DL-NLP Reading Group: SELF-SUPERVISED PRE-TRAINING FOR SPEECH RECOGNITION

Muhammad Abdul-Mageed

muhammad.mageed@ubc.ca

Natural Language Processing Lab

The University of British Columbia

WAV2VEC: UNSUPERVISED PRE-TRAINING FOR SPEECH RECOGNITION

Steffen Schneider, Alexei Baevski, Ronan Collobert, Michael Auli
Facebook AI Research

ABSTRACT

We explore unsupervised pre-training for speech recognition by learning representations of raw audio. wav2vec is trained on large amounts of unlabeled audio data and the resulting representations are then used to improve acoustic model training. We pre-train a simple multi-layer convolutional neural network optimized via a noise contrastive binary classification task. Our experiments on WSJ reduce WER of a strong character-based log-mel filterbank baseline by up to 36 % when only a few hours of transcribed data is available. Our approach achieves 2.43 % WER on the nov92 test set. This outperforms Deep Speech 2, the best reported character-based system in the literature while using two orders of magnitude less labeled training data.[1]

1 INTRODUCTION

Current state of the art models for speech recognition require large amounts of transcribed audio data to attain good performance (Amodei et al., 2016). Recently, pre-training of neural networks

Pre-Training

Pre-Training is Useful...

- Speech recognition SOTA requires large data (Amodei et al., 2016).
- **Pre-training** (What is it?) is effective, especially for low-resource settings.
- In speech processing, pre-training focused on **emotion recognition** (Lian et al., 2018), **speaker identification** (Ravanelli & Bengio, 2018), **phoneme discrimination** (Synnaeve & Dupoux, 2016a; van den Oord et al., 2018), and **transferring ASR representations from one language to another** (Kunze et al., 2017).
- **Goal:** Applying unsupervised pre-training to improve supervised speech recognition.

What is WAV2VEC?

WAV2VEC

- A CNN that takes raw audio as input and **computes a general representation** that can be input to a speech recognition system.
- **Objective function:** a **contrastive loss** that requires distinguishing a true future audio sample from negatives

How Good?

Experimental Results

- Results on the WSJ benchmark estimated on about 1000 hours of unlabeled speech: **substantially improves a character-based ASR system** and outperform the best character-based result in the literature, Deep Speech 2.
- Improves WER from 3.1% to 2.43%.
- On TIMIT, **matches SOTA**.
- In a simulated low-resource setup with **only eight hours of transcribed audio data**: reduces WER by up to 36% compared to a baseline relying on labeled data only.

Approach

A Speech Model

- **Modelling the data distribution $p(x)$ is challenging**
- Instead, **first encodes raw speech samples into a feature representation** at a lower temporal frequency
- Then **implicitly model a density ratio** $p(z_{i+k}|z_i \ldots z_{i-r})/p(z_{i+k})$ (similar to van den Oord et al., 2018).
- Abstractly, similar to estimating N-grams: $p(w_2|w_1)/p(w_2)$

Model

Model Overview

- Model **takes raw audio signal as input and then applies two networks**
- **The Encoder network:** embeds the audio signal in a latent space.
- **The context network:** combines multiple time-steps of the encoder to obtain contextualized representations.
- **Both networks are then used to compute the objective function.**

