An open-source end-to-end ASR system for Brazilian Portuguese using DNNs built from newly assembled corpora

Igor M. Quintanilha, Luiz W. P. Biscainho, *Senior Member, IEEE,* and Sergio L. Netto, *Senior Member, IEEE*

***Abstract*—In this work, we present a baseline end-to-end system based on deep learning for automatic speech recognition in Brazilian Portuguese. To build such a model, we employ a speech corpus containing 158 hours of annotated speech by assembling four individual datasets, three of them publicly available, and a text corpus containing 10.2 millions of sentences. We train an acoustic model based on the DeepSpeech 2 network, with two convolutional and five bidirectional recurrent layers. By adding a newly trained 15-gram language model at the character level, we achieve a character error rate of only 10.49% and a word error rate of 25.45%, which are on a par with other works in different languages using a similar amount of training data.**

***Index Terms*—speech recognition, deep learning, speech corpus, text corpus, Brazilian Portuguese**

I. Introduction

AUTOMATIC speech recognition (ASR) technology has been around for over sixty years, and it is embedded in many products, from automated calls to personal assistants [1], [2]. However, ASR systems are far from perfect: their performance degrades quickly with background noise or far speech, and they tend to have a higher error rate if insufficient annotated training data is available.

Current state-of-the-art ASR systems rely on deep learning techniques [3]. Such algorithms made possible to train speech recognizers with different levels of background noise and microphone distances, moving away from previous approaches based on Gaussian mixture models and hidden Markov models systems [4].

Unfortunately, deep-learning-based ASR systems are strongly data driven, requiring considerable amounts of data to produce good models. Building a large corpus is time-consuming and not a trivial task. Open-source efforts, such as the Common Voice [5] led by Mozilla, were able to gather more than $1,000$ hours of English speech, far away from its original $10,000$ goal. For non-English languages, such as Dutch, only less than 31 hours of annotated speech are available (from a $1,200$ hours goal). These numbers are orders of magnitude below the ones that private companies, such as Baidu and Google, have been reporting with their results [6], [7].

Igor M. Quintanilha is with the Electrical Engineering Program, Federal University of Rio de Janeiro, Brazil (e-mail: igor.quintanilha@smt.ufrj.br).

Luiz W. P. Biscainho and Sergio L. Netto are with the Department of Electronics and Computer Engineering and the Electrical Engineering Program, Federal University of Rio de Janeiro, Brazil (e-mails: {wagner, sergioln}@smt.ufrj.br).

Digital Object Identifier: 10.14209/jcis.2020.25

The lack of annotated speech and public corpora makes it difficult to evaluate ASR systems for several languages, whose accuracies are, therefore, much worse than the ones reported for English and Mandarin [8]. This is especially true for languages with only a few dozen hours of public data available, such as Brazilian Portuguese (PT-BR). This work aims to establish a new baseline system for ASR using deep neural networks for PT-BR. By doing so, this work addresses the lack of a large annotated speech corpus by studying and showing how English-trained ASR systems can be beneficial for under-represented languages by applying transfer learning techniques. As a result of this work, we make the following contributions:

- A new PT-BR text corpus, by assembling three text corpora totalling 10.2 million sentences, among which the WikiText-PT-BR—built for this work by scraping text from the Wikipedia;
- An assorted PT-BR speech corpus containing a total of 158 hours of speech, by gathering four smaller datasets (three of which are free to distribute);
- An open-source pre-trained model with weights fine-tuned from the DeepSpeech 2-based architecture, so that anyone can evaluate or further improve the baseline end-to-end PT-BR speech recognizer;
- New open-source character- and word-level language models based on deep-learning techniques for PT-BR.

All development code employed in this work will be open-sourced under the MIT License and available at http://github.com/igormq/speech2text, as well as the speech and text corpora.

The remaining sections of this paper are organized as follows. Section II reviews the related works on the field of deep-learning ASR. Section III details the acoustic model and the loss function, whereas Section IV details the decoding strategy and Section V, the language model. Section VI specifies all speech and text datasets employed in this work. Sections VII, VIII, and IX describe the experimental results for the developed language model, acoustic model, and final ASR system, respectively. Finally, Section X concludes the paper emphasizing its main contributions.

II. Development context and general diagram of the system

Most languages in the world lack the amount of text, speech, and/or linguistic resources required to build large

models based on deep neural networks. There has been an increasing research interest on how to build a high-accuracy ASR system for languages with insufficient annotated data (using from hours to a few dozen hours of annotated speech), such as Brazilian Portuguese [8]; Indian languages (Gujarati, Tamil, and Telugu) [9], [10]; and Seneca (an Indigenous North-American language) [11].

To overcome data scarsity, Swietojanski et al. [12] did an unsupervised pretraining on different languages while Dalmia et al. [13] shared the same weights from the recurrent layers over different languages and trained additional layers to develop a multilingual ASR system, which improved the final word error rate by over 6% when compared to monolingual systems. Renduchintala et al. [14] proposed a multimodal data augmentation scheme for attention-based models, which only requires text data. Zhou et al. [15] showed that a single transformer network performs well on reduced training data in a multilingual setting. While [14] consists on training an extra encoder for its augmentation scheme, and [15] uses a transformer topology with more than 200 million parameters, our DeepSpeech 2 based model has fewer parameters (42 millions) and does not require extra parameters for pre-training.

The present paper extends upon [16], [17] by: showing the advances on adding a newly-trained external language model based on over 10.2 million sentences scrapped from the Portuguese Wikipedia; considering a transfer-learning approach to train an acoustic model based on a large English dataset; showing the impacts of adding more 148 hours of PT-BR speech in the acoustic-model training stage.

The overall system described in this paper is represented in Figure 1, and detailed in the next three sections.

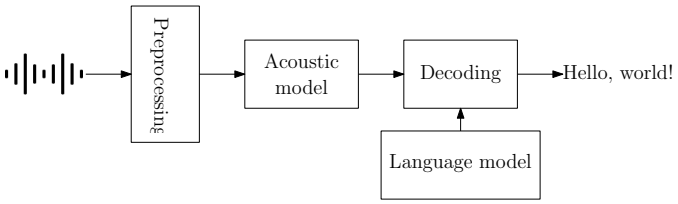


Fig. 1. The overall end-to-end automatic speech recognition system: the acoustic model is a DeepSpeech-2-based model trained using the connectionist temporal classification (CTC) loss function; the language model is an n-gram trained using the Kneser-Ney algorithm; and the decoding is the beam-search scheme adapted for the CTC algorithm.

III. Acoustic model and loss function

This section presents the acoustic model with the corresponding loss function employed in its development, similar to the work by Amodei et al. [6].

A. Acoustic model

Fig. 2 illustrates the deep-learning neural-network architecture considered in this work, which is based on the DeepSpeech 2 [6] model, introduced by Baidu, with two convolutional (Conv) and five bidirectional neural-network layers of the gated-recurrent unit (GRU) type.

Here, as in [6], we use a normalized power spectrogram calculated over the audio signal with $D = 161$ frequency bins as the network input. The first main layers are spatial convolutions, usually found in image-related tasks to increase the model capacity without exponentially increasing the number of parameters. In [6], the authors argue that a convolution in the frequency domain models the speakers' variability better than fully connected layers. Moreover, tuning the convolution-layer parameters, such as strides and kernel sizes, helps to release redundant information found in the input spectrogram, as well as to reduce the number of outputs to be fed into the subsequent and more expensive layers. Table I shows the parameters used in this work, which yield the best performance according to [6]. Each convolution layer is followed by a batch normalization layer [18], which improves the training speed by allowing higher learning rates, and a clipped ReLU non-linearity of the form $(\min(\max(x, 0), 20))$.

TABLE I
Description of convolutional layers in the DeepSpeech 2 model

	# channels	kernel	stride	padding	# parameters
Conv 1	32	(41, 11)	(2, 2)	(20, 5)	14,464
Conv 2	32	(21, 11)	(2, 1)	(10, 5)	236,576
TOTAL					**251,040**

The output from the convolutional layers is fed to a stack of five bidirectional recurrent layers. Different from [6], this work uses gated recurrent unit (GRU) instead of Elman's recurrent layer. The GRU is a simplified version of long short-term memories [19] where the forget and input gates are fused, thus reducing the overall number of parameters. In the bidirectional setting, there are two unidirectional GRUs for each layer, one proceeding forward and the other backward in time. Then, the two outputs are summed into a single output to be fed into the next layer.

After the bidirectional GRUs, one fully connected layer is employed to generate the unnormalized scores over the label set. At each timestep, the softmax layer predicts an augmented label given by the loss function. Finally, the predicted transcription is decoded from the given sequences according to the probability distributions. Given the input-output pair and the current network coefficients, the loss function and its gradient with respect to the network parameters can be calculated over the data batches. The gradient is then backpropagated through the network in order to update its coefficients.

