applied sciences

MDPI

Article

Bidirectional Representations for Low-Resource Spoken Language Understanding

Quentin Meeus [1,2,*], Marie-Francine Moens [1] and Hugo Van hamme [2]

[1] LIIR Lab, Computer Science Department, KU Leuven, 3001 Leuven, Belgium
[2] Speech Lab, Electrical Engineering Department, KU Leuven, 3001 Leuven, Belgium; hugo.vanhamme@kuleuven.be
* Correspondence: quentin.meeus@kuleuven.be

Featured Application: Models proposed in this research can be used to provide a spoken command interface for applications for which training data are limited and expensive.

Abstract: Speech representation models lack the ability to efficiently store semantic information and require fine tuning to deliver decent performance. In this research, we introduce a transformer encoder–decoder framework with a multiobjective training strategy, incorporating connectionist temporal classification (CTC) and masked language modeling (MLM) objectives. This approach enables the model to learn contextual bidirectional representations. We evaluate the representations in a challenging low-resource scenario, where training data is limited, necessitating expressive speech embeddings to compensate for the scarcity of examples. Notably, we demonstrate that our model's initial embeddings outperform comparable models on multiple datasets before fine tuning. Fine tuning the top layers of the representation model further enhances performance, particularly on the Fluent Speech Command dataset, even under low-resource conditions. Additionally, we introduce the concept of class attention as an efficient module for spoken language understanding, characterized by its speed and minimal parameter requirements. Class attention not only aids in explaining model predictions but also enhances our understanding of the underlying decision-making processes. Our experiments cover both English and Dutch languages, offering a comprehensive evaluation of our proposed approach.

Keywords: spoken language understanding; transformers; representation learning; attention

check for updates

Citation: Meeus, Q.; Moens, M.F.; Van hamme, H. Bidirectional Representations for Low-Resource Spoken Language Understanding. *Appl. Sci.* **2023**, *13*, 11291. https://doi.org/10.3390/app132011291

Academic Editors: Alexey Karpov, Ha-Jin Yu and Dongsuk Yook

Received: 29 August 2023
Revised: 5 October 2023
Accepted: 11 October 2023
Published: 14 October 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Commercial spoken language understanding (SLU) systems rely on cascading automatic speech recognition (ASR) with natural language understanding (NLU). We identify three main problems of such interfaces: (1) the ASR errors are cascaded to the NLU system; (2) discrepancies arise between training and inference, as the NLU model is often not trained on spoken language, which differs in terms of phrasing but also includes acoustic elements such as stress and pitch that are lost after transcription; and (3) the resulting systems can be large, as both recent ASR and NLU models contain an increasing number of parameters. However, it is undeniable that there is a strong need for lightweight, robust personal assistants able to understand the nuances and intricacies of spoken language. Those systems should also work out of the box, without requiring much effort from the user.

Since their introduction in 2017, transformers [1] have taken over the field of natural language processing (NLP). The key to their success is their ability to process long sequences, attend to relevant pieces of information present in the input and learn qualitative representations that can be used for other purposes. This last aspect is especially true for masked language models (MLMs), where a model is trained to reconstruct a partial input in a "fill in the gap" fashion. In this setting, words are masked from the input sentence, and the model must predict what was removed. While doing so, transformers learn syntactic and semantic information, such as parts of speech and language composition [2]. Unlike conditional language models (CLMs), which only have access to past information to generate a prediction, MLMs can benefit from left and right contexts [3,4]. Furthermore, whereas CLMs are limited to the generation of one token at a time conditioned on the previously generated tokens, MLMs can fill multiple masked tokens in parallel. Nonetheless, CLMs are still widely popular for language generation tasks.

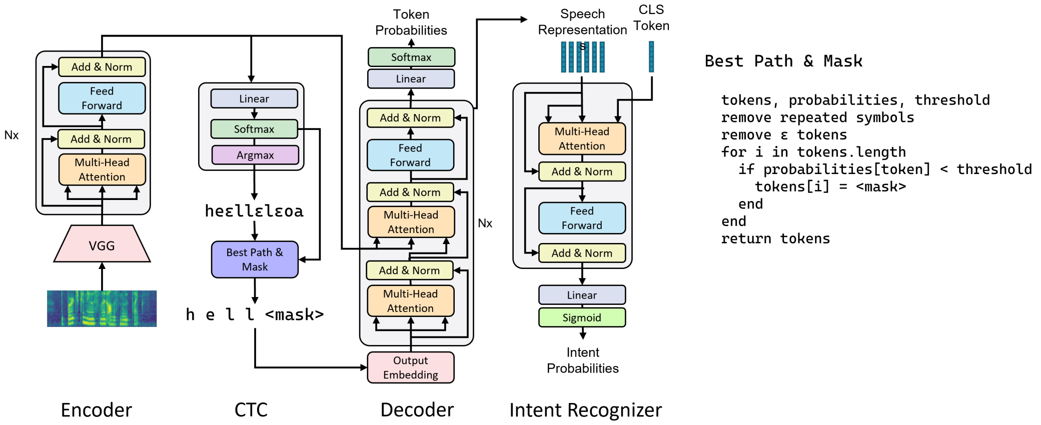
In speech processing, transformers have also been widely adopted by the research community [5]. A major difference with text-based transformers is the addition of a convolutional front end to process continuous sequences [6,7]. Its role is to learn local relationships and encode positional information. Both encoder-only and encoder–decoder transformers have been experimented with. Large self-supervised encoder-only models have shown impressive performance in many speech processing tasks [8,9]. However, the amount of semantic information contained in these encoders is limited or so deeply entangled that extracting it is challenging, and fine tuning the encoder is necessary to achieve satisfactory performance [10]. In contrast, encoder–decoder models split the representation task over two distinct modules: the encoder models the acoustic information contained in speech, while the decoder learns more language related features such as syntax and semantics [11]. The encoder–decoder relies only on attention to align speech and text. Watanabe et al. [12] proposed the integration of connectionist temporal classification (CTC) [13], a technique that has been well researched in speech recognition. The CTC objective in the hybrid transformer enforces a monotonic alignment between speech and text, resulting in improved robustness and faster convergence [12]. Another notable contribution is that of Higuchi et al. [4], who concatenated the outputs of a speech encoder and BERT, which was then passed to a prediction network. They also noted the benefit of using left and right context, as opposed to models learning only from past values. Finally, attempts have been made to augment large language models with speech capabilities, although more research is needed to achieve competitive performance [14]. Despite considerable improvement in speech modeling in recent years, speech representation models do not show the same ability as text-based language models to efficiently store semantic information, and a considerable amount of fine tuning is necessary to achieve decent performance [10].

