Learning Speech Representations with Variational Predictive Coding

Sung-Lin Yeh **Peter Bell** **Hao Tang**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
sunglin.yeh@ed.ac.uk Peter.Bell@ed.ac.uk hao.tang@ed.ac.uk

Abstract

Despite being the best known objective for learning speech representations, the HuBERT objective has not been further developed and improved. We argue that it is the lack of an underlying principle that stalls the development, and, in this paper, we show that predictive coding under a variational view is the principle behind the HuBERT objective. Due to its generality, our formulation provides opportunities to improve parameterization and optimization, and we show two simple modifications that bring immediate improvements to the HuBERT objective. In addition, the predictive coding formulation has tight connections to various other objectives, such as APC, CPC, wav2vec, and BEST-RQ. Empirically, the improvement in pre-training brings significant improvements to four downstream tasks: phone classification, f0 tracking, speaker recognition, and automatic speech recognition, highlighting the importance of the predictive coding interpretation.

1 Introduction

Self-supervised learning has been the most successful approach to semi-supervised learning, leveraging unlabeled data for various downstream tasks (Zhang et al., 2022). The impact of self-supervised learning in speech processing has now been extended to the pursuit of more accessible yet compact, discrete representations to interact with language models (Lakhotia et al., 2021; Borsos et al., 2023; Wang et al., 2023). Behind these successes, other than the ease of scaling with Transformers, the training objectives are the major factor that makes the advancement possible.

The training objective of HuBERT (Hsu et al., 2021) is undoubtedly the most successful self-supervised objective. The HuBERT objective combines quantization and masked prediction as its main components. However, those choices are supported by empirical evidence rather than an underlying principle, and it becomes difficult to further improve the HuBERT objective and to design novel objectives. This is evidenced by the fact that subsequent work has focused on data augmentation (e.g., WavLM (Chen et al., 2022)), simplification of training (e.g., BEST-RQ (Chiu et al., 2022) and MelHuBERT (Lin et al., 2023)), pairing with other objectives (e.g., w2v-BERT (Chung et al., 2021), DinoSR (Liu et al., 2024), and MS-HuBERT (Yadav et al., 2024)), training with different resolution (Shi et al., 2023); none has improved the HuBERT objective itself.

In this paper, we will show that predictive coding is the underlying principle of HuBERT. It is not difficult to see the lineage of predictive coding in HuBERT. HuBERT is inspired by wav2vec 2.0 (Baevski et al., 2020b) and DeepCluster (Caron et al., 2018), and wav2vec 2.0 is in turn influenced by BERT (Devlin et al., 2019) and contrastive predictive coding (CPC) (van den Oord et al., 2018). However, the HuBERT objective looks distinctively different from, say, the formulation in Atal and Hanauer (1971), Srinivasan et al. (1982), and Rao and Ballard (1999). What is missing is a general framework for predictive coding (to subsume masked prediction) and a discrete intermediate representation (to subsume quantization).

In this paper, the general framework we develop is a variational view of predictive coding and we present HuBERT as a special case. We will derive a training objective from first principles, and will show how it relates, not only to the HuBERT objective but also to various other objectives, such as VQ-APC (Chung et al., 2020), VQ-CPC (van Niekerk et al., 2020), wav2vec 2.0, and BEST-RQ. Moreover, our derivation can spawn new objectives, and we will give an example that brings immediate improvement over HuBERT. Once we have an objective, the optimization of it, i.e.,

Transactions of the Association for Computational Linguistics, vol. 14, pp. 1510–1525, 2026. https://doi.org/10.1162/TACL.a.738
Action Editor: Taro Watanabe. Submission batch: 9/2025; Revision batch: 12/2025; Published 7/2026.

the algorithm of learning representations, naturally decouples, and provides opportunities for improvement. We will give an example of how optimizing the HuBERT objective with a different algorithm brings immediate improvement over HuBERT. We will also validate how much the improvement on the self-supervised objective transfers to downstream performance.

2 A Variational Framework for Predictive Coding

To see how predictive coding relates to HuBERT, in this section, we briefly review predictive coding, and its variational formulation.

Given its long history, predictive coding comes in various forms. Predictive coding at the conceptual level, as described in Elias (1955), involves an encoder on one end and a decoder on the other. The encoder processes a signal as it comes in (e.g., frames of a speech utterance in streaming mode), predicts what comes next, and computes the residuals (or how off the prediction is). The decoder receives the residuals, makes a prediction, and combines the two to reconstruct the signal. The hope is that the residuals require fewer bits to send than the original signal, achieving the goal of coding.

When predictive coding later evolves into algorithms, the goal becomes predicting one part of a signal given the other. For example, it is predicting the next wave sample given the past samples in Atal and Hanauer (1971), and predicting the center pixel given the neighboring pixels in Srinivasan et al. (1982). A detailed exposition is beyond the scope of this paper and can be found in Makhoul (1975), Huang and Rao (2011), and Spratling (2017).

The general idea of predicting one part of a signal given the other can be formalized as follows. Let x be a signal, for example, wave samples of a speech utterance. Predictive coding is about learning $p(x_B|x_A)$, or minimizing $-\log p(x_B|x_A)$, where x_A and x_B forms a partition of x. To learn the entirety of x, the partition is not fixed but drawn stochastically, resulting in the objective

$$\mathbb{E}_{(x_A,x_B)\sim\mathcal{M}(x)}[-\log p(x_B|x_A)], \tag{1}$$

where $\mathcal{M}(x)$ is a distribution over many partitions of x. The formulation in Atal and Hanauer (1971) can be seen as choosing an $\mathcal{M}$ to partition wave samples in the future and the past, while in Srinivasan et al. (1982), $\mathcal{M}$ is chosen to partition the center pixel and the neighboring pixels. For the interest of this paper, the signal x is a sequence of acoustic frames $x_1, \ldots, x_T$, and we partition the signal into $x_A = \{x_i\}_{i\in A}$ and $x_B = \{x_i\}_{i\in B}$ based on two sets of time indices $A \subset \{1, \ldots, T\}$ and $B = \{1, \ldots, T\} \setminus A$. It is not difficult to see that x_A will be the masked frames and x_B will be the unmasked frames in HuBERT, and we willl make this explicit in the next section.

From the coding perspective, a few important components are missing in equation 1: the encoder, the decoder, and the message sent from the encoder to the decoder. Suppose the encoder encodes x_A into a message z and sends z to the decoder to infer x_B. We assume that knowing z is sufficient to infer x_B, i.e., $x_B \perp\!\!\!\perp x_A \mid z$. Otherwise, the compression is deemed lossy. A variational upper bound of equation 1 can then be written as

$$\mathbb{E}_{(x_A,x_B)\sim\mathcal{M}(x)}\Big[\mathrm{KL}\big[q(z|x_B)\|p(z|x_A)\big] - \mathbb{E}_{z\sim q(z|x_B)}[\log p(x_B|z)]\Big], \tag{2}$$

where $q(z|x_B)$ is an auxiliary distribution of our choice.[1] The second term $\mathbb{E}_{z\sim q(z|x_B)}[\log p(x_B|z)]$ is known as the reconstruction (or distortion in coding), where $p(z|x_A)$ is thought of as the encoder, $p(x_B|z)$ the decoder, and z the message. The variational formulation has the advantage of making the encoder, decoder, and message explicit in the objective. Equation 2 is known as the negative free energy or the negative evidence lower bound (negative ELBO), and this view of predictive coding is first made explicit in Friston and Kiebel (2009) and later generalized in Feldman and Friston (2010). Our treatment adheres more to the variational lower bound in Kingma and Welling (2014) and Sohn et al. (2015) with the additional assumption that $x_B \perp\!\!\!\perp x_A \mid z$.