B. Loss function

To build an ASR system, we need a dataset comprising audio clips and their corresponding transcriptions. Both input sequence $X = (x_1, \ldots, x_T)$ of size T, for $x_t \in \mathbb{R}^D$, where D is the input size, and its transcription $Y = (y_1, \ldots, y_U)$ of size U, for $y_u \in \mathbb{R}^L$, where L is the number of labels, may vary in length and ratio and may not have an accurate temporal alignment, which is not suitable for the usual supervised-learning setting. Performing manual alignment of the input sequence and its transcription requires intense human labor and it is too time-consuming.

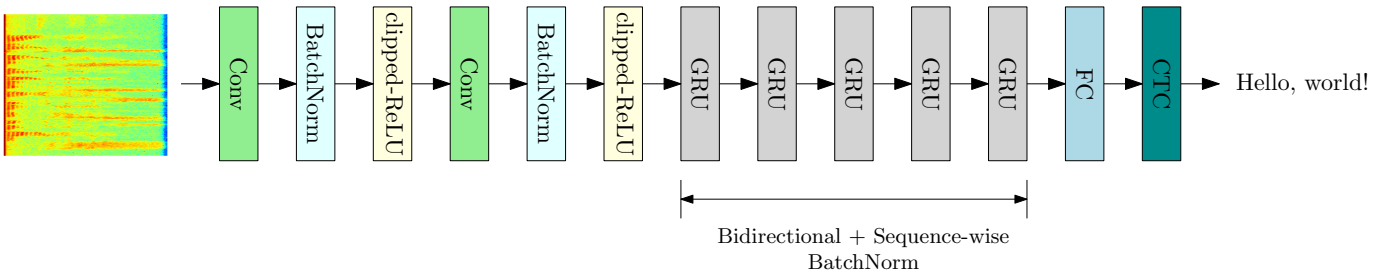


Fig. 2. DeepSpeech 2 model consisting of two convolutional (Conv) layers, each one followed by a batch normalization and clipped ReLU non-linearity, five gated-recurrent units (GRU), and one fully-connected (FC) layer, with interleaved batch-norm layers, totalling over 30 million parameters.

The well-known connectionist temporal classification (CTC) [20] approach is an alignment-free algorithm which considers all possible alignments between X and Y before opting for the most probable one. To handle repeated characters and regions of speech that do not contain any audio, the CTC introduces a special blank token ϵ. The alignments considered by CTC are of the same length as the input X and must map to the output Y after merging character repetitions and removing ϵ tokens. As an example, a valid alignment for $Y = [h, e, l, l, o]$ with input size of $T = 8$ can be $[h, h, e, l, \epsilon, l, o, \epsilon]$, while $[\epsilon, h, h, e, l, l, o, \epsilon]$ is an invalid alignment which leads to $[h, e, l, o]$. Therefore, the CTC alignment approach between X and Y is a many-to-one operation where the length of Y cannot be greater than the length of X.

The CTC approach aims at maximizing the log-likelihood $\log p(Y|X)$ of the label sequence given the inputs, such that

$$p(Y|X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X), \tag{1}$$

where $\mathcal{A}_{X,Y}$ is the set of all possible alignments $A = (a_1, \ldots, a_t)$ between X and Y and the output probabilities $p_t(a_t)$ at each time step t are assumed to be independent given X. The probability $p_t(a_t|X)$ can be estimated using any learning algorithm which produces a distribution over output classes (e.g., number of characters plus blank label) given a fixed-size slice of the input. Usually, as considered in this work, a recurrent neural network (RNN) is employed to estimate $p_t(a_t|X)$ [21].

Summing over all possible CTC alignments is computationally impractical. Fortunately, the likelihood evaluation can be efficiently computed via dynamic programming [4]. The overall CTC loss function is the colog probability of correctly labeling the entire training set $\mathcal{D}$, that is

$$J(\mathcal{D}) = -\sum_{(X,Y) \in \mathcal{D}} \log p(Y|X). \tag{2}$$

The loss $J(\mathcal{D})$ is differentiable with respect to the output probabilities, and can be used by any gradient-based optimization method to update the network coefficients.

The CTC algorithm can directly map speech into text without any alignment by considering all possible conversion paths. To do so, the CTC assumes that the output symbols are conditionally independent of each other given the input, i.e., no linguistic information is directly imposed in the process, which means that the acoustic and language models are separated. While this separation allows for domain independence and adaptation or reuse of some of the speech recognition components, this independence also brings at least one drawback to the decoding scheme: the model does not know what a word is and how the previous and current symbols correlate. This aspect, despite simplifying the ASR development, reduces the overall system performance in practice. The CTC decoding schemes discussed in the next section aim to overcome this issue.

IV. CTC DECODING WITH A LANGUAGE MODEL

This section describes a decoding procedure aided by an external language model that can potentially improve the performance of CTC-based ASR systems. We also described thoroughly the beam-search decoding algorithm for both character-level and word-level language models in a same framework, which has been previously handled separately by different works [22], [23].

There are several ways of decoding the output, each one with its pros and cons. The best-path or greedy decoding [24], for instance, is the fastest decoding scheme. It works by considering that the highest symbol-probability at each step yields the best hypothesis, that is,

$$A^* = \arg\max_{A} \prod_{t=1}^{T} p_t(a_t|X). \tag{3}$$

Then, it collapses the character duplicates and removes all blanks to get the final transcription. The best-path decoding is both fast and straightforward. For many applications, this approach works quite well, mainly when most of the probability mass is allocated to a single alignment, such as in handwriting recognition [24].

In practice, however, the primary goal of the decoding algorithm is not to find the best instantaneous match, but to find the final transcription with the highest probability, and a single transcription can have many paths. More precisely, we want to solve

$$Y^* = \arg\max_{Y} p(Y|X). \tag{4}$$

In the example illustrated in Fig. 3, alignments $[a, a]$, $[a, \epsilon]$, and $[\epsilon, a]$, which lead to the same CTC decoding

$Y = [a]$, have lower individual probabilities than $[\epsilon, \epsilon]$, which corresponds to $Y = []$ (empty output). However, the sum of their probabilities is greater than the probability of $[\epsilon, \epsilon]$ ($0.48 > 0.42$). In this situation, the best-path decoding would incorrectly define $Y = []$ as the most likely hypothesis, while a good decoding scheme should have chosen $Y = [a]$.

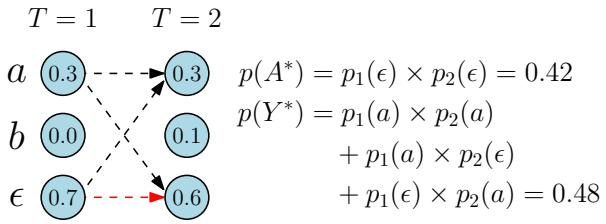


Fig. 3. CTC probabilities for time-steps 1 and 2 with an alphabet $\{a, b, \epsilon\}$. The best path algorithm will wrongly decode an empty output $Y = []$ (red dashed line) while the best output is $Y^* = [a]$.

As mentioned in the previous section, the standard CTC algorithm does not include the constraint of either a lexicon or a language model. Indeed, under lexicon/language model constraints the best hypothesis is defined as

$$Y^* = \arg\max_Y p(Y|X)\, p_{\text{lm}}(Y), \tag{5}$$

where $p_{\text{lm}}(Y)$ is the language-model prior distribution. In practice, this prior distribution may be too restrictive, and thus it is down-weighted by a factor of $\alpha > 0$ and a penalty (or bonus) $L(Y)$ controlled by $\beta \in \mathbb{R}$ is also included into the best hypothesis, which then becomes

$$Y^* = \arg\max_Y p(Y|X)\, p_{\text{lm}}(Y)^\alpha L(Y)^\beta, \tag{6}$$

where the hyperparameters α and β are set by cross-validation.

The most famous decoding algorithm, which takes into account the many-to-one mapping and may include lexicon/language model constraints, is the beam-search decoding [22], [23]. This scheme interactively searches for the best hypothesis in a tree of hypotheses and is flexible enough to handle both constrained and unconstrained vocabularies.

At each time step, the beam search computes a new set of hypotheses (beams), generated from the previous set by extending each hypothesis with all possible output symbols (labels) and keeping only the top candidates. Given its limited computation, the beam-search algorithm does not find the most probable output (Eq. (4)), but it empowers the machine-learning practitioner to trade-off computation (i.e., a larger beam size) for an asymptotically better solution.

The vanilla beam search needs some adjustments to handle the CTC many-to-one mapping. Instead of keeping a list of alignments in the beam, the CTC beam-decoding algorithm stores the output prefixes after the mapping (i.e., collapsing the repeats and removing the blank symbols). At each step, the algorithm accumulates the score of a given prefix.

By storing the outputs after mapping, a new hypothesis can now map to two different prefixes if the character is a repeat, as exemplified in Fig. 4 for $T = 3$. Both $[a]$ and $[a, a]$ are valid outputs for this new hypothesis. To generate $[a, a]$, a blank symbol is required between repeated characters; then, we must only consider in the new score the part of the alignment that ends with ϵ. Contrarily, to generate $[a]$, we must only consider the part of the previous score for alignments, which does not end with ϵ.

