A COMPARISON OF SELF-SUPERVISED SPEECH REPRESENTATIONS AS INPUT FEATURES FOR UNSUPERVISED ACOUSTIC WORD EMBEDDINGS

Lisa van Staden *Herman Kamper*

E&E Engineering, Stellenbosch University

18245471@sun.ac.za, kamperh@sun.ac.za

2021 IEEE Spoken Language Technology Workshop (SLT) | 978-1-7281-7066-4/20/$31.00 ©2021 IEEE | DOI: 10.1109/SLT48900.2021.9383625

ABSTRACT

Many speech processing tasks involve measuring the acoustic similarity between speech segments. *Acoustic word embeddings* (AWE) allow for efficient comparisons by mapping speech segments of arbitrary duration to fixed-dimensional vectors. For zero-resource speech processing, where unlabelled speech is the only available resource, some of the best AWE approaches rely on weak top-down constraints in the form of automatically discovered word-like segments. Rather than learning embeddings at the *segment level*, another line of zero-resource research has looked at representation learning at the *short-time frame level*. Recent approaches include self-supervised predictive coding and correspondence autoencoder (CAE) models. In this paper we consider whether these frame-level features are beneficial when used as inputs for training to an unsupervised AWE model. We compare frame-level features from contrastive predictive coding (CPC), autoregressive predictive coding and a CAE to conventional MFCCs. These are used as inputs to a recurrent CAE-based AWE model. In a word discrimination task on English and Xitsonga data, all three representation learning approaches outperform MFCCs, with CPC consistently showing the biggest improvement. In cross-lingual experiments we find that CPC features trained on English can also be transferred to Xitsonga.

Index Terms— acoustic word embeddings, speech represenation learning, self-supervised learning, zero-resource speech processing, crosslingual transfer.

1. INTRODUCTION

A number of speech processing tasks rely on measuring the acoustic similarity between speech segments [1, 2]. Usually, similarity is measured using dynamic time warping (DTW), an algorithm that finds an optimal alignment between speech segments [3]. However, DTW is computationally expensive. This has led to research in methods of finding fixed-dimensional speech representations, referred to as *acoustic word embeddings* (AWEs) [4–11]. These methods attempt to capture the acoustic information in speech segments of variable length and condense it in such a way that segments containing the same words are mapped to similar embeddings. Since speech segments can then be represented in the same fixed-dimensional space, measuring the acoustic similarity can be done with a computationally inexpensive distance calculation.

Many of the downstream tasks for which AWEs are useful can be performed in a setting where transcribed speech resources are unavailable. Such tasks include query-by-example search [12–14], where a speech segment is used as a query to search over a database of speech, and unsupervised term discovery (UTD) [2, 15–17], where the aim is to discover reoccurring patterns in untranscribed speech. Speech processing in settings without any labelled speech data is referred to as *zero-resource speech processing* and it has become a popular field of research [18–21]. Apart from practical tasks, this area is also closely related to modelling language acquisition in humans, since infants acquire language without access to transcribed speech data [22–24]. A number of studies have specifically focussed on developing AWEs for this zero-resource setting [4, 6, 8, 25, 26]. Several of these unsupervised AWE approaches rely on weak top-down constraints in the form of discovered words from a UTD system. For instance, the correspondence autoencoder recurrent neural network (CAE-RNN) [8] is trained to reconstruct one segment in a discovered pair given the other segment as input; embeddings are taken from an intermediate representation between the model's encoder and decoder RNNs. By using discovered words as input-output pairs, the CAE-RNN can then be trained in the absence of any labelled speech data.

While AWE approaches attempt to model speech at the *segment* level, several zero-resource studies have focussed instead on unsupervised representation learning at the short-time *frame* level [27–32]. Here the goal is to capture meaningful contrasts such as phone categories. Many of these methods could be described as *bottom-up*, learning representations directly from the lower-level features. This includes recent self-supervised predictive coding methods [33, 34].

In this paper we investigate and compare different learned speech representations as inputs to unsupervised AWE models. For learning frame-level features, we consider three approaches. The first two are recent predictive coding methods. In contrastive predictive coding (CPC) [33], representations are learned by predicting the correct future frames from a set

This work is supported in part by the National Research Foundation of South Africa (grant number: 120409) and a Google Faculty Award for HK.

containing negative examples. In autoregressive predictive coding (APC) [34], representations are learned by predicting future frames with an autoregressive model. Finally, we consider a frame-level correspondence autoencoder (CAE) neural network [35]. We find pairs of speech segments that are predicted to be of the same type using a UTD system [15]. DTW is then used to align frames between the pair of discovered speech segments. The frame-level CAE model is then trained to predict corresponding aligned frames from each other.

All three of these representation learning approaches attempt to capture linguistically meaningful information present at the short-time frame level (such as phone categories). But they do so in very different ways. The CPC tries to discriminate between frames in an utterance from negative examples, while the APC tries to reconstruct future frames based on a past history. The frame-level CAE follows quite a different methodology: trying to reconstruct frames from aligned segments predicted to contain the same word. As with the CPC and APC approaches, the CAE can also be described as self-supervised [26], since labels for a proxy task are automatically obtained from the data [36].