3 HuBERT as Predictive Coding

Given the variational framework of predictive coding, we now turn to the HuBERT objective and discuss how it relates to predictive coding. Recall that HuBERT training consists of two steps: first quantizing the acoustic frames and second training a Transformer to predict the cluster IDs of the

[1] See Appendix A.1 for the derivation of our objective based on the variational lower bound.

quantized frames. The HuBERT objective often refers to the cross entropy of predicting the cluster IDs of each frame (shown as KL in Figure 1). However, the cluster IDs of the quantized frames are produced by k-means, so there is an implicit ℓ_2 loss that measures the distortion (or reconstruction) of k-means (shown as MSE in Figure 1).

For the following subsections, we will detail how the partition $\mathcal{M}$ and the parameterization of $q(z|x_A)$, $p(x_B|z)$, and $p(z|x_A)$ are chosen, such that the variational objective in Equation 2 covers the cross entropy and the ℓ_2 loss. In particular, we will assume the latent variables are discrete and correspond to the cluster IDs of frames.

3.1 Masked Prediction

When training HuBERT, a mask is generated at random for every utterance, and frames in an utterance are partitioned into those that are masked and those that are not. The objective is to predict the cluster IDs of the masked frames given the unmasked frames. Formally, a mask is a subset of indices $M \subset \{1, \dots, T\}$, and forms a partition $x_M = \{x_i\}_{i \in M}$ and $x_{\setminus M} = \{x_i\}_{i \notin M}$. Let $\mathcal{M}(x)$ be the stochastic process of generating masks for the utterance x, where typically a frame has a small probability to be the start of a span of frames being masked (Baevski et al., 2020b). With this choice of $\mathcal{M}$, the predictive coding objective (equation 2) becomes

$$\mathcal{L}_{\text{Masked}} = \mathbb{E}_{(x_{\setminus M}, x_M) \sim \mathcal{M}(x)} [\mathcal{L}_{x_{\setminus M}, x_M}] \quad (3)$$

where

$$\mathcal{L}_{x_{\setminus M}, x_M} = \text{KL}\big[q(z|x_M) \| p(z|x_{\setminus M})\big] + \mathbb{E}_{z \sim q}[-\log p(x_M|z)]. \quad (4)$$

Since the HuBERT objective is frame-wise, we assume $q(z|x_M)$ and $p(x_M|z)$ factorize frame-wise, i.e., $q(z|x_M) = \prod_{i \in M} q(z_i|x_i)$ and $p(x_M|z) = \prod_{i \in M} p(x_i|z_i)$, where $z_1, \dots, z_T$ are discrete latent variables for frames $x_1, \dots, x_T$. After including the frame-wise independence, we have

$$\mathcal{L}_{x_{\setminus M}, x_M} = \sum_{i \in M} \mathbb{E}_{z_i \sim q}\big[\log q(z_i|x_i) - \log p(z_i|x_{\setminus M}) - \log p(x_i|z_i)\big]. \quad (5)$$

Each $z_i \in \{1, \dots, K\}$ corresponds to the cluster ID of x_i, where K is the total number of clusters. Note that the second term $\mathbb{E}_{z_i \sim q}[-\log p(z_i|x_{\setminus M})]$ is the cross entropy, and the last term $\mathbb{E}_{z_i \sim q}[-\log p(x_i|z_i)]$ is the reconstruction. We will discuss how these two terms become the cross entropy and the ℓ_2 loss of k-means in the HuBERT objective. We will also discuss why the first term $\mathbb{E}_{z_i \sim q}[\log q(z_i|x_i)]$, the negative entropy, does not appear in the HuBERT objective.

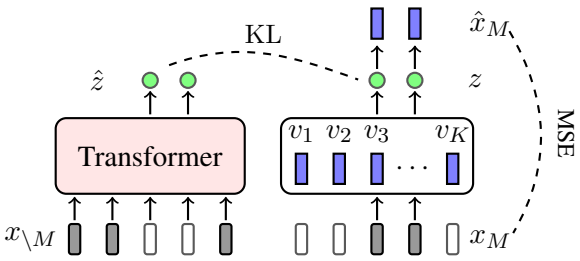

Figure 1: HuBERT as variational predictive coding. The set $v_1, \dots, v_K$ are the codewords in the codebook, $x_{\setminus M}$ are the unmasked frames, and x_M are the masked frames. There are two loss functions involved, the Kullback-Leibler divergence (KL) and the mean-squared error (MSE).

3.2 Quantization and Reconstruction

In HuBERT, cluster IDs of frames that later serve as targets for prediction are produced by k-means. The k-means algorithm finds the ID of the closest centroid for each individual frame, where closeness is defined by the ℓ_2 loss. To realize the quantization with k-means in our predictive coding framework, we assign a point mass to the minimum and let

$$q(z_i|x_i) = \mathbb{1}_{z_i = \text{argmin}_{k=1,\dots,K} \|x_i - v_k\|^2}, \quad (6)$$

where v_k is the k-th column of a matrix V, a codebook consisting of the centroids as columns. Note that q in principle can depend on x_M, but for this particular choice, q only depends on x_i.

The k-means objective is to minimize the distortion (or reconstruction) measured by the ℓ_2 loss. Given the codebook V, using the cluster z_i to reconstruct x_i with the centroid v_{z_i} gives a distortion $\|x_i - v_{z_i}\|^2$. Naturally, we choose to parameterize $p(x_i|z_i)$ with a Gaussian and let

$$p(x_i|z_i) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\|x_i - v_{z_i}\|^2\right), \quad (7)$$

where d is the dimension of an acoustic frame. The log of $p(x_i|z_i)$ gives the ℓ_2 loss, i.e., the mean-squared error (MSE). It is now clear that the term

$\mathbb{E}_{z_i \sim q}[-\log p(x_i|z_i)]$ in equation 5 involves both the quantization of a frame x_i and reconstructing it with the closest centroid v_{z_i}. The negative entropy term $\mathbb{E}_{z_i \sim q}[\log q(z_i|x_i)]$ becomes 0 due to q being a point mass.

3.3 Predicting Quantized Frames

Because $z_1, \ldots, z_T$ correspond to the targets for prediction, we parameterize $p(z_i|x_{\setminus M})$ as a softmax, i.e.,

$$p(z_i|x_{\setminus M}) = \frac{\exp\left(\text{enc}(x_{\setminus M})_i^\top u_{z_i}\right)}{\sum_{k=1}^{K} \exp\left(\text{enc}(x_{\setminus M})_i^\top u_k\right)}, \quad (8)$$

where $\text{enc}(\cdot)$ is an encoder (typically a Transformer encoder), $\text{enc}(x_{\setminus M})_i$ is the i-th frame of the encoder output after taking the unmasked frames $x_{\setminus M}$ as input, and u_k is the k-th column of the final linear layer U. The choice of $q(z_i|x_i)$ paired with the $p(z_i|x_{\setminus M})$ above completes the cross entropy $\mathbb{E}_{z_i \sim q}[-\log p(z_i|x_{\setminus M})]$ in equation 5 for predicting the cluster IDs of frames.

