OPTIMIZING BYTE-LEVEL REPRESENTATION FOR END-TO-END ASR

Roger Hsiao, Liuhui Deng, Erik McDermott, Ruchir Travadi, Xiaodan Zhuang

Apple

ABSTRACT

We propose a novel approach to optimizing a byte-level representation for end-to-end automatic speech recognition (ASR). Byte-level representation is often used by large scale multilingual ASR systems when the character set of the supported languages is large. The compactness and universality of byte-level representation allow the ASR models to use smaller output vocabularies and therefore, provide more flexibility. UTF-8 is a commonly used byte-level representation for multilingual ASR, but it is not designed to optimize machine learning tasks directly. By using auto-encoder and vector quantization, we show that we can optimize a byte-level representation for ASR and achieve better accuracy. Our proposed framework can incorporate information from different modalities, and provides an error correction mechanism. In an English/Mandarin dictation task, we show that a bilingual ASR model built with this approach can outperform UTF-8 representation by 5% relative in error rate.

Index Terms— byte-level representation, speech recognition

1. INTRODUCTION

End-to-end (E2E) neural networks are flexible and accurate models for multilingual automatic speech recognition (ASR). The output of such a multilingual model is often unions of characters or subwords of the supported languages. However, as the number of languages increases, the size of the output layer increases, which can negatively affect compute, memory usage and asset size. This problem is more prominent when the system supports languages that have large character sets, such as Chinese, Japanese and Korean (CJK). To tackle this problem, previous work proposed the use of byte level representation for E2E ASR [1, 2]. By using UTF-8 [3] codewords as the underlying base tokens, the output vocabulary is no longer constrained by the character sets of each language, allowing developers to choose a vocabulary size based on compute, and memory constraints. One well-known multilingual ASR system that uses UTF-8 subwords is Whisper [4].

UTF-8 aims to represent all the characters used in major languages. The encoding and decoding processes are designed to be simple and efficient. UTF-8 is a variable length prefix code where each character is represented by one to four bytes. Most byte sequences are not valid UTF-8 strings, and the UTF-8 decoder needs to detect invalid sequences. UTF-8 also provides backward compatibility, where ASCII characters are represented by a single byte and they are the same as the ASCII encoding. While UTF-8 has proven to be an effective output representation for ASR, it is unclear whether it is optimal. For example, characters with similar pronunciations or meaning are not guaranteed to share the same prefixes. In addition, the large number of invalid byte sequences means the model needs to identify valid UTF-8 strings, an additional burden.

In this work, we explore the possibility of optimizing a byte level representation for E2E ASR with a data driven approach. We believe an ideal byte level representation should:

1. Be optimized for target recognition/classification task;
2. Consider all available information;
3. Provide error correction.

For the first point, we think the representation should be optimized for the target machine learning task. The design should boost accuracy, and be data driven. For the second point, we think the design should consider all available information. For example, if the representation is designed for ASR, it should incorporate the information from both text and audio. Ideally, the framework should be flexible enough to incorporate any available information. For ASR, it means it should be able to take lexicon, phonemes or other useful information into account. For the last property, we believe the representation should provide an error correction mechanism to handle the cases when the model produces an invalid sequence, and the recovery should aim for minimal error.

In this paper, we propose a novel representation learning approach with a vector quantized auto-encoder. We show that we can optimize the byte level representation for ASR with available text and audio data. The framework is flexible enough so it can be extended to incorporate side information, such as a lexicon. Similar to UTF-8, we also provide a mechanism to recover from invalid sequences, and the recovery is optimized for accuracy instead of other metrics.

2. UTF-8 BASED REPRESENTATION

UTF-8 based models have been proposed for natural language processing (NLP) [5] [6] [7]. The idea is to convert text to a

优化端到端 ASR 的字节级表示

Roger Hsiao, Liuhui Deng, Erik McDermott, Ruchir Travadi, Xiaodan Zhuang

苹果

抽象的

我们提出了一种优化端到端自动语音识别（ASR）字节级表示的新方法。当支持的语言的字符集很大时，大规模多语言 ASR 系统通常使用字节级表示。字节级表示的紧凑性和通用性允许 ASR 模型使用更小的输出词汇表，从而提供更大的灵活性。 UTF-8 是多语言 ASR 常用的字节级表示形式，但它并不是为直接优化机器学习任务而设计的。通过使用自动编码器和矢量量化，我们表明我们可以优化 ASR 的字节级表示并实现更好的准确性。我们提出的框架可以合并来自不同模式的信息，并提供纠错机制。在英语/普通话听写任务中，我们表明使用这种方法构建的双语 ASR 模型的相对错误率比 UTF-8 表示高出 5%。

Index Terms— 字节级表示、语音识别

1. 简介

端到端 (E2E) 神经网络是用于多语言自动语音识别 (ASR) 的灵活而准确的模型。这种多语言模型的输出通常是受支持语言的字符或子词的联合。然而，随着语言数量的增加，输出层的大小也会增加，这可能会对计算、内存使用和资产大小产生负面影响。当系统支持中文、日文、韩文（CJK）等大字符集语言时，这个问题更加突出。为了解决这个问题，之前的工作提出使用字节级表示来进行 E2E ASR [1, 2]。通过使用 UTF-8 [3] 代码字作为底层基本标记，输出词汇不再受每种语言字符集的限制，允许开发人员根据计算和内存限制选择词汇大小。 Whisper [4] 是一种使用 UTF-8 子词的著名多语言 ASR 系统。

UTF-8 旨在表示主要语言中使用的所有字符。编码和解码过程被设计得简单而高效。 UTF-8是一种可变长度前缀码，其中每个字符由一到四个表示字节。大多数字节序列不是有效的 UTF-8 字符串，UTF-8 解码器需要检测无效序列。 UTF-8 还提供向后兼容性，其中 ASCII 字符由单个字节表示，并且与 ASCII 编码相同。虽然 UTF-8 已被证明是 ASR 的有效输出表示形式，但尚不清楚它是否是最佳的。例如，不保证具有相似发音或含义的字符共享相同的前缀。此外，大量无效字节序列意味着模型需要识别有效的 UTF-8 字符串，这是一个额外的负担。

在这项工作中，我们探索了使用数据驱动方法优化 E2E ASR 字节级表示的可能性。我们相信理想的字节级表示应该:

