Expert Systems With Applications 215 (2023) 119293

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A customized residual neural network and bi-directional gated recurrent unit-based automatic speech recognition model

Selim Reza [a], Marta Campos Ferreira [a], J.J.M. Machado [b], João Manuel R.S. Tavares [b,*]

[a] Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal
[b] Departamento de Engenharia Mecânica, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal

ARTICLE INFO

Keywords:
Deep learning
Layer normalization
Character error rate
Word error rate
LibriSpeech corpus
LJ speech dataset

ABSTRACT

Speech recognition aims to convert human speech into text and has applications in security, healthcare, commerce, automobiles, and technology, just to name a few. Inserting residual neural networks before recurrent neural network cells improves accuracy and cuts training time by a good margin. Furthermore, layer normalization instead of batch normalization is more effective in model training and performance enhancement. Also, the size of the datasets presents tremendous influences in achieving the best performance. Leveraging these tricks, this article proposes an automatic speech recognition model with a stacked five layers of customized Residual Convolution Neural Network and seven layers of Bi-Directional Gated Recurrent Units, including a logarithmic $softmax$ for the model output. Each of them incorporates a learnable per-element affine parameter-based layer normalization technique. The training and testing of the new model were conducted on the LibriSpeech corpus and LJ Speech dataset. The experimental results demonstrate a character error rate (CER) of 4.7 and 3.61% on the two datasets, respectively, with only 33 million parameters without the requirement of any external language model.

1. Introduction

Voice technology is currently employed in many industries, allowing businesses and consumers to facilitate digitization and automation. For example, speech recognition improves the safety and efficiency of vehicles by enabling voice-activated navigation systems and enhancing search capabilities. Besides, voice commands to access virtual agents, particularly on mobile devices, help the call centres transcribe a colossal number of phone calls and identify call patterns, queries and issues. Additionally, voice-based authentication generally provides an additional security level (Junqua & Haton, 2012).

However, speech recognition is one of the most challenging computer science topics due to the difficulties of separating similar phonetically sentences and smearing problems. Various algorithms and computational approaches have been proposed to convert speech to text and increase transcription accuracy (Yu & Deng, 2016). Neural networks, particularly Deep Learning (DL) algorithms, are best suited for these purposes. They can efficiently process vast amounts of data, identify hidden patterns between features, learn the mapping function, and then adjust it based on the loss function using the gradient descent technique. While neural networks are more accurate and can handle more inputs, they achieve lower performance efficiency than classic language models since they require longer training time and more computational resources (Benzeghiba et al., 2007).

Nowadays, transformer-based models are state-of-the-art as they have demonstrated superior performances compared to other baselines. However, these models are computationally expensive. Therefore, multiple $(8 - 16)$ graphical processing units (GPUs) in parallel with synchronous stochastic gradient (SGD) are generally applied to train this kind of model (Likhomanenko et al., 2021; Park et al., 2019). Residual Convolution Neural Networks (ResNets) reduce training time while improving the model's performance. Moreover, batch normalization is simply an additional network layer between two hidden layers. Its function is to normalize the outputs of the first hidden layer before passing them onto the next hidden layer as input, helping keep the network stable throughout training. Unlike batch normalization, layer normalization calculates the normalization statistics directly from the summed inputs to the neurons within a hidden layer, ensuring that no additional dependencies between training examples are introduced (Santurkar, Tsipras, Ilyas, & Madry, 2018).

Karita et al. (2019), Moritz, Hori, and Le Roux (2019) and Wang, Guan, and Li (2018) proposed different architectures for solving automatic speech recognition (ASR) problems on LibriSpeech datasets. Wang

* Corresponding author.
E-mail addresses: up202003355@fe.up.pt (S. Reza), mferreira@fe.up.pt (M.C. Ferreira), jjmm@fe.up.pt (J.J.M. Machado), tavares@fe.up.pt (J.M.R.S. Tavares).

https://doi.org/10.1016/j.eswa.2022.119293
Received 18 June 2022; Received in revised form 18 October 2022; Accepted 14 November 2022
Available online 18 November 2022

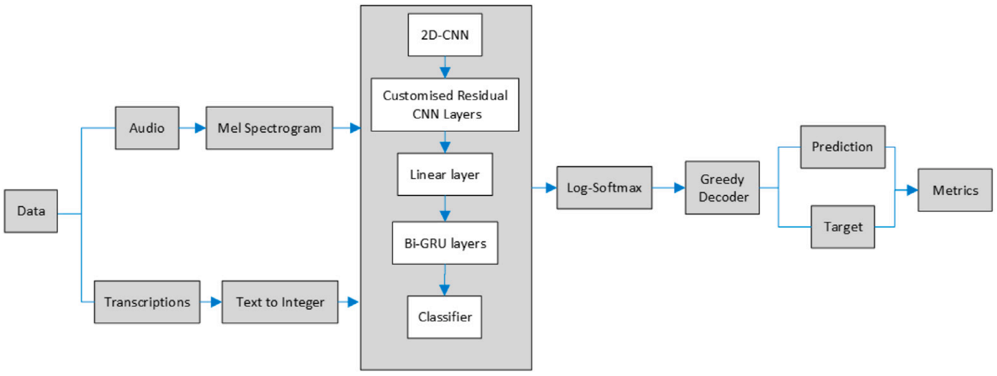

Fig. 1. Illustration of the overall concept of the proposed model.

et al. (2018) integrated two-dimensional invariant convolution layers and groups of residual convolution blocks and Bi-GRUs alongside the Connectionist Temporal Classification (CTC) loss. The authors used a subset of the LibriSpeech datasets for conducting the experiments. Karita et al. (2019) introduced a speech auto-encoder loss function and proposed an encoder–decoder architecture for speech recognition tasks, performing the model training and validation on a subset of the LibriSpeech datasets. Moritz et al. (2019) proposed a triggered attention mechanism with a CTC-based classifier to control the attention mechanism's activation. The proposed algorithm differs from the previously mentioned models in terms of architecture. Fig. 1 illustrates the overall concept of the proposed model. It relied on customized ResNets without any invariant convolution layers; particularly, it contains five layers of customized ResNets with seven layers of Bi-GRUs. For training and testing the new model, the LibriSpeech ASR corpus, which contains 1000 hours of audio speech, and the LJ Speech dataset, which includes approximately 24 hours of audio clips of a single speaker, were used. The experimental results demonstrated improvements in reducing the computational cost while providing a minimum CER of only 4.7 and 3.61%, respectively, on the used datasets without the need for any external language model. A CER of 4.7% means that 95.3% characters, which include not only letters but also punctuation and spaces, were correctly recognized. The main contributions of this research are as follows:

- It proposes a stacked five layers of customized ResNets and seven layers of Bi-GRUs, each including a layer normalization based on a learnable element-wise affine parameters approach without the requirement of external language models.
- The inclusion of the Gaussian error linear unit (GELU) layer and the dense and dropout layers for the classification tasks showed its worthiness in performance enhancement.
- It demonstrates that the volume of the training data significantly affects the model's output.
- It also demonstrates that the proposed architecture of ResNets helps the model to achieve performance improvement with five or more layers of Bi-GRUs, which is the opposite of the findings presented in Wang et al. (2018).