The three types of learned frame-level representations are used as input features to train unsupervised CAE-RNN AWE models. We evaluate the intrinsic quality of the resulting AWEs in a word discrimination task. We find that across two languages, English and Xitsonga, all three learned representations improve upon MFCCs, with CPC producing the best results when used as input features to the AWE model. We also perform crosslingual experiments, where we find that using frame-level representation models trained on a higher resourced language (English) to encode those of a low-resource language (Xitsonga), improves the AWEs of the latter.

2. SELF-SUPERVISED FRAME-LEVEL REPRESENTATION LEARNING

2.1. Contrastive predictive coding (CPC)

The aim of contrastive predictive coding (CPC) is to encode only the information that is shared between current and future acoustic observations [33]. This results in representations that better describe shared short-time information, like phone categories or speech intonation, depending on how far ahead future observations are. The original CPC study showed that it produced effective speech representation when trained on raw audio waveforms [33]. A more recent study [37] showed that it can also be successfully applied when predicting conventional frame-level acoustic features.

Figure 1 shows the architecture for learning CPC representations. A sequence of frames is received as input to the CPC model, with a single frame at time step t denoted as $\boldsymbol{x}_t$. The input frames are encoded by a function g_{enc} into latent variables, denoted as $\boldsymbol{z}_t$ at time step t.

In our case this encoder function consists of a sequence

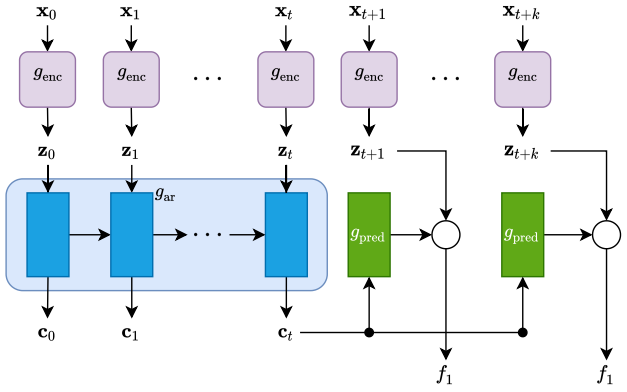

Fig. 1. The CPC model is trained to compute a score from the context variables, $\boldsymbol{c}_0,\boldsymbol{c}_1,...,\boldsymbol{c}_t$, and the future latent variables, $\boldsymbol{z}_{t+1},...,\boldsymbol{z}_{t+k}$.

of linear layers. Next, the latent variables are encoded by an autoregressive function g_{ar} into context variables $\boldsymbol{c}_t$. We use a recurrent neural network (RNN) layer for this purpose. Because this function is autoregressive it allows each $\boldsymbol{c}_t$ to be a summary of all $\boldsymbol{z}_{\leq t}$, such that $\boldsymbol{c}_t = g_{\text{ar}}(\boldsymbol{z}_{\leq t})$.

The next step is to determine a prediction score for every $\boldsymbol{c}_t$ at each prediction step. Let K be the chosen number of steps that we want to predict into the future. Then for each step $k \in [1,K]$ a function g^k_{pred} is used to transform $\boldsymbol{c}_t$. A log bilinear function is then used to calculate the score:

$$f_k(\boldsymbol{z}_t,\boldsymbol{c}_t) = \exp\left(\boldsymbol{z}_{t+k}^{\top} g^k_{\text{pred}}(\boldsymbol{c}_t)\right) \tag{1}$$

Let $\mathcal{Z}_t$ be a set that contains the true $\boldsymbol{z}_t$ along with $N-1$ negative samples of $\boldsymbol{z}_t$. We also calculate prediction scores for the negative samples. The model is then trained to maximise the score for $\boldsymbol{z}_t$ and minimise it for the negative samples. Concretely, the loss function used to do this is based on noise-contrastive estimation (NCE) [38]:

$$L_{\text{InfoNCE}}(\boldsymbol{x}_t,k) = -\log\left(\frac{f_k(\boldsymbol{z}_t,\boldsymbol{c}_t)}{\sum_{\boldsymbol{z}_i \in Z_t} f_k(\boldsymbol{z}_i,\boldsymbol{c}_t)}\right) \tag{2}$$

The final loss function applied to the CPC model for a sequence X of input frames can then be expressed as:

$$L_{\text{CPC}} = \frac{1}{K}\frac{1}{|X|} \sum_{k\in[1,K],\boldsymbol{x}_j \in X} L_{\text{InfoNCE}}(\boldsymbol{x}_j,k) \tag{3}$$

We sample negative frames from different utterances of the same speaker as the correct frames. This encourages the CPC model to normalise out speaker information, since it can't use this information to select the correct frame from among the negative examples [39].

Either $\boldsymbol{z}_t$ or $\boldsymbol{c}_t$ can be used as frame-level representations for a downstream task. But it is recommended to use $\boldsymbol{c}_t$ when extra context from the past is useful [33]. In our development experiments $\boldsymbol{c}_t$ did give better results, and we therefore use it as our input representations to the AWE models.