1.针对目标识别/分类任务进行优化； 2. 考虑所有可用信息； 3. 提供错误修正。

对于第一点，我们认为应该针对目标机器学习任务来优化表示。设计应该提高准确性，并由数据驱动。对于第二点，我们认为设计应该考虑所有可用的信息。例如，如果表示是为 ASR 设计的，则它应该包含来自文本和音频的信息。理想情况下，该框架应该足够灵活，能够包含任何可用的信息。对于 ASR 来说，这意味着它应该能够考虑词汇、音素或其他有用的信息。对于最后一个属性，我们认为表示应该提供纠错机制来处理模型产生无效序列的情况，并且恢复应该以最小错误为目标。

在本文中，我们提出了一种带有矢量量化自动编码器的新颖表示学习方法。我们表明，我们可以利用可用的文本和音频数据来优化 ASR 的字节级表示。该框架足够灵活，因此可以扩展以包含辅助信息，例如词典。与 UTF-8 类似，我们还提供了一种从无效序列中恢复的机制，并且恢复针对准确性而不是其他指标进行了优化。

2. 基于 UTF-8 的表示

基于 UTF-8 的模型已被提出用于自然语言处理 (NLP) [5] [6] [7]。这个想法是将文本转换为

sequence of variable-length UTF-8 codewords, and to have the model predict one byte at each decoding step. The advantages of byte-level representation are compactness and universality, as any combination of languages may be represented with an output dimension of only 256. However, a sequence represented at byte level is often longer than its character-level counterpart, especially for CJK languages [8]. This is because while Latin characters are represented by a single byte, many CJK characters and accented characters are represented by multiple bytes. As a result, a byte-level model can be error-prone since it needs to make multiple predictions for many single characters, and each prediction might make a mistake.

To compensate for the drawback of making byte level mistakes, [1, 2] propose byte-level subwords for E2E ASR. The idea is to apply byte pair encoding (BPE) [9] to UTF-8 codeword sequences to create UTF-8 subwords. As subwords are in general longer than byte-level tokens, this approach reduces the number of steps required by the decoding process. However, BPE does not guarantee that the output will be a valid UTF-8 sequence. To repair an invalid byte sequence, [1] proposes a dynamic programming algorithm to recover as many characters as possible given any byte sequence. While this dynamic programming approach ensures the output sequence is always valid, it optimizes for the number of valid characters, not ASR quality.

3. OPTIMIZING BYTE-LEVEL REPRESENTATION

3.1. Formulation as a latent variable optimization problem

We first formulate the representation problem as an optimization problem with latent variables. Suppose W is a sequence of label tokens (e.g. words, subwords or characters), X is a sequence of acoustic features and Q is a sequence of discrete latent variables, $[\cdots, q_i, \cdots]$, from a vocabulary $\mathcal{V}_Q$, then we can write the formula of posterior probability as,

$$P(W|X) = \sum_Q P(W|Q, X)P(Q|X)\ . \tag{1}$$

Assuming Q captures sufficient information to infer W, we can then simplify Equation 1 to:

$$P(W|X) = \sum_Q P(W|Q)P(Q|X)\ . \tag{2}$$

The representation problem becomes the optimization problem of finding the set of latent variables $\mathcal{V}_Q$, such that the posterior probability of Equation 2 is maximized. In our case, the sequence Q is the byte-level representation to be optimized for ASR, and it forms the output of the ASR model. In this formulation, $P(Q|X)$ is the ASR model that describes how acoustic features will be mapped to these latent variables. $P(W|Q)$ plays a role similar to a lexicon, which describes how Q can be used to infer W. One difference is that this lexicon will be jointly optimized. In this work, we assume that the relationship between W and Q is deterministic, so we can convert label tokens to and from latent variables.

Optimizing Equation 2 directly is not trivial since Q is hidden and this equation marginalizes over all possible Q. In practice, it means we would need to sample Q from $P(Q|X)$ to approximate the equation, which is still expensive and difficult. At a high level, the challenge here is to solve the representation problem, $P(W|Q)$, and the alignment problem, $P(Q|X)$, at the same time. To circumvent this challenge, we propose to reformulate the task as an auto-encoding problem.

3.2. Formulation as auto-encoding optimization problem

The auto-encoder consists of the following components:

1. a label encoder, $P_L(Q|W)$;
2. an acoustic encoder, $P_A(Q|X)$;
3. a label decoder, $P_D(W|Q)$;
4. a vector quantizer, VQ.

Figure 1 illustrates the architecture of this auto-encoder. This

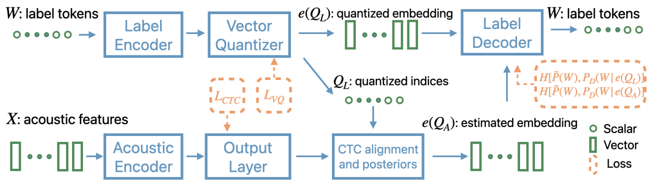

Fig. 1. Overview of the proposed auto-encoder. The label tokens could be words/subwords/characters. In our experiments, the label tokens are characters.

auto-encoder uses vector quantization (VQ) as its bottleneck, with the indices of the quantized embeddings serving as the latent variables. We use Q to represent the sequence of embedding indices and define $e(q)$ as a function that returns the quantized embedding vector given an embedding index q. Each encoder branch represents the information that contributes to building the latent representation, and the decoder would recover the label sequences from the latent variables. This auto-encoder can be optimized by the loss function,

$$\begin{aligned} L_{AE} &= H[\tilde{P}(W), P_D(W|e(Q_L))] \\ &+ H[\tilde{P}(W), P_D(W|e(Q_A))] \\ &+ L_{CTC}[P_A(Q_L|X)] \\ &+ L_{VQ}\ . \end{aligned} \tag{3}$$

This loss function consists of four terms. The first term is the cross entropy loss between the empirical distribution of the reference word sequence and the distribution induced by the label decoder given the output of the label encoder,

可变长度 UTF-8 码字序列，并让模型在每个解码步骤预测一个字节。字节级表示的优点是紧凑性和通用性，因为任何语言组合都可以用仅 256 的输出维度来表示。然而，在字节级表示的序列通常比其字符级对应的序列长，特别是对于 CJK 语言 [8]。这是因为拉丁字符由单个字节表示，而许多 CJK 字符和重音字符则由多个字节表示。因此，字节级模型可能容易出错，因为它需要对许多单个字符进行多次预测，并且每个预测都可能出错。

为了弥补字节级错误的缺点，[1, 2]提出了 E2E ASR 的字节级子字。这个想法是将字节对编码 (BPE) [9] 应用于 UTF-8 码字序列来创建 UTF-8 子字。由于子字通常比字节级标记长，因此这种方法减少了解码过程所需的步骤数。但是，BPE 不保证输出是有效的 UTF-8 序列。为了修复无效的字节序列，[1] 提出了一种动态编程算法，以在给定任何字节序列的情况下恢复尽可能多的字符。虽然这种动态编程方法可确保输出序列始终有效，但它会优化有效字符的数量，而不是 ASR 质量。