The remaining sections of this article are the following: Section 2 presents selected state-of-the-art works along with their performances and limitations; Section 3 introduces the proposed ResNets and the Bi-GRUs based model; the experimental setups and results are presented in Section 4; Section 6 is devoted to discuss the overall performance and feasibility of proposed model in automatic speech recognition tasks; and finally, Section 7 outlines the conclusions.

2. Related works

Three types of ASR models are common in the literature: (i) CTC-based, (ii) transformer-based, and (ii) recurrent neural networks (RNNs)-based models. Lee and Watanabe (2021) presented a conformer model with an intermediate CTC loss connected to an encoder network's intermediate layer showing that training regularization and performance improvements reached a Word error rate (WER) and CER of 9.9 and 5.2% on the WSJ and AISHELL-1 corpus, respectively. The comparisons showed that state-of-the-art performance is achievable with only 12 layers of conformer (118 million parameters) compared with 48 layers of a transformer-based model. Although the authors reduced the computational expenses to a good margin, more improvements are still required.

Li, Xu, and Zhang (2021) proposed a linear attention-based conformer that reduced the number of parameters of the conformer-based model by 50%, guaranteeing faster training, but still achieving a CER of 4.88% on the same datasets. Two factors were responsible for reducing the number of parameters: the application of (i) multi-head linear self-attention and (ii) low-rank matrix factorization techniques. Majumdar et al. (2021) proposed the Citrinet model, which consists of a deep residual neural model using one-dimensional time-channel separable convolutions. The authors combined it with sub-word encoding, squeeze-and-excitation and convolutional CTC. The results demonstrated very high accuracy with only 37.2 million parameters on LibriSpeech, MLS, TED-LIUM 2, and AISHELL datasets without any requirement of external language models.

Zhao et al. (2021) proposed a semantic correction model using pretrained BERT (Liu et al., 2019) with six encoder and decoder layers. It effectively reduced the CER by 21.7% compared with 3gram, 4gram, and RNN-based models. Fan, Chu, Chang, Xiao, and Alwan (2021), for both the encoder and decoder architectures, proposed convolution-augmented self-attention blocks. Expanding the acoustic boundary for each token demonstrated an increase in CTC alignment's robustness, and using an iterative loss function enhanced the gradient update for low-layer parameters. These concepts led to a 7 to 21% improvement compared to the other baselines by achieving WER of 3.1 and 7.2% on Librispeech $Test\_Clean$ and $Test\_Other$ datasets, respectively, and a CER of 5.4% on the AISHELL-1 test dataset.

Karita et al. (2019) introduced semi-supervised speech and text autoencoders to improve the performance further. The work presented a loss function called maximum mean discrepancy, which achieved a CER and a WER of 10.4 to 8.4% and 20.6 to 18.0% on the LibriSpeech $Test\_Clean$ dataset, respectively. Baidu's Deep Speech (Hannun et al., 2014) and Google's Listen Attend Spell (Chan, Jaitly, Le, & Vinyals, 2016) are two of the most famous pioneering speech recognition models on RNNs. However, the authors employed distinct techniques to model speech recognition problems. The first one predicted speech transcriptions using the CTC loss function, which achieved a WER of 13.25% on the Librispeech datasets. The second model for prediction employed a sequence-to-sequence network architecture, which demonstrated a WER of only 6.5% on the same datasets.

Han et al., 2020 and Kriman et al., 2020 incorporated depthwise separable convolution layers to further reduce the computational costs while maintaining adequate accuracy on the Librispeech datasets. The

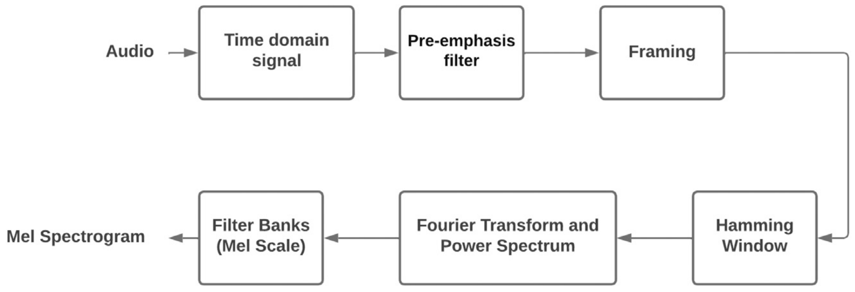

Fig. 2. Process of converting audio data to the Mel Spectrogram.

standard convolution requires around 9 times more multiplications than the depthwise separable convolution. Hence, the authors achieved tremendous accuracy with much fewer parameters; for example, a WER of 7% on Librispeech $Test\_Other$ with only $10.8M$ parameters. However, for small networks, these models might end up with too few parameters and fail to learn appropriately during training. Also, the depthwise separable convolution technique solely relies on cross-kernel correlations and a more efficient separation of regular convolutions is required to address the demands (Haase & Amthor, 2020). Besides, these two models need much longer training steps; for example, the model proposed by Kriman et al. (2020) took 1200 epochs with a batch size of 32 per GPU to achieve state-of-the-art accuracy. The model proposed in this article intends to address these drawbacks in more a manageable way.

Inspired by the transformer architecture and the Deep Speech model, the current study employs the concepts of customized ResNets and Bi-GRUs with layer normalization to reduce the computational costs further while maintaining a very interesting performance without any need for external language models.

3. Methodology

This section presents the formulation of the proposed automatic speech recognition model architecture based on customized ResNets with Bi-GRUs.