Algorithm 1 outlines the employed decoding procedure. Instead of one score for each beam, the CTC beam-search algorithm has to keep track of two probabilities for each prefix in the beam, $p_{\text{b}}(\hat{Y}|X_{1:t})$ and $p_{\text{nb}}(\hat{Y}|X_{1:t})$, the probabilities of the candidate prefix $\hat{Y}$ ending in ϵ or not, respectively, given the first t time steps of the input X. The final score for a given prefix $\hat{Y}$ is the sum of these two probabilities, i.e., $p(\hat{Y}|X_{1:t}) = p_{\text{b}}(\hat{Y}|X_{1:t}) + p_{\text{nb}}(\hat{Y}|X_{1:t})$. The hypothesis sets $\mathcal{H}_{t-1}$ and $\mathcal{H}_t$ maintain a list of active prefixes at the previous and current time steps, respectively, and $\mathcal{H}_{t-1}$ is never larger than the beam width k. Also, variable $\hat{Y} + l$ is the concatenation of the label l with the prefix $\hat{Y}$, while $\hat{Y}_{t-1}$ is the last label in the prefix $\hat{Y}$.

The language model constraint is only added for a new prefix $\hat{Y}^+$ if the language model is character-based or if l is a space in the case of a word-based language model. Finally, the overall probability of a prefix is a product of a language model insertion term and the sum of non-blank and blank probabilities, that is

$$(p_{\text{nb}}(Y|X_{1:t}) + p_{\text{b}}(Y|X_{1:t}))L(Y)^\beta, \tag{7}$$

where $L(Y)$ is the number of characters in Y for a character-based language model or the number of words for a word-based language model.

The final decoding scheme is significantly simpler and faster compared to weighted finite-state transducers [25], can naturally handle out-of-vocabulary words, and uses an arbitrary language model (i.e., character/word level, rule/neural-network based).

V. Language Model

In the previous section, we described the beam-search decoding for CTC, which may add linguistic information through a language model. This section walks through one of the main approaches to construct a language model $p_{\text{lm}}(\cdot)$ using a non-parametric model based on counting statistics, the Kneser-Ney algorithm [26], [27]. In the end, we make a brief discussion about other language models based on neural networks [28], [29], [30], [31].

A. Statistical language model

A language model computes the probability over a sequence W of N words [32], w_n, i.e.,

$$p(W) = p(w_1, w_2, \ldots, w_N), \tag{8}$$

which can be factorized, using the chain rule, as

$$p(W) = \prod_{k=1}^{N} p(w_k|w_1, w_2, \ldots w_{k-1}) = \prod_{k=1}^{N} p(w_k|W_{1:k-1}). \tag{9}$$

Given a text corpus, each conditional probability in this equation can be estimated by counting the number of occurrences of the sequence of words, $C(W_{1:k})$, normalized by the

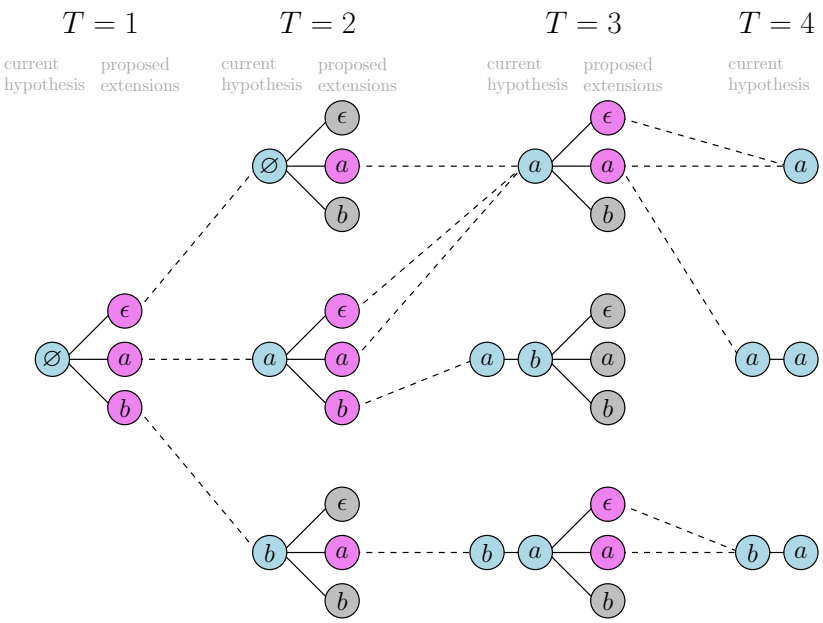


Fig. 4. Connectionist temporal classification (CTC) beam search with an alphabet $\{\epsilon, a, b\}$ and a beam size of 3: (i) multiple extensions can merge to the same hypothesis (e.g., in $T = 2$, ϵ and a merge to a in $T = 3$); (ii) an extension can split into two hypotheses (e.g., in $T = 3$, a splits into a and $[a, a]$ in $T = 4$); (iii) multiple extensions can merge to the same prefix (e.g., in $T = 3$, ϵ and a merge to $[b, a]$ in $T = 4$). Adapted from [21].

Algorithm 1 Connectionist Temporal Classification (CTC) beam-search decoding, a unified approach for both [22], [23].

Input CTC likelihoods $p_t(a_t|X_t)$, $\forall t$.
Parameters beam width k
Initialize $\mathcal{H}_{t-1} = \{\varnothing\}$, $p_{\text{b}}(\varnothing|X_{1:0}) = 1$, $p_{\text{nb}}(\varnothing|X_{1:0}) = 0$

for $t = 1, \ldots, T$ **do**
  $\mathcal{H}_t \leftarrow \{\}$
  for $\hat{Y}$ **in** $\mathcal{H}_{t-1}$ **do**
    $p_{\text{b}}(\hat{Y}|X_{1:t}) \leftarrow p_{\text{b}}(\hat{Y}|X_{1:t}) + p_t(\epsilon|X_t)p(\hat{Y}|X_{1:t-1})$ ▷ Handle blanks
    $p_{\text{nb}}(\hat{Y}|X_{1:t}) \leftarrow p_{\text{nb}}(\hat{Y}|X_{1:t}) + p_t(\hat{Y}_{t-1}|X_{1:t})\, p_{\text{nb}}(\hat{Y}; X_{1:t-1})$ ▷ Handle repeated character collapsing
    Add $\hat{Y}$ to $\mathcal{H}_t$
    for l **in** $\mathcal{L} \setminus \{\epsilon\}$ **do**
      $\hat{Y}^+ \leftarrow \hat{Y} + l$ ▷ Extend the current prefix with character l
      if l **is equal** $\hat{Y}_{t-1}$ **then**
        $p_{\text{temp}}(\hat{Y}^+|X_{1:t}) \leftarrow p_t(l|X_t)\, p_{\text{b}}(\hat{Y}|X_{1:t-1})$ ▷ Repeated labels have "ϵ" between
      else
        $p_{\text{temp}}(\hat{Y}^+|X_{1:t}) \leftarrow p_t(l|X_t)p(\hat{Y}|X_{1:t-1})$
      end if
      if p_{lm} **is** character based **or** (l **is** space **and** $\hat{Y}_{t-1}$ **is not** space) **then**
        $p_{\text{temp}}(\hat{Y}^+|X_{1:t}) \leftarrow p_{\text{temp}}(\hat{Y}^+|X_{1:t})\, p_{\text{lm}}(\hat{Y}^+)^{\alpha}$
      end if
      $p_{\text{nb}}(\hat{Y}^+|X_{1:t}) \leftarrow p_{\text{nb}}(\hat{Y}^+|X_{1:t}) + p_{\text{temp}}(\hat{Y}^+|X_{1:t})$
      Add $\hat{Y}^+$ to $\mathcal{H}_t$
    end for
  end for
  $\mathcal{H}_t \leftarrow k$ most probable $\hat{Y}$ by $p(\hat{Y}|X_{1:t})L(\hat{Y})^{\beta}$ in $\mathcal{H}_t$
end for
Return $\arg\max_{Y \in \mathcal{H}_t} p(Y|X_{1:T})L(Y)^{\beta}$

total times of the sequence of words minus the last word, $C(W_{1:k-1})$, that is,

$$p(w_k|W_{1:k-1}) = \frac{C(W_{1:k})}{C(W_{1:k-1})}, \tag{10}$$

However, computing the conditional probability would require virtually infinite data. Under a Markov assumption, we can by approximation condition it to just a few words, that is

$$p(w_k|W_{1:k-1}) \approx p(w_k|W_{k-n+1:k-1}), \tag{11}$$

such that the probability over a sequence of words becomes

$$P(W) = \prod_k p(w_k|W_{k-n+1:k-1}), \tag{12}$$

which is the definition of a vanilla n-gram model. For the unigram model ($n = 0$), the simplest case, the conditional probability is approximated by the probability of the current word, whereas in a bigram model ($n = 1$) the conditional probability is only conditioned to the previous word.

