2\pi, y_1, y_2 \rightarrow (y_1, y_2)) \) this is easy because the exponential factors are Gaussians, leading to the result

\[
P(y_1'|y_1, y_2) \approx P_0(y_1, y_2) \left( 1 + \frac{(y_1' - y_1) \partial \rho(y_1, y_2)}{\rho(y_1, y_2)} \right) \exp(-\pi \rho(y_1, y_2) (y_1' - y_1)^2)
\]

(4.10)

with an analogous result for \( P(y_2'|y_1, y_2) \). These results may be used to model the marginals in Figure 5.

Recall that \( P(y_1'|y_1, y_2) \) and \( P(y_2'|y_1, y_2) \) serve as topographic neighbourhood functions for optimising the \( x_1 \rightarrow y_1 \cdots y_1' \rightarrow x_1' \) and \( x_2 \rightarrow y_2 \cdots y_2' \rightarrow x_2' \) transformations. In the ensemble average model, these neighbourhood functions emerge naturally from the ensemble properties of \( (y_1, y_2) \rightarrow z \rightarrow (y_1', y_2') \), so we do not need to supply them manually. We call this property self-supervision, because the topographic neighbourhood functions that are required by one part of the network are automatically supplied by another part of the network.

E. Optimisation of the joint output density

The update prescription depends on the biased marginals \( P(y_1'|y_1, y_2) \) and \( P(y_2'|y_1, y_2) \), which causes a migration of \( P(y_1, y_2) \) as shown in Figure 5, where the arrows represent the direction of migration. The widths of the marginals \( P(y_1'|y_1, y_2) \) and \( P(y_2'|y_1, y_2) \) determine the widths of the vertical and horizontal bands of \( P(y_1, y_2) \) that are affected.

Strictly speaking, the change to \( P(y_1, y_2) \) is not restricted entirely to the vicinity of the two regions indicated in Figure 5. For instance, the movement of the code vectors in the topographic neighbourhood of \( y_1(x) \) and \( y_2(x) \) can change the shape of the quantisation cells of other code vectors, which, in turn, causes other changes
to $P(y_1,y_2)$. However, this is a second order effect.

We see from Figure 5 that the net migration averaged over all inputs has the affect of squeezing the $P(y_1,y_2)$ distribution. This inward pressure is counterbalanced by the stretching tendency of each marginal $P(y_1)$ and $P(y_2)$ to become approximately uniform, as normally occurs in VQs. Informally, we can interpret this competition as tending to maximise the mutual information $I[y_1:y_2]$ between $y_1$ and $y_2$. Thus $I[y_1:y_2] = H[y_1] + H[y_2] - H[y_1,y_2] \geq 0$, where $H[\cdot]$ is the entropy of its argument, and stretching causes $H[y_1]$ and $H[y_2]$ to increase, whereas squeezing causes $H[y_1,y_2]$ to decrease, hence $I[y_1:y_2]$ tends to increase, although this is not absolutely guaranteed.

Mutual information can be used as our basic optimisation criterion instead of $L_2$ distortion minimisation. An example of this approach and its relationship to the optimisation of a novel class of hierarchical Gibbs distributions can be found in Luttrell [12].

V. CONCLUSIONS

This paper makes two main points. Firstly, we introduce an ensemble hierarchical VQ model, where the lower layers of the hierarchy use knowledge of the ensemble properties of the higher layers of the hierarchy, rather than their exact properties. Secondly, we present an analytically solvable model, where the ensemble properties of the higher layers give rise to data dependent topographic neighbourhood functions, which influence the optimisation of the lower layers. We call this effect self-supervision.

We may express the concept of self-supervision in a language that is more appropriate to unsupervised neural networks. The higher layers of a network respond to longer space/time scale features in the input data, and issue backpropagating signals that cause the lower layers to process the data in such a way that overall the network responds more strongly to the input data.