In this research, we explore representation models of speech and demonstrate their usefulness for SLU, with a particular focus on low-resource solutions, when only a handful of examples is available to train and validate the models. We propose an architecture for pretraining of bidirectional speech transformers on a surrogate ASR task with the goal of learning qualitative representations that prove useful for solving language understanding tasks. We evaluate the resulting embeddings in an SLU task, intent recognition, where the objective is to, given a spoken command, identify the intent and any argument necessary. For example, the sentence “switch the lights in the kitchen to blue” would result in an intent of “lights” and arguments of “room = kitchen” and “color = blue”, while “play Beyonce” has an intent of “music” and an argument of “artist = Beyonce”. For completeness and comparison purposes, we also include results obtained after partially fine tuning the model.

We are particularly interested in low-resource scenarios, where we control the number of examples used for training and validation to study the limits of what can be learned from the representations. When information is stored efficiently, few examples are necessary to learn how to extract it. In contrast, noisy embeddings require many examples to extract relevant information. For the SLU module, we propose class attention as a drop-in replacement of LSTMs. Class attention assigns a score to each position that relates to the importance of the corresponding token to predict the intent and arguments. Aside from the advantages of its small footprint, it provides us with a method to understand why a model makes predictions. In the remainder of this article, we first detail the architecture of the different components of the model. Then, we lay out the experimental setup and methodology. Finally, we evaluate the models and analyze the representations in detail.

2. Methods

Building on the work of Higuchi et al. [15], we propose an encoder–decoder architecture with a multiobjective training strategy to learn bidirectional representations of speech (Figure 1). We evaluate the features learned in a downstream SLU task: intent prediction. The encoder, as presented in Section 2.1, learns acoustic representations of speech. The sequences of acoustic unit representations are used in two modules in parallel: they are mapped to output symbols with a classification layer and optimized with CTC (Section 2.2), and they are processed by a bidirectional transformer decoder to learn linguistic features (Section 2.3). These features learned by the decoder are processed by an intent recognizer to obtain the final output (Section 2.4).



```
Best Path & Mask

  tokens, probabilities, threshold
  remove repeated symbols
  remove ε tokens
  for i in tokens.length
    if probabilities[token] < threshold
      tokens[i] = <mask>
    end
  end
  return tokens
```

Figure 1. The encoder processes speech features. The CTC module makes a rough prediction of the text, where low-probability tokens are masked. The decoder attends to the masked transcription and to the encoder's output and predicts missing symbols. The generated representations are used as input to the downstream model to predict the intent.

2.1. Encoder

The encoder processes Mel-scaled filter banks and transforms them into a sequence of acoustic embeddings (h_{enc}). The module is composed of a VGG-like convolutional front end followed by a transformer encoder. The role of the CNN is twofold: decrease the the frequency and time dimensions and learn the local relationships in the sequence of speech features. This replaces the positional encoding traditionally required by transformers to keep track of the sequence order [7]. The CNN front end compresses a sequence of speech features of length T to a smaller sequence of length $T' \approx T/4$. The transformer encoder is composed of multiple blocks of multihead self-attentions and position-wise fully connected feedforward networks. Each block has a residual connection around both operations, followed by layer normalization. In this setting, the encoder cannot be trained in a standalone process.

2.2. CTC

Connectionist temporal classification [13] is a method for optimizing sequence-to-sequence models. For each unit in the input sequence, CTC predicts a probability distribution over the vocabulary, consisting of the set of output symbols and a blank token. The predictions are assumed to be conditionally independent. A simple algorithm allows a sequence of CTC tokens to be reduced to a sentenceby first removing repeated symbols, then removing blank tokens (Figure 1). The CTC loss is computed by summing the negative log likelihood of all the alignments that result in the correct sentence. For decoding, we opt for a greedy algorithm, which takes the token predicted with the highest probability at each time step. Although more advanced techniques such as beam search or weighted finite-state transducers [16] can improve the CTC prediction considerably, Zenkel et al. [16] found that the largest gains were observed in substitution errors, with substantially lower gains for insertions and deletions. For this reason, we prefer a fast decoding technique and let the decoder fix the substitution errors in the rough prediction.

2.3. Decoder

Although an encoder equipped with CTC is perfectly able to transcribe speech, it does so by assigning a token to each position independently. Watanabe et al. [12] reported that the addition of a decoder integrates language modeling in the learning process. Indeed, the decoder combines the acoustic cues of the encoder with the previous tokens to predict the next one. Here, we opt for a bidirectional transformer decoder instead of a left-to-right model. Left-to-right decoders predict one token at a time and stop after producing a special end-of-sequence token. This method is slower because a prediction depends on the previously predicted tokens. Additionally, the network only has access to previous tokens, thus limiting the amount of information available to make a prediction. In modern NLP, bidirectional architectures have become popular to learn powerful representations that make use of both left and right contexts [3,17]. For instance, BERT [3] is optimized with a masked language modeling objective by randomly masking some tokens in the input sequence. In ASR, a similar approach was adopted by Higuchi et al. [15], who obtained a rough text prediction with CTC, which was refined by masking low-probability tokens and letting the decoder predict the missing items in the sequence, similarly to Ghazvininejad et al. [18]. One issue with these models is the decoder's initial state definition. As the output sequence is unknown at the inference time, it is necessary to provide the decoder with a template consisting of a sequence of mask tokens of the same length as the target sequence. The decoder is not able to add or remove symbols to adjust the sequence length, and the correct length is unknown at the inference time. Ghazvininejad et al. [18] presented a solution that involves predicting the length as part of the prediction task of the encoder. Then, during inference, the decoder predicts multiple candidates of different lengths corresponding to the predicted lengths with the highest probability. Higuchi et al. [15] assumed the greedy CTC prediction to be of correct length, although they showed that it leads to weaker transcriptions. In this work, we focus on learning rich representations for spoken language understanding. We thus work around this issue, as we assume that the length is not essential for our purpose. We follow Higuchi et al. [15] for the architecture of the decoder. We use the output of the penultimate layer as the linguistic representations of speech.