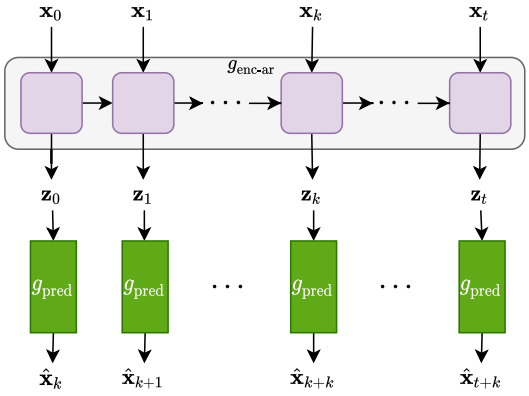

Fig. 2. The APC model is trained to predict the frame k steps ahead of the input frame from a latent variable.

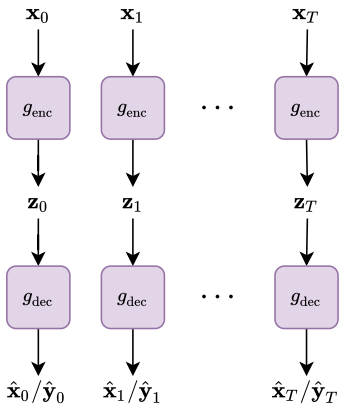

Fig. 3. The AE is trained to reconstruct the input frames $\boldsymbol{x}_0,\boldsymbol{x}_1,...,\boldsymbol{x}_T$ directly. The CAE is trained to reconstruct frames from another segment $\boldsymbol{y}_0,\boldsymbol{y}_1,...,\boldsymbol{y}_T$, predicted to be a different instance of the same word as the input.

2.2. Autoregressive predictive coding (APC)

Similarly to CPC, the aim of autoregressive predictive coding (APC) is to encode only the information that is shared between current and future frames. The original APC paper [34] argues that when learned representations are encouraged to throw out nuisance information (like speaker identity or noise) there is a risk that useful information might also be lost. So instead of encouraging the model to normalise out non-discriminative features, as with the score maximisation of CPC, an autoregressive function is used to decode the predicted future frame from a latent variable containing more general information. Figure 2 shows the APC architecture. A sequence of frames are encoded by an autoregressive function $g_{\text{enc-ar}}$. In our case the autoregressive function is a stack of RNN layers. The last layer's hidden states at each time step is then used as the latent variables, denoted as $\boldsymbol{z}_t$ for time step t. Next, a prediction function g_{pred} transforms each $\boldsymbol{z}_t$ to the predicted input frame k steps ahead, such that it can be described as $\hat{\boldsymbol{x}}_{t+k} = g_{\text{pred}}(\boldsymbol{z}_t)$. For a sequence of input frames $X = (\boldsymbol{x}_0,\boldsymbol{x}_1,...,\boldsymbol{x}_T)$ the model is trained to minimise the mean absolute error (MAE) between the true and predicted future frames:

$$L_{\text{MAE}}(X) = \frac{1}{|X|} \sum_{\boldsymbol{x}_t \in X} ||\boldsymbol{x}_{t+k} - \hat{\boldsymbol{x}}_{t+k}||_1 \tag{4}$$

A follow-up study on APC proposed adding an auxiliary loss as a regularisation term [40]. This loss encourages the latent variables to include information from previous frames in the sequence. Concretely, M different frames are chosen at random from X to use as anchors. An anchor at position m is denoted by $\boldsymbol{x}_{a_m}$. For each anchor we take a slice of X, denoted by A_m, that contains n frames that start s time steps behind a_m, such that $A_m = (\boldsymbol{x}_{a_m-s},\boldsymbol{x}_{a_m-s+1},...,\boldsymbol{x}_{a_m-s+n-1})$. Then the auxiliary loss is given as the MAE loss for every A_m:

$$L_{\text{aux}}(X) = \frac{1}{M} \sum_{A_m \in \mathcal{A}} L_{\text{MAE}}(A_m) \tag{5}$$

where $\mathcal{A}$ denotes the set that contains every A_m sequence sliced from X.

In our development experiments we found that adding the auxiliary loss does result in a small improvement for the AWEs. The final loss function for our APC model is therefore

$$L_{\text{APC}} = L_{\text{MAE}} + \lambda L_{\text{aux}} \tag{6}$$

with λ a hyper-parameter.

The hidden states from any of the layers of $g_{\text{enc-ar}}$ can be used as frame representations for downstream tasks. Previous research on autoregressive textual word embedding models showed that the information in hidden states are hierarchical across across layers [41]. The original APC study [34] concluded that earlier layers in the autoregressive function contain more speaker information and later layers more phonetic information. Therefore we use the hidden states of the last layer $\boldsymbol{z}_t$ as speech representations in our AWE experiments.