B. Perplexity

It is expected that a trained language model would assign higher probabilities to frequently observed sequences than to rarely observed ones. In practice, however, how can we evaluate a trained model? One way is using the extrinsic evaluation, i.e., testing it in the task it was designed for (e.g., ASR) and comparing the resulting evaluation metrics (e.g., WER–word error rate). This method is time-consuming; besides, many factors can impact the performance of the task, and hide some training problems. An alternative way, commonly used in the ASR context, is the intrinsic evaluation using the so-called perplexity measure, which is the inverse probability of the test set, normalized by the number of words, that is

$$\begin{aligned} PP(W) &= p(w_1, w_2, \ldots, w_N)^{-1/N} \\ &= \sqrt[N]{\frac{1}{p(W)}} \\ &= \sqrt[N]{\prod_k \frac{1}{p(w_k|W_{k-n+1:k-1})}}. \end{aligned} \tag{13}$$

In other words, perplexity measures how well the model can predict the next word. A better text model is one that assigns a higher probability to the word that actually occurs. Better models mean minimizing the perplexity or maximizing the $P(W)$ probability.

C. Generalization

For any n-gram that occurred a significant number of times, the trained model usually provides a good probability estimate. As the training data are limited, however, many acceptable n-grams are bound to be missing, being assigned zero probability in spite of having non-zero probability in real-world scenarios. This means that our model is underestimating the probability of several word sequences, which can hurt its performance, and that the entire perplexity of the test set cannot be calculated due to divisions by zero.

One way to overcome the zero probabilities is called smoothing discounting. The intuition in this approach is to adjust the probability mass to generalize better. The simplest algorithm is the 1-add smoothing (or Laplacian smoothing), which adds one to all n-gram counts. All the counts that used to be zero will now have a count of one, the counts of one will be two, and so on. However, there is additional information that the model can rely on instead of "adding one". If an n-gram model does not find a particular n-gram, the model can instead estimate the probability by using the $(n-1)$-gram. Similarly, if $(n-1)$-gram model does not have a particular $(n-1)$-gram count, the model can look at $(n-2)$-gram and so on down to the unigram model. In this so-called backoff approach, the n-gram evidence is used when it is sufficient; otherwise, the language model uses lower-order n-gram information. For a backoff n-gram model to yield a correct probability distribution, one has to discount the higher-order n-grams to save some probability mass for the lower order n-grams.

One of the most commonly used and best performing n-gram smoothing methods is the interpolated Kneser-Ney algorithm, which is based on the absolute discount [33] and subtracts a fixed (absolute) discount d from each count. The Kneser-Ney discounting augments absolute discounting with a more sophisticated way to handle the lower-order n-gram distribution. A standard n-gram model assigns higher probabilities to frequent occurrences, but the Kneser-Ney discounting estimates the probability of occur the n-gram as a novel continuation in a new unseen context.

For example, consider that for a training data the word *Francisco* is more common than *glasses*, since *San Francisco* is a very frequent word. The Kneser-Ney method captures the intuition that although *Francisco* is more frequent, it is mainly only frequent alongside *San*, whereas the word *glasses* has a much wider distribution. The number of times that a word w appears as a novel continuation can be expressed as

$$p_{\text{CONT}}(w) = \frac{|\{w_i : C(w,w_i) > 0\}|}{|\{(w_i, w_j) : C(w_i, w_j) > 0\}|}, \tag{14}$$

where the numerator is the cardinality of the set containing all the non-zero counts, i.e. $C(\cdot) > 0$, of the sequence $W = [w, w_i]$ for all i, and the denominator is the normalization factor.

Then, a frequent word *Francisco* occurring in only one context (*San*) will have a low continuation probability p_{CONT}. The final equation for interpolated Kneser-Ney smoothing is then

$$\begin{aligned} p_{\text{KN}}(w_k|W_{k-n+1:k-1}) = \frac{\max(C_{\text{KN}}(W_{k-n+1:k}) - d, 0)}{\sum_{w_i} C_{\text{KN}}(W_{k-n+1:k-1w_i})} + \\ \lambda(W_{k-n+1:k-1}) p_{\text{CONT}}(w_k|W_{k-n+2:k-1}), \end{aligned} \tag{15}$$

where

$$\begin{aligned} &\lambda(W_{k-n+1:k-1}) = \\ &\quad \frac{d}{\sum_{w_i} C(W_{k-n+1:k-1}w_i)} |\{w : C(W_{k-n+1:k-1}w) > 0\}| \end{aligned} \tag{16}$$

is a normalizing constant that distributes the discounted probability mass, $\frac{d}{\sum_{w_i} C(W_{k-n+1:k-1}w_i)}$ is the normalized discount, and $|\{w : C(w_{k-1}w) > 0\}|$ is the number of times the normalized discount was applied. In these expressions, C_{KN} is given by

$$C_{\text{KN}}(\cdot) = \begin{cases} C(\cdot), & \text{for the highest order,} \\ C_{\text{CONT}}(\cdot), & \text{for lower orders,} \end{cases} \tag{17}$$

where $C_{\text{CONT}}(\cdot)$ is the number of unique single-word contexts for "·".

A modified version of the Kneser-Ney smoothing [27] employed in this work uses three different discounts d_1, d_2, and d_{3+} for n-grams with counts of one, two, and three or more, respectively.

Neural language models address both n-gram data sparsity and limited context issues. Due to their parametric model, such models require more data and take longer than statistical language models to train. The n-gram data sparsity problem is addressed through word embeddings (representing each word as a real-valued vector instead of a one-hot vector) and using them as inputs to a neural network [28], [29]. The key idea behind word embedding is to create semantic relationships between words in the feature space, i.e., words like "cat" and "dog" should somehow be close in the embedding space, since they tend to appear in similar contexts. Many language model methods use RNNs [29], [34] and, more recently, transformers-like architectures [35], [36] were proposed in the literature, achieving state-of-the-art results on many datasets. However, in this work, a statistical language model using the Kneser-Ney algorithm is employed, as it leads to comparable results under data-resource constraints, which are inherent in the PT-BR scenario, leaving the use of a neural language model as a future work.

VI. Datasets

So far, we described how to build, to train, and to evaluate both the acoustic and language models employed in this work. As explained in Section I, the key idea of this paper is to train a backbone acoustic model using an open available English dataset, and then fine-tune it using our PT-BR speech dataset. Finally, we use a PT-BR language model to increase even further the system accuracy. This section describes all datasets employed to train both the acoustic (backbone and fine-tuned) and the language models. For the acoustic model, three datasets were used:

1) LibriSpeech: an English dataset containing nearly 1000 hours of speech and freely available, to train the backbone model;
2) Brazilian Portuguese speech dataset (BRSD) v1 [8], with approximately 14 hours of PT-BR speech;
3) Brazilian Portuguese speech dataset (BRSD) v2, containing the first version (BRSD v1) plus extra 144 hours of speech data.

The separation of BRSD v1 and v2 is employed in this work to illustrate the effect of additional training data in the final performance of the ASR system and how the use of language models can mitigate the problem of having a few hours of annotated speech corpus.

For the language model, we built a single PT-BR text dataset (BRTD) by combining three datasets publicly available: LapsNews [37]; CETENFolha [38]; New WikiText PT-BR, including scraped data from the Wikipedia website, retrieving more than 8 million sentences, which is 71 times larger than LapsNews and 5.5 times larger than CETENFolha.

A. Acoustic model datasets

1) LibriSpeech: The LibriSpeech [39] is a speech corpus derived from reading audiobooks from the LibriVox project, totalling almost $1,000$ hours of reading speech sampled at 16 kHz. Due to its massive amount of data, the LibriSpeech corpus is a perfect candidate to pre-train end-to-end ASR models.

2) BRSD v1: This dataset is an ensemble of three publicly available datasets (Sid, VoxForge, LapsBM) and one paid (PT-BR Spoltech) [8]. It contains almost 14 hours of non-conversational speech, totalling 425 different speakers, and more than $12,000$ utterances sampled at 16 kHz in a non-controlled environment:

- Sid dataset: kindly provided by Dr. Sidney dos Santos for research purposes, this dataset contains recordings by 72 speakers (20 women) from 17 to 59 years old with fields such as place of birth, age, gender, education, and occupation. Recorded at 22.05 kHz in a non-controlled environment, its $5,777$ utterances were transcribed at word level without time alignment. Contents span from spoken digits, single words, complex sequences, spelling of name, and local of birth to phonetic covering, and semantically unpredictable sentences. Some excerpts were discarded due to a systematic transcription error found.
- Voxforge [40] dataset: its intent is distributing transcribed speech audio under general public license to aid the development of acoustic models. Everyone can record and (anonymously or not) send specific utterances, which makes for the most heterogeneous *corpus*. Its Brazilian Portuguese section contains recordings by at least 111 speakers, not always with gender/age information, at different sample rates ranging from 16 kHz to 44.1 kHz, many with a low signal-to-noise ratio. Its $4,130$ utterances are transcribed at the word level.
- LapsBM1.4 [41] dataset: the "Fala Brasil" group at the Federal University of Pará uses this *corpus* to evaluate PT-BR ASR systems. It contains recordings of 20 unique utterances by each of 35 speakers (10 women), totaling 700 excerpts, at a 22.05 kHz sample rate without environment control.
- CSLU Spoltech dataset version 1.0 [42]: distributed by the Linguistic Data Consortium (LDC) under catalog number LDC2006S16, this *corpus* includes recordings by 477 speakers from several regions in Brazil in either reading speech (for phonetic coverage) or (spontaneous) responses to questions. From its $8,080$ utterances recorded at a 44.1 kHz sampling rate in a non-controlled environment, $2,540$ have been transcribed at word level alignment, and $5,479$ at phoneme level with time alignment. As pointed in [43], some audio recordings have lacking or erroneous transcriptions.