3.4 Two-Step Optimization

We have now instantiated the HuBERT objective from predictive coding. The cross entropy term $\mathbb{E}_{z_i \sim q}[-\log p(z_i|x_{\setminus M})]$ and the reconstruction term $\mathbb{E}_{z_i \sim q}[-\log p(x_i|z_i)]$ are both present in the objective (equation 5). In principle, both terms should be optimized together, but HuBERT takes a two-step approach, first finding the codebook by optimizing the reconstruction (running k-means) and then finding the Transformer parameters by optimizing the cross entropy. The two-step approach is reminiscent to the variational view of expectation maximization (Neal and Hinton, 1998; Attias, 1999). However, the codebook in HuBERT is never updated again after the first k-means, and is said to be optimized offline as opposed to jointly with the Transformer parameters. This provides an opportunity for improving the optimization of the HuBERT objective.

4 Extensions

Given how HuBERT is a special case of predictive coding, in this section, we discuss several immediate extensions that are made possible by our framework.

4.1 Softening the Point Mass

Even though the two-step optimization in HuBERT likely leads to suboptimal solutions, it turns out to be difficult to optimize both the cross entropy and the reconstruction together. The difficulty stems from q being a point mass, which cannot be optimized with gradient descent. Instead of exact minimization, we can use soft-min and let

$$q(z_i|x_i) = \frac{\exp\left(-\|x_i - v_{z_i}\|^2/\tau\right)}{\sum_{k=1}^{K} \exp\left(-\|x_i - v_k\|^2/\tau\right)}, \quad (9)$$

where τ is the temperature. As $\tau \to 0$, the distribution collapses to minimization or a hard k-means assignment (Kulis and Jordan, 2011), where each x_i is assigned to the code v_{z_i} with the smallest squared Euclidean distance. With this parameterization, it is now possible to optimize the entire objective in equation 5 with gradient descent.

Since q is no longer a point mass, we now have an additional negative entropy term $\mathbb{E}_{z_i \sim q}[\log q(z_i|x_i)]$ to optimize in equation 5. As entropy is maximized when q is uniform, this term, when being minimized with other terms, encourages a diverse set of codes to be used and serves as a regularizer. This is reminiscent to the diversity loss in wav2vec 2.0, and we will discuss the differences in later sections.

4.2 Approximating the Expectation with Sampling

Since $p(x_i|z_i)$ is parameterized in a relatively simple form, computing the expectation $\mathbb{E}_{z_i \sim q}[-\log p(x_i|z_i)]$ by enumerating all values of z_i, i.e., exact marginalization, is feasible. However, exact marginalization is not always feasible when $p(x_i|z_i)$ becomes expensive to compute. An alternative is to approximate the expectation with sampling. There are various approaches to optimizing a function that involves sampling, and the simplest solution is to use Gumbel softmax (Jang et al., 2017). We will compare marginalization and Gumbel softmax in the experiments. Note that, the expectation is optimized offline with k-means in HuBERT, which will also be included in the comparisons.

4.3 Future Prediction

Our framework of predictive coding can also instantiate future prediction given the past.[2] The partition of a signal is a simpler than the masked prediction in that choosing a time point partitions

[2]Next-token prediction in language modeling, e.g., is a form of future prediction.

the signal into the past and the future. Formally, let $\mathcal{M}(x)$ be the stochastic process of choosing a time point t in a signal x. The past $x_{<t}$ and the future $x_{\geq t}$ forms a partition of x, and our autoregressive objective becomes

$$\mathcal{L}_{\text{Future}} = \mathbb{E}_{(x_{<t}, x_{\geq t}) \sim \mathcal{M}(x)} [\mathcal{L}_{x_{<t}, x_{\geq t}}] \tag{10}$$

where

$$\mathcal{L}_{x_{<t}, x_{\geq t}} = \sum_{i=t}^{T} \mathbb{E}_{z_i \sim q} \big[\log q(z_i|x_i) - \log p(z_i|x_{<i}) - \log p(x_i|z_i) \big]. \tag{11}$$

The form of equation 11 is nearly identical to masked prediction in equation 5, except the term $p(z_i|x_{<i})$.[3] In terms of parameterization, $p(z_i|x_{<i})$ is typically modeled with a unidirectional LSTM or a causal Transformer; the rest of the terms remain the same. Note that only the suffix of a signal participates in the objective, and $\mathcal{M}$ needs to be carefully chosen to avoid putting too much weight on the suffixes. In practice, instead of choosing a time point at random, all frames are predicted with an equal amount of times (van den Oord et al., 2018; Chung et al., 2019).

Since speech is generally smooth in time, an additional assumption $z_i \perp\!\!\!\perp x_{i-\kappa+1:i} \mid x_{\leq i-\kappa}$ is commonly made for a small $\kappa > 0$ (e.g., in van den Oord et al. (2018) and Chung et al. (2019)). In other words, once we know the past frames $x_{\leq i-\kappa}$ close enough to the current time point i, knowing additional few frames $x_{i-\kappa+1:i}$ does not add much information to z_i. The term $p(z_i|x_{<i})$ becomes $p(z_i|x_{\leq i-\kappa})$ under this assumption.

5 Connections to Prior Work

We have demonstrated in Section 3 how our framework instantiates HuBERT. Given the generality of our framework, we can also instantiate other objectives that are similar to other self-supervised objectives. In this section, we will discuss what our framework can achieve and how it differs from prior work, including approaches based on likelihood and contrastive learning. We do not consider combined objectives in this section, such as w2v-BERT (Chung et al., 2021).

[3]The derivation of future prediction can be found in Appendix A.1.

5.1 APC and VQ-APC

We begin with other variants of predictive coding for speech representation that optimizes the likelihood equation 1, in which x_A is $x_{<t}$, x_B is $x_{\geq t}$. First, autoregressive predictive coding (Chung et al., 2019; Yang et al., 2022) (APC) is a generalization of linear predictive coding (Atal and Hanauer, 1971; Saito et al., 1967) (LPC), where a model is trained to predict future frames given the past. Its vector-quantized variant, VQ-APC (Chung et al., 2020), includes a quantization layer in the neural network while optimizing APC. Both APC and VQ-APC optimize the likelihood, while our framework is based on the variational bound in equation 2. APC and VQ-APC in their original form in Chung et al. (2019) and Chung et al. (2020) do not have the latent variables explicitly stated. Our framework makes the choice of latent variables explicit, and the proposed auxiliary distribution offers more flexibility to estimate the likelihood, especially when marginalization of latent variables is not tractable.

Similar to our approach, Yang et al. (2022) and Yeh and Tang (2022) make the latent variables explcit, leading to various other interpretations of APC with respect to, e.g., mutual information and co-training (McAllester, 2018).

5.2 MPC and DeCoAR

MPC (Jiang et al., 2019; Zhang et al., 2021) and DeCoAR (Ling et al., 2020) are both generalizations of APC, replacing future prediction with masked prediction. DeCoAR 2.0 (Ling and Liu, 2020) adds quantization similar to VQ-APC. Both optimize the likelihood in equation 1. The difference between them lies in how the frames are masked—MPC masks many small spans while DeCoAR masks a single large span.

5.3 HuBERT, WavLM, and BEST-RQ

We have shown that HuBERT is a special case of our framework. Subsequent work, such as WavLM (Chen et al., 2022) and BEST-RQ (Chiu et al., 2022), uses the same HuBERT objective. WavLM improves over HuBERT with data augmentation. BEST-RQ avoids updating the codebook, but requires a careful initialization.