Model Illustration

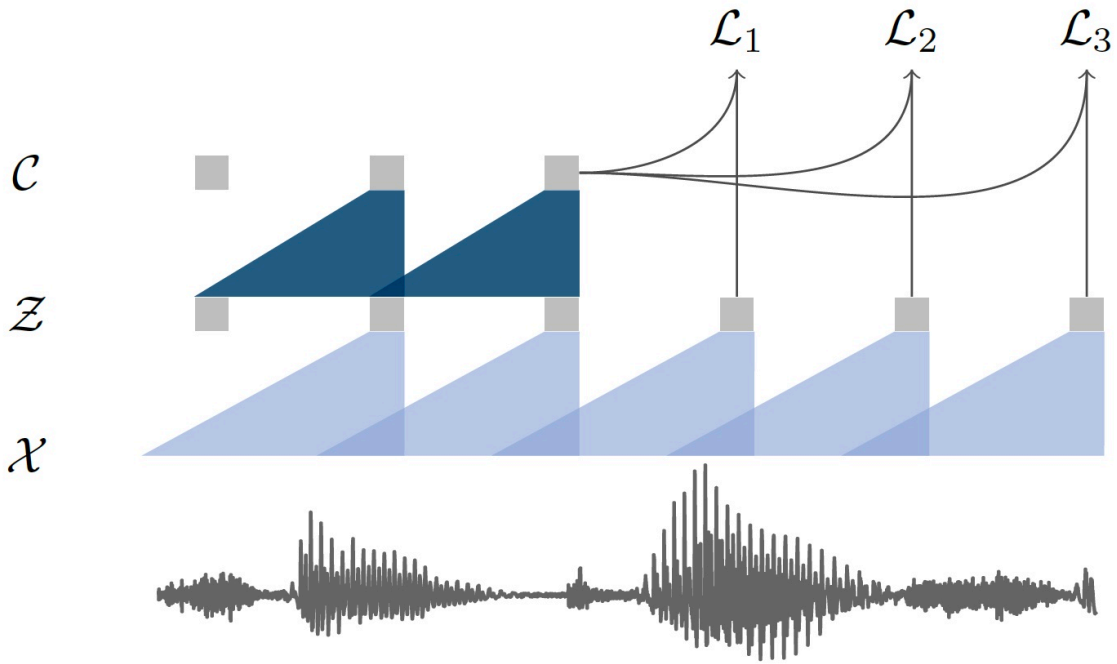


Figure 1: Illustration of pre-training from audio data $\mathcal{X}$ which is encoded with two convolutio neural networks that are stacked on top of each other. The model is optimized to solve a next ti step prediction task.

Encoder

Given raw audio samples $\mathbf{x}_i \in \mathcal{X}$, we apply the *encoder* network $f : \mathcal{X} \mapsto \mathcal{Z}$ parameterized as a five-layer convolutional network (van den Oord et al., 2018). Alternatively, one could use other architectures such as the trainable frontend of Zeghidour et al. (2018a). The encoder layers have kernel sizes $(10, 8, 4, 4, 4)$ and strides $(5, 4, 2, 2, 2)$. The output of the encoder is a low frequency feature representation $\mathbf{z}_i \in \mathcal{Z}$ which encodes about 30 ms of 16 kHz of audio and the striding results in representations $\mathbf{z}_i$ every 10ms.

Context Network

Next, we apply the *context* network $g : \mathcal{Z} \mapsto \mathcal{C}$ to the output of the encoder network to mix multiple latent representations $\mathbf{z}_i \dots \mathbf{z}_{i-v}$ into a single contextualized tensor $\mathbf{c}_i = g(\mathbf{z}_i \dots \mathbf{z}_{i-v})$ for a receptive field size v. The context network has nine layers with kernel size three and stride one. The total receptive field of the context network is about 210 ms.

Objective

We train the model to distinguish a sample $\mathbf{z}_{i+k}$ that is k steps in the future from distractor samples $\tilde{\mathbf{z}}$ drawn from a proposal distribution p_n, by minimizing the contrastive loss for each step $k = 1, \ldots, K$:

True samples ↓ negative samples ↓

$$\mathcal{L}_k = -\sum_{i=1}^{T-k} \left(\log \sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i)) + \lambda \mathop{\mathbb{E}}_{\tilde{\mathbf{z}} \sim p_n} [\log \sigma(-\tilde{\mathbf{z}}^\top h_k(\mathbf{c}_i))] \right) \quad (1)$$

where we denote the sigmoid $\sigma(x) = 1/(1+\exp(-x))$, and where $\sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i))$ is the probability of $\mathbf{z}_{i+k}$ being the true sample. We consider a step-specific affine transformation $h_k(\mathbf{c}_i) = W_k \mathbf{c}_i + \mathbf{b}_k$ for each step k, that is applied to $\mathbf{c}_i$ (van den Oord et al., 2018). We optimize the loss $\mathcal{L} = \sum_{k=1}^{K} \mathcal{L}_k$, summing (1) over different step sizes. In practice, we approximate the expectation by sampling ten negatives examples by uniformly choosing distractors from each audio sequence, i.e., $p_n(\mathbf{z}) = \frac{1}{T}$, where T is the sequence length and we set λ to the number of negatives.[2]

After training, we input the representations $\mathbf{c}_i$ produced by the context network to the acoustic model instead of log-mel filterbank features.