3. 优化字节级表示

3.1.作为潜变量优化问题的公式化

我们首先将表示问题表述为具有潜变量的优化问题。假设 W 是标签标记序列（例如单词、子词或字符），X 是声学特征序列，Q 是来自词汇表 $\mathcal{V}_Q$ 的离散潜在变量 [$\cdots, q_i, \cdots$] 序列，那么我们可以将后验概率公式写为：

$$P(W|X) = \sum_Q P(W|Q,X)P(Q|X)\,. \tag{1}$$

假设 Q 捕获足够的信息来推断 W，我们可以将方程 1 简化为：

$$P(W|X) = \sum_Q P(W|Q)P(Q|X)\,. \tag{2}$$

表示问题变成了寻找潜在变量集$\mathcal{V}_Q$的优化问题，使得方程2的后验概率最大化。在我们的例子中，序列 Q 是针对 ASR 优化的字节级表示，它形成了 ASR 模型的输出。在此公式中，$P(Q|X)$ 是 ASR 模型，描述如何将声学特征映射到这些潜在变量。

$P(W|Q)$ 起到类似于词典的作用，它描述了如何使用 Q 来推断 W。一个区别是这个 lex- icon 将被联合优化。在这项工作中，我们假设 W 和 Q 之间的关系是确定性的，因此我们可以将标签标记与潜在变量相互转换。

直接优化方程 2 并非易事，因为 Q 是隐藏的，并且该方程边缘化了所有可能的 Q。实际上，这意味着我们需要从 $P(Q|X)$ 中采样 Q 来近似方程，这仍然是昂贵且困难的。在高层次上，这里的挑战是同时解决表示问题 $P(W|Q)$ 和对齐问题 $P(Q|X)$。为了规避这一挑战，我们建议将任务重新表述为自动编码问题。

3.2.作为自动编码优化问题的公式化

自动编码器由以下组件组成：

1. 标签编码器，$P_L(Q|W)$； 2. 声学编码器，$P_A(Q|X)$； 3. 标签解码器，$P_D(W|Q)$； 4.矢量量化器，VQ。

图 1 展示了该自动编码器的架构。这

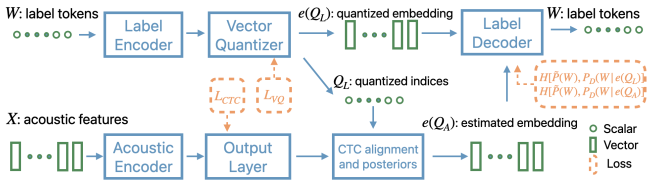

图 1.所提出的自动编码器概述。标签标记可以是单词/子单词/字符。在我们的实验中，标签标记是字符。

自动编码器使用矢量量化（VQ）作为其瓶颈，量化嵌入的索引作为潜在变量。我们使用 Q 来表示嵌入索引的序列，并将 $e(q)$ 定义为一个函数，该函数返回给定嵌入索引 q 的量化嵌入向量。每个编码器分支表示有助于构建潜在表示的信息，并且解码器将从潜在变量中恢复标签序列。该自动编码器可以通过损失函数进行优化，

$$\begin{aligned} L_{AE} \quad &= \quad H[\tilde{P}(W), P_D(W|e(Q_L))] \\ &+ \quad H[\tilde{P}(W), P_D(W|e(Q_A))] \\ &+ \quad L_{CTC}[P_A(Q_L|X)] \\ &+ \quad L_{VQ}\,. \end{aligned} \tag{3}$$

该损失函数由四项组成。第一项是参考词序列的经验分布与给定标签编码器的输出的标签解码器引起的分布之间的交叉熵损失，

$Q_L = [\cdots, q_i, \cdots]$. This term is the standard loss of an auto-encoder, where it first uses the label encoder, $P_L(Q|W)$, to decide Q_L. Then, the corresponding sequence of embeddings, $e(Q_L) = [\cdots, e(q_i), \cdots]$, is fed to the label decoder to compute $P_D(W|e(Q_L))$. This term would optimize the label decoder and the label encoder end-to-end.

The second cross entropy loss in Equation 3 represents the loss of the decoder when it uses the information from the acoustic encoder. Q_A is a sequence of latent variables chosen by the acoustic encoder, and has the same length as Q_L. However, instead of sampling from $P_A(Q|X)$ to obtain Q_A explicitly, a sequence of corresponding embeddings, $[\cdots, e_i(Q_A), \cdots]$, is obtained via alignment of the acoustic features with Q_L from the label encoder, followed by a linear combination of the label embeddings,

$$\underbrace{[\cdots, e_i(Q_A), \cdots]}_{\text{same length as } Q_L} = [\cdots, \sum_{q \in \mathcal{V}_Q} P(q|x_{t(i)}) e(q), \cdots] \ . \quad (4)$$

Here, for each token $q_i \in Q_L$, we use the aligned feature, $x_{t(i)}$, for which q_i is first emitted. Then, we compute the posterior probabilities across all embedding vectors in the VQ codebook, $P(q|x_{t(i)})$, and use them to construct the embedding to be fed to the label decoder, $e_i(Q_A)$. This term would optimize the label decoder and the acoustic encoder end-to-end.

The third term is a CTC loss [10] for the acoustic encoder. This term computes how well the acoustic features are aligned to the latent variables, Q_L, which are decided by the label encoder. The CTC loss would also produce the alignment information required to compute the second term. This term would optimize only the acoustic encoder. Finally, the fourth term is the quantization loss for VQ.

3.3. Vector quantization variational auto-encoder

In this work, we use a vector quantization variational auto-encoder (VQ-VAE) [11] as our vector quantizer. VQ-VAE discretizes real-valued vectors into some discrete variables while still allowing a neural network to be trained end-to-end. Standard version of VQ-VAE consists of a codebook of M embedding vectors. During inference, an input vector, z, will be compared to all embedding vectors in the codebook. The closest one based on euclidean distance will be chosen and VQ-VAE would output the embedding vector $e(q)$ and its index q. During training, it optimizes the following loss function,

$$\mathit{VQLoss}(z) = ||sg[z] - e(q_z)||_2 + \beta||z - sg[e(q_z)]||_2 \ , \quad (5)$$

where z is the input to the VQ module; q_z is the index of the closest embedding vector $e(q_z)$ to z; sg is the stop gradient operator and β is a tunable parameter. This loss function aims to keep the embedding vectors close to the inputs, and this design helps the training to be more stable [11]. To enable back-propagation, it uses a straight-through estimator that copies the gradient from the downstream to the upstream directly, as if the VQ component does not exist [11].