When training HuBERT in Hsu et al. (2021), there are multiple iterations, each of which trains a Transformer encoder from scratch. What differs for each iteration are the training targets. In

the first iteration, quantized MFCC is used as targets, while in the second iteration, quantized hidden vectors from layer 9 of the first iteration are used as targets. Our framework can also instantiate the second iteration with a pre-trained Transformer encoder from the first iteration. We can simply let

$$q(z_i|h_i) = \frac{\exp\left(-\|h_i - v_{z_i}\|^2/\tau\right)}{\sum_{k=1}^{K}\exp\left(-\|h_i - v_k\|^2/\tau\right)}, \quad (12)$$

where the i-th frame h_i now comes from the intermediate layer (e.g., 6-th layer) of the encoder from the first iteration.

5.4 CPC, VQ-CPC, and wav2vec 2.0

CPC (van den Oord et al., 2018), its vector-quantized variants, VQ-CPC (van Niekerk et al., 2020), wav2vec (Schneider et al., 2019), and wav2vec 2.0 (Baevski et al., 2020b), are all based on noise contrastive estimation (NCE) (Smith and Eisner, 2005; Gutmann and Hyvärinen). In this section, we show how wav2vec 2.0 can be instantiated with our framework.

We derive their contrastive objective in wav2vec 2.0 from the cross entropy in the variational objective $\mathbb{E}_{z_i\sim q}[-\log p(z_i|x_{\setminus M})]$. Recall that wav2vec 2.0 has a CNN followed by VQ and a Transformer, where the VQ produces targets for contrastive learning. We first choose

$$q(z_i|x_M) = \delta_{z_i = g(x_M)} \quad (13)$$

$$p(z_i|x_{\setminus M}) = \frac{\exp(\mathrm{sim}(\mathrm{enc}(x_{\setminus M})_i, z_i))}{\int \exp(\mathrm{sim}(\mathrm{enc}(x_{\setminus M})_i, z))dz}, \quad (14)$$

where $g(\cdot)$ is the CNN with the VQ, $\mathrm{enc}(\cdot)$ is the Transformer, δ being the Dirac delta function, and $\mathrm{sim}(x, y) = \cos(x, y)$. The choice of $p(z_i|x_{\setminus M})$ is similar to that of Equation 8 in our framework, except a similarity function and a continuous z. The cross entropy becomes

$$\mathbb{E}_{z_i\sim q}[-\log p(z_i|x_{\setminus M})] = \log \frac{\exp(\mathrm{sim}(\mathrm{enc}(x_{\setminus M})_i, z_i))}{\int \exp(\mathrm{sim}(\mathrm{enc}(x_{\setminus M})_i, z'))dz}. \quad (15)$$

Note that z_i here is the codeword after quantization, i.e., a continuous vector (Baevski et al., 2020b). The denominator in the cross entropy is in general difficult to compute due to the integral.[4]

[4]In fact, even when $\mathrm{sim}(x, y) = x^\top y$, i.e., $p(z_i|x_{\setminus M})$ being a von-Mises–Fisher distribution, the denominator of p is still difficult to compute.

Inspired by NCE, van den Oord et al. (2018) and Baevski et al. (2020b) approximate the integral by summing over a set of negative samples S_i. The cross entropy becomes

$$\log \frac{\exp(\mathrm{sim}(\mathrm{enc}(x_{\setminus M})_i, z_i))}{\sum_{z'\in S_i}\exp(\mathrm{sim}(\mathrm{enc}(x_{\setminus M})_i, z'))}. \quad (16)$$

which is the objective used in wav2vec 2.0. Consequently, minimizing the contrastive loss is analogue to minimizing the cross entropy between the masked frames and unmasked frames after quantization.

Several design choices in wav2vec 2.0 are taken verbatim by HuBERT, but they have a few key differences. Both use a softmax to parameterize $p(z_i|x_{\setminus M})$, but wav2vec 2.0 assumes z_i to be a continuous vector, while HuBERT assumes z_i to be discrete. As a consequence, wav2vec 2.0 uses a softmax over the negative and positive samples, while HuBERT uses a softmax over the codewords in a codebook. The codebook in wav2vec 2.0 is trained together with the rest of the model. In other words, there is no reconstruction and we can simply parameterize $p(x_i|z_i)$ as a uniform distribution. In contrast, the codebook in HuBERT is trained with an offline k-means. Compared to our framework, we have the choice to optimize the codebook offline or jointly with the Transformer.

There is a additional diversity loss in wav2vec 2.0 that computes the entropy of codeword usage within a batch, to encourage the use of individual codewords. The diversity loss was not meant to be part of the main objective (Baevski et al., 2020b), but more of a regularizer that improves training. However, our instantiation on HuBERT naturally has an entropy term per frame that serves a similar purpose as the diversity loss.

6 Experiments

We have shown that our framework subsumes the HuBERT objective and the two are numerically identical. In the experiments, we will study how the extensions in Section 4 can lead to immediate improvements over HuBERT.

6.1 Experimental Settings

Unless otherwise stated, models are built with a Transformer encoder, a linear projection after the encoder, and a codebook (consisting of the centroids in k-means). We follow Chung et al. (2019), Jiang et al. (2019), Ling and Liu (2020),

Model	$p(z\|x_A)$	$q(z\|x_B)$	$p(x_B\|z)$	Codebook	$\mathbb{E}_{z\sim q(z\|x_B)}[\cdot]$
MASKED PREDICTION					
HuBERT Obj (eq 5)	softmax (eq 8)	point-mass (eq 6)	Gaussian (eq 7)	offline	single point
Masked-VPC (eq 5)	softmax (eq 8)	soft-min (eq 9)	Gaussian (eq 7)	joint opt.	Gumbel / marginal
Masked-NCE (eq 5)	softmax (eq 16)	point-mass (eq 13)	uniform	joint opt.	single point
FUTURE PREDICTION					
Future-VPC (eq 11)	softmax (eq 8)	soft-min (eq 9)	Gaussian (eq 7)	joint opt.	Gumbel / marginal
VQ-APC (eq 1)	softmax (eq 8)	n/a	Gaussian (eq 7)	joint opt.	n/a

Table 1: A summary of objectives that can instantiated from our framework. Each loss is charaterized by how the probability distributions are parameterized, how the code is optimized, and how the expectation is computed. When q is a point-mass, the expectation becomes an assignment to the point (single point). The codebook can be either optimized with k-means (offline) or jointly optimized (online) with the Transformer.

Misra et al. (2021), Chiu et al. (2022), and Lin et al. (2023), using Mel spectrograms as acoustic frames. We extract 40-dimensional log Mel features with a 25 ms window and a 10 ms hop. We concatenate every two contiguous frames, having 80-dimensional features with a 20 ms frame rate. For masked prediction, we use a masking span of 4 frames (80 ms) with every frame being the start of a span with a probability of 0.2; spans may overlap. For future prediction, we use a time shift (the κ in Section 4.3) of 2 frames (40 ms).

In the experiments, we consider a BASE setting similar to that described by Baevski et al. (2020b); Hsu et al. (2021), using 12-layer Transformers with 6 attention heads instead of 8. We set the codebook size to 100 for all models, following the cluster size in Hsu et al. (2021). We train models on the 360-hour subset of LibriSpeech (Panayotov et al., 2015) for 150 epochs. We detail hyperparameters in Appendix A.3.