Figure: After training, **representations produced by the context network** c_i are fed to the **acoustic model** instead of log-mel filterbank features.

VQ-WAV2VEC

VQ-WAV2VEC: SELF-SUPERVISED LEARNING OF DISCRETE SPEECH REPRESENTATIONS

Alexei Baevski[*,△] **Steffen Schneider**[*,▽,†] **Michael Auli**[△]
[△] Facebook AI Research, Menlo Park, CA, USA
[▽] University of Tübingen, Germany

ABSTRACT

We propose vq-wav2vec to learn discrete representations of audio segments through a wav2vec-style self-supervised context prediction task. The algorithm uses either a Gumbel-Softmax or online k-means clustering to quantize the dense representations. Discretization enables the direct application of algorithms from the NLP community which require discrete inputs. Experiments show that BERT pre-training achieves a new state of the art on TIMIT phoneme classification and WSJ speech recognition.[1]

1 INTRODUCTION

Learning discrete representations of speech has gathered much recent interest (Versteegh et al., 2016; Dunbar et al., 2019). A popular approach to discover discrete units is via autoencoding (Tjandra et al., 2019; Eloff et al., 2019; Chorowski et al., 2019) sometimes coupled with an autoregressive model (Chung et al., 2019). Another line of research is to learn continuous speech representations in a self-supervised way via predicting context information (Chung & Glass, 2018; van den Oord

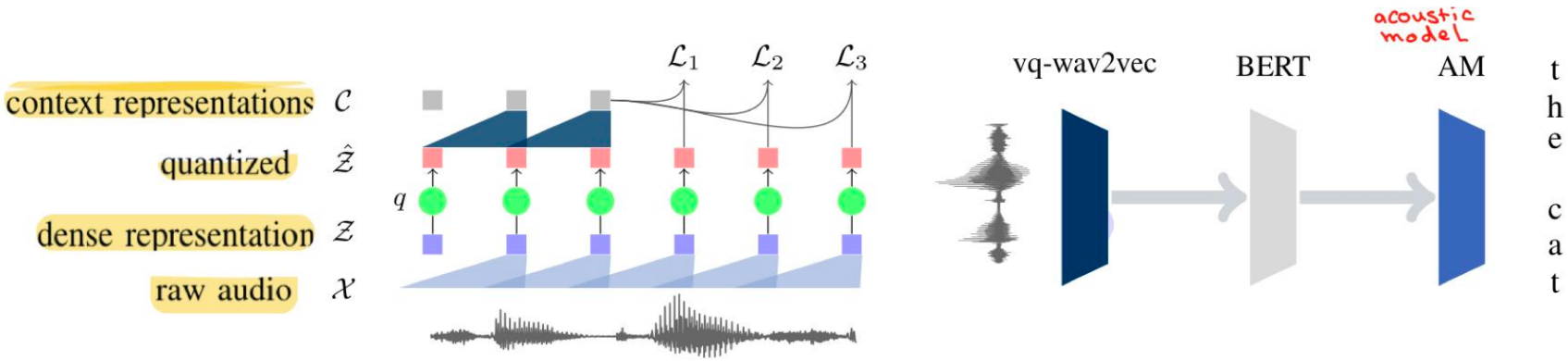
Approach

Model Overview

- **Learns vector quantized (VQ) representations** of aduio data using a **future step prediction task**
- Like wav2vec, uses two CNNs one as **encoder** and another as **context**, yet with a **quantization module in between**
- Quantization: **(a) Gumble-Softmax** and (b) **K-means clustering**
- **Encoder:** Does feature extraction and aggregation.
- **Context net:** Combines multiple encoder time-steps to obtain *contextualized representations*

Our approach, vq-wav2vec, learns vector quantized (VQ) representations of audio data using a future time-step prediction task. We follow the same architectual choices as wav2vec (§2.1) with two convolutional networks $f : \mathcal{X} \mapsto \mathcal{Z}$ and $g : \hat{\mathcal{Z}} \mapsto \mathcal{C}$ for feature extraction and aggregation, as well as a new *quantization* module $q : \mathcal{Z} \mapsto \hat{\mathcal{Z}}$ to build discrete representations (Figure 1a).