3.1. Mel spectrogram

Humans cannot recognize frequencies on a linear scale. Lower frequency differences are easier to notice than higher frequency variances; for example, humans can readily distinguish between 1000 and 1500 Hz, but would struggle to distinguish between 10,000 and 10,500 Hz, although the difference is the same (Qasim, Fried, & Jacobs, 2021). The Mel scale was proposed in 1937 to solve this problem. It demonstrates that a logarithmic transformation (natural log or 10 base log) can serve the purpose of making sounds of equal distance to be of equal distance to humans (Thys, Treviño, & Nadkarni, 2021). Accordingly, the transformation from the Hertz scale to the Mel Scale is defined as:

$$M = 2595 \log(1 + f/700). \tag{1}$$

In the Mel spectrogram, the frequency components of the audio signal are converted to the Mel scale. First, the raw audio data are represented as a time-domain signal. Then, a pre-emphasis filter is used to bring the frequency spectrum into equilibrium; generally, the higher frequencies possess smaller magnitudes than the lower frequencies and make the Fourier transform operation numerically efficient. After pre-emphasis, the signal is split into multiple short-time frames, followed by the application of a window function, i.e., the Hamming window, to each frame with the following form:

$$W_f[x] = 0.54 - 0.46 \cos(2\pi x/N - 1), \tag{2}$$

where $0 \leq x \leq (N-1)$ and N is the window length.

The frequency and power spectrum are then obtained using an n-point Fast Fourier Transform (FFT) on each frame. The final step is to apply the triangular filters to the power spectrum on a Mel-scale to extract frequency bands (Hwang et al., 2020), as can be perceived from the block diagram shown in Fig. 2.

3.2. Residual neural network

ResNets employs skip connections to leap over some layers, usually implemented by skipping double or triple layers with non-linearities, e.g., ReLU, and batch normalization between the skipped layers. This technique avoids vanishing gradients and accuracy saturation problems, which occur when more layers are added to a sufficiently deep model that ends up with an increased training error (Szegedy, Ioffe, Vanhoucke, & Alemi, 2017).

Let z be the input. Then, z will travel through different layers, i.e., Convolution, Dropout and normalization layers, of each block of the ResNets cell. To get the output, y, the loss function is calculated on the basis of input, z, using $y = f(z)$, where $f(z)$ is the loss function. The input, z, is added to the output using $y = f(z) + z$ following the skip connection. The goal is to make $f(z) = 0$ so that the network can learn from the difference between the input and the output.

After several trial-error tests, within each ResNets block (identity blocks), the convolution kernel of 3×3 shape and 32 filters provided the best overall performance. It also included two Dropout layers, with each having a 10% probability ($p = 0.1$) of randomly making some components of the input tensor equal to zero, and two-layer normalization within each ResNets block. The used ResNets architecture is illustrated in Fig. 3.

3.3. Bi-directional Gated Recurrent Units

Gated recurrent units (GRUs) are an updated version of the long short-term memory (LSTM) network that recognize sequential or temporal information uniquely and more straightforwardly (Chung, Gulcehre, Cho, & Bengio, 2014). GRUs combine two gates, the update and the reset gates, with a different working mechanism than LSTM. For instance, the GRU's update gate combines the LSTM's forget and input gates, but the reset gate is the same. Like the LSTM network, a GRU model modifies the information inside the units, but it does not require different memory cells (see Fig. 4).

Let x_t, z_t, r_t, $\hat{h}_t$, h_{t-1}, and h_t be the current input, update gate, reset gate, candidate hidden state of the currently hidden node, hidden state at the previous moment, and current hidden state, respectively. The update gate, z_t, determines how much information in h_{t-1} will be passed along the future, and is formulated as:

$$z_t = \sigma(W_{zx} x_t + U_{zh} h_{t-1}). \tag{3}$$

Here, both x_t and h_{t-1} are multiplied by their W_{zx} and U_{zh} weights, respectively, which are learned during the algorithm training. Both results are added together and passed through a $sigmoid$ function to determine their values between 0 and 1. A value of 0 means that the

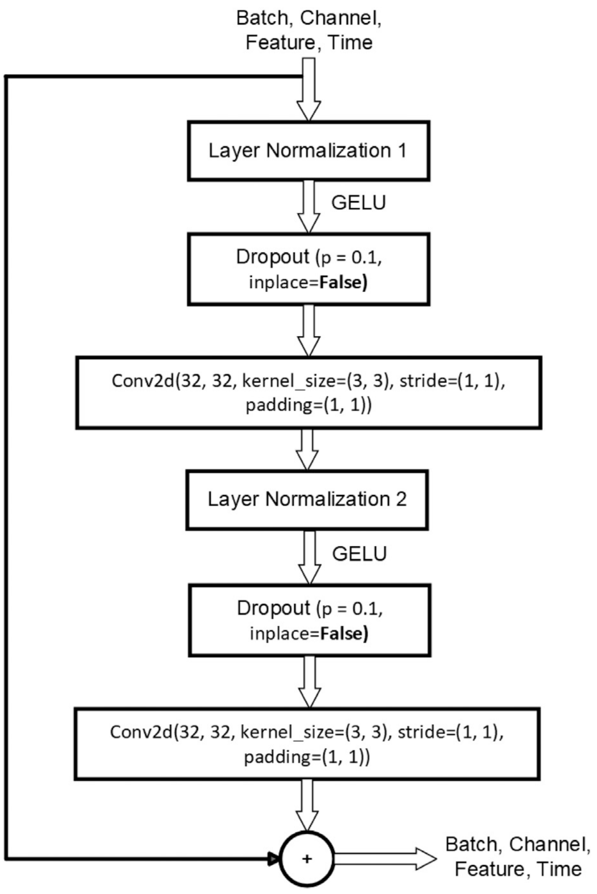
Batch, Channel, Feature, Time
Layer Normalization 1
GELU
Dropout (p = 0.1, inplace=False)
Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
Layer Normalization 2
GELU
Dropout (p = 0.1, inplace=False)
Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
+
Batch, Channel, Feature, Time

Fig. 3. ResNets architecture of the proposed model based on the concept of skip connections.

information will be forgotten, otherwise is retained in the current hidden state. The reset gate, r_t, decides how much of the past information to forget, and is formulated as:

$$r_t = \sigma(W_{rx}x_t + U_{rh}h_{t-1}). \tag{4}$$

Eq. (4) is identical to Eq. (3) for the update gate with the exception of the weights and the use of gates. The candidate hidden state of the currently hidden node, $\hat{h}_t$, uses the reset gate to store relevant information from the past using:

$$\hat{h}_t = tanh(W_{hx}x_t + r_t \odot U_{hh}h_{t-1}). \tag{5}$$

Here, firstly, x_t and h_{t-1} are multiplied by their W_{hx} and U_{hh} weights and secondly, the Hadamard product ($\odot$), i.e., the element-wise multiplication, between r_t and $U_{hh}h_{t-1}$ is computed. The results of these two steps are summed up and passed through a non-linear activation function, for example, $tanh$ function, to get $\hat{h}_t$. The final step involves the calculation of the current hidden state, h_t, using:

$$h_t = (1 - z_t) \odot \hat{h}_t + z_t \odot h_{t-1}. \tag{6}$$

Eq. (6) calculates the Hadamard products (element-wise multiplications) between $(1-z_t)$ and $\hat{h}_t$, and z_t and h_{t-1}, respectively. The inability of unidirectional GRUs to simultaneously consider past and future data is detrimental to the improvement of accuracy (Li et al., 2020). Hence, this research adopted a Bi-GRU structure to meet the objectives. Fig. 5 presents a Bi-GRU schematic diagram.