6.2 Model Variants and Feature Summary

Before comparing different optimization strategies and downstream performance, we summarize model variants evaluated in our experiments in Table 1. Each model corresponds to a particular instantiation of our framework, characterized by a combination of choices such as prediction task, codebook optimization and hard (point-mass) or soft assignments (soft-min) of q. Specifically, we denote our variational predictive coding framework as Masked-VPC or Future-VPC, depends on the prediction task. The framework thus allows us to compare different features in isolation. Note that the original HuBERT model (Hsu et al., 2021) consists of two or more training iterations. Since we concerns training objective rather than the models, we use HuBERT Obj to indicate its training "objective" to avoid confusions. The first few sets experiments compares the different losses in the first iteration, and we leave the results of the second iteration to last subsection.

6.3 Comparing Optimization Methods

We first focus on comparing the first two variants (HuBERT Obj, Masked-VPC) in Table 1, which optimize the same loss. As pointed out in Section 4, softening the point mass, i.e., moving away from hard assignment in k-means to soft assignment, provides an opportunity to jointly optimize the cross entropy and the reconstruction. Not only could we jointly optimize both terms, we also have the option to choose between exact marginalization and sampling. When sampling, we only take a single sample and use Gumbel softmax to compute the gradient. There are a total of three options: the original hard assignment in HuBERT Obj, soft assignment with exact marginalization, and soft assignment with sampling. HuBERT uses k-means++ (Arthur and Vassilvitskii, 2007) as initialization, so we also compare random initialization and k-means++ initialization for the codebook.

Figure 2 (top) shows the variational objective (the negative ELBO in equation 5) of training BASE models on the 360-hour subset of Librispeech. The HuBERT objective starts off low because of the offline k-means, but both exact marginalization and sampling improve over the HuBERT objective after training. Similar to how HuBERT and BEST-RQ are sensitive to initialization (Chiu et al., 2022), marginalization ends up

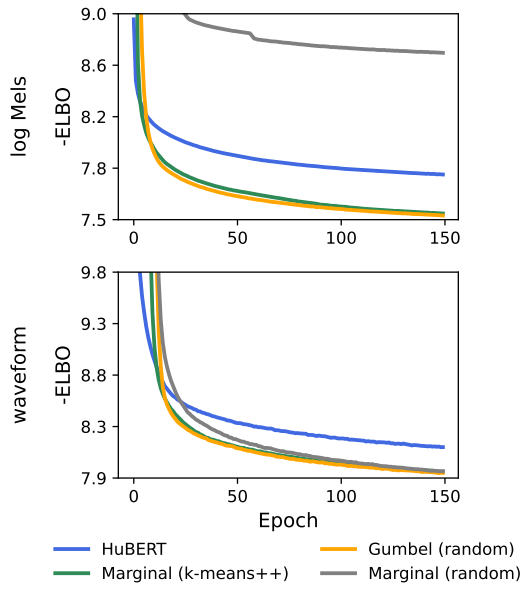

Figure 2: The training losses of different optimization approaches implemented by our own BASE (top) and in fairseq BASE (bottom). The final loss values can be found in Table 6.

at drastically different final values depending on whether k-means++ is used. Sampling (and taking the gradient with Gumbel softmax) turns out to be both efficient and fast converging.

To confirm the findings, we run the same experiments in fairseq[5] , in which CNN is used to aggregate frame-wise features. Figure 2 (bottom) shows the same trend, except that marginalization works out of the box with a randomly initialized codebook. The final losses are presented in Table 6. For the rest of the experiments, if not otherwise stated, we will only use Mel spectrograms as input. We will also use sampling (and Gumbel softmax) to optimize our objective due to its faster convergence.

6.4 Downstream Evaluation

Given that we can improve pre-training, the next question is whether the improvement transfer to downstream tasks. Besides, our framework allows for controlled experiments that only differ in loss functions while holding everything else, such as the model architecture and training hyperparameters, fixed. Based on the connections to prior work in Section 5, we will study two important design choices, i.e., comparing masked prediction to future prediction and comparing the contrastive loss in wav2vec 2.0 to the softmax in the HuBERT objective.

[5]The experiments mostly follow the default HuBERT configuration at `hubert_base_librispeech.yaml`, using waveform as input.

Table 2: Downstream probing results for different speech representations on phone classification, speaker verification, and f0 tracking.

	PER ↓		EER ↓	RMSE ↓
	dev93	eval92	voxceleb	eval92
log Mel	49.8	50.0	24.6	38.4
i-vector	-	-	15.7	-
HuBERT Obj	12.8	12.6	18.3	23.4
Masked-VPC	11.8	11.3	14.4	20.9

The utility of speech representations on downstream tasks are evaluated with probing (van den Oord et al., 2018; Chung et al., 2019; Yang et al., 2022, 2021), where the pre-trained models are frozen and small classifiers are trained to complete tasks given the hidden vectors of one of the layers.

We conduct four probing tasks, phone classification, speaker verification, fundamental frequency tracking, and automatic speech recognition. All settings follow Yang et al. (2022) unless otherwise noted. For phone classification, we train a linear classifier to predict phone labels obtained from forced alignments on the Wall Street Journal (Paul and Baker, 1992) (WSJ). We report phone error rates (PERs) on `dev93` and `eval92`, using 10% of the training set `si284` for development. For f0 tracking, we train a linear regression model to predict the f0 extracted with PYIN (Mauch and Dixon, 2014) on WSJ. We report the root-mean-square error (RMSE) in Hz on `eval92`. For speaker verification, we train a two-layer classifier to predict speaker identities on VoxCeleb1 (Nagrani et al., 2020) and use the intermediate layer as the speaker embedding for speaker verification. We report equal error rates (EERs) on the test set. The reported numbers are with the best layer for each task and for each model. More details are in Appendix A.4.

Better pre-training leads to better downstream performance We have shown that using soft assignment and sampling gives us the best pre-training loss. To answer whether the improved pre-training transfer to downstream tasks, we compare HuBERT Obj with the soft assignment variant optimized with sampling and Gumbel softmax (termed Masked-VPC). Results on three tasks are shown in Table 2. We see that improving pre-

Table 3: Downstream probing results comparing future prediction (Future-VPC) and masked prediction (Masked-VPC) on phone classification, speaker verification, and f0 tracking.

	PER ↓		EER ↓	RMSE ↓
	dev93	eval92	voxceleb	eval92
Masked-VPC	11.8	11.3	14.4	20.9
Future-VPC	16.0	15.5	13.6	20.5

Table 4: Downstream probing results comparing NCE (Masked-NCE), i.e., using a contrastive loss with the HuBERT approaches (HuBERT Obj and Masked-VPC) on phone classification, speaker verification, and f0 tracking.

	PER ↓		EER ↓	RMSE ↓
	dev93	eval92	voxceleb	eval92
Masked-NCE	12.2	12.4	20.2	24.4
HuBERT Obj	12.8	12.6	18.3	23.4
Masked-VPC	11.8	11.3	14.4	20.9

training indeed improves downstream tasks across the board. ASR results are deferred to later sections, but the conclusion stands.