VQ-WAV2VEC Illustrated

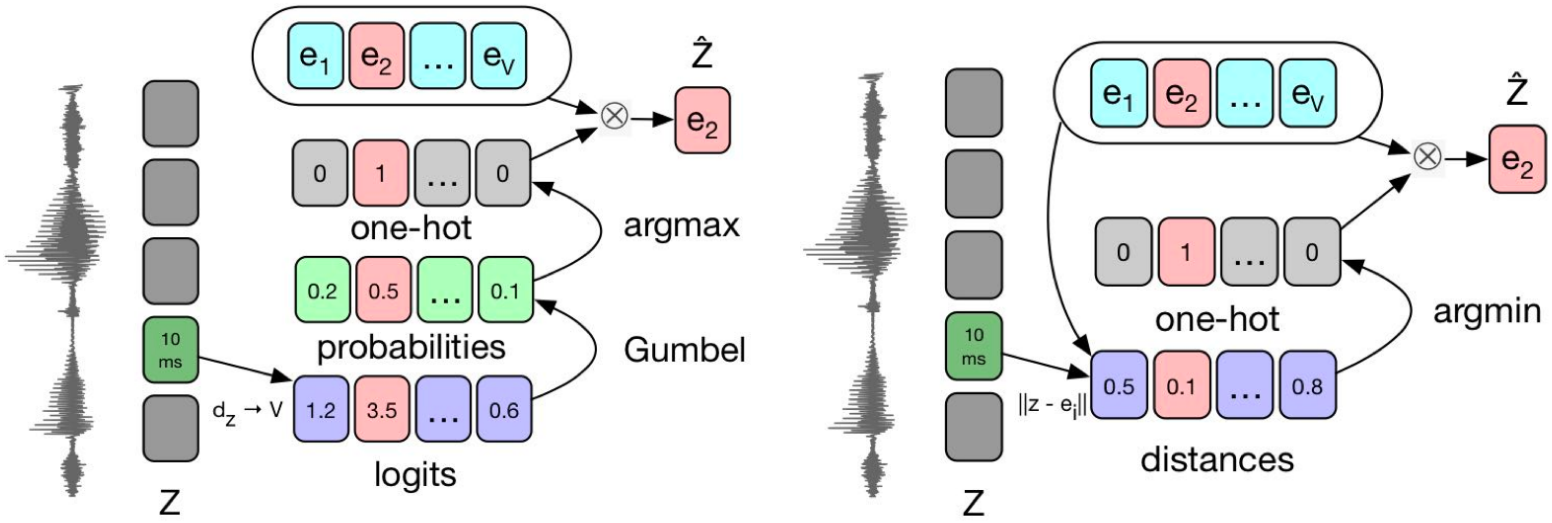


(a) vq-wav2vec

(b) Discretized speech training pipeline

Figure 1: (a) The vq-wav2vec encoder maps raw audio ($\mathcal{X}$) to a dense representation ($\mathcal{Z}$) which is quantized (q) to $\hat{\mathcal{Z}}$ and aggregated into context representations ($\mathcal{C}$); training requires future time step prediction. (b) Acoustic models are trained by quantizing the raw audio with vq-wav2vec, then applying BERT to the discretized sequence and feeding the resulting representations into the acoustic model to output transcriptions.

Discretization of Variables

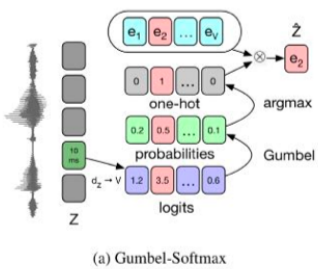


(a) Gumbel-Softmax

(b) K-means clustering.

Figure 2: (a) The Gumbel-Softmax quantization computes logits representing the codebook vectors ($\mathbf{e}$). In the forward pass the argmax codeword ($\mathbf{e}_2$) is chosen and for backward (not shown) the exact probabilities are used. (b) K-means vector quantization computes the distance to all codeword vector and chooses the closest (argmin).