The representational power of VQ-VAE depends on the size of the codebook. Given a codebook of M embeddings, it assumes the inputs can be grouped into M clusters. Residual VQ-VAE (RVQ-VAE) improves by applying multiple VQ-VAE iteratively to an input [12]. For RVQ-VAE, we have N codebooks and each codebook has M embeddings. Given an input, it is quantized by the first VQ-VAE module and the difference between the input and quantized embedding would then be quantized by the next VQ-VAE module. This process continues until it has used all the codebooks in RVQ-VAE. Theoretically, RVQ-VAE can represent up to M^N different inputs.

In this paper, we use RVQ-VAE with two or three ($N = 2, 3$) codebooks and the codebook size M is always 256. Therefore, each embedding index can be represented by one byte, and each label token is represented by N bytes. We choose this setting so we can compare to UTF-8 encoding, though in practice, one can pick any number of codebooks and codebook size.

3.4. Convert bytes to labels with error correction

When using a byte-level representation, errors made by the ASR model could create invalid byte sequences. For example, the byte sequence might contain substitution, deletion or insertion errors. Therefore, we need an error correction mechanism, and such a mechanism should optimize for accuracy. In our proposed VQ representation, the label decoder at inference time can perform error correction by estimating the most likely label sequence, as shown in Algorithm 1. In this algorithm, errors in a byte sequence would affect the quality of the embeddings, but the label decoder still has the potential to map those embeddings to the correct label. Note that this algorithm is also used at training time with the multiple bytes per label token produced by RVQ-VAE, but since the byte sequence is decided by the label encoder applied to the reference text, optimization does not directly target insertion or deletion errors.

3.5. Comparing UTF-8 and VQ based representation

While UTF-8 and our proposed VQ based representations are both byte-level representations, they have a few key differences. First, UTF-8 is a variable length prefix code, but our proposed VQ representation is a fixed length code. Second, VQ representation can be optimized for a specific machine learning task while UTF-8 is fixed and not optimized for machine learning. Third, UTF-8 recovers invalid sequences with the sequences with most characters [1], while VQ's label decoder would try to find the most likely output given any input sequence. Finally, as a learned representation, it is possible to

$Q_L = [\cdots, q_i, \cdots]$。该术语是自动编码器的标准损失，它首先使用标签编码器 $P_L(Q|W)$ 来决定 Q_L。然后，相应的嵌入序列 $e(Q_L) = [\cdots, e(q_i), \cdots]$ 被馈送到标签解码器以计算 $P_D(W|e(Q_L))$。该术语将端到端地优化标签解码器和标签编码器。

等式 3 中的第二个交叉熵损失表示解码器使用来自声学编码器的信息时的损失。 Q_A是声学编码器选择的潜在变量序列，并且与Q_L具有相同的长度。然而，不是从 $P_A(Q|X)$ 中采样来显式获得 Q_A，而是通过将声学特征与来自标签编码器的 Q_L 对齐来获得相应的嵌入序列 $[\cdots, e_i(Q_A), \cdots]$，然后是标签嵌入的线性组合，

$$\underbrace{[\cdots, e_i(Q_A), \cdots]}_{\text{same length as } Q_L} = [\cdots, \sum_{q \in \mathcal{V}_Q} P(q|x_{t(i)})e(q), \cdots] \,. \quad (4)$$

在这里，对于每个标记 $q_i \in Q_L$，我们使用对齐的特征 $x_{t(i)}$，首先发出 q_i。然后，我们计算 VQ 码本中所有嵌入向量的后验概率，$P(q|x_{t(i)})$，并使用它们来构建要馈送到标签解码器的嵌入，$e_i(Q_A)$。该术语将端到端地优化标签解码器和声学编码器。

第三项是声学编码器的 CTC 损失 [10]。该术语计算声学特征与潜在变量 Q_L 的对齐程度，该变量由标签编码器决定。 CTC 损失还将产生计算第二项所需的对齐信息。该术语将仅优化声学编码器。最后，第四项是 VQ 的量化损失。

3.3.矢量量化变分自动编码器

在这项工作中，我们使用矢量量化变分自动编码器（VQ-VAE）[11]作为矢量量化器。 VQ-VAE 将实值向量离散化为一些离散变量，同时仍然允许神经网络进行端到端训练。 VQ-VAE 的标准版本由 M 嵌入向量的码本组成。在推理过程中，输入向量 z 将与码本中的所有嵌入向量进行比较。将选择基于欧氏距离的最接近的一个，VQ-VAE 将输出嵌入向量 $e(q)$ 及其索引 q。在训练过程中，它优化以下损失函数，

$$\mathit{VQLoss}(z) = ||\mathit{sg}[z] - e(q_z)||_2 + \beta||z - \mathit{sg}[e(q_z)]||_2 \,, \quad (5)$$

其中 z 是 VQ 模块的输入； q_z 是最接近 $e(q_z)$ 到 z 的嵌入向量的索引； sg 是停止梯度运算符，β 是可调参数。这种损失函数旨在保持嵌入向量接近输入，这种设计有助于训练更加稳定[11]。到启用反向传播，它使用直通估计器将梯度从下游直接复制到上游，就好像 VQ 分量不存在一样[11]。

VQ-VAE 的表征能力取决于码本的大小。给定 M 嵌入的码本，它假设输入可以分为 M 集群。残余 VQ-VAE (RVQ-VAE) 通过对输入迭代应用多个 VQ-VAE 进行改进 [12]。对于 RVQ-VAE，我们有 N 码本，每个码本都有 M 嵌入。给定一个输入，它由第一个 VQ-VAE 模块量化，然后输入和量化嵌入之间的差异将由下一个 VQ-VAE 模块量化。这个过程一直持续到用完 RVQ-VAE 中的所有码本为止。理论上，RVQ-VAE 可以表示最多 M^N 个不同的输入。

在本文中，我们使用具有两个或三个（$N = 2, 3$）码本的RVQ-VAE，并且码本大小M始终为256。因此，每个嵌入索引可以由一个字节表示，每个标签令牌由N字节表示。我们选择此设置是为了与 UTF-8 编码进行比较，但实际上，我们可以选择任意数量的码本和码本大小。

3.4.将字节转换为带有纠错的标签

使用字节级表示时，ASR 模型所犯的错误可能会创建无效的字节序列。例如，字节序列可能包含替换、删除或插入错误。因此，我们需要一种纠错机制，并且这种机制应该针对准确性进行优化。在我们提出的 VQ 表示中，推理时的标签解码器可以通过估计最可能的标签序列来执行纠错，如算法 1 所示。在该算法中，字节序列中的错误会影响嵌入的质量，但标签解码器仍然有可能将这些嵌入映射到正确的标签。请注意，该算法也在训练时使用 RVQ-VAE 生成的每个标签标记多个字节，但由于字节序列由应用于参考文本的标签编码器决定，因此优化并不直接针对插入或删除错误。