2.3. Frame-level correspondence autoencoder (CAE)

Unlike with the predictive coding models, the correspondence autoencoder (CAE) model has no autoregressive component [35]. Therefore the model does not encode information that is shared temporally. The model is rather encouraged to encode only the information that is shared between frames from different instances of the same (predicted) word, as outlined detail below. The intuition is that this encourages the model to normalise out noise and speaker information, since these properties could be different for the input and output frames.

The CAE model produces the best results if its weights are initialised with those of a trained autoencoder (AE) [31, 35]. The architectures of these two models are similar and are shown in Figure 3.

The AE takes a frame $\boldsymbol{x}_t$ as input. A function $g_{\text{enc}}^{(\text{AE})}$ encodes the input into a latent variable $\boldsymbol{z}_t$. This latent variable is then

decoded by the function $g_{\text{dec}}^{(\text{AE})}$, yielding the model output $\hat{\boldsymbol{x}}_t = g_{\text{dec}}^{(\text{AE})}(\boldsymbol{z}_t)$. The target for this output is the input frame itself (from there the $\hat{\boldsymbol{x}}_t$ notation). For a batch of input frames X the AE is trained to minimise the mean square error (MSE) between the input and predicted frames.

$$L_{\text{AE}} = \frac{1}{|X|} \sum_{\boldsymbol{x}_t \in X} ||\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t||_2^2 \tag{7}$$

For the CAE model, we first have to create pairs of similar frames. Given an unlabelled speech collection, we use a UTD system to find speech segments which are predicted to be of the same unknown type [2]. Pairs of discovered word segments are then aligned using DTW, producing input-output frame pairs for the CAE. Since the segments may differ in length, some frames could be paired with multiple other frames.

The architecture of the CAE model is the same as that of the AE model, but instead of decoding $\boldsymbol{z}_t$ in order to predict the input frame itself, we use it to predict the corresponding frame in the pair. Formally, for an input-output pair $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, the model takes $\boldsymbol{x}_t$ as input, produces the latent representation $\boldsymbol{z}_t = g_{\text{enc}}^{(\text{CAE})}(\boldsymbol{z}_t)$, and then decodes $\boldsymbol{z}_t$ to obtain the model output $\hat{\boldsymbol{y}}_t = g_{\text{dec}}^{(\text{CAE})}(\boldsymbol{z}_t)$. For a batch of input-output frame pairs Y, the model is trained to minimise the mean square error (MSE) between the input and output frames.

$$L_{\text{CAE}} = \frac{1}{|Y|} \sum_{(\boldsymbol{x}_t, \boldsymbol{y}_t) \in Y} ||\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t||_2^2 \tag{8}$$

We use the latent variables $\boldsymbol{z}_t$ from the CAE model as the representations in our AWEs.

3. ACOUSTIC WORD EMBEDDING MODEL

Above we introduced three frame-level representation learning methods. We will use each of these methods to produce input features for an unsupervised AWE model, and then compare the results. The idea is that the resulting AWE model would be better able to discriminate at the segment (word) level by taking advantage of features learned at the frame level.

Concretely, we will use the correspondence autoencoder recurrent neural network (CAE-RNN) AWE model of [8]. Note that, although they are related, this AWE model is different from the frame-level CAE of Section 2.3. The unsupervised CAE-RNN was shown to give comparable or slightly better performance compared to a DTW approach [8], making it one of the best unsupervised AWE models.

The weights of the CAE-RNN are initialised with those of a autoencoder recurrent neural network (AE-RNN) [6]. Both models are based on encoder-decoder model architecture [42, 43], as illustrated in Figure 4. The encoder and decoder each consists of a stack of RNN layers. The encoder maps an input sequence X of variable length into a fixed-dimensional latent variable $\boldsymbol{z}$. This latent variable could be the last hidden state of the last encoder RNN layer, but in our case we add a linear layer after the encoder to transform the last hidden state into $\boldsymbol{z}$. We use these latent variables $\boldsymbol{z}$ as acoustic embeddings. The decoder then maps $\boldsymbol{z}$ to an output sequence, denoted by $\hat{X}$ for the AE-RNN and $\hat{Y}$ for the CAE-RNN.

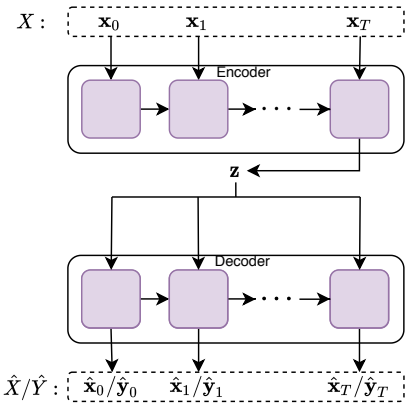

Fig. 4. The AE-RNN is trained to reconstruct the input sequence $X = (\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_T)$ from a latent variable $\boldsymbol{z}$. The CAE-RNN is trained to reconstruct a different sequence $Y = (\boldsymbol{y}_0, \boldsymbol{y}_1, ..., \boldsymbol{y}_T)$ predicted to contain the same word as the input X.