All datasets above were pruned of samples with no/wrong transcription, too short recordings, and other defects that could produce wrong results. All audio files were re-sampled at 16 kHz. Recording lengths concentrate around 3 s but can reach up to 25 s.

3) BRSD v2: This second BRSD version includes its previous version, BRSD v1, plus the CETUC dataset [44], which totals almost 145 hours of speech signals performed by 50 male and 50 female speakers, each one pronouncing $1,000$ phonetically balanced sentences selected from the CETEN-

Folha corpus [38]. The CETUC dataset was recorded in a controlled environment at a sampling rate of 16 kHz.

For the reader's convenience, a summary of all acoustic datasets employed in this work is provided in Table II. For the sake of comparison, the most important datasets employed in the ASR literature are the 5.4-h TIMIT [45], the 73-h Wall Street Journal (WSJ) [46], [47], the 300-h Switchboard [48], and the 1,000-h Librispeech [39], all in English. For its non-controlled environmental conditions, multiplicity of acquiring hardware, and distinct speaker dialects, the 158-h BRSD v2 is far more stringent than the WSJ. In addition, Switchboard contains conversational speech, which is also not found in BRSD v2.

TABLE II
SUMMARY OF ALL SPEECH CORPORA USED IN THIS WORK (IN THE THIRD COLUMN, HH:MM = HOURS:MINUTES; THE LAST COLUMN GIVES THE NUMBER OF DIFFERENT SPEAKERS IN EACH DATASET.)

			Words		Speakers	
Dataset	Subset	HH:MM	Total	Unique	M	F
Librispeech	-	1,000	-	-	1,283	1,201
BRSD v1	Sid	7:23	33,189	5,676	52	20
	VoxForge	4:14	20,879	729	111	
	Spoltech	1:35	16,776	558	477	
	LapsBM	0:54	7,228	2,731	25	10
	Total	14:07	78,072	7,772	-	
BRSD v2	CETUC	144:39	1,040,278	3,528	50	50
	Total	158:47	1,118,350	8,328	-	

B. Language model datasets

The Brazilian Portuguese text dataset (BRTD) comprises the following three text datasets:

- LapsNews [37] is a text corpus dataset consisting of automatic crawling of the top ten daily Brazilian newspapers available on the Internet in 2010. It was post-processed to convert to lowercase letters, to remove tags and punctuation marks, and to expand numbers and well-know acronyms to the written form, resulting in a corpus with approximately 120k sentences.
- CETENFolha [38] dataset is a corpus containing over 24 million Brazilian words crawled from the 1994 editions of Folha de São Paulo newspaper, resulting in approximately 1.6M sentences.
- New WikiText PT-BR dataset is a collection of over 8 million sentences extracted from the Wikipedia articles and was built for this work. The dataset is available under Creative Commons Attribution-ShareAlike[1] license. The dataset was pre-processed to remove all punctuation and tags and convert numbers into their written form, to be well suited for ASR models.

A summary of all language model datasets used in this work is given in Table III.

[1]https://en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License

TABLE III
SUMMARY OF ALL TEXT CORPORA USED IN THIS WORK

		Words	
Datasets	#sentences	Total	Unique
LapsNews	119k	2.7M	66k
CETENFolha	1.5M	26M	214k
WikiText PT-BR	8.5M	194M	1.1M
Total	10.2M	223M	1.2M

VII. LANGUAGE-MODEL EXPERIMENTS

We consider as a baseline model the LapsLM [37], which is a word-level 3-gram model trained with the modified Kneser-Ney smoothing technique and is freely available to use.

When training the PT-BR language model, we used the BRTD, keeping all words with at least three occurrences in the dataset, totalling almost 512k words. Words not in the vocabulary were replaced by 'unknown'. We also removed sentences that were in common with the LapsBM and CETUC datasets so as not to bias our result. We split the text corpus into two sets: the training set, containing 90% of all sentences, and the test set with the remaining ones. We evaluate the word perplexity in both test sets and the LapsBM dataset (by extracting the utterances).

All n-gram models were trained with the KenLM [49] algorithm, and differently from [6], the use of word- and character-level language models is studied in the context of a much smaller speech corpus. Two n-gram word models, with $n = 3$ and $n = 5$, were trained using the same parameters as the LapsLM. For the character-level language models, we study how the context influences perplexity by training n-grams with $n = 5, 10, 15, 20$ following the same procedure as in [50]. In the $n = 15$ and $n = 20$ cases, rare sequences were pruned according to: $6, 7, 8$-grams appearing only once, 9-grams appearing once or twice, and for $n \geq 10$ all n-grams with less than 4 appearances were dropped.

In order to compare word-level and character-level models, one may estimate the word-level perplexity in all cases. In this strategy, following [50], given a previous context C, the word probability can be estimated:

$$p(\mathrm{w}|C) = p(l_1|C) \prod_{i=2}^{n} p(l_i|C, l_1, \ldots, l_{i-1}), \qquad (18)$$

where $l_1, \ldots, l_n$ are the letters in a word, w. This approach, used by [50], does not take into account that word-level models are constrained to a fixed-size lexicon, while character-based models have virtually an infinite vocabulary size. Therefore, Eq. (18) shall be used only as an upper bound for the true word-level perplexity in the character-level models.

The word-level perplexity results attained by different language models, along with their respective receptive fields (average characters seen in all n-grams) and storage size, are shown in Table IV. As one can see, the LapsLM presents the best perplexity results over the LapsBM dataset, but it does not generalize over the BRTD test set as well as some of the other models. The LapsLM model might have been trained on a smaller dataset probable in a similar text domain

of the LapsBM test set, which may explain the difference in perplexity between the two test sets (138 PP gap), while ours was trained on more text, exhibiting better generalization over different test sets. As expected, increasing the context increases the model complexity and decreases the perplexity, indicating a better performance. For $n = 15, 20$ both character-level models exhibit competitive performances against the trained word-level models while having virtually infinite vocabulary.

TABLE IV
GENERAL CHARACTERISTICS FOR SEVERAL LANGUAGE MODELS: RECEPTIVE FIELD (RP, MEASURED IN CHARACTERS), NUMBER OF PARAMETERS, AND PERPLEXITY RESULTS OVER THE LAPS BENCHMARK (LAPSBM) AND THE NEWLY BRAZILIAN TEXT DATASET (BRTD) TEST SET (MODELS MARKED WITH * WERE PRUNED DURING TRAINING, AS EXPLAINED IN THE MAIN TEXT. THE FOUR LAST LINES GIVE UPPER BOUNDS FOR THE TRUE PERPLEXITY VALUE.)

Language model	RP	Size	LapsBM	BRTD
LapsLM (3-gram)	25	124M	103.54	297.20
word 3-gram	25	1.9G	173.79	161.29
word 5-gram	42	7.8G	136.50	135.12
char 5-gram	5	41M	≤ 2,334.48	≤ 2,694.51
char 10-gram	10	4.7G	≤ 271.86	≤ 323.71
char 15-gram*	15	5.4G	≤ 239.59	≤ 198.49
char 20-gram*	20	8.8G	≤ 227.84	≤ 189.53

In Section IX, we investigate how these newly-trained language models may improve the ASR performance when combined to the proposed acoustic model.

VIII. ACOUSTIC-MODEL EXPERIMENTS

In this section, we develop new acoustic models based on the DeepSpeech 2 deep neural-network architecture. All hyperparameter tuning was performed on validation sets, and the final performance was evaluated on the test sets. We use tempo and gain perturbation as data augmentation in all experiments [51]. We randomly modified the tempo in about 85% to 115% of the original rate, while the gain was randomly altered from −6 dB to 8 dB from the original one. System performances are evaluated with respect to their final word error rate (WER) and character error rate (CER), which are the edit distances at word and character levels, respectively. In each case, we consider as the best model the one that achieves the lowest WER in the validation set.

Over the next sections, we show the results for ASR models trained on BSRD v1 and v2, discuss how using a well-trained language model can further improve the results and even reduce the gap between the speech recognition systems trained on a smaller and a bigger speech corpora, respectively.

A. Backbone model training

When training the backbone acoustic model for English, we employed the same training, validation, and test sets as provided in the original publication [39]. In particular, there are two versions for each of the validation and test sets, containing either clean or noisy speech, respectively.

Table V summarizes the backbone model architecture and related hyperparameters. The network input is the normalized spectrogram, as described in Section III, calculated using a Hamming window of 320 samples and a hop size of 160 samples, resulting in $D = 161$ frequency bins. Each recurrent layer has $H = 800$ hidden units, and the output alphabet contains $C = 29$ labels corresponding to the {A, B, ..., Z} letters plus the apostrophe, space, and blank characters.