2.4. Intent Recognizer

In the SLU datasets, intent is represented as a certain action (e.g., move, grab, turn, etc.) to which are attached a number of arguments (forward, fast, left, etc.). We encode the full intent string as a multihot vector, where each bit corresponds to either one of the possible actions or an argument value. The representations obtained from the decoder are summarized with a stack of class attention layers [19]. Class attention combines a sequence of representations into a class embedding (x_{CLS}), that is, a learned vector of the same dimensions as the representations. During training, the module learns to identify patterns in the input features that are predictive of the class labels. Similarly to other transformers, a class attention layer is composed of two sublayers: an attention layer and a point-wise feedforward layer. The input to the layer is normalized before processing to stabilize training, as prescribed by Xiong et al. [20]. We define the queries in the original notation according to Vaswani et al. [1] as the following class embedding: $Q = x_{\mathrm{CLS}}$. The keys and values are linear projections of the elements in the input sequence of features (x): $K = W_k \cdot x + b_k$ and $V = W_v \cdot x + b_v$. In contrast to Touvron et al. [19], we do not include the CLS token in the keys and values. We compute the attention weights by applying the softmax function to the inner product of the keys and queries, scaled by the square root of the dimension of each attention head: $A = \sigma(Q \cdot K^T / \sqrt{d/h})$, where d is the total dimension of x_{CLS}, and h is the number of attention heads. The output corresponds to $y = W_o AV + b_o$. W_k, where b_k, W_v, b_v, W_o and b_o are learned parameters. We train the model by minimizing a binary cross-entropy loss. Since not all combinations are possible, we enforce the output structure by selecting the valid combination that minimizes the cross-entropy with the predicted vector.

3. Experiments

3.1. Datasets

Corpus Gesproken Nederlands [21] is a collection of recordings in Dutch and Flemish collected from various sources, such as readings, lectures, news reports, conferences, telephone conversations, etc., totaling more than 900 h. They are divided into 15 components (a to o) based on their nature. After removing short and overlapping utterances, we divide each component into three subsets to serve as training, validation and test sets, respectively. We leave out three components (a, c and d) because the quality of the transcriptions differs considerably from that of the other components. The subsets from the remaining components, totaling 415 h of speech, are concatenated to form the training, validation and test sets.

Librispeech [22] contains about a thousand hours of read English speech derived from audiobooks. The authors provide official splits for training, validating and testing the models, and this dataset serves as a reference in ASR. We use all 960 h for training.

Grabo [23] is composed of spoken commands, mostly in Dutch (one speaker speaks in English) intended to control a robot. There are 36 commands that were repeated 15 times by each of the 11 speakers. More precisely, the robot can perform eight actions (i.e., approach, move (relative), move (absolute), turn (relative), turn (absolute), grab, point and lift), of which is further defined by some attributes (e.g., the robot can move forward or backward and rapidly or slowly). We follow the methodology applied in [23]: For each speaker, we create nine datasets of different sizes and divide each dataset in five folds for cross validation. The target is represented as a binary vector of 31 dimensions.

Patience [24] is derived from a card game in which a player provides vocal instructions to move cards between different tiles on the table. The dataset contains recordings from eight different Flemish speakers. The intent of the player is represented as a binary vector. Again, we use the same splitting methodology as Renkens and Van hamme [23], keeping only the 31 most represented classes. It should be noted that without visual input, this task is quite difficult, even for a human operator.

Fluent Speech Commands [25] contains 30,043 utterances from 97 English speakers. Each utterance is a command to control smart home appliances or a virtual assistant. The challenge splits that we use in this work were proposed by Arora et al. [26]. Two test sets are provided, in which specific speakers or utterances are separated from the training and validation sets. We also explore data shortage scenarios, where only a portion of the training set is available.

SmartLights [27] is composed of 1660 commands to control smart lights uttered by speakers with various accents and of various origins. Again, we use the challenge split proposed by Arora et al. [26].

3.2. Pretraining

We pretrain our encoder–decoder model on an ASR task with a hybrid objective that is the weighted sum of the CTC loss ($Ł_{\text{ctc}}$) and the label-smoothing cross-entropy loss ($Ł_{\text{dec}}$: $Ł = \rho\, Ł_{\text{ctc}} + (1 - \rho)\, Ł_{\text{dec}}$, where ρ and $1 - \rho$ are the weights of the CTC loss and the decoder loss in the total loss, respectively). We use CGN (220 h) to pretrain the Dutch model and Librispeech (960 h) for the English model. Our objective is to train a representation model of speech that captures elements of language and semantics while remaining robust to irrelevant aspects such as speaker identity or background noise. Additionally, the information stored in the encodings should be readily accessible for processing by a downstream model. We choose ASR as a proxy task because of the availability of large datasets and because it is easier to generate rich representations containing elements of language than with self-supervised approaches, which typically require considerably more resources to achieve the same goal. However, the supervised objective is likely to discard information not directly useful for solving the task. Addressing this neglected information will be the focus of future work. During pretraining, we use the real tokens as input to be masked for the decoder. During inference, we use the CTC module to generate a rough prediction that is used as a template, and we mask out tokens predicted with a probability of less than 90%. Similarly to MaskPredict [18], we iteratively refine the input for a maximum of 10 steps. We pretrain the models on a GPU for 200 epochs, with batches of 32 examples, and accumulate the gradients over 8 iterations, which provides an effective batch size of 256. We use the Noam learning rate scheduler [1]. We experimentally set ρ to 0.3 by measuring the ASR accuracy on a held-out validation set.