3.5.比较 UTF-8 和基于 VQ 的表示

虽然 UTF-8 和我们提出的基于 VQ 的表示都是字节级表示，但它们有一些关键的区别。首先，UTF-8是变长前缀码，但我们提出的VQ表示是定长码。其次，VQ 表示可以针对特定的机器学习任务进行优化，而 UTF-8 是固定的并且没有针对机器学习进行优化。第三，UTF-8 使用包含最多字符的序列来恢复无效序列 [1]，而 VQ 的标签解码器将尝试在给定任何输入序列的情况下找到最可能的输出。最后，作为学习的表示，可以

Algorithm 1 The bytes to labels conversion procedure for our proposed VQ based representation. The label decoder would produce the most likely labels even if the input byte sequence contains errors

```
function BYTES_TO_LABELS(Q: List[int])
    $W \leftarrow []$                      ▷ the label sequence to be returned
    $v \leftarrow \vec{0}$                      ▷ a variable to store the embedding
    $k \leftarrow -1$
    for $i = 1, \ldots, len(Q)$ do
        $j \leftarrow cb(Q[i])$          ▷ get the codebook index of $Q[i]$
        if $k == -1$ or $k < j$ then
            $v \leftarrow v + e(Q[i])$          ▷ add the $Q[i]$'s emb to $v$
        else
            $w \leftarrow \arg\max_w P_D(w|v)$          ▷ label decoder
            $W \leftarrow W \cdot w$                    ▷ append $w$ to $W$
            $v \leftarrow e[Q[i]]$                   ▷ reset $v$ to $Q[i]$'s emb
        end if
        $k \leftarrow j$
    end for
    $w \leftarrow \arg\max_w P_D(w|v)$                ▷ label decoder
    $W \leftarrow W \cdot w$                          ▷ append $w$ to $W$
    return W
end function
```

have collision, i.e. different characters can be mapped to the same byte sequences, which is not the case for UTF-8.

4. EXPERIMENTAL RESULTS

We evaluate our approach through our proprietary English and Mandarin dictation tasks. Since our proposed approach is new, we would like to focus on simpler cases where the ASR system only handles two languages. We build these bilingual English and Mandarin ASR models using a mixture of monolingual English and Mandarin supervised data, which is randomly sampled and anonymized. The English training set consists of 10k hours of data while the Mandarin training set has 14k hours of data. Our training sets cover domains like dictation and assistant use cases. For test sets, we use the dictation test sets for English and Mandarin, which are randomly sampled and anonymized. The English test set consists of 27 hours of data and the Mandarin test set has 13 hours of data. Both train and test sets cover platforms like smart phones, tablets and laptops. When we report accuracy numbers on these test sets, we report word error rate (WER) for English and character error rate (CER) for Mandarin. We use the term token error rate (TER) when we discuss the error rates of both languages.

The model we built is a CTC-AED model similar to the one described in [13]. The model consists of an acoustic encoder and a cross attention decoder (AED). The input to the CTC-AED model is 80 dimensional mel filterbank features with 25ms window and 10ms shift. A depthwise separable convolutional layer [14] would downsample the features by 6x. The acoustic encoder consists of 12 conformer blocks [15] with eight attention heads and the hidden dimension is 512. The attention decoder consists of six layers of bi-directional transformer blocks. Each transformer block has eight attention heads and the hidden dimension is also 512. The entire model has around 120M parameters. The decoding process consists of two passes. In the first pass, we perform CTC prefix beam search with the acoustic encoder to generate N-best hypotheses. Then, these hypotheses are rescored by the attention decoder for the final output.

We compare three types of output representations for this bilingual setup,

1. Character based output
2. UTF-8 subword output with BPE
3. Our proposed VQ based subword output with BPE

Both UTF-8 and VQ representation use BPE to create subword outputs. The character based representation has all English and simplified Chinese characters, and the size is around 8000.

For VQ based representation, the auto-encoder consists of three components: acoustic encoder, label encoder and label decoder. The acoustic encoder has the same model architecture as the acoustic encoder of the CTC-AED model. It has an output layer that predicts the embeddings used in the VQ module. This output layer is used for the CTC loss in Equation 3. The label encoder consists of six uni-directional transformer blocks, and the label decoder is a linear transform that converts the input embedding to the output tokens. The label encoder and decoder are connected by the VQ module. After we train the auto-encoder, we use the label encoder to convert all the transcripts in our training data into byte sequences. Then, we run BPE with SentencePiece to create byte-level subwords similar to the UTF-8 approach. These byte-level subwords will be used as the output of the CTC-AED model.

Table 1 shows the results of the three bilingual CTC-AED models built with these output representations. Across different number of subwords in the ASR output, VQ based representation has improvement over UTF-8 based representation. With 8000 subwords, proposed VQ based approach has 5.8% relative reduction in WER for English and 3.7% relative reduction in CER for Mandarin. This suggests it is possible to optimize a byte-level representation for a target task. Compared to character based representation, both VQ and UTF-8 are better on English but similar accuracy on Chinese. With 8000 subwords, which has the same size as the character based output, VQ has 14.8% and 2.3% relative reduction in error rate for English and Mandarin respectively. Character based output is expected to have worse English accuracy since it does not have common English subwords in the output. In comparison, VQ and UTF-8 can have an output dimension smaller than the size of the character set. They provide more flexibility to system design.

Algorithm 1 The bytes to labels conversion procedure for our proposed VQ based representation. The label decoder would produce the most likely labels even if the input byte sequence contains errors

```
function BYTES_TO_LABELS(Q: List[int])
    $W \leftarrow []$                          ▷ the label sequence to be returned
    $v \leftarrow \vec{0}$                          ▷ a variable to store the embedding
    $k \leftarrow -1$
    for $i = 1, \ldots, len(Q)$ do
        $j \leftarrow cb(Q[i])$                 ▷ get the codebook index of $Q[i]$
        if $k == -1$ or $k < j$ then
            $v \leftarrow v + e(Q[i])$          ▷ add the $Q[i]$'s emb to $v$
        else
            $w \leftarrow \arg\max_w P_D(w|v)$        ▷ label decoder
            $W \leftarrow W \cdot w$                  ▷ append $w$ to $W$
            $v \leftarrow e[Q[i]]$                    ▷ reset $v$ to $Q[i]$'s emb
        end if
        $k \leftarrow j$
    end for
    $w \leftarrow \arg\max_w P_D(w|v)$                ▷ label decoder
    $W \leftarrow W \cdot w$                          ▷ append $w$ to $W$
    return W
end function
```