The input data sequence is first concurrently fed into a forward GRU, $\overrightarrow{GRU}$ and a backward GRU, $\overleftarrow{GRU}$. Then, the concatenation of the hidden states $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$, which are produced by the forward and the backward GRU at time t, respectively, will yield the hidden state at that instant using:

$$h_t = concatenate(\overrightarrow{h_t}, \overleftarrow{h_t}), \tag{7}$$

where $\overrightarrow{h_t} = \overrightarrow{GRU}(h_{t-1}, x_t)$ and $\overleftarrow{h_t} = \overleftarrow{GRU}(h_{t+1}, x_t)$.

3.4. Speech recognition model

The first step of the proposed speech recognition model is to transform the raw audio into Mel Spectrograms. Then, the model maps the character labels for each audio sample into integer labels. The heart of the model consists of two deep neural network modules: $N = 5$ layers of ResNets architecture according to the description of Section 3.2, and a stack of 7 Bi-GRUs blocks using both a layer normalization and a dropout layer with $p = 0.1$ within each block. The ResNets extracted and learned relevant audio features, and the Bi-GRUs block used those learnt audio features for the speech recognition task. A sequential model containing two linear layers, a GELU layer and a dropout layer, provided the best results after a few trial-error tests for classifying characters per time step. Fig. 6 illustrates the architecture of the proposed model. The logarithmic $softmax$ was integrated into the output layer to get the probability matrix of characters. The final step to predict the most likely character is to use a decoder. Developing a greedy decoder allowed the extraction of the most likely characters spoken from the audio for this study.

4. Experiments

The training of the proposed model was accomplished using an NVIDIA DGX Station with 4 NVIDIA Tesla V100 tensor core GPUs having 128GB of memory, the CUDA library in its 10.1 version, and the open-source PyTorch machine learning library for Python coding. The code implementation comprised five steps. First, the datasets were collected using the ’torchaudio.datasets’ module. The PyTorch ‘DataLoader’ function facilitated the building of the train and test datasets. The second step included mapping the characters to integers and vice-versa by defining a ’Text_Transformation’ class. It also performed data processing tasks. The ’torchaudio.transforms’ module transformed the data samples’ waveform into Mel Spectrograms and a pre-defined function using ’torch.nn.utils.rnn.pad_sequence’ helped to pad the Spectrograms and labels (from the transcriptions) into equal-length sequences. This step was concluded by defining a greedy decoder that returned the decoded and target transcriptions. The third step comprised defining the model’s architecture. This study built a class named ’Layer_Norm’ using the ’torch.nn’ module for the CNNs input. Then, a customized ResNets class was built using the ’torch.nn’ module. A pre-defined forward function within the class returned the batch, channel, feature, and time utilizing Layer_Norm, GELU, 2-dimensional CNN and dropout layers based on residual connection. For determining the Bi-GRU cells, this study built a class named ’Bi_Directional_GRU’ using the ’torch.nn’ module. Within it, a pre-defined forward function returned the Bi-GRU cell based on Layer_Norm, GELU, Bi-GRU and dropout layers. The last part of the third phase combined all the previously defined classes into a speech recognition class, including a classifier comprised of Linear, GELU and dropout layers using the ’torch.nn.Sequential’ module. A pre-defined forward function returned the batch, time, and features for classification tasks. The fourth stage covered the model training procedures. A pre-defined function named ’model_train’ was used for model training purposes, and the Spectrograms and labels were fed into the model. Initially, the ’torch.optim’ module aimed to set the gradients to zero. The logarithmic $softmax$ from the ’torch.nn.functional’ module worked as the output layer to determine the probabilities of each character of the output transcriptions. The CTC loss function from the ’torch.nn’ module calculated the probability of the correct labels. The fifth phase included the testing procedures of the model. The python ‘jiwer’ package facilitated

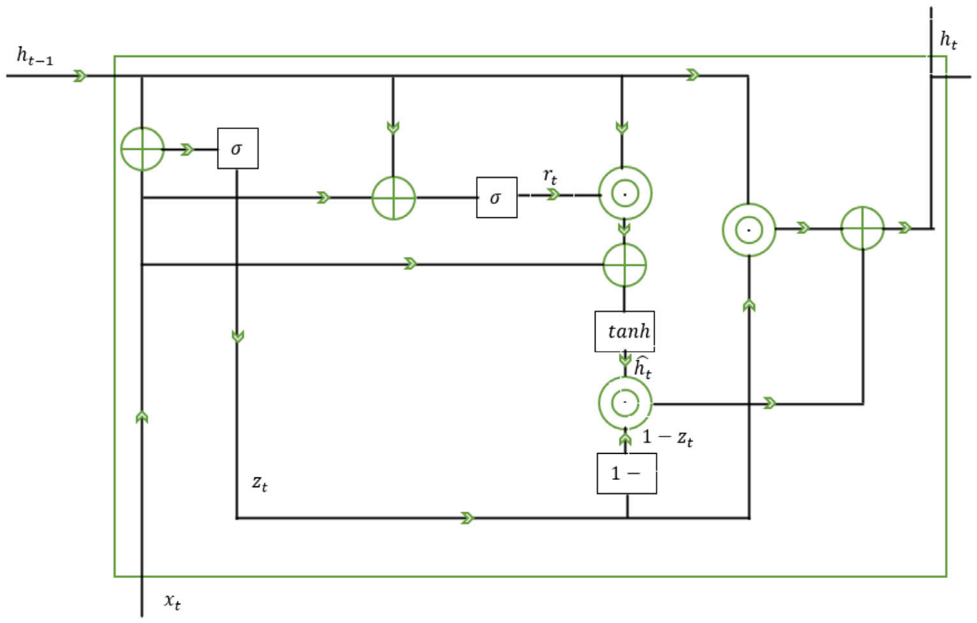

Fig. 4. A simplified architecture of a GRU cell (σ, $\oplus$ and $\odot$ represent the *sigmoid* activation function, summation and Hadamard product, respectively).