3.3. Training

After pretraining the representation model, we freeze all the layers to train the spoken language understanding module. The training objective is to minimize the binary cross entropy between the predictions and the multihot-encoded targets. Freezing the representation model gives us the opportunity to evaluate the predictive power of the representations. At this point, the pretrained encoder–decoder from described in Section 3.2 has not been exposed to any training example from the SLU datasets, nor does it know about the output structure of the underlying task (number of classes, etc.). We train the models with the Adam optimizer [28] for a maximum of 200 epochs, with early stopping. The batch size is 512. We set the learning rate to 0.005, except for the Smartlights dataset, for which we found that a slower learning rate of 0.001 achieved better results.

3.4. Fine Tuning

Our main objective with this research is to learn representations that perform well without fine tuning. However, to provide a good basis for comparison with previous research and to quantify how much can be gained by updating the main model, we perform a few fine-tuning steps at a low learning rate while unfreezing some or all layers of the representation model's decoder. This operation can be seen as specializing the representation model for the specific aspects of the downstream task, often at the expense of generality. In practice, we found that unfreezing the whole decoder was more tedious and sometimes led to overtraining. Unfreezing the last four layers generally yielded the best results. As discussed in Section 2.3, we use the CTC module to produce the initial sequence in which low-probability tokens are masked. We perform only one pass with the decoder to generate the representations used by the SLU module.

3.5. Evaluation

We choose specific metrics to assess the performance of our models, primarily focusing on accuracy and F1 score. Accuracy measures the proportion of correctly predicted instances, encompassing both intents and slots. The F1 score, which is computed as the harmonic mean of precision and recall, provides a more holistic evaluation. While we consider the F1 score to offer a more comprehensive assessment, we also examine accuracy to facilitate comparisons with established models.

In our investigation into the model's performance within a low-resource setting, we deliberately limit the number of available training examples. This allows us to assess its adaptability and effectiveness under conditions of data scarcity.

Furthermore, to gain deeper insights into the representations generated by our model, we conduct a qualitative analysis. We achieve this by visualizing the representations using t-SNE [29], a technique that helps identify clusters of examples, shedding light on the structure and organization of the learned features. We also visualize the attention weights to understand how the model makes certain predictions.

3.6. Hyperparameters

The transformer has 18 layers (12 encoder layers and 6 decoder layers) each with 4 attention heads, a hidden layer size of 256 and 2048 hidden units in the linear layer. We use dropout with a probability of 0.1 after each transformer layer. The front end has two 2D convolutional layers with kernel sizes of 3 × 3 and a stride of 2 × 2. Then, the input dimension is divided by a factor of four along the time and frequency dimensions. The hyperparameters related to the architecture of the encoder–decoder were chosen according to Higuchi et al. [15]. The encoder–decoder model has 30.9 million parameters. The weights for scaling pf the different losses in the final loss function are chosen through experimentation using the pretraining data. We train our models for 200 epochs with a batch size of 256. The models are trained with the Adam optimizer [28]. The learning rate linearly increases until reaching a maximum value of 0.4 after 25,000 steps, then decreases according to the Noam schedule [1]. The downstream datasets were not used in any way to determine the value of the abovementioned hyperparameters.

The downstream models are composed of two class attention layers with four attention heads with 32 dimensions. The fully connected layer has 1024 units. The models are trained with batches of 512 examples for 100 epochs, with a learning rate equal to 0.005. The intent classification module has 890 thousand parameters.

4. Results

We compare our models on basis of the accuracy score to remain consistent with previous research. We select four baselines because they propose a similar approach as ours and present results on at least one of our selected SLU datasets. End-to-end SLU [25] proposes a model composed of a phoneme module, a word module and an intent module that can all be trained or fine-tuned independently. The authors experimented with four settings: with or without pretraining and fine tuning the word module only or all modules together. The two best models (pretrained on ASR and the fine-tuned word module only) are displayed in Tables 1–3. ST-BERT [30] also uses an MLM objective. However, it is pretrained with cross-modal language modeling on speech and text data in two different ways: using MLM or with conditional language modeling (CLM), where the goal is to predict one modality given the other. Additionally, the model uses large text corpora pretraining and domain adaptation pretraining with the SLU transcripts to further improve the results. However, the use of text transcripts from the downstream task is not compatible with our use case, as we assume that only the speech is available for the SLU task. Since domain adaptation provides an unfair advantage, we do not report the results related to this experiment. The pretrained model corresponds to the model that was pretrained on speech-only data, and the fine-tuned model corresponds to the model pretrained on text with CLM and fine-tuned on SLU. The performance of these two baselines were reported after training on 1% and 10% of the data, which is not the case for the following. Wav2Vec2-Classifier [31] is an encoder-only model with a fully connected layer as the output. The model is entirely fine-tuned, and the performance when freezing Wav2Vec2 is not reported, although we know from [10] that Wav2Vec does not store semantics in its internal representations and that fine tuning is necessary to achieve decent performance. ESPnet-SLU [11] uses a pretrained HuBERT as a feature extractor and a transformer decoder for the SLU module. The complete model is fine-tuned on SLU.

Table 1 shows the results on Fluent Speech Commands when the models are trained on varying amounts of training data. For completeness, we also report the results of the models fine-tuned on the entire training set (Table 2). In Table 3, the challenge splits [26] correspond to improved splits, where specific speakers or specific utterances are partitioned. For Smartlights, no previous splits are available, so we follow the same approach as Arora et al. [26] and use a random splitting strategy.