Future prediction can be as strong as masked prediction on some downstream tasks As shown in Section 4.3, under our framework, it is possible (and relatively simple) to switch from masked prediction to future prediction while holding everything else fixed. Future prediction is more relevant than masked prediction in certain scenarios, such as streaming or pre-training decoder-only Transformers. Misra et al. (2021), for example, show that future prediction can be as strong as masked prediction for streaming ASR. Results comparing future prediction (Future-VPC) and masked prediction (Masked-VPC) are shown in Table 3. We find that, not surprisingly, future prediction is worse at phone classification compared to masked prediction. However, future prediction is better than masked prediction on speaker verification and fundamental frequency tracking.

NCE is on par with HuBERT Obj To compare to the contrastive loss, we parameterize $q(z_i|x_i)$ and $p(z_i|x_{\setminus M})$ as detailed in Section 5.4. We characterize the model in Table 1 and denote it as Masked-NCE. The entropy term is a constant because of the Dirac delta. We follow Baevski et al. (2020b) and take the negative samples for NCE from frames within the same batch. The codebook is updated together with the contrastive loss with Gumbel softmax as in Baevski et al. (2020b). Results are shown in Table 4. NCE alone performs surprisingly well and closely matches HuBERT's results, while our approach is still superior.

Table 5: Probing results with lexicon-free seq2seq models on WSJ. No language models are used.

	CER ↓		WER ↓	
	dev93	eval92	dev93	eval92
BASELINE				
wav2letter++ †	6.3	4.1	19.5	13.9
18L Transformer‡	-	-	22.2	17.9
PROBING WITH OUR SEQ2SEQ				
log Mel	6.8	5.1	18.2	14.7
VQ-APC	5.8	5.2	16.8	14.9
Future-VPC	5.4	4.0	15.5	12.4
Masked-NCE	5.0	4.9	14.5	14.4
HuBERT Obj	5.2	5.0	15.2	14.5
Masked-VPC	4.4	3.6	13.6	11.4

† Numbers taken from Baevski et al. (2020a).
‡ Numbers taken from Higuchi et al. (2020).

6.5 Automatic Speech Recognition

Automatic speech recognition (ASR) has been the main driving force behind speech representation learning. In this section, we evaluate the quality of speech representations with two ASR settings, a probing setting with sequence-to-sequence (seq2seq) models and a fine-tuning setting with connectionist temporal classification (CTC). We report CERs and WERs averaged over three runs.

Probing with seq2seq In the probing setting, we follow Yang et al. (2022), extracting the layer that achieves the best phone classification, and train a seq2seq model with character outputs on WSJ.[6] Following the spirit of probing, we make sure that the improvement is solely from the learned representations and that the phonetic information is accessible, by not using a language model or a lexicon, and by using a lightweight model. Beam search is used with a beam size of 5.

Table 5 reports the ASR performance in terms of character error rates (CERs) and word error rates (WERs). Despite being lightweight, our baseline model on log Mel features is by no means weak, and is on par with wav2letter++ in Pratap et al. (2019) and better than a 18-layer Trans-

[6] More training details are in Appendix A.4.

Table 6: Results of ASR fine-tuned on WSJ using CTC for different optimization approaches during pre-training. No language models are used.

	−ELBO	dev93	eval92
FINE-TUNING (OURS)			
HuBERT Obj	7.79	15.5	12.5
Marginal (k-means++)	7.50	14.3	11.0
Marginal (random)	8.70	15.0	11.7
Gumbel (random)	7.48	14.1	11.7
FINE-TUNING (FAIRSEQ)			
HuBERT Obj	8.14	14.9	11.8
Marginal (k-means++)	7.91	14.5	11.1
Marginal (random)	7.91	14.2	10.9
Gumbel (random)	7.89	14.3	11.0

former baseline in Higuchi et al. (2020). We then compare representations learned from the approaches characterized in Table 1, including two future prediction approaches (VQ-APC, Future-VPC) and three masked prediction approaches (Masked-NCE, HuBERT Obj, Masked-VPC). We first find that representations learned from future prediction are not far behind those from masked prediction. The second is that the improvement from HuBERT Obj to Masked-VPC (in Section 6.3) transfers well to ASR as well.

Fine-tuning with CTC To further confirm that improvement in pre-training transfers to ASR, we fine-tune the pre-trained models on WSJ (Paul and Baker, 1992) using CTC (Graves et al., 2006) with characters output. We fine-tune our BASE for 100 epochs and fairseq BASE for 200 epochs on `si284`. Similar to Lee and Watanabe (2021); Higuchi et al. (2020), we average models from last 10 epochs after training to obtain final model parameters. Table 6 summarizes the pre-training losses (-ELBO) reported in Figure 2 and the corresponding WERs on `dev93` and `eval92`. The inference is done with greedy decoding. The improvement in objective indeed transfers well to ASR in both settings.

To see whether the improvement is still present when decoding with a language model, Table 7 compared the results after using a 4-gram word language model (Heafield et al., 2013). A beam size of 2000 is used for decoding with the 4-gram language model. The improvement transfer well even with the use of a language model.

Lastly, to see the effect on small datasets, we follow Baevski et al. (2020b) and evaluate phonetic recognition on TIMIT. We observe the same trend in Table 8 that our approach consistently performs better than HuBERT Obj.

Table 7: Comparing ASR fine-tuned on WSJ using CTC with and without a 4-gram language model.

	w/o LM		w/ LM	
	dev93	eval92	dev93	eval92
BASELINE				
12L Transformer[‡]	20.1	16.5	-	-
wav2letter++ [†]	19.5	8.57		
FINE-TUNING				
Masked-NCE	14.7	11.1	7.8	4.4
HuBERT Obj	15.5	12.5	7.3	4.6
Masked-VPC	14.1	10.3	6.8	4.6

[‡] Numbers taken from Lee and Watanabe (2021).
[†] Numbers taken from Baevski et al. (2020a).

Table 8: Phone recognition fine-tuned with CTC on TIMIT.

	dev PER	test PER ↓
BASELINE		
12L Transformer	22.1	24.1
FINE-TUNING		
Masked-NCE	11.2	12.9
HuBERT Obj	10.2	11.3
Masked-VPC	9.5	11.0

6.6 Second-Iteration Training

As detailed in Section 5.3, our framework can be extended to the second iteration as in HuBERT (Hsu et al., 2021). Following Hsu et al. (2021), the Transformer encoder from the previous iteration is held fixed to derive targets from its intermediate layer for the next iteration training. We adopt the 6-th layer of Transformers of HuBERT and Masked-VPC since the layer that provides the best downstream ASR performance in Table 5. A new Transformer encoder is reinitialized and optimized using HuBERT objective or Masked-VPC.

For second-iteration training, we find that codebook initialization becomes critical, whereas it is not sensitive during the first iteration. With random initialization, the model often suffers from codebook collapse, i.e., all frames are assigned to a single class. The issue does not occur in the first iteration, indicating that optimizing vector quantization layer becomes challenging with high-dimensional representations. To address this issue, we initialize the codebook in Masked-VPC with

Table 9: Second-iteration results of ASR fine-tuned on WSJ using CTC with or without an ngram LM.

	w/o LM		w/ LM	
	dev93	eval92	dev93	eval92
FIRST ITERATION				
HuBERT Obj	14.9	11.8	7.9	6.0
Masked-VPC	14.5	11.1	7.8	5.3
SECOND ITERATION				
HuBERT Obj	11.1	8.2	6.2	4.2
Masked-VPC	10.4	7.9	5.9	3.8

k-means++ initialization over 6-th layer representations. For consistency, the same method used in the first iteration over logMels is applied in the second iteration.