Gumble-Softmax



(a) Gumbel-Softmax

The Gumbel-Softmax (Gumbel, 1954; Jang et al., 2016; Maddison et al., 2014) enables selecting discrete codebook variables in a fully differentiable way and we use the straight-through estimator of Jang et al. (2016). Given the dense representation $\mathbf{z}$, we apply a linear layer, followed by a ReLU and another linear which outputs $\mathbf{l} \in \mathbb{R}^V$ logits for the Gumbel-Softmax. At inference, we simply pick the largest index in l. At training, the output probabilities for choosing the j-th variable are

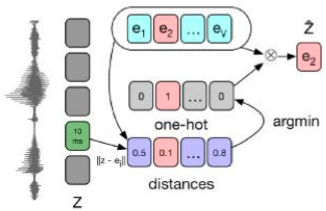
$$p_j = \frac{\exp(l_j + v_j)/\tau}{\sum_{k=1}^{V} \exp(l_k + v_k)/\tau}, \tag{2}$$

where $v = -\log(-\log(u))$ and u are uniform samples from $\mathcal{U}(0, 1)$. During the forward pass, $i = \text{argmax}_j p_j$ and in the backward pass, the true gradient of the Gumbel-Softmax outputs is used.

K-Means

The vector quantization approach of van den Oord et al. (2017) is an alternative to making the index selection procedure fully differentiable. Different to their setup, we optimize a future time step prediction loss instead of the reconstruction loss of an autoencoder.

We choose the codebook variable representation by finding the closest variable to the input features $\mathbf{z}$ in terms of the Euclidean distance, yielding $i = \text{argmin}_j \|\mathbf{z} - \mathbf{e}_j\|_2^2$. During the forward pass, we select $\hat{\mathbf{z}} = \mathbf{e}_i$ by choosing the corresponding variable from the codebook. We obtain gradients for the encoder network by back-propagating $\mathrm{d}\mathcal{L}^{\text{wav2vec}}/\mathrm{d}\hat{\mathbf{z}}$ (van den Oord et al., 2017). The final loss has two additional terms:



(b) K-means clustering.

$$\mathcal{L} = \sum_{k=1}^{K} \mathcal{L}_k^{\text{wav2vec}} + \left(\|\text{sg}(\mathbf{z}) - \hat{\mathbf{z}}\|^2 + \gamma \|\mathbf{z} - \text{sg}(\hat{\mathbf{z}})\|^2 \right), \tag{3}$$

where $\text{sg}(x) \equiv x$, $\frac{\mathrm{d}}{\mathrm{d}x}\text{sg}(x) \equiv 0$ is the stop gradient operator and γ is a hyperparameter. The first term is the future prediction task and gradients do not change the codebook because of the straight-through gradient estimation of mapping $\mathbf{z}$ to $\hat{\mathbf{z}}$. The second term $\|\text{sg}(\mathbf{z}) - \hat{\mathbf{z}}\|^2$ moves the codebook vectors closer to the encoder output, and the third term $\|\mathbf{z} - \text{sg}(\hat{\mathbf{z}})\|^2$ makes sure that the encoder outputs are close to a centroid (codeword).

Training BERT

Discrete BERT

- Next paper trains a ***continuous*** **BERT**
- Only train on the **masked language modeling (MLM)** task (So, **ROBERTA** not **BERT** no next sentence prediction)
- They mask *spans of tokens* (not a single token)
- Choose token for masking **based on a probability of 0.05**, then expand token sequence to a span of 10 tokens
- **Spans may *overlap***
- Compute **cross-entropy loss** (attempts to maximize the likelihood of predicting the true span for each masked span)

Continuous BERT

Training Continuous BERT

- A MLM task **cannot be performed with continuous inputs and outputs**. Why?
- **Reason:** There are **no targets to predict in place of the masked tokens**
- They **classify the masked positive example among a set of negatives**
- **Model input:** dense wav2vec features (**MFCC** or **FBANK** features) representing 10ms of audio data
- Replace some input with a ***mask* embedding**, and feed into Transformer encoder
- Then compute dot product between **outputs corresponding to each masked input**, the **true input that was masked**, and a **set of negatives sampled from other masked inputs within the same batch** (dot product attention)

Continuous BERT Contd.