In Table 1, we observe significant improvements compared to Lugosch et al. [25], especially when few training examples are used, both before and after fine tuning. Both Lugosch et al. [25] and Seo et al. [31] noted that fine tuning the entire model does not always improve performance because exposition to new data leads to catastrophic forgetting and, in some cases, overfitting. We make the same observation and find that unfreezing the encoder leads to its degradation. Indeed, the encoder was pretrained on large amounts of data and is robust to variations in the input features. When it is fine-tuned with a different objective for a handful of examples, the parameter updates following the new setting often lead to overfitting and loss of generality. This is also observed for ESPnet-SLU [11] in Table 3, where we observe a much larger difference between the performance on unknown speakers and unknown utterances than with the other models. Updating the encoder also deteriorates its ability to generate a good template for the decoder, which leads to weaker performance. Consequently, we decided to keep the encoder frozen but update the decoder during the fine-tuning stage. With this setting, the performance improves slightly, although consistently compared to the model before fine tuning. Our model shows a similar pretraining accuracy as Kim et al. [30], but small improvements are observed after fine tuning. The good performance of the frozen models is encouraging, as it suggests that the pretrained model stores semantic information adequately enough for the SLU module to make use of it. Both approaches using an MLM objective are particularly efficient when trained on 1% of the training data, which leads us to conclude that bidirectional context embeddings are particularly useful in a low-resource setting.

Our model also fares well against Seo et al. [31] and Watanabe et al. [11], although our methodology does not include data augmentation (Table 2). Unfortunately, we cannot compare the quality of the representations produced by the pretrained models, owing to a lack of reported results.

In Table 3, we observe a 12% relative improvement compared to [25] on FSC (Challenge) but a deterioration on Smartlights. We speculate that the fact that the MLM objective produces similar embeddings for word pieces that belong to the same semantic scope (such as blue and green or bedroom and bathroom) hurts the performance on this particular dataset. The few examples in the training set do not suffice to generalize to unknown phrasings for a specific target (i.e., when the same target is expressed differently).

Table 1. Test accuracies on Fluent Speech Commands (original splits). In the first stage, the pretrained model is frozen, and the SLU layers are trained. In the second stage, the pretrained model is (partially) fine-tuned.

Stage	% for Training	MLM	E2E SLU [25]	ST-BERT [30]
Frozen	100	**99.5**	98.8	99.4
	10	99.0	98.0	99.0
	1	**91.5**	82.8	89.8
Fine tuning	100	**99.8**	99.1	99.5
	10	**99.5**	97.9	99.1
	1	95.7	-	95.7

Table 2. Accuracy on the Fluent Speech Commands original test set (100%) after fine tuning.

Model	Accuracy
MLM (finetune)	**99.8**
E2E SLU [25]	99.1
ST-BERT [30]	99.5
Wav2Vec2-Classifier [31]	99.7
ESPnet-SLU [11]	99.6

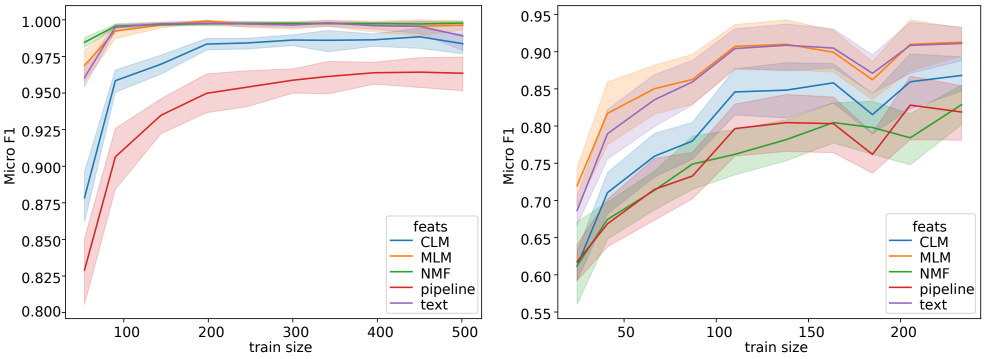
Table 3. Test accuracies on the Fluent Speech Commands (challenge splits) and Smartlights datasets with random and challenge splits. In the first stage, the pretrained model is frozen, and the SLU layers are trained. In the second stage, the pretrained model is partially (or totally in [11]) fine-tuned.

		FSC (Ch.)		**Smartlights**	**Smartlights (Ch.)**	
		Spk.	**Utt.**	**Random**	**Spk.**	**Utt.**
Frozen	MLM	**95.2**	**86.3**	**84.0**	**79.5**	**69.0**
	E2E SLU [25]	90.9	73.4	83.2	73.2	67.4
	ST-BERT [30]	-	-	81.2	-	-
Fine tuning	MLM	**98.8**	**88.0**	85.8	81.9	74.6
	E2E SLU [25]	92.3	78.3	**88.0**	**82.6**	**78.5**
	ST-BERT [30]	-	-	84.7	-	-
	ESPnet-SLU [11]	97.5	78.5	-	-	-

5. Analysis

5.1. Low-Resource Scenario

To explore low-resource scenarios, we split Grabo and Patience per speaker. For each speaker, we randomly select a fixed number of examples per class for the training set of the SLU model. By gradually increasing the size of the training set, we are able to measure the learning curve. This operation is performed three times per speaker, with different splits each time. We compute the micro-averaged F1 score for each experiment and report the average F1 score with its standard deviation in Figure 2.



(**a**) (**b**)

Figure 2. Comparison of different models trained with increasing train set sizes on (**a**) Grabo and (**b**) Patience. Each curve represents the F1 score on the test set as a function of the size of the training set (utterance count). The colored areas represent the 68% confidence interval. We use the F1 score for comparison with the NMF baseline [32].

