A Appendix

A.1 Broader Impact and Limitations

This work, alone with the related prior works in self-supervised speech representation learning discussed, focused on English speech, thereby implying the potential risks associated with neglecting other languages, especially the low-resource languages. Nevertheless, we provide a preliminary result on other languages in §A.5 to mitigate the problem as much as possible.

A.2 Discussion on online clustering

Codebook initialization. The initialization of codebook $\mathbf{E}^k \in \mathbb{R}^{V \times D}$ can be critical in vector quantization methods [11]. We tried different initializations including

- $\mathbf{E}^k \sim \mathcal{N}(0, \mu)$ where $\mu \in \{1, \frac{1}{\sqrt{D}}\}$,
- $\mathbf{E}^k \sim \mathcal{N}(0, 1)$ followed by L2 normalization $\frac{\mathbf{e}_v^k}{\|\mathbf{e}_v^k\|_2}$.

In practice, we found different initialization leads to different codebook perplexity at the beginning of training. Nevertheless, the methods all lead to similar codebook perplexity at the end of training and downstream performance. This also demonstrated the stability of gradient-free online VQ in oppose to standard VQ.

Input for quantization. We found normalizing the teacher model representation $\tilde{\mathbf{z}}_t^k$ is necessary for stable clustering. We briefly summarized the results using different normalization methods:

- Instance normalization (IN; default): this can be interpreted as a parameter-free utterance-wise normalization for each channel, we found it stable.
- Batch normalization (BN): this can be viewed as a dataset-wise version of IN which yields a similar result but introduces additional parameters tracking the running stats.
- L2 normalization: a frame-wise normalization along the feature dimension, results in a more unstable codebook perplexity and code collapse occasionally.

In addition, we also tried combining targets from different layers (i.e, $\sum_k \tilde{\mathbf{z}}_t^k$ with a single codebook across all layers) before clustering following Baevski et al. [9]. This model performed significantly worse in the downstream fine-tuning task.

Dealing with inactive codewords. Note that the codebook update policy

$$\begin{aligned} \mathbf{s}_v^k &\longleftarrow \tau\, \mathbf{s}_v^k + (1-\tau) \sum \tilde{\mathbf{Z}}_v^k, \\ n_v^k &\longleftarrow \tau\, n_v^k + (1-\tau) \left|\tilde{\mathbf{Z}}_v^k\right|, \\ \mathbf{e}_v^k &\longleftarrow \frac{\mathbf{s}_v^k}{n_v^k}, \end{aligned} \tag{3}$$

updates each codeword $\mathbf{e}_v^k$ regardless of whether the codeword activate (i.e., having at least one neighbor $\left|\tilde{\mathbf{Z}}_v^k\right| \geq 1$) or not. In the extreme case where a codeword remains inactive for a long period, we have $\mathbf{s}_v^k \to \vec{0}$ and $n_v^k \to 0$ which results in numerical instability or code collapse. In practice, we found freezing the inactivate codewords with

$$\tau_v^k = \begin{cases} \tau, & \text{if } \left|\tilde{\mathbf{Z}}_v^k\right| \geq 1 \\ 1, & \text{otherwise} \end{cases}, \tag{5}$$

leads to a slightly better codebook perplexity but the improvement diminishes as the batch size increases.

Simplifying codeword update policy. A simplified version of online clustering described in Eq.3 is to only track the averaged embedding without the size

$$\mathbf{e}_v^k \longleftarrow \tau\, \mathbf{e}_v^k + (1-\tau)\frac{\sum \tilde{\mathbf{Z}}_v^k}{\left|\tilde{\mathbf{Z}}_v^k\right|}. \tag{6}$$

The simplified version enforces equal momentum for each step regardless of the size of the neighbor set $\left|\tilde{\mathbf{Z}}_v\right|$. In practice, we found using Eq. 3 more stable and results in slightly better performance with a negligible cost.

A.3 Pseudo-code for DinoSR training

Algorithm 1 PyTorch pseudocode for DinoSR

```
# teacher, student: student and teacher networks
# phi[k]: DxV cluster prediction matrix for k-th layer codebook
# codebook[k]: VxD codebook matrix for k-th layer
# code_sum[k]: VxD unnormalized codebook matrix for k-th layer
# code_cnt[k]: Vx1 codeword counter for k-th layer
# lbd, tau: decay rates of teacher network, codebook
teacher.weight = student.weight

for x in dataset: # mini audio batch BxT

    # Eq.1: teacher EMA
    teacher.weight = lbd * teacher.weight + \
                     (1-lbd) * student.weight

    z = student(mask(x))      # BxTxD, last layer only
    z = z[masked_position]    # MxD
    with torch.no_grad():     # gradient-free syntax
        z_tilde = teacher(x) # KxBxTxD, all K layers
        z_tilde = z_tilde[:,masked_position] # KxMxD

    loss = 0
    for k in range(K-N,K):

        with torch.no_grad():
            # Eq.2: online clustering
            d = -framewiseL2(z_tilde[k],codebook[k])
            target_cls = hardmax(d, dim=-1) # MxV

            # Eq.3: codebook learning
            code_sum[k] = tau * code_sum[k] + \
                          (1-tau) * matmul(target_cls.T,z_tilde)
            code_cnt[k] = tau * code_cnt[k] + \
                          (1-tau) * target_cls.sum(dim=0)
            codebook[k] = code_sum[k] / code_cnt[k]

        # Eq.4: cluster prediction
        p_v = phi[k](z) # MxV
        loss += cross_entropy(p_v, target_cls)

    loss.backward()
    student.step()
```