Training Continuous BERT

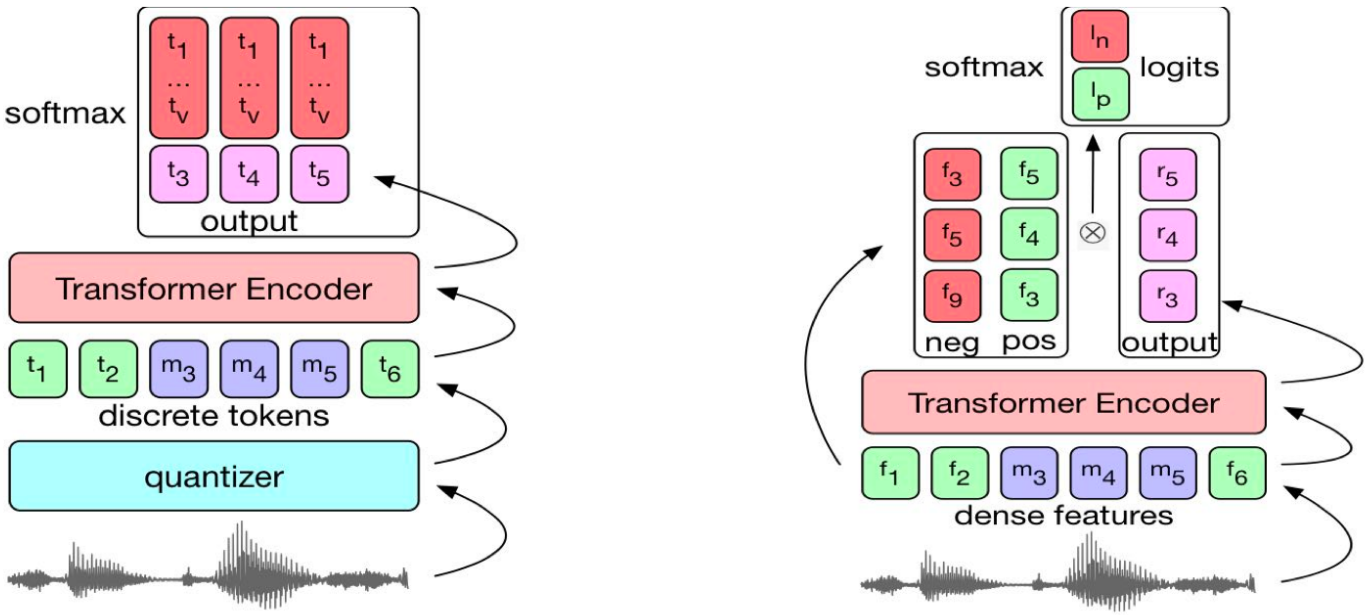
- Dot product between **outputs corresponding to each masked input**, **true input that was masked**, and a **negative samples**

the same batch. The model is optimized with the InfoNCE loss [11] where given one positive sample $\mathbf{z}_i$ and N negative samples $\tilde{\mathbf{z}}$ we minimize:

$$\mathcal{L}_k = \sum_{i=1}^{T} \frac{\exp(\mathbf{z}_i)}{\sum_{j=1}^{N} \exp(\tilde{\mathbf{z}}^j)} \tag{2}$$

where each sample $\mathbf{z}_i$ is computed as a dot product of the output of the model at timestep i and the true unmasked value of positive example at timestep i or a randomly sampled negative example. To stabilize training, we add the squared sum of logits produced by the dot-product to the loss, and then apply a soft clamp $\hat{s_i} = \lambda \tanh(s_i/\lambda)$ for each logit s_i to prevent the model's tendency to continually increase the magnitude of logits during training [28]. We use the same kind of convolutional positional layer as described in section 3.1.

Illustration of Two BERT Variants



(a) Quantized Inputs

(b) Continuous Inputs

Fig. 1: Illustration of BERT pre-training. m_i refers to masked time-steps chosen at random during each forward pass. (a) Inputs are quantized with a vq-wav2vec quantizer or, for MFCC and FBANK variants, by finding the closest centroids and are then used for training a BERT model with a masked language model objective. (b) wav2vec, MFCC or FBANK inputs are masked, encoded by a transformer, and then fed into a classifier to predict which features were masked at each time-step.