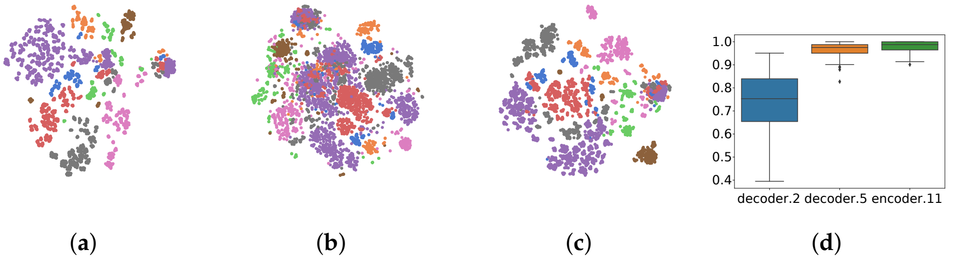
We compare our representations (MLM in Figure 2) with three types of features: NLP features generated by encoding the gold transcriptions with bert-base-dutch-cased [33], Pipeline features resulting from the encoding of ASR transcripts predicted by ESPnet's hybrid ASR model [12] trained on CGN with bert-base-dutch-cased and CLM features corresponding to the output of the penultimate layer of the ASR model. To provide a fair comparison, neither the NLP, ASR nor MLM model encountered training examples from the SLU dataset. In other words, we do not fine tune any of the representation models on the downstream datasets at this stage. Both the ASR and MLM models are trained on CGN, and the NLP model is trained on a collection of five text corpora for a total of about 2.4B tokens.

Considering the amount of training data, it is no surprise to see that the gold transcript encoded with the NLP model performs very well in all data regimes. As expected, the ASR transcripts show the weakest performance. Converting features into discrete symbols forces the model to make decisions, resulting in potential errors from which the model cannot recover. In contrast, the CLM and MLM models use the hidden representations produced by the model, thus avoiding this problem. Nonetheless, we see that the CLM features perform worse than our model. The masked language modeling objective allows the MLM model to look at both right and left contexts, whereas the CLM model only has access to previous predictions. Because each predicted unit is conditioned on the whole sentence, the model is able to derive better representations than by only looking at the past.

We also compare our results with the state of the art on both datasets [32]. This model combines ASR pretraining and an NMF decoder for intent recognition [32,34]. The NMF decoder uses a bag-of-words approach with multihot intent representation, thus ignoring the order of the words in the sentence. Although this model was presented for dysarthric speech, results are available for both the Grabo and Patience datasets [32]. The main advantage of the NMF decoder is its low computational requirements, which makes it particularly well-suited when very few training data are available, as can be observed in Figure 2a. This advantage disappears as the size of the training set increases. However, for the Patience corpus, in which the order of words is important for prediction, our approach achieves better performance compared to the model proposed by Wang and Van hamme [34], in which the sequence order is ignored (Figure 2b).

5.2. Representation Content

To better understand the characteristics of the representations, we average the sequences to a unique representation per utterance and visualize them in two dimensions using the t-SNE algorithm [29]. For this experiment, we focus on Grabo and, in particular, on the eight actions that the robot can perform. We compare the representations produced by different layers, namely the last encoder layer, the third decoder layer and the last decoder layer (encoder.11, decoder.2 and decoder.5 in Figure 3, respectively). Both encoder.11 and decoder.5 form meaningful clusters, while decoder.2 is ill-defined. This is also observed in the classification accuracy of those representations on SLU, where we observe a clear difference between decoder.2 and the other features. We expected that decoder.5 would generate better features than encoder.11, although they seem to lead to similar performance, with a small advantage for the encoder's output. We conjecture that the CTC component helps to define the acoustic units at the encoder's output, which translates into well-defined representation sequences, albeit much longer ones.



(**a**) (**b**) (**c**) (**d**)

Figure 3. t-SNE representations in different layers. Intents: approach (blue), grab (orange), lift (green), move_abs (red), move_rel (purple), pointer (brown), turn_abs (pink) and turn_rel (grey). The right-most graph shows the accuracy of the models with the representations of different layers as input. Best viewed in color. (**a**) Encoder.11; (**b**) decoder.2; (**c**) decoder.5; (**d**) SLU accuracy.

We also observe that approach and move_abs end up in the same region, which means that a classifier might often confuse them. This makes sense from a language perspective, and the fact that this is more the case in decoder.5 than in encoder.11 leads us to conclude that the implicit language model from the decoder generates similar embeddings for these two commands, although they do not sound similar.

5.3. Interpreting Model Predictions

Finally, we want to explore an aspect that is often overlooked in deep learning, namely elucidating the factors in the input features that drive the model's predictions. A key aspect of our class attention layer is its ability to establish a connection between the final output prediction and the input sequence through the attention weights. These weights can be interpreted as the relevance of each input unit in determining the correct label for the utterance. As shown in Figure 4, a limited subset of tokens (e.g., "turn", "the", "light", "on" and "kitchen") substantially influences the prediction process. In instances in which the model's predictions may deviate in terms of accuracy, these attention weights serve as valuable starting points for in-depth investigations into the underlying causes.

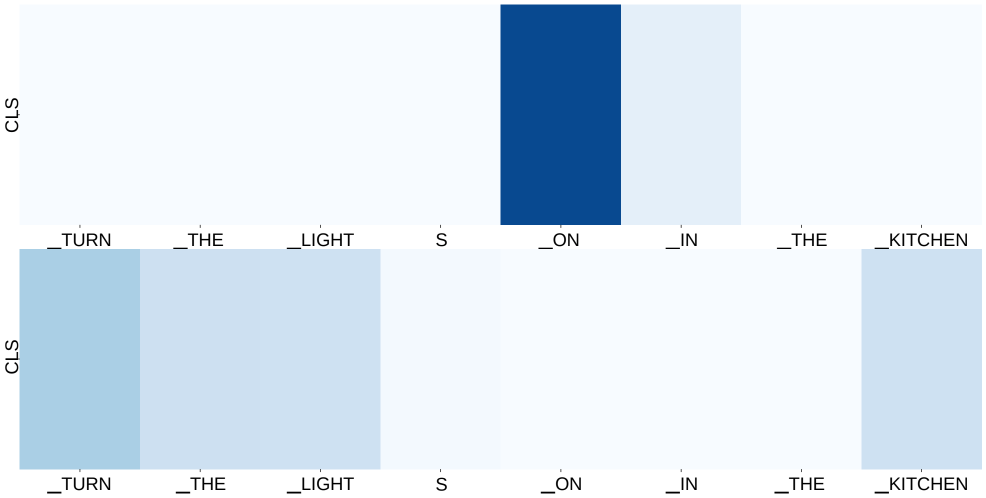


Figure 4. Class attention weight visualization of the two layers of the downstream class attention model. Although the model receives sequences of embeddings, we label the graph with the corresponding tokens for demonstration purposes. The two layers focus on different positions to predict the intent and arguments.