存在冲突，即不同的字符可以映射到相同的字节序列，而 UTF-8 则不然。

4 实验结果

我们通过专有的英语和普通话听写任务来评估我们的方法。由于我们提出的方法是新的，因此我们希望重点关注 ASR 系统仅处理两种语言的更简单的情况。我们使用单语英语和普通话监督数据的混合来构建这些双语英语和普通话 ASR 模型，这些数据是随机采样和匿名的。英语训练集包含 10k 小时的数据，而普通话训练集包含 14k 小时的数据。我们的训练集涵盖听写和助理用例等领域。对于测试集，我们使用英语和普通话的听写测试集，这些测试集是随机采样和匿名的。英语测试集包含 27 小时的数据，普通话测试集包含 13 小时的数据。训练集和测试集都涵盖智能手机、平板电脑和笔记本电脑等平台。当我们报告这些测试集的准确率数字时，我们会报告英语的单词错误率 (WER) 和普通话的字符错误率 (CER)。当我们讨论两种语言的错误率时，我们使用术语令牌错误率（TER）。

我们构建的模型是类似于[13]中描述的 CTC-AED 模型。该模型由声学编码器和交叉注意解码器（AED）组成。 CTC-AED 模型的输入是 80 维梅尔滤波器组特征，具有 25ms 窗口和 10ms 偏移。深度分离耕地卷积层 [14] 会将特征下采样 6 倍。声学编码器由 12 个一致块 [15] 组成，具有 8 个注意力头，隐藏维度为 512。注意力解码器由六层双向变换器块组成。每个 Transformer 块有 8 个注意力头，隐藏维度也是 512。整个模型有大约 120M 参数。解码过程由两遍组成。在第一遍中，我们使用声学编码器执行 CTC 前缀波束搜索，以生成 N 个最佳假设。然后，这些假设由注意力解码器重新评分以获得最终输出。

我们比较了这种双语设置的三种类型的输出表示形式，

1. 基于字符的输出 2. 使用 BPE 的 UTF-8 子字输出 3. 我们提出的基于 VQ 的使用 BPE 的子字输出

UTF-8 和 VQ 表示都使用 BPE 来创建子字输出。基于字符的表示有全英文和简体中文字符，大小在8000左右。

对于基于 VQ 的表示，自动编码器由三个组件组成：声学编码器、标签编码器和标签解码器。声学编码器与 CTC-AED 模型的声学编码器具有相同的模型架构。它有一个输出层，用于预测 VQ 模块中使用的嵌入。该输出层用于等式3中的CTC损失。标签编码器由六个单向转换器块组成，标签解码器是将输入嵌入转换为输出标记的线性变换。标签编码器和解码器通过VQ模块连接。训练自动编码器后，我们使用标签编码器将训练数据中的所有转录本转换为字节序列。然后，我们使用 SentencePiece 运行 BPE 以创建类似于 UTF-8 方法的字节级子词。这些字节级子字将用作 CTC-AED 模型的输出。

表 1 显示了使用这些输出表示构建的三个双语 CTC-AED 模型的结果。在 ASR 输出中不同数量的子词中，基于 VQ 的表示比基于 UTF-8 的表示有改进。所提出的基于 VQ 的方法有 8000 个子词，英语的 WER 相对减少 5.8%，普通话的 CER 相对减少 3.7%。这表明可以优化目标任务的字节级表示。与基于字符的表示相比，VQ 和 UTF-8 在英语上都更好，但在中文上的准确性相似。VQ 有 8000 个子词，其大小与基于字符的输出相同，因此英语和普通话的错误率分别相对降低了 14.8% 和 2.3%。基于字符的输出预计英语准确性较差，因为它的输出中没有常见的英语子词。相比之下，VQ 和 UTF-8 的输出维度可以小于字符集的大小。它们为系统设计提供了更大的灵活性。

Table 1. TER(%) of CTC-AED model using character based, UTF-8 and our proposed VQ based representation. The VQ based representation has three codebooks and each codebook has 256 embeddings ($N = 3$, $M = 256$). Both UTF-8 and VQ use BPE to create subword output with their corresponding error correction algorithms. Each column uses different number of subwords in the ASR output.

rep.	lang	2000	4000	8000	16000
Char	EN	-	-	9.61	-
UTF-8	EN	9.25	8.84	8.70	8.41
VQ	EN	8.77	8.29	8.19	8.14
Char	ZH	-	-	9.77	-
UTF-8	ZH	10.53	10.27	9.92	9.43
VQ	ZH	10.48	10.06	9.55	9.14

Table 2. TER(%) of CTC-AED model using our proposed VQ based representation with different VQ configurations.

VQ config	lang	2000	4000	8000
$N = 2, M = 256$	EN	8.51	8.30	8.40
$N = 3, M = 256$	EN	8.77	8.29	8.19
$N = 2, M = 256$	ZH	11.32	10.96	10.70
$N = 3, M = 256$	ZH	10.48	10.06	9.55

4.1. Ablation study

We would like to look at how factors, like the number of codebooks in VQ, and the four losses used in training the auto-encoder would affect the ASR accuracy. Table 2 shows the results of comparing different number of codebooks. We are interested in the case of two codebooks since effectively, each label token would be represented by two bytes and theoretically, it should be enough to cover 8000 English and Chinese characters. However, as shown by the results, two codebooks are not enough and give suboptimal accuracy. It is due to the low utilization rate of the codebooks, where many embeddings in the codebooks are inactive. This problem, known as index collapse [16], is thoroughly studied in [17] and we will explore those techniques in our future work.

Table 3 shows the results of applying different weights to the cross entropy loss with respect to the acoustic encoder (the second term of Equation 3). When the weight is zero, it means the acoustic encoder would not contribute to representation learning. In this case, the auto-encoder would ignore acoustic information and the acoustic encoder would fit to the latent variables decided by the label encoder. The results in Table 3 show that acoustic information helps, and this supports the core concept of this paper, where optimizing a representation with all available information is beneficial.

Table 3. TER(%) of CTC-AED model when the underlying VQ representations are trained with different weights to the cross entropy loss with respect to the acoustic encoder. For this experiment, the VQ module has three codebooks and each codebook has 256 embeddings ($N = 3, M = 256$). BPE is performed to generate 8000 byte-level subwords

lang	0.0	0.5	1.0
EN	8.61	8.49	8.19
ZH	9.71	9.89	9.55

5. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel algorithm to optimize a byte-level representation for ASR and compare it with UTF-8 representation. Our proposed representation can be optimized with both audio and text data, and the error correction mechanism is optimized for accuracy. On our English/Mandarin dictation test sets, our auto-encoder and VQ based approach can achieve 5% relative reduction in TER.