Network training was carried out using the stochastic gradient-descent method with momentum [3] with a learning rate of 4.8×10^{-4}, a momentum of 0.9, an annealing rate of 0.9091, and a gradient norm clipping [52] of 400 for over 20 epochs with a batch size of 16. In the first epoch, we sort out the utterances by their lengths, accelerating the network training, as proposed in [6]. After the first epoch, the batches are randomly organized. The predicted sequence is decoded using the greedy search [20]. Table VI shows the results of the backbone model, which are comparable to the ones found in the literature without a proper language model for decoding and extra data. The Paddle Paddle [53] implementation differs from others by adopting larger recurrent layers (with 2,048 hidden units each) and a different activation function, while Sean Naren's implementation [54] employs a different padding scheme in the convolutional layers.

B. PT-BR model training

For the PT-BR acoustic model, two experiments were performed, using v1 and using v2 of the BRSD dataset (see Subsection VI-A), in order to evaluate the impact of their sizes (approximately 14 and 159 hours, respectively) in the final results.

For the BRSD v1 experiment, we follow the previous work [16], [17] by using same validation set (termed `v1-val`), containing 21 speakers from LapsBM dataset, whereas the training set (`v1-train`) comprised the Sid, VoxForge, and CSLU datasets. It is worth mentioning that both LapsBM and CETUC datasets were built using sentences gathered from the CETENFolha dataset, so they are not utterance independent, which may bias our results.

To mitigate such utterance contamination in the BRSD v2 experiment, we considered a new test set (`v2-test`) using 20 speakers from the CETUC dataset, each one speaking only 200 sentences out of the possible 1000 ones instead of the remaining speakers of the LapsBM, as used in [16], [17]. We use the `v2-test` in both BRSD v1 and v2 experiments to report the final results. The other 80 speakers in the CETUC set were used to expand the `v1-train` set into the `v2-train` set with the remaining 800 sentences not included in the test set. We made three different 200 : 800 random splits, and the results are reported in the form of mean and variance. Finally, the validation set (`v2-val`) in the BRSD v2 experiment remained the same as in the v1 case.

We conduct an experiment to compare the performances attained without pre-training (i.e., training from scratch) and with pre-training (i.e., fine-tuning the model obtained in Subsection VIII-A).

The PT-BR model has a broader character set to include Brazilian Portuguese accents, so the number of characters is $C = 43$. Since the number of characters is different in the

TABLE V
DEEPSPEECH 2 BACKBONE ARCHITECTURE AND HYPERPARAMETERS (THE NETWORK INPUT IS THE NORMALIZED SPECTROGRAM USING A HAMMING WINDOW OF 320 SAMPLES AND A HOP SIZE OF 160 SAMPLES, RESULTING IN 161 BINS. EACH RECURRENT LAYER HAS 800 HIDDEN UNITS, AND THE OUTPUT CONTAINS $C = 29$ LABELS. FOR TRAINING, WE USE THE STOCHASTIC GRADIENT-DESCENT METHOD WITH MOMENTUM OF 0.9, A LEARNING RATE OF 4.8×10^{-4}, AN ANNEALING RATE OF 0.9091, AND GRADIENT NORM CLIPPING OF 400 FOR 20 EPOCHS WITH A BATCH SIZE OF 16. VIEW OPERATION IS A RESHAPE OVER THE INPUTS.)

	Operation	Kernel size	Stride	Feature maps	Padding	Nonlinearity
Network - Input	$B \times 1 \times 161 \times T$					
	Convolution	41×11	2×2	32	20×5	BN-clippedReLU
	Convolution	21×11	2×1	32	10×5	BN-clippedReLu
	View	$B \times 32 \times 41 \times T_{\text{out}} \rightarrow T_{\text{out}} \times B \times 32 * 41$				
x 5	BatchRNN	*hidden size:* 800				
	BN					
	View	$T \times B \times 800 \rightarrow B \times T \times 800$				
	FC	*output size:* $B \times T \times 29$				softmax + CTC
BatchRNN Module						
	Sequence-wise BN					
	Bidirectional GRU					tanh
Sequence-wise BN Module	t, b, d					
	View	$t \times b \times d \rightarrow t * b \times d$				
	BN					
	View	$t * b \times d \rightarrow t \times b \times d$				
	Preprocessing	Normalized linear spectrogram (window size = 320, hop size = 160)				
	Optimizer	SGD with momentum (lr= 4.8×10^{-4}, momentum= 0.9), SortaGrad enabled				
	Max gradient norm	400				
	Learning rate annealing	0.9091				
	Batch size	16				
	Epochs	20				
	Decoding	Greedy search decoder				

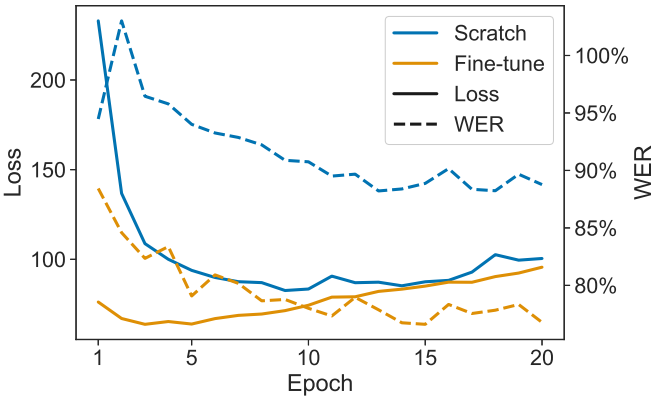
TABLE VI
BACKBONE MODEL WORD ERROR RATE (WER) COMPARED WITH OTHER IMPLEMENTATIONS FOUND IN THE RELATED LITERATURE (THE PADDLE PADDLE [53] IMPLEMENTATION DIFFERS FROM OURS BY ADOPTING LARGER RECURRENT LAYERS, THEREFORE MORE PARAMETERS, AND A DIFFERENT ACTIVATION FUNCTION. TEST-CLEAN CONTAINS A CLEANER, WHEREAS TEST-OTHER CONTAINS A MORE CHALLENGING SPEECH TEST SET [39].)

	Ours	Sean Naren [54]	Paddle Paddle [53]
test-clean	14.41%	11.27%	6.85%
test-other	35.18%	30.74%	21.18%

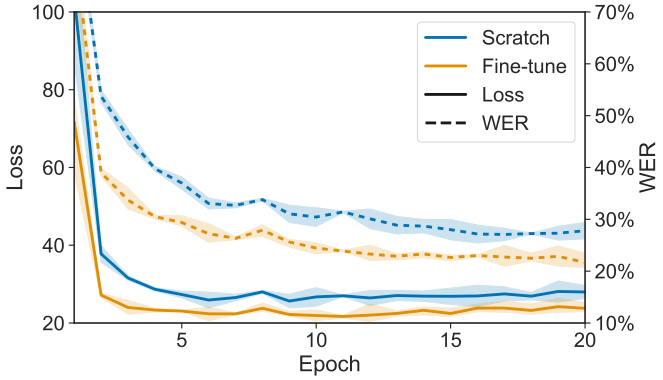
backbone and in the fine-tuned models, the last fully-connected weights must be initialized from scratch.

The PT-BR acoustic models are trained using the same procedure as the backbone model, except for a batch size of 32 and a learning rate decay of 0.99. We trained the model from scratch for 100 epochs and the fine-tuned model for 50 epochs. The training curves are depicted in Fig. 5, where one notices how the model performances in both cases did not improve after only 20 epochs. As one can see, using a pre-trained model accelerates training and reaches a better WER, besides a lower bias.

The final comparison between the acoustic PT-BR models is shown in Table VII for both BRSD v1 and v2 experiments. From these results, it is clear that fine-tuning is advantageous, reducing the final CER by 11.51% and 1.41% for the v1 and v2 cases, respectively. Also, it is noteworthy that increasing the dataset size was quite beneficial, significantly reducing the CER by 8.21%.



(a) BRSD v1



(b) BRSD v2

Fig. 5. Loss and WER comparison over the epochs, for both fine tuned and from scratch settings. The pre-trained model accelerates training and reaches a better WER, besides a lower bias.

IX. COMPLETE ASR EXPERIMENTS

In this section, we combine the language and acoustic models using the beam search decoder discussed in Section IV and the fine-tuned models from the previous experiment.

TABLE VII
PT-BR MODEL COMPARISON BETWEEN TRAINING FROM RANDOM INITIALIZATION (SCRATCH) AND TRAINING FROM A PRE-TRAINED BACKBONE MODEL (THE NUMBER BETWEEN PARENTHESES IS THE STANDARD DEVIATION CALCULATED OVER DIFFERENT `v2-test` SPLITS. IT IS CLEAR THAT FINE-TUNING IS ADVANTAGEOUS, REDUCING THE FINAL CER FOR BOTH V1 AND V2 CASES.)