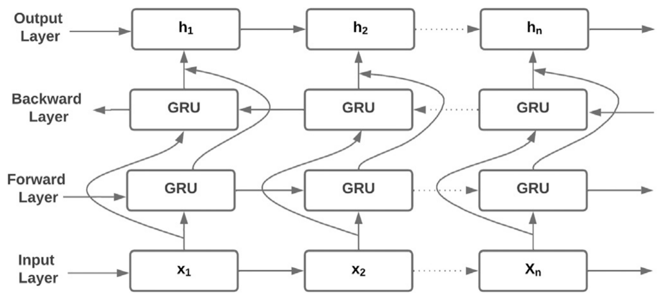

Fig. 5. Schematic diagram of a Bi-GRU architecture.

calculating the WER and CER metrics. The previously defined greedy decoder picked up the character with the highest probability from the output of the logarithmic *softmax* layer using 'torch.nn.functional' class.

4.1. Connectionist temporal classification loss

The CTC Loss function calculates a differentiable loss value for each input node by adding all potential input alignments to the target. It employs a bidirectional LSTM design with peepholes and forgets gates having logistic *sigmoid* ranging from 0 (zero) to 1 (one). The hyperbolic tangent function is necessary for activation. The main difference between the CTC and other temporal classifiers is that CTC segments the input sequences indirectly, which has several advantages, including avoiding the need to locate label borders that are inherently ambiguous and allowing grouping of the label predictions (Zeyer, Irie, Schlüter, & Ney, 2018).

If I and O are the input and output, respectively, then the CTC objective for (I, O) is defined as:

$$p(O|I) = \sum_{X \epsilon X_{I,O}} \prod_{t=1}^{T} p_t(x_t|I), \tag{8}$$

where $p(O|I)$ and $p_t(x_t|I)$ represent CTC conditional probability and per time-step probability for a single alignment, respectively (Graves, Fernández, Gomez, & Schmidhuber, 2006).

4.2. Dataset

Due to good baselines for character-based ASR, the LibriSpeech corpus (Panayotov, Chen, Povey, & Khudanpur, 2015) was used to assess the proposed model. This study utilizes all the three training datasets: $Train\_Clean\_100$, $Train\_Clean\_360$, and $Train\_Other\_500$ for the model training, and the test dataset was $Test\_Clean$. The number suffix in the dataset name denotes the dataset's time in hours.

Also, to investigate the natural generalization of the proposed model, i.e., to verify that it can perform well on other datasets, the current study used the LJ Speech dataset (Ito & Johnson, 2017), which contains around $13K$ short audio clips from a single speaker spanning over approximately 24 hours.

4.3. Model training

The following hyper-parameters provided the best results after several trial-error tests: $batch\_size = 10$, $p = 0.1$, $epochs = 100$, $\alpha = 5.0E-4$, $N = 29/31$, $N\_cnn\_layers = 5$, $F\_N = 128$, $N\_rnn\_layers = 7$, $D\_rnn = 512$, and $stride = 2$. Here, p, α, N, N_cnn_layers, F_N, N_rnn_layers and D_rnn denote the likelihood of a component being zeroed within the dropout layers, the learning rate, number of classes (29 and 31 for the LibriSpeech and LJ Speech datasets, respectively), number of ResNets layers, number of features, number of Bi-GRU layers, and GRU dimension, respectively.

Spectrograms and labels were fed into the model to train it. AdamW was used as the optimizer because traditional Adam contains a wrong implementation of weight decay Loshchilov & Hutter, 2017. Furthermore, the CTC loss function was applied to calculate the loss. For the LibriSpeech corpus, it was trained on three different train datasets: $Train\_Clean\_100$, $Train\_Clean\_360$, and $Train\_Clean\_500$. Each contains 100, 360 and 500 hours long audio clips. On the other hand, for the LJ Speech dataset, the number of audio samples of the train and test datasets were 11,790 and 1310, respectively. The algorithm was trained up to 100 epochs, and training time was proportional to the volume of datasets. For example, $Train\_Clean\_500$ dataset took around 1.5 h on average for each epoch. Fig. 7 illustrates the training loss comparison of the proposed model for the different used datasets. Clearly, the model gradually learned without undergoing over-fitting or under-fitting problems. The minimum-maximum values of the training loss for