The AE-RNN is trained so that $\hat{X}$ gives a reconstruction of the original input sequence [6]. We do this by minimising the MSE between the true and reconstructed sequences:

$$L_{\text{AE-RNN}} = \frac{1}{|X|} ||X - \hat{X}||_{2,1}^2 \tag{9}$$

Instead of reconstructing the input sequence, the CAE-RNN is trained to reconstruct another instance of the same (predicted) word as the input sequence [8]. Since our training data is unlabelled, we again (like with the frame-level CAE in Section 2.3), use a UTD system to automatically discover speech segments which are predicted to be of the same type. For a given pair of sequences (X, Y), the CAE-RNN is fed with input X and then trained to reconstruct Y at its output. We do this by minimising the MSE between the true sequence Y and the predicted output $\hat{Y}$.

$$L_{\text{CAE-RNN}} = \frac{1}{|Y|} ||Y - \hat{Y}||_{2,1}^2 \tag{10}$$

The intuition behind the CAE-RNN is that the model learns to only encode information that is shared between the input-output segments (such as the word identity) while throwing out nuisance information. This is similar to the idea behind the frame-level CAE (Section 2.3) but here we use whole segments containing a sequence of frames instead of single frames.

4. EXPERIMENTAL SETUP AND EVALUATION

4.1. Data

As in [8], we train our models on datasets from two languages: English, from the Buckeye corpus [44], and Xitsonga, from

the NCHLT corpus [45]. The English training, validation and test sets each contain around six hours of speech. For Xitsonga, we have a single set of 2.5 hours. For both the frame-level CAE (Section 2.3) and segment-level CAE-RNN (Section 3), we use the UTD system of [15] to discover pairs of speech segments. In the English training set, 14k unique pairs are discovered. In the Xitsonga set, around 6k unique pairs are discovered. For the CPC model we assume that speaker labels are available. As in [8, 19], no validation data is available for Xitsonga; we therefore perform all development experiments on the English validation data and then use exactly the same hyperparameters on the Xitsonga set, replicating a true zero-resource setting.

All speech audio are transformed to 13-dimensional Mel-frequency cepstral coefficients (MFCCs). These are used as input to the predictive coding feature learning models (Sections 2.1 and 2.2). For the frame-level CAE (Section 2.3), it was found beneficial to additionally include velocity and acceleration coefficients (39-dimensional input).

4.2. Representation learning model implementations

For our CPC model (Section 2.1), we use an encoder of six 512-unit linear layers with layer normalisation and ReLU activation functions in between. A dropout layer with a rate of 0.5 is added after the third ReLU activation. In development experiments we found that the dropout layer does not improve results, but it does stabilise training. We choose a long-short term memory (LSTM) layer as our summarising autoregressive function [46]. The dimensions of $\boldsymbol{z}_t$ and $\boldsymbol{c}_t$ are 64 and 356, respectively. For the contrastive loss in (2), based on validation experiments, we choose 31 negative examples from the same batch and we predict three steps ahead. The model is trained with a learning rate of $1 \cdot 10^{-5}$. All our neural networks are optimised using Adam [47]. Each batch contains nine utterances from nine different speakers. We train the English model for a maximum of 15k epochs, but stop at the epoch that produced the best results on validation data. We find that this happens when the model is at a training loss of around 0.93, and so we train the Xitsonga model until it reaches this loss value.

The autoregressive encoder of our APC model (Section 2.2) consists of a stack of three gated recurrent unit (GRU) [43] layers with a hidden state dimensionality of 512, which is thus also the dimensionality of $\boldsymbol{z}_t$. The predictor function is one linear layer which, based on validation experiments, predicts two steps ahead. For the auxiliary loss (5), we choose twelve anchors that we use to create sequences of seven frames from 14 steps back and we predict five steps ahead. We use an auxiliary loss weight of $\lambda = 0.1$. In development experiments we found the number of epochs that produces the best AWEs on the English validation data to be 50, and also use this many epochs on the Xitsonga data. We use a learning rate of $1 \cdot 10^{-3}$.

We set up our frame-level AE and CAE models (Section 2.3) as in [31]. The encoder and decoder functions both consist of six 100-unit linear layers with a 39-dimensional latent variable in between. Through development experimentation we found that on the English datasets the best results are achieved if the AE model is trained for five epochs and the CAE for ten epochs, and therefore again do the same on Xitsonga. A learning rate of $1 \cdot 10^{-3}$ is used for both models.

4.3. Unsupervised AWE model implementation

We compare MFCCs, CPC, APC and CAE representations (Section 2) as input features to the unsupervised CAE-RNN AWE model (Section 3). This model is pre-trained as an AE-RNN using (9) before switching to the CAE-RNN loss of (10). We follow the model setup of [8]. The dimensionality of the embeddings is set to 130. The encoder and decoder functions each consist of a stack of three GRUs with a hidden state dimensionalityof 512. We use learning rates of $1 \cdot 10^{-3}$ and $1 \cdot 10^{-4}$ for the AE-RNN and CAE-RNN, respectively, and both models use a batch size of 256. For the English dataset, we train the AE-RNN for 150 epochs and the CAE-RNN for 25 epochs and use early-stopping on validation data. For the Xitsonga dataset, we do not have validation data, so we average the number of epochs that it takes to produce the best AWEs on the English validation data for each of the different types of input representations.