	BRSD v1		BRSD v2	
	scratch	fine-tuning	scratch	fine-tuning
CER	34.65% (0.52%)	**23.14% (0.54%)**	16.34% (0.81%)	**14.93% (0.54%)**
WER	87.65% (0.42%)	**71.74% (1.49%)**	52.55% (2.42%)	**47.41% (1.73%)**

The beam search hyperparameters were selected using a grid search. We set a beam size of 100 for all experiments. The language-model parameters in Eq. (6) were selected among 25 linearly spaced points within the interval $(0, 3)$ for α, and 4 linearly spaced values within $(0, 0.5)$ for β. Hyperparameter values that lead to the best WER in the `v1-val` set were chosen for the final evaluation on the test sets.

The resulting WER and CER values for the trained models are reported in Tables VIII and IX for the BRSD v1 and v2 datasets, respectively. The best ASR system, trained with the BSRD v2 dataset and using a 15-gram character-based language model, achieved a CER and a WER of 10.49% and 25.45%, respectively. This is similar to the result found in [9] (15.2% of WER) for Tamil language, and comparable to the improvement reported by [13] and [14] using a multilingual setting training and multimodal data augmentation, respectively. From these tables, one clearly verifies the advantage of incorporating the language model onto the acoustic models obtained in Section VIII, with the WER dropping from 71.62% to 30.50% for the BRSD v1 set, and a 21.96% improvement for the BRSD v2 set. Comparing the v1 and v2 results, one concludes that more training data makes the final ASR system less dependent on an external language model, as also observed in [6]. The ASR system based on the n-gram character-level language models with $n \geq 10$ achieves better performance than the ones using word-level language models, most probably due to the former virtually infinite vocabulary. It is worth mentioning that the use of a well-trained language model greatly reduces the gap between the BRSD v1 and v2. This is due to the α parameter in the language model. In our experiments, a worse acoustic model (i.e., trained on v1 data) has a higher α, indicating that the overall ASR system is relying more on the language model to transcribe the audio than on the audio itself.

Some of the transcriptions provided by our best ASR system are shown in Table X. As one can see, some errors arise from grammatical mistakes (example 1), have phonetic similarities with the expected transcriptions (examples 2 and 3), or are due to proper names (example 4). Overall, most of the transcriptions are plausible and can be easily understandable by human readers.

TABLE VIII
CHARACTER AND WORD ERROR RATES (%) ON BRSD V1 DATA (ONE CAN SEE THAT INCOPORATING A LANGUAGE MODEL ONTO THE ACOUSTIC MODELS IS CLEARLY ADVANTAGEOUS, BY DROPPING THE WER FROM 71.62% IN TABLE VII TO 30.50% WITH A 20-GRAM CHAR LM.)

		v2-test	
	Lexicon	CER	WER
LapsLM	Yes	**16.11% (0.42%)**	**35.53% (1.58%)**
word 3-gram	Yes	18.36% (0.46%)	41.25% (1.43%)
word 5-gram	Yes	17.92% (0.48%)	40.96% (1.58%)
char 5-gram	No	19.08% (0.52%)	50.84% (1.64%)
char 10-gram	No	14.13% (0.39%)	33.03% (0.80%)
char 15-gram	No	13.34% (0.41%)	30.88% (1.01%)
char 20-gram	No	**13.25% (0.34%)**	**30.50% (0.91%)**

TABLE IX
CHARACTER AND WORD ERROR RATES (%) ON BRSD V2 DATA (WE CAN SEE THE SAME BEHAVIOUR AS IN V1 DATA, BY DROPPING THE WER FROM 47.41% IN TABLE VII TO 25.45% FOR A 15-GRAM CHAR LM. DIFFERENTLY FROM TABLE VIII, THE OVERALL IMPROVEMENT IS LOWER DUE TO THE BETTER ACOUSTIC MODEL. THE FIRST ROW REPORTS THE RESULT ON THE LAPS LANGUAGE MODEL (LAPSLM) DATASET, FREELY DISTRIBUTED BY ITS OWNERS.)

		v2-test	
	Lexicon	CER	WER
LapsLM	Yes	**11.50% (0.30%)**	**26.18% (0.74%)**
word 3-gram	Yes	12.31% (0.54%)	28.92% (0.84%)
word 5-gram	Yes	12.17% (0.55%)	28.65% (0.93%)
char 5-gram	No	13.02% (0.46%)	37.14% (1.38%)
char 10-gram	No	10.91% (0.21%)	27.03% (0.39%)
char 15-gram	No	**10.49% (0.13%)**	**25.45% (0.75%)**
char 20-gram	No	10.46% (0.18%)	25.50% (1.08%)

X. CONCLUSIONS

Through this work, we described all steps necessary to build an end-to-end ASR system for Brazilian Portuguese. Our proposal employs a DeepSpeech-2-based architecture and a transfer-learning approach from a backbone model trained on the English LibriSpeech dataset. To do so, we introduced a new version of the Brazilian Portuguese speech dataset and we built a new text corpus comprising 10.2 million sentences scraped from the Wikipedia Portuguese website. We showed the advantages of combining the language model with the acoustic model using the beam search decoder, significantly improving the performance of the overall system, which achieved CER and WER values of 10.49% and 25.45%, respectively.

Future works include testing the BRSD v2 on other deep neural networks, such as transformers, seq2seq with attention, and recurrent neural network transducers, as well as testing the BRTD on neural language models.

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Na-

TABLE X
TRANSCRIPTION ANALYSIS FOR BEST ASR SYSTEM, TRAINED WITH THE BSRD V2 DATASET AND USING A 15-GRAM CHARACTER-BASED LANGUAGE MODEL (SOME OF THE TRANSCRIPTIONS ERRORS FOUND WERE DUE TO GRAMMATICAL ERRORS (1); PHONETIC PLAUSIBLE ERRORS (2 AND 3); AND PROPER NAMES (2 AND 4).)

1	Reference:	apenas nove por cento afirma**m** que v**ã**o recorrer a empréstimos
	Transcript:	apenas nove por cento afirma que vai recorrer a empréstimos
2	Reference:	mari**z** está na segunda metade do seu primeiro mandato de senador
	Transcript:	mari**s** está na segunda metade do seu primeiro mandato de senador
3	Reference:	depois a **conta é encerrada** por falta de movimentação
	Transcript:	depois **acontecerrada** por falta de movimentação
4	Reference:	o acusado do crime **é** o ator guilherme de **pádua**
	Transcript:	o acusado do crime o ator guilherme de fado

tional Council for Scientific and Technological Development (CNPq), and Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ).

The authors would like to thank Roberto de Moura Estevão Filho, Matheus Araújo Marins, Lucas Pinheiro Cinelli, and many others for their support, assistance, and fruitful discussions.

REFERENCES

[1] Y. Leviathan and Y. Matias, "Google duplex: An AI system for accomplishing real-world tasks over the phone," http://bit.ly/2keb5Om, accessed: 2020-03-19.

[2] G. Anders, "Alexa, understand me," http://bit.ly/2x2Yx5F, accessed: 2020-03-19.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, UK: MIT Press, 2016.

[4] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, February 1989, doi:10.1109/5.18626.

[5] Mozilla, "Common voice," https://voice.mozilla.org, 2018, accessed: 2020-03-19.

[6] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, A. Y. Hannun, B. Jun, T. Han, P. LeGresley, X. Li, L. Lin, S. Narang, A. Y. Ng, S. Ozair, R. Prenger, S. Qian, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, C. Wang, Y. Wang, Z. Wang, B. Xiao, Y. Xie, D. Yogatama, J. Zhan, and Z. Zhu, "Deep speech 2: end-to-end speech recognition in English and Mandarin," in *International Conference on Machine Learning*, vol. 48, New York, USA, June 2016, pp. 1–10.

[7] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S. Chang, K. Rao, and A. Gruenstein, "Streaming end-to-end speech recognition for mobile devices," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Brighton, UK, May 2019, pp. 6381–6385, doi:10.1109/ICASSP.2019.8682336.

[8] I. M. Quintanilha, "End-to-end speech recognition applied to Brazilian Portuguese using deep learning," M.Sc. Dissertation, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 2017.

[9] B. Pulugundla, M. K. Baskara, S. Kesiraju, E. Egorova, and J. C. Martin Karafiát, Lukás Burget, "BUT system for low resource Indian language ASR," in *Interspeech*, Hyderabad, India, September 2018, pp. 3182–3186, doi:10.21437/Interspeech.2018-1302.

[10] J. Billa, "ISI ASR system for the low resource speech recognition challenge for Indian languages," in *Interspeech*, Hyderabad, India, September 2018, pp. 3207–3211, doi:10.21437/Interspeech.2018-2473.

[11] R. Jimerson and E. Prud'hommeaux, "ASR for documenting acutely under-resourced indigenous languages," in *Language Resources and Evaluation Conference*, Miyazaki, Japan, May 2018, pp. 4161–4166.