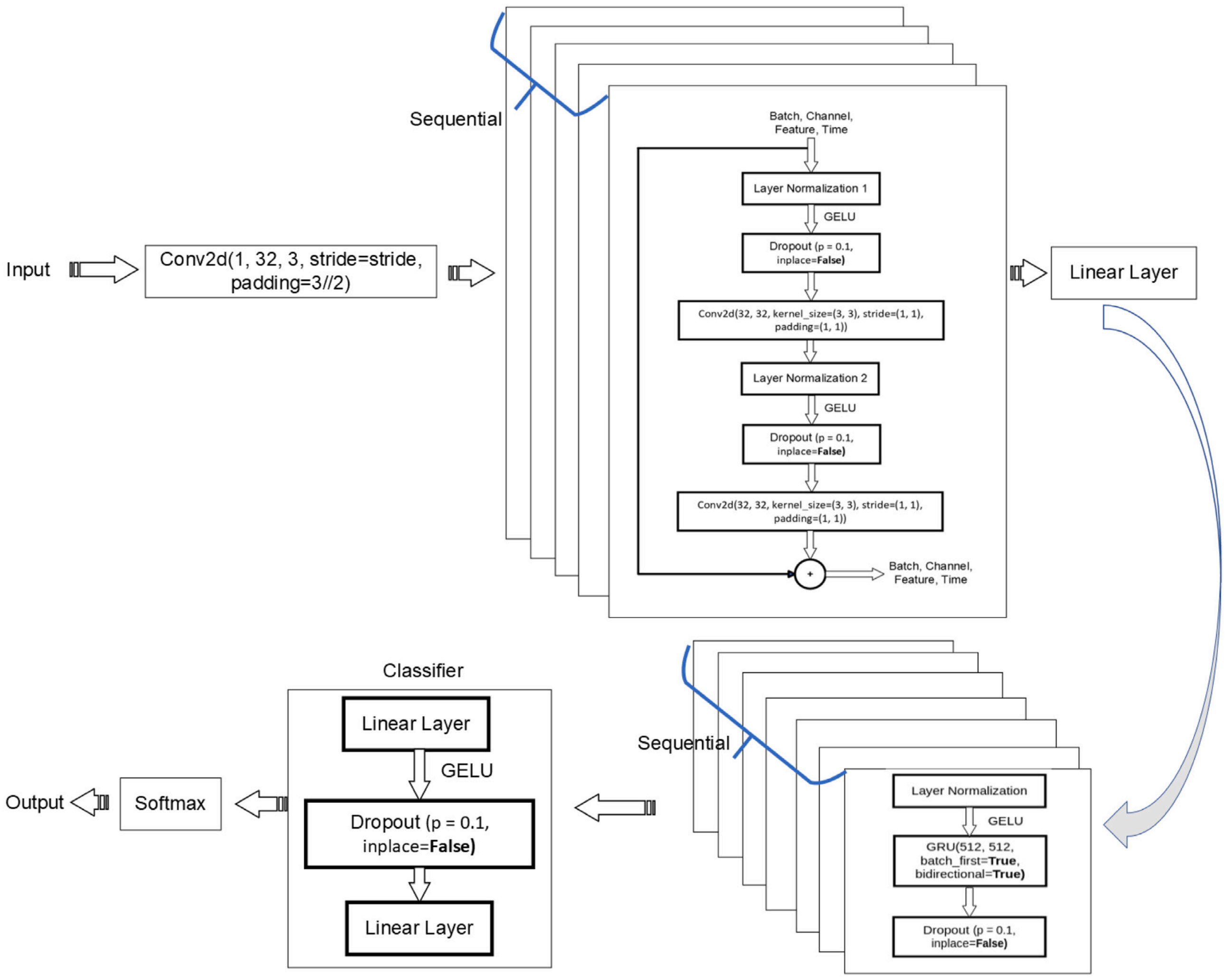
Input
Conv2d(1, 32, 3, stride=stride, padding=3//2)
Sequential
Batch, Channel, Feature, Time
Layer Normalization 1
GELU
Dropout (p = 0.1, inplace=False)
Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
Layer Normalization 2
GELU
Dropout (p = 0.1, inplace=False)
Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
+
Batch, Channel, Feature, Time
Linear Layer
Sequential
Layer Normalization
GELU
GRU(512, 512, batch_first=True, bidirectional=True)
Dropout (p = 0.1, inplace=False)
Classifier
Linear Layer
GELU
Dropout (p = 0.1, inplace=False)
Linear Layer
Softmax
Output

Fig. 6. Architecture of the proposed model for speech recognition.

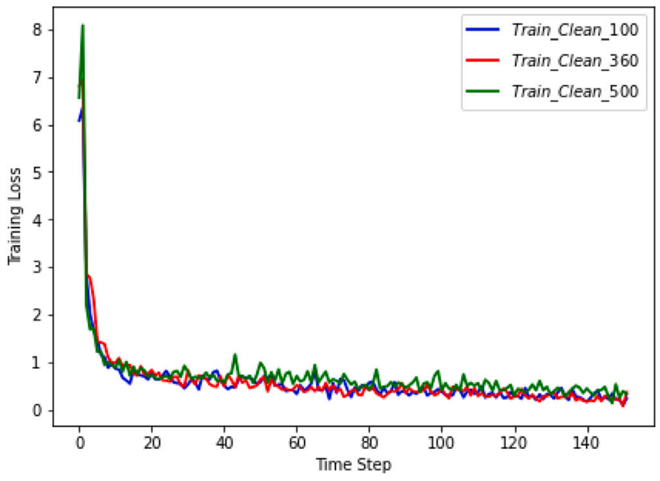

Fig. 7. Obtained Training loss along time for each used dataset.

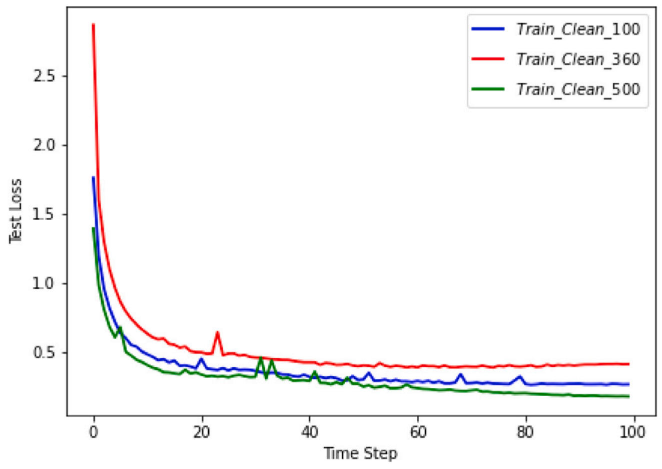

Fig. 8. Obtained test loss along time for each used dataset.

$Train\_Clean$_100, $Train\_Clean$_360, and $Train\_Clean$_500 datasets were equal to $(0.078-6.945)$, $(0.097-6.351)$, and $(0.141-8.078)$, respectively.

In the case of the LibriSpeech dataset, the model testing used a separate dataset from the same corpus: $Test\_Clean$. The Greedy Decoder, which selects the word/character with the highest probability, i.e., acts greedily, was implemented for decoding purposes. Fig. 8 illustrates the test loss of the model in terms of the different used datasets. The Minimum-Maximum test loss values were equal to 0.39 – 2.862, 0.266 – 1.758, and 0.182 – 1.393, for $Train\_Clean$_100, $Train\_Clean$_360, and $Train\_Clean$_500 datasets, respectively.

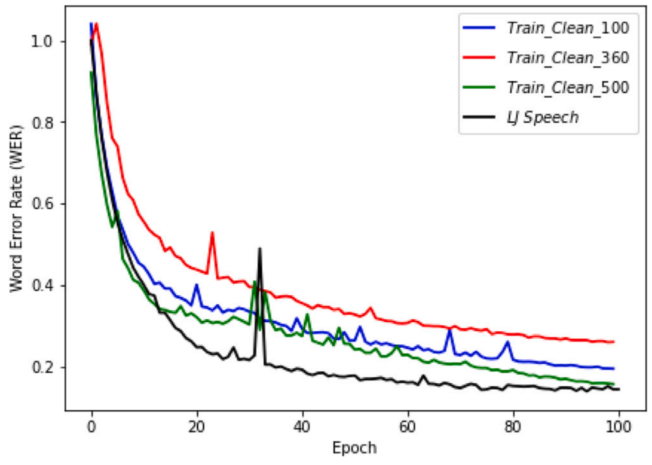

Fig. 9. Obtained word error rate for each used dataset.