6. Conclusions

Despite the significant advancements in speech processing, there remains a crucial gap in the field concerning the development of speech representation models that capture semantics akin to the capabilities exhibited by language models. In this paper, we introduce a novel bidirectional representation model pretrained on an ASR task, which demonstrates remarkable effectiveness in transferring learned features to the domain of intent recognition, all without necessitating additional fine tuning. Our pretraining strategy incorporates CTC, together with an MLM objective, yielding speech features endowed with bidirectional contextual awareness.

Throughout our research, we examined the representations in various layers of the model, particularly with regard to their performance when applied to previously unseen datasets. Our findings show the versatility of these representations in training an effective SLU model. Notably, our representations either match or surpass the performance of state-of-the-art models on the SLU task, especially prior to the fine-tuning phase. We have also introduced a novel approach leveraging class attention mechanisms to summarize sequences of representations. This approach not only proves highly efficient, owing to its transformer-like architecture and minimal parameter requirements, but also offers insights into the model's decision-making process, shedding light on the input patterns that significantly influence predictions. Furthermore, our combination of class attention and ASR pretraining exhibits substantial gains in terms of data efficiency, as evidenced in a low-resource scenario in which we intentionally constrained the number of training examples available for the SLU task. In an era of increasingly massive and data-hungry models, our research underscores the importance of pursuing resource-friendly and efficient solutions.

However, it is important to acknowledge a limitation of our presented model, namely its challenges when confronted with SLU datasets that differentiate between terms seldom encountered or entirely absent in the pretraining data. In such cases in which two words are largely interchangeable except for a few instances in the pretraining dataset, our model's representations may become overly similar, hindering the ability to discriminate between these distinct classes without fine tuning. Addressing this limitation will be a focal point of our future research endeavors.

Author Contributions: Conceptualization, Q.M., H.V.h. and M.-F.M.; methodology, Q.M.; software, Q.M.; validation, Q.M., H.V.h. and M.-F.M.; formal analysis, Q.M.; investigation, Q.M.; resources, H.V.h.; data curation, Q.M.; writing—original draft preparation, Q.M.; writing—review and editing, Q.M.; visualization, Q.M.; supervision, H.V.h. and M.-F.M.; project administration, H.V.h. and M.-F.M.; funding acquisition, H.V.h. and M.-F.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiele Intelligentie (AI) Vlaanderen" program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in this research can be accessed online: https://taalmaterialen.ivdnt.org/download/tstc-corpus-gesproken-nederlands/ (accessed on 2 June 2022) (Corpus Gesproken Nederlands); https://www.openslr.org/12 (accessed on 2 June 2022) (Librispeech); https://www.esat.kuleuven.be/psi/spraak/downloads/ (accessed on 2 June 2022) (Grabo and Patience Corpus); https://fluent.ai/fluent-speech-commands-a-dataset-for-spoken-language-understanding-research/ (accessed on 2 June 2022) (Fluent Speech Commands); and https://github.com/sonos/spoken-language-understanding-research-datasets (accessed on 2 June 2022) (SmartLights).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ASR	Automatic speech recognition
CNN	Convolutional neural network
CLM	Conditional language modeling
CTC	Connectionist temporal classification
LSTM	Long short-term memory
MLM	Masked Language modeling
NLP	Natural language processing
NLU	Natural language understanding
NMF	Non-negative matrix factorization
SLU	Spoken Language understanding

References

1. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017 ; Curran Associates: Red Hook, NY, USA, 2017.
2. Jawahar, G.; Sagot, B.; Seddah, D. What Does BERT Learn about the Structure of Language? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Association for Computational Linguistics), Florence, Italy, 28 July–2 August 2019. [CrossRef]
3. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Association for Computing Machinery), Minneapolis, MN, USA, 2–7 June 2019; Volume 1. [CrossRef]