4.4. Evaluation

We use the same-different task to evaluate the intrinsic quality of the AWE models [48]. This task works as follows. First an evaluation set of isolated words is embedded using a particular AWE model. A decision of whether two embedded segments contain the same or different words can then be made based on the distance between the embeddings. By varying a threshold, we create a curve of precision versus recall measuring correct predictions. The average precision (AP) is the area under this curve. We use AP to measure the quality of the AWEs, where higher scores are better.

As an AWE baseline, we use downsampling [4, 11]. In this method, we choose ten equally spaced frame representations from a sequence and interpolate them to form an AWE. We obtain downsampled AWEs from each of the representation learning methods considered. As an additional way of performing the same-different task, we also consider using DTW over the speech segments, each segment represented using the frame-level representation under consideration. This approach has access to the full sequences without any compression.

All experiments are repeated three times. We report averaged AP scores along with the standard deviations.

5. EXPERIMENTS

Our main research question is whether improved self-supervised frame-level feature learning is beneficial when used in combination with a segment-level model for producing AWEs. We therefore compare MFCC, CPC, APC and CAE

Table 1. AP (%) results on the English and Xitsonga test data for DTW and the CAE-RNN and downsampling AWE approaches. On the right is the crosslingual AP (%) results obtained by training the frame-level representations on English before applying it to the Xitsonga data to train a CAE-RNN AWE model.

	English			Xistonga			Crosslingual
	DTW	Downsampled	CAE-RNN	DTW	Downsampled	CAE-RNN	CAE-RNN
MFCC	35.90	19.40	30.18±0.34	28.15	18.36	22.52±0.29	–
CPC	16.03±0.03	25.38±0.48	**36.83±0.92**	7.65±0.32	18.66±1.24	**40.93±0.77**	**41.79±0.60**
APC	30.68±0.26	20.48±0.08	33.55±1.03	18.65±0.42	16.16±0.22	38.96±0.74	40.07±1.13
CAE	41.49±1.97	21.27±1.51	31.31±1.17	48.32±1.96	26.01±0.33	29.61±3.21	34.25±1.61

features when used as input representations to an CAE-RNN model. For comparison, we also use these frame-level features in downsampled AWEs as well as a direct DTW approach. Additionally, we are interested to see if the learned representations can be used across languages. This is related to previous studies applying frame-level features learned on one language to another, e.g. [49–51]. But here we are specifically interested in the resulting AWEs, which have not been considered before. We train the frame-level representations on English data and then use the trained models to encode the Xitsonga data is then fed to the CAE-RNN.

Table 1 shows the AP scores of the English, Xistonga and crosslingual test experiments. First focusing only on the AWE results of the English and Xitsonga experiments (downsampling and CAE-RNN columns), we see that in both cases all the learned representations improve upon MFCCs. In both AWE approaches, best results are consistently achieved when using the CPC representations. Overall, the best AWE approach is the CAE-RNN taking in CPC representations as input, outperforming the downsampled CPC AWEs by more than 10% and 20% absolute in AP for the English and Xitsonga data, respectively.

Somewhat surprisingly, when the features are used directly to do the same-different task (DTW columns), the only features to outperform MFCCs, for both languages, are the frame-level CAE features. Moreover, for both the CPC and APC features, the corresponding CAE-RNN actually outperforms its DTW counterpart (e.g. 16.03% vs. 36.83% for the English CPC features and 38.96% vs. 18.65% for the Xitsonga APC features). This is despite DTW having access to the full sequences while the CAE-RNN needs to compress the sequences into an AWE. One reason for this could be that the weak top-down constraints used in the frame-level CAE are the same as those used in the CAE-RNN model (obtained from the UTD system), and therefore does not provide any additional signal. In contrast, the self-supervision signal for the CPC and APC models are obtained in a bottom-up fashion which is different from that of the CAE-RNN—the top-down signal used in the CAE-RNN seems to be complementary to the bottom-up approach of CPC and APC. But these are speculations and further investigation is required.

The right-most column shows the cross-lingual test results of the resulting AWEs. Again, the CPC features perform best.

Table 2. Speaker classification accuracy (%) results of the English development AWEs

	Speaker acc. (%)
MFCC	64.05±3.89
CPC	57.83±1.65
APC	60.76±3.20
CAE	51.43±1.95

Surprisingly, the representations learned on English perform better than using those trained on Xitsonga. The reason for this is likely due to the English dataset containing more speech data. There is therefore potential for even larger improvements by using more substantial amounts of unlabelled data, as e.g. in [52].

Finally, we consider a speaker information probing task on the English development data. A linear classifier is used to predict the speaker of the final AWEs produced by the CAE-RNN for each type of frame-level feature. The speaker classification results, seen in Table 2, show that all the learned representations lead to reduced speaker information in the AWEs compared to those produced by the MFCCs, with the CAE representations leading to the biggest reduction.