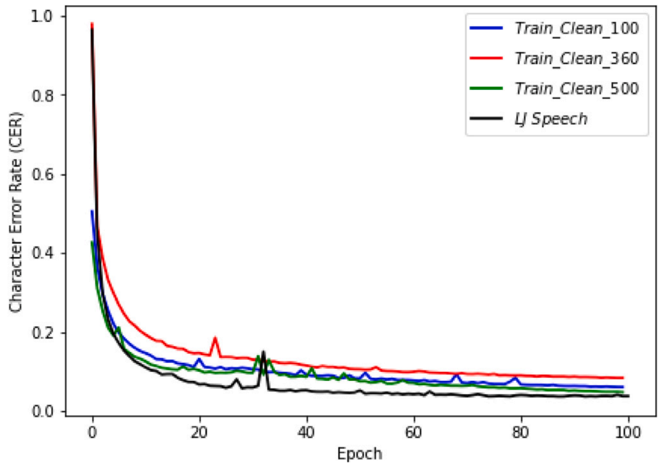

Fig. 10. Obtained Character error rate for each used dataset.

5. Results

The model evaluation used state-of-the-art metrics for speech recognition tasks: Word Error Rate and Character Error Rate. It is to be noted that, in the literature, almost all the researchers use these two metrics for ASR model evaluation.

5.1. Word and character error rates

These two error functions measure how much a model misreads text in a transcription. WER computes the Levenshtein distance (Yujian & Bo, 2007) between the reference sequence and hypothesis sequence in word level, while CER computes it in character level. If S_w is the number of words substituted, D_w the number of words deleted, I_w the number of words inserted, and N_w the number of words in the reference, then WER is:

$$WER = (S_w + D_w + I_w)/N_w. \tag{9}$$

Similarly, if S_c is the number of characters substituted, D_c the number of characters deleted, I_c the number of the character inserted, and N_w the number of characters in the reference, then CER can be expressed as:

$$CER = (S_c + D_c + I_c)/N_c. \tag{10}$$

The lower these error metric values, the better the model's performance, with a CER or WER of 0 (zero) being a perfect score. Because words, unlike characters, are of varying lengths and do not offer a clear comparison, the WER value might have little importance for the model's quality, i.e., a word is already incorrectly recognized if just one letter in it is not correct. As a result, the WER should rarely describe a model's worthiness.

This study performed the model evaluation based on the aforementioned experimental setups and parameters. Figs. 9 and 10 present the WER and CER values obtained by the proposed model for the different used datasets. After a few trial-error tests, the obtained hyper-parameters described in Section 4.3 provided the following model outcomes. For the LibriSpeech dataset, the minimum value of WER of the model were equal to 25.9, 19.4, and 15.6%, and for CER, they were 8.3, 6.0, and 4.7% on $Train_Clean$_100, $Train_Clean$_360, and $Train_Clean$_500 datasets, respectively. The tested dataset was the $Test\_Clean$, which contains 5.4 h of clean audio speech in all those cases. On the other hand, based on the test dataset obtained from data splitting as described in Section 4.3, the model exhibited minimums CER and WER of 3.61 and 13.86% on the LJ Speech dataset, respectively, as can be verified in Table 1. The model was trained up to 100 epochs for each of the aforementioned cases, and it might even exhibit better performance gains for training over 100 epochs.

Table 1
Performance comparison in terms of different metrics on different datasets.

Train datasets	CER (MIN%)	WER (MIN%)	Duration
$Train_Clean$_100	8.3	25.9	40 : 26 : 24
$Train_Clean$_360	6.0	19.4	71 : 35 : 22
$Train_Clean$_500	4.7	15.6	158 : 51 : 15
LJ Speech	3.61	13.86	21 : 15 : 27

Number of model parameters = 33196445, duration is represented in hour : minute : second).

Table 2
Prediction performance of the model on the LJ Speech dataset.

Attributes	Transcription
Target 1	*that he compared the four empty cartridge cases found near the scene of the shooting with a test cartridge fired from the weapon in Oswald's possession when he was arrested*
Predicted 1	*that he compared the four empty cartridge cases found near the scene of the shooting with a test cartridge fired from the weapon in Oswald's possession when he was arrested*
Target 2	*in an instant our faces were covered we cocked our pistols and with drawn swords stood waiting to receive the enemy*
Predicted 2	*in an instant **are** faces were covered we **cockedar** pistols and with drawn **sords** stood waiting to receive the enemy*

The bold words represents wrongly predicted words.

The proposed model demonstrated its effectiveness in predicting the target transcriptions accurately. Table 2 depicts the model's performance in predicting the speech transcriptions on the LJ Speech dataset. Two arbitrary target and predicted transcriptions were taken from the test dataset when the CER and WER values were 3.7 and 14.39%, respectively. The second row presents the correctly predicted transcriptions, and the last contains three wrongly predicted words within the target. The model failed to correctly predict characters within the words 'our', 'cocked our', and 'swords'. Nevertheless, the proposed model acquired a minimum CER of only 3.61% on the LJ Speech dataset, meaning that 96.39% characters, including letters, punctuation, special symbols and spaces of the target transcriptions, were correctly predicted. Hence, the results of Table 2 justify its effectiveness in the current study context.

6. Discussion

This study aimed to build a deep neural network-based model that is computationally efficient and produces good outcomes. As a solution,

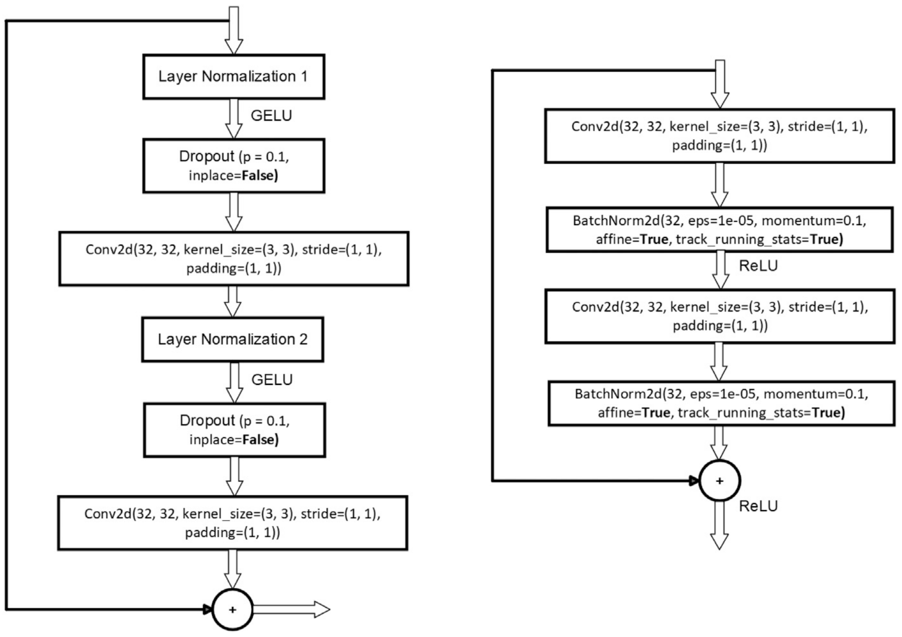
Layer Normalization 1
GELU
Dropout (p = 0.1, inplace=False)
Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
Layer Normalization 2
GELU
Dropout (p = 0.1, inplace=False)
Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
+
Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
ReLU
Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
+
ReLU