4. Higuchi, Y.; Ogawa, T.; Kobayashi, T.; Watanabe, S. BECTRA: Transducer-Based End-To-End ASR with Bert-Enhanced Encoder. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023. [CrossRef]
5. Karita, S.; Wang, X.; Watanabe, S.; Yoshimura, T.; Zhang, W.; Chen, N.; Hayashi, T.; Hori, T.; Inaguma, H.; Jiang, Z.; et al. A Comparative Study on Transformer vs RNN in Speech Applications. In Proceedings of the Automatic Speech Recognition and Understanding Workshop (ASRU), Singapore, 14–18 December 2019. [CrossRef]
6. Dong, L.; Xu, S.; Xu, B. Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018. [CrossRef]
7. Mohamed, A.; Okhonko, D.; Zettlemoyer, L. Transformers with convolutional context for ASR. *arXiv* **2019**, arXiv:1904.11660. [CrossRef]
8. Baevski, A.; Zhou, H.; Mohamed, A.; Auli, M. Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In Proceedings of the 34th International Conference on Neural Information Processing Systems, Online, 6–12 December 2020; Curran Associates: Red Hook, NY, USA, 2020. [CrossRef]
9. Hsu, W.N.; Bolte, B.; Tsai, Y.H.; Lakhotia, K.; Salakhutdinov, R.; Mohamed, A. HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2021**, *29*, 3451–3460. [CrossRef]
10. Wang, Y.; Boumadane, A.; Heba, A. A Fine-tuned Wav2vec 2.0/HuBERT Benchmark For Speech Emotion Recognition, Speaker Verification and Spoken Language Understanding. *arXiv* **2021**, arXiv:2111.02735. [CrossRef]
11. Watanabe, S.; Hori, T.; Karita, S.; Hayashi, T.; Nishitoba, J.; Soplin, Y.; Unno, N.E.Y.; Heymann, J.; Wiesner, M.; Chen, N.; et al. ESPnet: End-to-End Speech Processing Toolkit. In Proceedings of the International Speech Communication Association, Hyderabad, India, 2–6 September 2018. [CrossRef]
12. Watanabe, S.; Hori, T.; Kim, S.; Hershey, J.R.; Hayashi, T. Hybrid CTC/Attention Architecture for End-to-End Speech Recognition. *IEEE J. Sel. Top. Signal Process.* **2017**, *11*, 1240–1253. [CrossRef]
13. Graves, A.; Fernández, S.; Gomez, F.; Schmidhuber, J. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; Association for Computing Machinery: New York, NY, USA, 2006. [CrossRef]
14. Gao, H.; Ni, J.; Qian, K.; Zhang, Y.; Chang, S.; Hasegawa-Johnson, M. WavPrompt: Towards Few-Shot Spoken Language Understanding with Frozen Language Models. *Proc. Interspeech* **2022**, *2022*, 2738–2742. [CrossRef]
15. Higuchi, Y.; Watanabe, S.; Chen, N.; Ogawa, T.; Kobayashi, T. Mask CTC: Non-Autoregressive End-to-End ASR with CTC and Mask Predict. *Proc. Interspeech* **2020**, *2020*, 3655–3659. [CrossRef]
16. Zenkel, T.; Sanabria, R.; Metze, F.; Niehues, J.; Sperber, M.; Stüker, S.; Waibel, A. Comparison of Decoding Strategies for CTC Acoustic Models. In Proceedings of the 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, 20–24 August 2017. [CrossRef]
17. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, New Orleans, LA, USA, 1–6 June 2018. [CrossRef]
18. Ghazvininejad, M.; Levy, O.; Liu, Y.; Zettlemoyer, L. Mask-Predict: Parallel Decoding of Conditional Masked Language Models. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019. [CrossRef]
19. Touvron, H.; Cord, M.; Sablayrolles, A.; Synnaeve, G.; Jégou, H. Going deeper with Image Transformers. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021. . [CrossRef]
20. Xiong, R.; Yang, Y.; He, D.; Zheng, K.; Zheng, S.; Xing, C.; Zhang, H.; Lan, Y.; Wang, L.; Liu, T.Y. On Layer Normalization in the Transformer Architecture. In Proceedings of the 37th International Conference on Machine Learning, Virtual, 13–18 July 2020.
21. Oostdijk, N. Het Corpus Gesproken Nederlands. *Ned. Taalkunde* **2000**, *5*, 280–285.
22. Panayotov, V.; Chen, G.; Povey, D.; Khudanpur, S. Librispeech: An ASR corpus based on public domain audio books. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia, 19–24 April 2015. [CrossRef]
23. Renkens, V.; Van hamme, H. Capsule Networks for Low Resource Spoken Language Understanding. *Proc. Interspeech* **2018**, *2018*, 601–605. [CrossRef]
24. Tessema, N.; Ons, B.; van de Loo, J.; Gemmeke, J.F.; De Pauw, G.; Daelemans, W.; Van hamme, H. *Metadata for Corpora Patcor and Domotica-2*; KU Leuven, ESAT: Leuven, Belgium, 2013.
25. Lugosch, L.; Ravanelli, M.; Ignoto, P.; Tomar, V.S.; Bengio, Y. Speech Model Pre-Training for End-to-End Spoken Language Understanding. *Proc. Interspeech* **2019**, *2019*, 814–818. [CrossRef]
26. Arora, S.; Ostapenko, A.; Viswanathan, V.; Dalmia, S.; Metze, F.; Watanabe, S.; Black, A.W. Rethinking End-to-End Evaluation of Decomposable Tasks: A Case Study on Spoken Language Understanding. *Proc. Interspeech* **2021**, *2021*, 1264–1268. [CrossRef]
27. Saade, A.; Dureau, J.; Leroy, D.; Caltagirone, F.; Coucke, A.; Ball, A.; Doumouro, C.; Lavril, T.; Caulier, A.; Bluche, T.; et al. Spoken Language Understanding on the Edge. In Proceedings of the Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS), Vancouver, BC, Canada, 13 December 2019. [CrossRef]

28. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, 7–9 May 2015. [CrossRef]
29. van der Maaten, L.; Hinton, G.E. Visualizing High-Dimensional Data Using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
30. Kim, M.; Kim, G.; Lee, S.W.; Ha, J.W. ST-BERT: Cross-Modal Language Model Pre-Training for End-to-End Spoken Language Understanding. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021. [CrossRef]
31. Seo, S.; Kwak, D.; Lee, B. Integration of Pre-Trained Networks with Continuous Token Interface for End-to-End Spoken Language Understanding. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 22–27 May 2022. [CrossRef]
32. Wang, P.; Van hamme, H. Pre-training for low resource speech-to-intent applications. *arXiv* **2021**, arXiv:2103.16674. [CrossRef]
33. de Vries, W.; van Cranenburgh, A.; Bisazza, A.; Caselli, T.; van Noord, G.; Nissim, M. BERTje: A Dutch BERT Model. *arXiv* **2019**, arXiv:1912.09582. [CrossRef]
34. Wang, P.; Van hamme, H. Benefits of pre-trained mono- and cross-lingual speech representations for spoken language understanding of Dutch dysarthric speech. *EURASIP J. Audio Speech Music Process.* **2023**, *2023*, 15. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.