A.4 Additional results and analysis

Visualizing phone-code correlation. To further demonstrate the difference between DinoSR and prior works with online codebook learning, we visualized the conditional probability $P(\text{phone}|\text{code})$ computed using Co-training APC [2][29] and Vector-Quantized Autoregressive Predictive Coding (VQ-APC[3];[12]) following the exact same setup used in Figure 4. Clearly, DinoSR is able to capture the long-tail distribution better, and the codewords tend to be more concentrated to the most correlated phone when compared against the prior works.

[2]Model checkpoint from https://github.com/30stomercury/autoregressive-co-training

[3]Code from https://github.com/iamyuanchung/VQ-APC

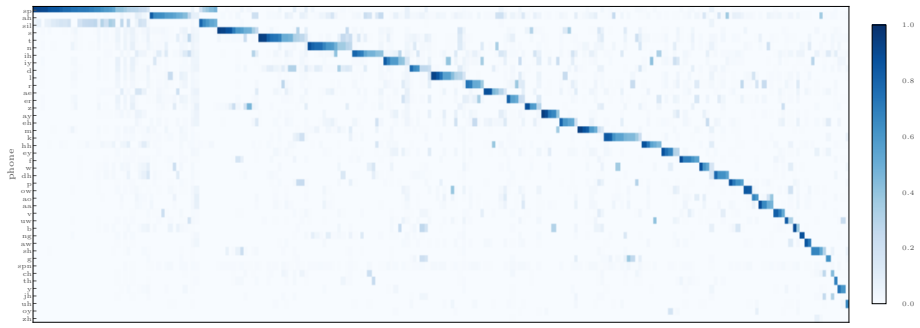

Figure 6: $P(\text{phone}|\text{code})$ from DinoSR with 217 codewords activated out of 256.

Figure 7: $P(\text{phone}|\text{code})$ from Co-training APC [29] with 164 codewords activated out of 256.

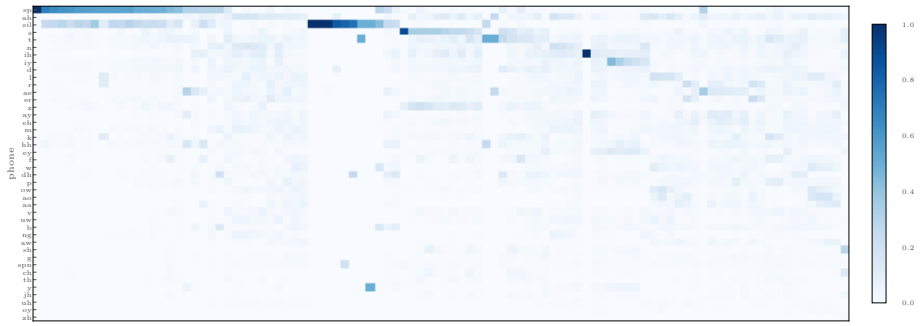

Figure 8: $P(\text{phone}|\text{code})$ from VQ-APC [12] with 98 codewords activated out of 512.

Figure 9 and figure 10 provided another view of the codeword distribution over the phone set. Each codeword is assigned to a single phone based on the most correlated phone (i.e., $\text{argmax}_{\text{phone}} P(\text{phone}|\text{code})$). We derive the learned phone distribution by accumulating the occurrence of all codewords and compare to the ground truth. Results show the distribution of codewords from DinoSR is very close to the ground truth, while other methods failed to capture the underlying distribution by over-assigning codewords to the more frequent phones and dropping the less frequent phones.

Finding the best codebook. By examining phone-normalized mutual information, code perplexity, and ABX score in acoustic unit discovery in each layer, we can see that the 5th layer of the teacher model consistently performed the best. However, this is only considering phones as the ideal discrete unit. A more throughout study on the content of each layer is left as future work.