6. CONCLUSION

We considered how unsupervised acoustic word embedding (AWE) models can be improved by taking in frame-level features from self-supervised approaches. Concretely, we compared contrastive predictive coding (CPC), autoregressive coding and a correspondence autoencoder (CAE) when used as input to a recurrent CAE-based AWE model. In a word discrimination task on two languages CPC features outperformed MFCCs as well as the other learned representations.

However, different trends were observed when using the features to perform the task directly using DTW: in this case, CPC performed worst while the frame-level CAE performed best. Future work will investigate this discrepancy. This might also be related to recent work [26] showing the limitations of the same-different task as an intrinsic AWE metric. Future work will therefore also consider other downstream tasks.

7. REFERENCES

[1] S. Settle, K. Levin, H. Kamper, and K. Livescu, "Query-by-example search with discriminative neural acoustic word embeddings," in *Proc. Interspeech*, 2017.

[2] A. S. Park and J. R. Glass, "Unsupervised pattern discovery in speech," *IEEE Trans., Audio, Speech, Language Process.*, vol. 16, no. 1, pp. 186–197, 2008.

[3] L. Rabiner, A. Rosenberg, and S. Levinson, "Considerations in dynamic time warping algorithms for discrete word recognition," *IEEE Trans., Audio, Speech, Language Process.*, vol. 26, no. 6, pp. 575–582, 1978.

[4] K. Levin, K. Henry, A. Jansen, and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *Proc. ASRU*, 2013.

[5] S. Bengio and G. Heigold, "Word embeddings for speech recognition," in *Proc. Interspeech*, 2014.

[6] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H.-Y. Lee, and L.-S. Lee, "Audio Word2Vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder," in *Proc. Interspeech*, 2016.

[7] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," in *Proc. ICASSP*, 2016.

[8] H. Kamper, "Truly unsupervised acoustic word embeddings using weak top-down constraints in encoder-decoder models," in *Proc. ICASSP*, 2019.

[9] A. L. Maas, S. D. Miller, T. M. O'Neil, A. Y. Ng, and P. Nguyen, "Word-level acoustic modeling with convolutional vector regression," in *Proc. ICML Workshop Representation Learn.*, 2012.

[10] Y.-A. Chung and J. Glass, "Speech2Vec: A sequence-to-sequence framework for learning word embeddings from speech," in *Proc. Interspeech*, 2018.

[11] N. Holzenberger, M. Du, J. Karadayi, R. Riad, and E. Dupoux, "Learning word embeddings: unsupervised methods for fixed-size representations of variable-length speech segments," in *Proc. Interspeech*, 2018.

[12] K. Levin, A. Jansen, and B. Van Durme, "Segmental acoustic indexing for zero resource keyword search," in *Proc. ICASSP*, 2015.

[13] S.-F. Huang, Y.-C. Chen, H.-Y. Lee, and L.-s. Lee, "Improved audio embeddings by adjacency-based clustering with applications in spoken term detection," *arXiv preprint arXiv:1811.02775*, 2018.

[14] Y. Yuan, C.-C. Leung, L. Xie, H. Chen, B. Ma, and H. Li, "Learning acoustic word embeddings with temporal context for query-by-example speech search," in *Proc. Interspeech*, 2018.

[15] A. Jansen and B. Van Durme, "Efficient spoken term discovery using randomized algorithms," in *Proc. ASRU*, 2011.

[16] L. Ondel, H. K. Vydana, L. Burget, and J. Černockỳ, "Bayesian subspace hidden markov model for acoustic unit discovery," *arXiv preprint arXiv:1904.03876*, 2019.

[17] O. Räsänen and M. A. C. Blandón, "Unsupervised discovery of recurring speech patterns using probabilistic adaptive metrics," *arXiv preprint arXiv:2008.00731*, 2020.

[18] A. Jansen, E. Dupoux, S. Goldwater, M. Johnson, S. Khudanpur, K. Church, N. Feldman, H. Hermansky, F. Metze, R. Rose, M. Seltzer, P. Clark, I. McGraw, B. Varadarajan, E. Bennett, B. Borschinger, J. Chiu, E. Dunbar, A. Fourtassi, D. Harwath, C. Lee, K. Levin, A. Norouzian, V. Peddinti, R. Richardson, T. Schatz, and S. Thomas, "A summary of the 2012 JHU CLSP workshop on zero resource speech technologies and models of early language acquisition," in *Proc. ICASSP*, 2013.

[19] M. Versteegh, X. Anguera, A. Jansen, and E. Dupoux, "The zero resource speech challenge 2015: Proposed approaches and results," in *Proc. SLTU*, 2016.

[20] E. Dunbar, X.-N. Cao Kam, J. Benjumea, J. Karadayi, M. Bernard, L. Besacier, X. Anguera, and E. Dupoux, "The zero resource speech challenge 2017," in *Proc. ASRU*, 2017.