As this proposed algorithm is new, we focus on simpler bilingual ASR in this paper. To learn a representation that can cover all languages, we believe there are multiple issues that need to be addressed, like the index collapse issue [16]. The index collapse issue is well studied within the machine learning community [17] and we will explore some of the ideas in that space. We hope that as the algorithm becomes more mature, we can learn a universal byte-level representation for all languages that functions like UTF-8.

Comparing UTF-8 and our proposed algorithm, while our approach has accuracy advantages, UTF-8 has some desirable properties, such as, being a prefix code, not having a collision issue. In addition, the variable length design is more flexible, though ideally, the length would be correlated with token frequency. In the future, we will explore VQ methods that allow variable length. If code length can be optimized jointly, we would be able to bypass BPE and use that to generate target subwords directly. While as a machine learning approach, it cannot guarantee no collision in the learned representation, we will look into whether we can derive a prefix code from the auto-encoder. This would also make encoding and decoding more efficient. Then, we will also look into using other side information, such as a phonemic lexicon, to improve the representation.

6. ACKNOWLEDGMENT

The authors would like to thank Pawel Swietojanski, Ossama Abdelhamid, Takaaki Hori, and Barry Theobald for their support and useful discussions.

表 1. 使用基于字符、UTF-8 和我们提出的基于 VQ 表示的 CTC-AED 模型的 TER(%)。基于 VQ 的表示具有三个码本，每个码本具有 256 个嵌入（$N=3$，$M=256$）。UTF-8 和 VQ 都使用 BPE 及其相应的纠错算法来创建子字输出。每列在 ASR 输出中使用不同数量的子字。

rep.	lang	2000	4000	8000	16000
Char	EN	-	-	9.61	-
UTF-8	EN	9.25	8.84	8.70	8.41
VQ	EN	8.77	8.29	8.19	8.14
Char	ZH	-	-	9.77	-
UTF-8	ZH	10.53	10.27	9.92	9.43
VQ	ZH	10.48	10.06	9.55	9.14

表 2. CTC-AED 模型的 TER(%)，使用我们提出的具有不同 VQ 配置的基于 VQ 的表示。

VQ config	lang	2000	4000	8000
$N=2, M=256$	EN	8.51	8.30	8.40
$N=3, M=256$	EN	8.77	8.29	8.19
$N=2, M=256$	ZH	11.32	10.96	10.70
$N=3, M=256$	ZH	10.48	10.06	9.55

4.1.消融研究

我们想看看 VQ 中的码本数量以及训练自动编码器时使用的四个损失等因素如何影响 ASR 准确性。表2显示了不同数量的码本的比较结果。我们对两个密码本的情况感兴趣，因为实际上，每个标签标记将由两个字节表示，理论上，它应该足以覆盖 8000 个英文和中文字符。然而，如结果所示，两个码本是不够的，而且精度不高。这是由于码本的利用率较低，码本中的许多嵌入是不活跃的。这个问题被称为索引崩溃[16]，在[17]中进行了深入研究，我们将在未来的工作中探索这些技术。

表 3 显示了对声学编码器的交叉熵损失应用不同权重的结果（等式 3 的第二项）。当权重为零时，意味着声学编码器不会对表示学习做出贡献。在这种情况下，自动编码器将忽略声学信息，并且声学编码器将适合标签编码器决定的潜在变量。表 3 中的结果表明声学信息有帮助，这支持了本文的核心概念，即使用所有可用信息优化表示是有益的。

表 3. 当底层 VQ 表示使用与声学编码器相关的交叉熵损失的不同权重进行训练时，CTC-AED 模型的 TER(%)。对于本实验，VQ 模块具有三个码本，每个码本具有 256 个嵌入（$N=3, M=256$）。执行BPE生成8000字节级子字

lang	0.0	0.5	1.0
EN	8.61	8.49	8.19
ZH	9.71	9.89	9.55

5. 结论和未来工作

在本文中，我们提出了一种新的算法来优化 ASR 的字节级表示，并将其与 UTF-8 表示进行比较。我们提出的表示可以使用音频和文本数据进行优化，并且纠错机制也针对准确性进行了优化。在我们的英语/普通话听写测试集上，我们的自动编码器和基于 VQ 的方法可以实现 TER 相对减少 5%。

由于这个算法是新的，我们在本文中重点关注更简单的双语 ASR。为了学习可以覆盖所有语言的表示，我们认为需要解决多个问题，例如索引崩溃问题[16]。机器学习社区[17]对索引崩溃问题进行了深入研究，我们将探讨该领域的一些想法。我们希望随着算法变得更加成熟，我们可以为所有具有类似 UTF-8 功能的语言学习通用的字节级表示。

比较 UTF-8 和我们提出的算法，虽然我们的方法具有准确性优势，但 UTF-8 具有一些理想的属性，例如作为前缀代码，不存在冲突问题。此外，可变长度设计更加灵活，尽管理想情况下，长度将与令牌频率相关。将来，我们将探索允许可变长度的 VQ 方法。如果代码长度可以联合优化，我们将能够绕过 BPE 并使用它直接生成目标子词。虽然作为一种机器学习方法，它不能保证学习的表示不会发生冲突，但我们将研究是否可以从自动编码器中导出前缀代码。这也将使编码和解码更加高效。然后，我们还将研究使用其他辅助信息（例如音素词典）来改进表示。

6.致谢

作者要感谢 Pawel Swietojanski、Ossama Abdelhamid、Takaaki Hori 和 Barry Theobald 的支持和有益的讨论。

7. REFERENCES

[1] Bo Li, Yu Zhang, Tara Sainath, Yonghui Wu, and William Chan, "Bytes are all you need: End-to-end multilingual speech recognition and synthesis with bytes," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 5621–5625.

[2] L. Deng, R. Hsiao, and A. Ghoshal, "Bilingual end-to-end ASR with byte-level subwords," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2022.

[3] Julie D. Allen, Deborah Anderson, Joe Becker, Richard Cook, Mark Davis, Peter Edberg, Asmus Freytag, Richard Ishida, John H. Jenkins, Rick McGowan, Lisa Moore, Eric Muller, Addison Phillips, Michel Suignard, and Ken Whistler, "The unicode standard version 6.0 – core specification," Tech. Rep., The Unicode Consortium, 2011.