We run the second iteration using fairseq, based on the first iteration presented in Figure 2 (bottom). Since the distribution of q is based on the Euclidean distance between a frame and the codevectors, once the dimensional increases, the distance scales up. After softmax, the distribution becomes nearly point-mass. We use a temperature τ of 10 to avoid q being one-hot in equation 12. We train second-iteration HuBERT and Masked-VPC for 160 epochs, and evaluate them with CTC finetuning as in the previous section.

As shown in Table 9, the second iteration improves both approaches by a large margin. In particular, we observe 4.1% reduction of WER on `dev93` with our approach. Moreover, the improvements in objective transfers to the second iteration, in which Masked-VPC obtains 0.7% and 0.3% lower WERs than HuBERT on each set. Similar trend is observed with an ngram language model for CTC decoding.

7 Conclusion

In this work, we provide an underlying principle for the HuBERT objective—a variational view of predictive coding. We show the utility of this framework by identifying opportunities to improve the HuBERT objective, i.e., its parameterization and optimization. Evaluating across several downstream tasks using probing and fine-tuning, we empirically show that the improved pre-training learns better speech representations. We further show that the predictive coding framework is general and has many connections to existing self-supervised objectives. Predictive coding has fruitful and mature theoretical underpinnings, while the theory of self-supervised learning is still in its infancy. We hope the making the connections among self-supervised learning and predictive coding clear as is done in this work helps advance the understanding of both.

References

David Arthur and Sergei Vassilvitskii. 2007. k-means++: the advantages of careful seeding. In *SODA*.

Bishnu S Atal and Suzanne L Hanauer. 1971. Speech analysis and synthesis by linear prediction of the speech wave. *The journal of the acoustical society of America*.

Hagai Attias. 1999. A variational Baysian framework for graphical models. *NeurIPS*.

Alexei Baevski, Steffen Schneider, and Michael Auli. 2020a. vq-wav2vec: Self-supervised learning of discrete speech representations. In *ICLR*.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020b. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *NeurIPS*.

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. 2023. Audiolm: a language modeling approach to audio generation. *IEEE/ACM transactions on audio, speech, and language processing*.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *ECCV*.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, et al. 2022. WavLM: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*.

Chung-Cheng Chiu, James Qin, Yu Zhang, Jiahui Yu, and Yonghui Wu. 2022. Self-supervised

learning with random-projection quantizer for speech recognition. In *ICML*.

Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James R. Glass. 2019. An unsupervised autoregressive model for speech representation learning. In *Interspeech*.

Yu-An Chung, Hao Tang, and James R. Glass. 2020. Vector-quantized autoregressive predictive coding. In *Interspeech*.

Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. 2021. W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In *ASRU*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. ACL.

Peter Elias. 1955. Predictive coding–i. *IRE transactions on information theory*.

Zhiyun Fan, Meng Li, Shiyu Zhou, and Bo Xu. 2020. Exploring wav2vec 2.0 on speaker verification and language identification. In *Interspeech*.

Harriet Feldman and Karl J Friston. 2010. Attention, uncertainty, and free-energy. *Frontiers in human neuroscience*.

Karl Friston and Stefan Kiebel. 2009. Predictive coding under the free-energy principle. *Philosophical transactions of the Royal Society B: Biological sciences*.

Jonas Geiping and Tom Goldstein. 2023. Cramming: Training a language model on a single GPU in one day. In *ICML*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *ACL*.

Yosuke Higuchi, Shinji Watanabe, Nanxin Chen, Tetsuji Ogawa, and Tetsunori Kobayashi. 2020. Mask CTC: Non-autoregressive end-to-end ASR with CTC and mask predict. In *Interspeech*.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. HuBERT: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE Transactions on Audio, Speech, and Language Processing*.

Yanping Huang and Rajesh PN Rao. 2011. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with Gumbel-softmax. In *ICLR*.

Dongwei Jiang, Xiaoning Lei, Wubo Li, Ne Luo, Yuxuan Hu, Wei Zou, and Xiangang Li. 2019. Improving transformer-based speech recognition using unsupervised pre-training. *arXiv preprint arXiv:1910.09932*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR*.

Brian Kulis and Michael I. Jordan. 2011. Revisiting k-means: New algorithms via bayesian nonparametrics. In *ICML*.

Kushal Lakhotia, Eugene Kharitonov, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Benjamin Bolte, Tu-Anh Nguyen, Jade Copet, Alexei Baevski, Abdelrahman Mohamed, et al. 2021. On generative spoken language modeling from raw audio. *Transactions of the Association for Computational Linguistics*.

Jaesong Lee and Shinji Watanabe. 2021. Intermediate loss regularization for CTC-based speech recognition. In *ICASSP*.

Tzu-Quan Lin, Hung-yi Lee, and Hao Tang. 2023. MelHubert: A simplified Hubert on Mel spectrograms. In *ASRU*.

Shaoshi Ling and Yuzong Liu. 2020. Decoar 2.0: Deep contextualized acoustic representations with vector quantization. *arXiv preprint arXiv:2012.06659*.

Shaoshi Ling, Yuzong Liu, Julian Salazar, and Katrin Kirchhoff. 2020. Deep contextualized acoustic representations for semi-supervised speech recognition. In *ICASSP*.

Alexander H Liu, Heng-Jui Chang, Michael Auli, Wei-Ning Hsu, and Jim Glass. 2024. Dinosr: Self-distillation and online clustering for self-supervised speech representation learning. In *NeurIPS*.

John Makhoul. 1975. Linear prediction: A tutorial review. *Proceedings of the IEEE*.

Matthias Mauch and Simon Dixon. 2014. PYIN: A fundamental frequency estimator using probabilistic threshold distributions. IEEE.

David McAllester. 2018. Information theoretic co-training. *arXiv preprint arXiv:1802.07572*.

Ananya Misra, Dongseong Hwang, Zhouyuan Huo, Shefali Garg, Nikhil Siddhartha, Arun Narayanan, and Khe Chai Sim. 2021. A comparison of supervised and unsupervised pre-training of end-to-end models. In *Interspeech*.

Arsha Nagrani, Joon Son Chung, Weidi Xie, and Andrew Zisserman. 2020. Voxceleb: Large-scale speaker verification in the wild. *Computer Speech & Language*.

Radford M Neal and Geoffrey E Hinton. 1998. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*. Springer.

Benjamin van Niekerk, Leanne Nortje, and Herman Kamper. 2020. Vector-quantized neural networks for acoustic unit discovery in the ZeroSpeech 2020 challenge. In *Interspeech*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. LibriSpeech: an asr corpus based on public domain audio books. In *ICASSP*.

Douglas B Paul and Janet Baker. 1992. The design for the Wall Street Journal-based CSR corpus. In *Speech and Natural Language Workshop*.

Vineel Pratap, Awni Hannun, Qiantong Xu, Jeff Cai, Jacob Kahn, Gabriel Synnaeve, Vitaliy Liptchinsky, and Ronan Collobert. 2019. wav2letter++: A fast open-source speech recognition system. In *ICASSP*.

Rajesh PN Rao and Dana H Ballard. 1999. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*.

S Saito, F Itakura, et al. 1967. Theoretical consideration of the statistical optimum recognition of the spectral density of speech. *J. Acoust. Soc. Japan*.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. 2019. wav2vec: Unsupervised pre-training for speech recognition. In *Interspeech*.