[21] E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic, C. Dugrain, L. Ondel, A. Black, L. Besacier, S. Sakti, and E. Dupoux, "The zero resource speech challenge 2019: TTS without T," in *Proc. Interspeech*, 2019.

[22] O. Räsänen, "Computational modeling of phonetic and lexical learning in early language acquisition: Existing models and future directions," *Speech Commun.*, vol. 54, no. 9, pp. 975 – 997, 2012.

[23] M. Elsner and C. Shain, "Speech segmentation with a neural encoder model of working memory," in *Proc. EMNLP*, 2017.

[24] Y. Matusevych, T. Schatz, H. Kamper, N. H. Feldman, and S. Goldwater, "Evaluating computational models of infant phonetic learning across languages," in *Proc. CogSci*, 2020.

[25] L. van Staden and H. Kamper, "Improving unsupervised acoustic word embeddings using speaker and gender information," in *Proc. SAUPEC/RobMech/PRASA*, 2020.

[26] R. Algayres, M. Zaiem, B. Sagot, and E. Dupoux, "Evaluating the reliability of acoustic speech embeddings," in *Interspeech*, 2020.

[27] L. Badino, A. Mereta, and L. Rosasco, "Discovering discrete subword units with binarized autoencoders and hidden-Markov-model encoders," in *Proc. Interspeech*, 2015.

[28] M. Heck, S. Sakti, and S. Nakamura, "Learning supervised feature transformations on zero resources for improved acoustic unit discovery," *IEICE T. Inf. Syst.*, vol. 101, no. 1, pp. 205–214, 2018.

[29] T. Tsuchiya, N. Tawara, T. Ogawa, and T. Kobayashi, "Speaker invariant feature extraction for zero-resource languages with adversarial learning," in *Proc. ICASSP*, 2018.

[30] R. Riad, C. Dancette, J. Karadayi, N. Zeghidour, T. Schatz, and E. Dupoux, "Sampling strategies in Siamese networks for unsupervised speech representation learning," in *Proc. Interspeech*, 2018.

[31] P.-J. Last, H. A. Engelbrecht, and H. Kamper, "Unsupervised feature learning for speech using correspondence and siamese networks," *IEEE Signal Proc. Let.*, vol. 27, pp. 421–425, 2020.

[32] B. van Niekerk, L. Nortje, and H. Kamper, "Vector-quantized neural networks for acoustic unit discovery in the zerospeech 2020 challenge," in *Proc. Interspeech*, 2020.

[33] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[34] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. R. Glass, "An unsupervised autoregressive model for speech representation learning," in *Proc. Interspeech*, 2019.

[35] H. Kamper, M. Elsner, A. Jansen, and S. Goldwater, "Unsupervised neural network based feature extraction using weak top-down constraints," in *Proc. ICASSP*, 2015.

[36] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," in *Proc. ICCV*, 2017.

[37] M. A. C. Blandón and O. Räsänen, "Analysis of predictive coding models for phonemic representation learning in small datasets," in *Proc. ICML Workshop Self-Supervision Audio Speech*, 2020.

[38] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proc. AISTATS*, 2010.

[39] B. van Niekerk, L. Nortje, and H. Kamper, "Vector-quantized neural networks for acoustic unit discovery in the ZeroSpeech 2020 challenge," *arXiv preprint: arXiv:2005.09409*, 2020.

[40] Y.-A. Chung and J. Glass, "Improved speech representations with multi-target autoregressive predictive coding," in *Proc. ACL*, 2020.

[41] M. Peters, M. Neumann, L. Zettlemoyer, and W.-t. Yih, "Dissecting contextual word embeddings: Architecture and representation," in *Proc. EMNLP*, 2018.

[42] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014.

[43] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN Encoder–Decoder for statistical machine translation," in *Proc. EMNLP*, 2014.

[44] M. A. Pitt, K. Johnson, E. Hume, S. Kiesling, and W. Raymond, "The Buckeye corpus of conversational speech: labeling conventions and a test of transcriber reliability," *Speech Commun.*, vol. 45, no. 1, pp. 89–95, 2005.

[45] N. Vries, M. Davel, J. Badenhorst, W. Basson, E. Barnard, and A. de Waal, "A smartphone-based ASR data collection tool for under-resourced languages," *Speech Commun.*, vol. 56, pp. 119–131, 2014.

[46] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–80, 1997.

[47] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2014.

[48] M. A. Carlin, S. Thomas, A. Jansen, and H. Hermansky, "Rapid evaluation of speech representations for spoken term discovery," in *Proc. Interspeech*, 2011.

[49] E. Hermann and S. J. Goldwater, "Multilingual bottleneck features for subword modeling in zero-resource languages," in *Proc. Interspeech*, 2018.

[50] A. Conneau, A. Baevski, R. Collobert, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition," *arXiv preprint: arXiv:2006.13979*, 2020.

[51] M. Rivière, A. Joulin, P.-E. Mazaré, and E. Dupoux, "Unsupervised pretraining transfers well across languages," in *Proc. ICASSP*, 2020.

[52] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.