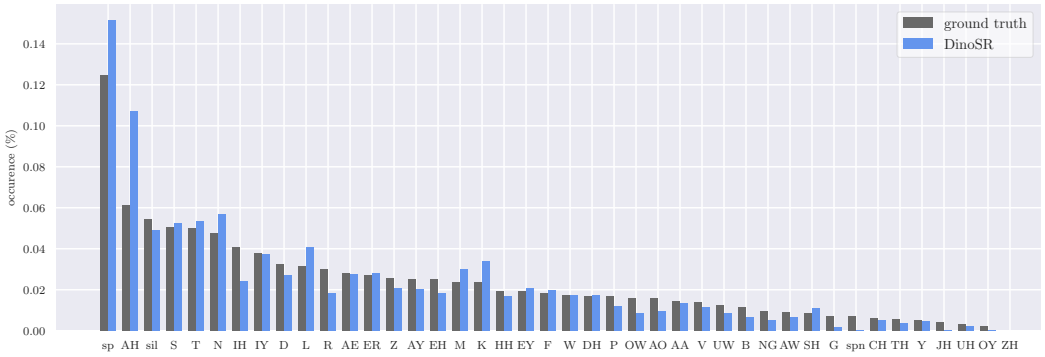

Figure 9: Histogram of phones and codewords.

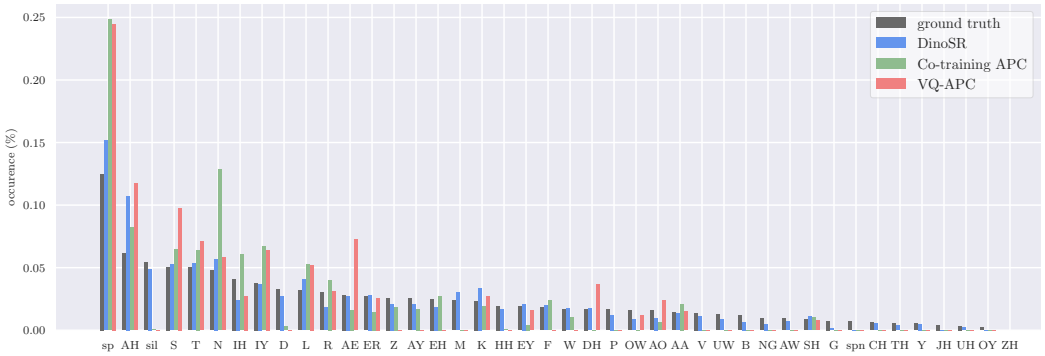

Figure 10: Histogram of phones and codewords.

Figure 11: Layer-wise phone-normalized mutual information, code perplexity, and ABX score in acoustic unit discovery.

A.5 Preliminary Result on Multi-lingual Speech

We conducted a preliminary experiment under limited computing budget to showcase that our method can be generalized to other languages. We followed the setting in multi-lingual speech representation learning [42, 43] to pre-train DinoSR on 10 different languages (Bengali, Cantonese, Georgian, Haitian, Kurmanji, Pashto, Tamil, Turkish, Tokpisin, Vietnamese) on the BABEL dataset [44] and fine-tune on 4 different unseen languages (Assamese, Tagalog, Swahili, Lao). In this setup we trained our model for 200k steps on 4 GPUs with a total batch size of 16 minutes. We report Character Error Rate (CER) on the fine-tuning languages in Table 6.

Table 6: Character Error Rate (WER) on BABEL dataset.

Model	Pre-training steps	Batch size (minutes)	Fine-tuning Language			
			Assamese	Tagalog	Swahili	Lao
Seq-to-seq ASR without pre-training [42]	-	-	41.3	37.9	29.1	38.7
XLSR-10 [43]	250k	96	29.4	21.9	16.6	23.3
DinoSR	200k	16	27.2	19.4	14.6	22.8

Inspired by the International Phonetic Alphabet [45] which defined a universal set of acoustic units across different languages, we take a look into how DinoSR acoustic units derived from the 10 pre-training languages are distributed. Interestingly, we found the acoustic units are more likely to be shared across different languages as shown in Table 7.

Table 7: Codeword distribution over different languages from BABEL dataset. Each cell corresponded to the proportion of codewords activated in n languages, e.g., 86.5% of the codewords emerged in all 10 languages for the 5th layer.

Layer	n languages									
	1	2	3	4	5	6	7	8	9	10
5th	0.0%	0.0%	4.5%	1.7%	1.1%	0.6%	1.1%	3.4%	1.1%	86.5%
6th	0.0%	0.0%	2.9%	1.7%	1.7%	1.7%	1.7%	1.7%	2.9%	85.5%
7th	0.7%	0.7%	4.0%	1.3%	1.3%	0.7%	3.4%	4.0%	7.4%	76.5%
8th	0.0%	0.0%	1.9%	2.6%	1.3%	0.6%	1.3%	2.6%	1.9%	87.8%
9th	0.0%	0.0%	2.4%	1.8%	1.8%	1.2%	0.6%	1.2%	3.6%	87.4%
10th	0.0%	0.0%	1.8%	1.2%	1.2%	0.0%	0.6%	1.2%	1.8%	92.1%
11th	0.0%	0.0%	1.9%	0.6%	0.0%	0.6%	0.0%	2.5%	1.9%	92.5%
12th	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.6%	1.2%	0.6%	97.6%

In conclusion, preliminary results on BABEL show that DinoSR can be applied to languages other than English. Moreover, learning acoustic units from different languages is possible even with a shared vocabulary.