Jiatong Shi, Hirofumi Inaguma, Xutai Ma, Ilia Kulikov, and Anna Sun. 2023. Multi-resolution HuBERT: Multi-resolution speech self-supervised learning with masked unit prediction. In *ICLR*.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL*.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *NeurIPS*.

Michael W Spratling. 2017. A review of predictive coding algorithms. *Brain and cognition*.

Mandyam Veerambudi Srinivasan, Simon Barry Laughlin, and Andreas Dubs. 1982. Predictive coding: a fresh view of inhibition in the retina. *Proceedings of the Royal Society of London. Series B. Biological Sciences*.

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. 2023. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *ICML*.

Hemant Yadav, Sunayana Sitaram, and Rajiv Ratn Shah. 2024. MS-Hubert: Mitigating pre-training and inference mismatch in masked language modelling methods for learning speech representations. In *Interspeech*.

Gene-Ping Yang, Sung-Lin Yeh, Yu-An Chung, James Glass, and Hao Tang. 2022. Autoregressive predictive coding: A comprehensive study. *IEEE Journal of Selected Topics in Signal Processing*.

Shu-wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y. Lin, Andy T. Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, Tzu-Hsien Huang, Wei-Cheng Tseng, Ko tik Lee, Da-Rong Liu, Zili Huang, Shuyan Dong, Shang-Wen Li, Shinji Watanabe, Abdelrahman Mohamed, and Hung yi Lee. 2021. SUPERB: Speech Processing Universal PERformance Benchmark. In *Interspeech*.

Sung-Lin Yeh and Hao Tang. 2022. Autoregressive co-training for learning discrete speech representations. In *Interspeech*.

Ruixiong Zhang, Haiwei Wu, Wubo Li, Dongwei Jiang, Wei Zou, and Xiangang Li. 2021. Transformer based unsupervised pre-training for acoustic representation learning. In *ICASSP*.

Yu Zhang, Daniel S Park, Wei Han, James Qin, Anmol Gulati, Joel Shor, Aren Jansen, Yuanzhong Xu, Yanping Huang, Shibo Wang, et al. 2022. Bigssl: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition. *IEEE Journal of Selected Topics in Signal Processing*.

A Appendix

A.1 Negative Free Energy of Predictive Coding

The proof of equation 2 mirrors those in variational autoencoders (Kingma and Welling, 2014). We start with the KL divergence between the posterior distribution $p(z|x_A, x_B)$ and the auxiliary distribution $q(z|x_B)$,

$$\begin{aligned} &\mathrm{KL}\big[q(z|x_B)\|p(z|x_A, x_B)\big] \qquad (17)\\ &= \mathbb{E}_{z\sim q}\left[\log \frac{q(z|x_B)}{p(x_B|z)p(z|x_A)}\right] + \log p(x_B|x_A) \end{aligned}$$

where we assume $x_B \perp\!\!\!\perp x_A \mid z$ or $p(x_B|x_A, z) = p(x_B|z)$. Because the KL divergence is always positive, we obtain

$$\begin{aligned} &-\log p(x_B|x_A) \qquad (18)\\ &\leq \mathrm{KL}\big[q(z|x_B)\|p(z|x_A)\big] - \mathbb{E}_{z\sim q}[\log p(x_B|z)] \end{aligned}$$

A.2 Future Prediction

The proof of equation 11 involves unrolling $p(x_B, z|x_A)$ or $p(x_{\geq t}, z_{\geq t}|x_{<t})$, where $x_B = x_{\geq t}$ and $x_A = x_{<t}$. Based on the definition of conditional probability, we have

$$p(x_{\geq t}, z_{\geq t}|x_{<t}) = \prod_{i=t}^{T} p(x_i, z_i|x_{<i}, z_{t:i-1}) \quad (19)$$

$$= \prod_{i=t}^{T} p(x_i|x_{<i}, z_{t:i})p(z_i|x_{<i}, z_{t:i-1}) \quad (20)$$

$$= \prod_{i=t}^{T} p(x_i|z_i)p(z_i|x_{<i}), \quad (21)$$

where the last line makes two reasonable assumptions, $x_i \perp\!\!\!\perp z_{t:i-1} \mid z_i$ and $z_i \perp\!\!\!\perp z_{t:i-1} \mid x_{<i}$. The variable z_i should represent x_i well without relying on previous latent variables $z_{t:i-1}$. Given the history $x_{<i}$, computing the representation z_i should not depend on the previous latent variables $z_{t:i-1}$.

A.3 Pre-Training Recipes

We set a maximum length of 1400 frames per utterance, corresponding to about 28 seconds. The learning rate is fixed to 10^{-4} under the Adam optimizer, no learning rate schedule is applied. We pre-train BASE models on a single A40 with a batch size of 16 for 150 epochs. We set model dimension to 768, with inner dimension of feedforward netowrks being 3072. A dropout probability of 0.1 is applied.

The temperature is set to 1 without annealing for variational training. In wav2vec 2.0, the similarity value is re-scaled by dividing it with 0.1, the temperature for Gumbel-Softmax (Jang et al., 2017) is annealed from 2 to a minimum of 0.5 by a decay rate of 0.999995, i.e., the τ in Equation 9. We use 100 negative samples following Baevski et al. (2020b).

There are a few architectural differences between our setting and Baevski et al. (2020b). We employ a single codebook in our wav2vec 2.0 for quantization. Rather than Post-LN Transformers, we use Pre-LN Transformers for pre-training and remove the warm-up stage (Geiping and Goldstein, 2023; Xiong et al., 2020). We use sinusoidal positional embeddings rather than relative position embeddings used in Baevski et al. (2020b).

A.4 Downstream Evaluation

We provide additional details on the experimental setups of downstream tasks. The layers chosen for each task is noted in Table 10.

Phone Classification (PC) We freeze the pre-trained model and only train a linear layer with a learning rate of 10^{-3} for 10 epochs.

Speaker Verification (SV) We average frame representations to obtain utterance-level representations, and simply employ two linear layers to predict 1251 speakers following (Fan et al., 2020). The linear classifier is optimized with a learning rate of 10^{-3} for 10 epochs. After training, we take the output of the first linear layer as speaker vectors for speaker verification. We set the dimension of speaker vectors to 512.

F0 Tracking (F0) We train a linear regression layer with a learning rate of 10^{-3} for 10 epochs. We set the minimum and maximum frequency to 50 Hz and 600 Hz respectively.

Automatic Speech Recognition (ASR) We use a lightweight sequence-to-sequence encoder for downstream ASR. The encoder contains two convolutional layers with $(32, 32)$ channels and $(2, 1)$ strides, and a 4-layer, 256-dimensional bidirectional GRU. The decoder is a unidirectional 256-dimensional GRU.

We adopt a fixed scheduled sampling probability of 0.4 during training. We use Adam with learning rates of 10^{-4} for all s2s models. We employ a dropout rate of 0.2, and a label smoothing

	VQ-APC	Future-VPC	Masked-NCE	HuBERT	Masked-VPC
PC/ASR	8	8	9	8	8
f0	3	3	2	2	3
SV	4	3	3	3	3

Table 10: Layers selected for downstream experiments for each model variant.

rate of 0.1 for regularization. We train seq2seq models for 100 epochs, and lower the learning rate with a factor of 0.1 for another 20 epochs.