[4] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proceedings of the 40th International Conference on Machine Learning*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, Eds. 23–29 Jul 2023, vol. 202 of *Proceedings of Machine Learning Research*, pp. 28492–28518, PMLR.

[5] Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya, "Multilingual language processing from bytes," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, 2016, pp. 1296–1306.

[6] Marta Ruiz Costa-Jussà, Carlos Escolano Peinado, and José Adrián Rodríguez Fonollosa, "Byte-based neural machine translation," in *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, 2017, pp. 154–158.

[7] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel, "Byt5: Towards a token-free future with pre-trained byte-to-byte models," in *arXiv preprint arXiv:2105.13626*, 2021.

[8] Changhan Wang, Kyunghyun Cho, and Jiatao Gu, "Neural machine translation with byte-level subwords," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 9154–9160.

[9] Rico Sennrich, Barry Haddow, and Alexandra Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 1715–1725.

[10] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the International Conference on Machine Learning*, 2006, pp. 369–376.

[11] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems*, 2017, vol. 30.

[12] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han, "Autoregressive image generation using residual quantization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 11523–11532.

[13] Zhuoyuan Yao, Di Wu, Xiong Wang, Binbin Zhang, Fan Yu, Chao Yang, Zhendong Peng, Xiaoyu Chen, Lei Xie, and Xin Lei, "Wenet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit," in *Proc. Interspeech*, Brno, Czech Republic, 2021, IEEE.

[14] François Chollet, "Xception: Deep learning with depthwise separable convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1800–1807.

[15] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech 2020*, 2020, pp. 5036–5040.

[16] L. Kaiser, S. Bengio, A. Roy, A. Vaswani, N. Parmar, J. Uszkoreit, and N. Shazeer, "Fast decoding in sequence models using discrete latent variables," in *Proceedings of the International Conference on Machine Learning*, 2018.

[17] Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola, "Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks," in *Proceedings of the 40th International Conference on Machine Learning*, 2023.

7. 参考文献

[1] Bo Li、Yu Zhang、Tara Sainath、Yonghui Wu 和 William Chan，“字节就是你所需要的：端到端多语言语音识别和字节合成”，载于 *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*，2019 年，第 5621-5625 页。

[2] L. Deng、R. Hsiao 和 A. Ghoshal，“具有字节级子词的双语端到端 ASR”，载于 *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*，2022 年。

[3] Julie D. Allen、Deborah Anderson、Joe Becker、Richard Cook、Mark Davis、Peter Edberg、Asmus Freytag、Richard Ishida、John H. Jenkins、Rick McGowan、Lisa Moore、Eric Muller、Addison Phillips、Michel Suignard 和 Ken Whistler，“Unicode 标准版本 6.0 – 核心规范”，Tech。代表，Unicode 联盟，2011 年。

[4] Alec Radford、Jong Wook Kim、Tao Xu、Greg Brockman、Christine Mcleavey 和 Ilya Sutskever，“通过大规模弱监督进行鲁棒语音识别”，载于 *Proceedings of the 40th International Conference on Machine Learning*，Andreas Krause、Emma Brunskill、Kunghyun Cho、Barbara Engelhardt、Sivan Sabato 和 Jonathan Scarlett，编辑。2023 年 7 月 23-29 日，卷。*Proceedings of Machine Learning Research* 202，第 28492–28518 页，PMLR。

[5] Dan Gillick、Cliff Brunk、Oriol Vinyals 和 Amarnag Subramanya，“字节多语言处理”，载于 *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*，2016 年，第 1296-1306 页。

[6] Marta Ruiz Costa-Jussà、Carlos Escolano Peinado 和 José Adrián Rodr´guez Fonollosa，“基于字节的神经机器翻译”，*Proceedings of the First Workshop on Subword and Character Level Models in NLP*，2017 年，第 154-158 页。

[7] Linting Xu、Aditya Barua、Noah Constant、Rami Al-Rfou、Sharan Narang、Mihir Kale、Adam Roberts 和 Colin Raffel，“Byt5：通过预训练的字节到字节模型迈向无代币的未来”，*arXiv preprint arXiv:2105.13626*，2021 年。

[8] Changhan Wang、Kyunghyun Cho 和 Jiatao Gu，“具有字节级子词的神经机器翻译”，载于 *Proceedings of the AAAI Conference on Artificial Intelligence*，2020 年，第 9154-9160 页。

[9] Rico Sennrich、Barry Haddow 和 Alexandra Birch，“带有子词单元的稀有词的神经机器翻译”，载于 *Proceedings of the Annual Meeting of the Association for Computational Linguistics*，2016 年，第 1715-1725 页。

[10] Alex Graves、Santiago Fernández、Faustino Gomez 和 Jürgen Schmidhuber，“连接主义时间分类：用循环神经网络标记未分段的序列数据”，*Proceedings of the International Conference on Machine Learning*，2006 年，第 369-376 页。

[11] Aaron van den Oord、Oriol Vinyals 和 Koray Kavukcuoglu，“神经离散表示学习”，*Advances in Neural Information Processing Systems*，2017 年，第 1 卷。30.

[12] Doyup Lee、Chiheon Kim、Saehoon Kim、Minsu Cho 和 Wook-Shin Han，“使用残差量化的自回归图像生成”，载于 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*，2022 年 6 月，第 11523-11532 页。

[13] Zhuoyuan Yao、Di Wu、Xiong Wang、Binbin 张、Fan Yu、Chao Yang、Zhendong Peng、Xiaoyu Chen、Lei Xie 和 Xin Lei，“Wenet：面向生产的流式和非流式端到端语音识别工具包”，*Proc. Interspeech*，捷克共和国布尔诺，2021 年，IEEE。

[14] François Chollet，“Xception：深度学习与深度可分离卷积”，*IEEE Conference on Computer Vision and Pattern Recognition*，2017 年，第 1800-1807 页。

[15] Anmol Gulati、Jamesqin、Chung-ChengChiu、NikiParmar、YuZhang、JiahuiYu、WeiHan、ShiboWang、ZhengdongZhang、YonghuiWu 和 RuomingPang，“Conformer：用于语音识别的卷积增强变压器”，*Proc. Interspeech 2020*，2020 年，第 5036-5040 页。

[16] L. Kaiser、S. Bengio、A. Roy、A. Vaswani、N. Parmar、J. Uszkoreit 和 N. Shazeer，“使用离散潜在变量的序列模型中的快速解码”，in *Proceedings of the International Conference on Machine Learning*，2018。

[17] Minyoung Huh、Brian Cheung、Pulkit Agrawal 和 Phillip Isola，“理顺直通估计器：克服矢量量化网络中的优化挑战”，*Proceedings of the 40th International Conference on Machine Learning*，2023 年。