[12] P. Swietojanski, A. Ghoshal, and S. Renals, "Unsupervised cross-lingual knowledge transfer in DNN-based LVCSR," in *IEEE Spoken Language Technology Workshop*, Miami, USA, December 2012, pp. 246–251, doi:10.1109/SLT.2012.6424230.

[13] S. Dalmia, R. Sanabria, F. Metze, and A. W. Black, "Sequence-based multi-lingual low resource speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Calgary, Canada, April 2018, pp. 4909–4913, doi:10.1109/ICASSP.2018.8461802.

[14] A. Renduchintala, S. Ding, M. Wiesner, and S. Watanabe, "Multi-modal data augmentation for end-to-end ASR," in *Interspeech*, Hyderabad, India, September 2018, pp. 2394–2398, doi:10.21437/Interspeech.2018-2456.

[15] S. Zhou, S. Xu, and B. Xu, "Multilingual end-to-end speech recognition with a single transformer on low-resource languages," June 2018, eprint arXiv:1806.05059.

[16] I. M. Quintanilha, L. W. P. Biscainho, and S. L. Netto, "Towards an end-to-end speech recognizer for Portuguese using deep neural networks," in *XXXV Brazilian Symposium on Telecommunications and Signal Processing*, São Pedro, Brazil, September 2017.

[17] ——, "A new automatic speech recognizer for Brazilian Portuguese based on deep neural networks and transfer learning." in *AES Latin-American Conference on Audio*, Montevideo, Uruguay, September 2018.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, vol. 37, Lille, France, July 2015, pp. 1–9.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, November 1997, doi:10.1162/neco.1997.9.8.1735.

[20] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Heidelberg, Germany: Springer Verlag, 2012, doi:10.1007/978-3-642-24797-2.

[21] A. Hannun, "Sequence modeling with CTC," *Distill*, November 2017, doi:10.23915/distill.00008.

[22] A. L. Maas, A. Y. Hannun, D. Jurafsky, and A. Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs," December 2014, eprint arXiv:1408.2873.

[23] A. L. Maas, Z. Xie, D. Jurafsky, and A. Y. Ng, "Lexicon-free conversational speech recognition with neural networks," in *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, USA, May 2015, pp. 345–354, doi:10.3115/v1/N15-1038.

[24] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *International Conference on Machine Learning*, Pittsburgh, USA, June 2006, pp. 369–376, doi:10.1145/1143844.1143891.

[25] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, October 2002, doi:10.1006/csla.2001.0184.

[26] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, Miami, USA, May 1995, pp. 181–184, doi:10.1109/ICASSP.1995.479394.

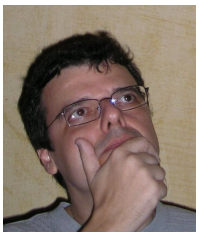
[27] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Association for Computational Linguistics*, California, USA, June 1996, pp. 310–318, doi:10.3115/981863.981904.

[28] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, March 2003.

[29] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur,

"Recurrent neural network based language model," in *Interspeech*, Makuhari, Japan, September 2010, pp. 1045–1048.
[30] R. Józefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," February 2016, eprint arXiv:1602.02410v2.
[31] J. G. Zilly, R. K. Srivastava, J. Koutník, and J. Schmidhuber, "Recurrent highway networks," in *International Conference on Machine Learning*, vol. 70, Sydney, Australia, August 2017, pp. 4189–4198.
[32] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, pp. 179–190, February 1983, doi:10.1109/TPAMI.1983.4767370.
[33] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependences in stochastic language modelling," *Computer Speech & Language*, vol. 8, no. 1, pp. 1– 38, January 1994, doi:10.1006/csla.1994.1001.
[34] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Interspeech*, Portland, USA, September 2012, pp. 194–197.
[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, California, USA, December 2017, pp. 5998–6008.
[36] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," October 2018, eprint arXiv:abs/1810.04805.
[37] N. Neto, C. Patrick, A. Klautau, and I. Trancoso, "Free tools and resources for brazilian portuguese speech recognition," *Journal of the Brazilian Computer Society*, vol. 17, no. 1, pp. 53–68, November 2011, doi:10.1007/s13173-010-0023-1.
[38] Linguateca, "CETENFolha," https://www.linguateca.pt/cetenfolha/index_info.htmll, accessed: 2020-03-19.
[39] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: An ASR corpus based on public domain audio books," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Brisbane, Australia, April 2015, pp. 5206–5210, doi:10.1109/ICASSP.2015.7178964.
[40] "Voxforge," https://http://www.voxforge.org, accessed: 2020-03-19.
[41] "Falabrasil - UFPA," https://github.com/falabrasil/gitlab-resources, accessed: 2020-03-19.
[42] M. Schramm, L. F. Freitas, A. Zanuz, and D. Barone, "CSLU: Spoltech Brazilian Portuguese version 1.0 LDC2006S16," Philadelphia, 2006, Linguistic Data Consortium.
[43] N. Neto, P. Silva, A. Klautau, and A. Adami, "Spoltech and OGI-22 baseline systems for speech recognition in Brazilian Portuguese," in *International Conference on Computational Processing of Portuguese Language*, vol. 5190, Aveiro, Portugal, September 2008, pp. 256–259, doi:10.1007/978-3-540-85980-2_33.
[44] V. F. S. Alencar and A. Alcaim, "LSF and LPC - derived features for large vocabulary distributed continuous speech recognition in Brazilian Portuguese," in *Asilomar Conference on Signals, Systems and Computers*, October 2008, pp. 1237–1241, doi:10.1109/ACSSC.2008.5074614.
[45] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, N. L. Dahlgren, and V. Zue, "Timit acoustic-phonetic continuous speech corpus LDC93S1," Philadelphia, 1993, Linguistic Data Consortium.
[46] J. Garofolo, D. Graff, D. Paul, and D. Pallett, "CSR-I (WSJ0) Sennheiser LDC93S6B," Philadelphia, 1993, Linguistic Data Consortium.
[47] "CSR-II (WSJ1) Sennheiser LDC94S13B," Philadelphia, 1994, Linguistic Data Consortium.
[48] J. Godfrey and E. Holliman, "Switchboard-1 release 2 LDC97S62," Philadelphia, 1993, Linguistic Data Consortium.
[49] K. Heafield, "KenLM: Faster and smaller language model queries," in *Sixth Workshop on Statistical Machine Translation*, Edinburgh, UK, July 2011, pp. 187–197.
[50] T. Likhomanenko, G. Synnaeve, and R. Collobert, "Who needs words? Lexicon-free speech recognition," April 2019, eprint arXiv:1904.04479v1.
[51] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Annual Conference of the International Speech Communication Association*, Dresden, Germany, September 2015, pp. 3586–3589.
[52] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, vol. 28, Atlanta, USA, June 2013, pp. 1310–1318.
[53] "A PaddlePaddle implementation of DeepSpeech2 architecture for ASR," https://github.com/PaddlePaddle/DeepSpeech, accessed: 2020-03-19.
[54] S. Naren, "Speech recognition using DeepSpeech2," https://github.com/SeanNaren/deepspeech.pytorch, accessed: 2020-03-19.

Igor M. Quintanilha was born in Rio de Janeiro, Brazil, in 1991. He received the B.Sc. degree of electronic and computing engineer from the Federal University of Rio de Janeiro (UFRJ), Brazil, in 2015, and the M.Sc. degree in electrical engineering from the COPPE/UFRJ, in 2017. He is currently pursuing the D.Sc. degree in electrical engineering at COPPE/UFRJ. His research interests are deep learning, machine learning, speech processing, computer vision, and signal processing.



Luiz W. P. Biscainho was born in Rio de Janeiro, Brazil, in 1962. He received the diploma of electronic engineer (magna cum laude) from the EE (now Poli) at Universidade Federal do Rio de Janeiro (UFRJ), Brazil, in 1985, and the M.Sc. and D.Sc. degrees in electrical engineering from the COPPE at UFRJ in 1990 and 2000, respectively. Having worked in the telecommunication industry between 1985 and 1993, Dr. Biscainho is now Associate Professor at DEL/Poli and PEE/COPPE, at UFRJ. His research area is digital signal processing, particularly audio processing. He is currently a member of IEEE (Institute of Electrical and Electronics Engineers), AES (Audio Engineering Society), SBrT (Brazilian Telecommunications Society), and SBC (Brazilian Computer Society).

Sergio L. Netto was born in Rio de Janeiro, Brazil. He received the B.Sc. (cum laude) degree from the Federal University of Rio de Janeiro (UFRJ), Brazil, in 1991, the M.Sc. degree from COPPE/UFRJ in 1992, and the Ph.D. degree from the University of Victoria, BC, Canada, in 1996, all in electrical engineering. Since 1997, he has been with the Department of Electronics and Computer Engineering, Poli/UFRJ, and since 1998, he has been with the Program of Electrical Engineering, COPPE/UFRJ. He is the Co-Author (with P. S. R. Diniz and E. A. B. da Silva) of Digital Signal Processing: System Analysis and Design (Cambridge University Press, second edition, 2010). His research and teaching interests lie in the areas of digital signal processing, adaptive filtering, speech processing, information theory, computer vision, and machine learning.