Fig. 11. Model comparison as to the used residual neural network architecture.

this research suggests five layers of customized ResNets with seven layers of a Bi-GRUs-based ASR model. The LibriSpeech ASR corpus, which has 1000 hours of audio speech, and the LJ Speech datasets, were used for training and testing the proposed model.

During the development of the proposed model, different architectures of the ResNets layers were examined. The layer normalization instead of bath normalization within the ResNets block produced better results. The batch normalization in the network reduced both the train and test losses; nonetheless, there was a significant reduction in terms of WER and CER. Applying ResNets without batch normalization layers improved the CER and WER by 73.3 and 53.93%, respectively, reducing the train and test losses by 12.46 and 36.95%, respectively. Fig. 11 presents two different examined architectures of the ResNets layers. The left residual block consists of two-layer normalization, dropout, convolution layers using the GELU activation function, and a skip connection according to the provided arrangement. Furthermore, the residual block on the right employs two 2-dimensional (2D) batch normalization convolution layers using the ReLU activation function and a skip connection according to the illustrated arrangement. Interestingly, the residual block on the left provided much better results; hence, the proposed model implements the ResNets architecture according to it.

This study suggests replacing batch normalization layer with layer normalization to achieve better ASR accuracy. As the batch size highly influences batch normalization, the proposed model was trained with different batch sizes to examine whether the obtained good performances are sustained with an increase in the used batch size. A batch normalization layer must compute the mean and variance to normalize the previous outputs across the batch. Hence, for small batch sizes, its estimation will be less accurate. Here, layer normalization is proposed to address the shortcoming. This study examined three different batch sizes: 10, 16 and 32. The variations of different batch sizes exhibited some small effects on the model performance, as can be seen in Fig. 12. For $Train\_Clean\_100$ and $Test\_Clean$ as the train and test datasets, the proposed model achieved a slight improvement in terms of CER by 4.4 and 7.56% with a batch size of 10 compared to its value of 16 and 32, respectively. Overall, the model is capable of maintaining its good performances regardless of the batch size.

The training data significantly influences the model's accuracy and increases the computational demands. The training on the $Train\_Clean\_500$ dataset containing 500 hours of English audio speech

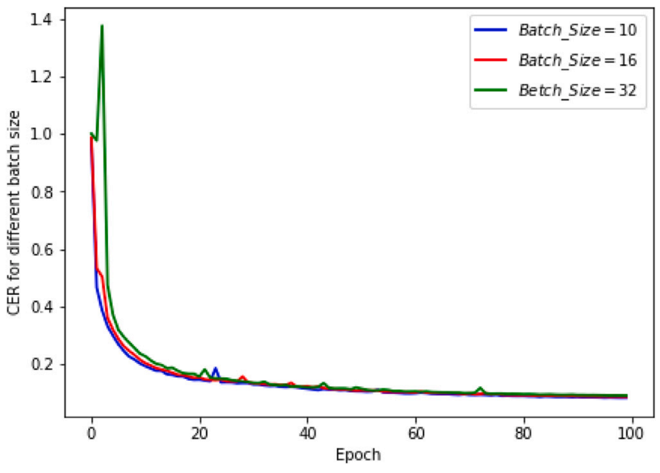

Fig. 12. Obtained CER for different values of batch size.

led to the best performance. For 500 hours of audio data, the CER had an increase of 43.37 and 26.67% relatively to the 100 and 500 hours of audio data, respectively.

The number of ResNets and Bi-GRUs layers also demonstrated performance variation. Using 3 and 5 layers of ResNets and Bi-GRUs resulted in a 79.12% performance degradation in terms of CER compared to the current 5 and 7 layers of the parameters previously mentioned. However, in Wang et al. (2018), it was demonstrated that the increase of the GRUs layers beyond 5 or more fails to enhance the performance of a model based on Deep Speech 2 (Amodei et al., 2016). The opposite findings of the current study are due to the insertion of the proposed ResNets layers.

Replacing Adam for AdamW as the optimizer, and GELU instead of ReLU as the activation function, demonstrated their usefulness in the problem under study. AdamW overcame the limitation of Adam of its wrong implementation of the weight decay, and hence, it showed better performance. GELU combines approximated numbers and various functions, e.g., hyperbolic tangent ($tanh$), and has a negative coefficient that changes to a positive coefficient as time goes on; for example, when x

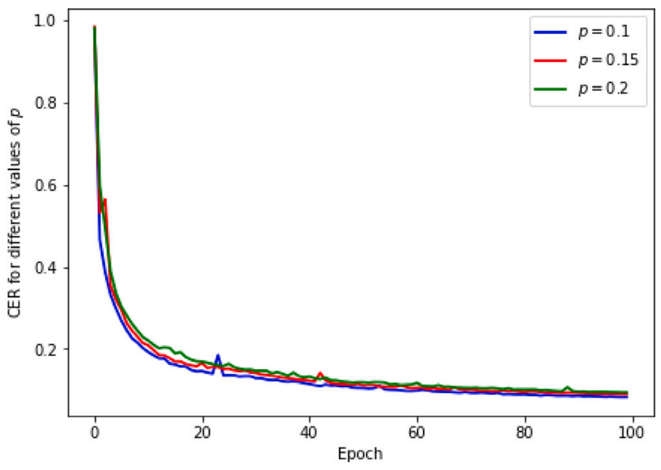

Fig. 13. Obtained CER for different values of p.

is more significant than zero, the output will be x, except for the range 0 (zero) to 1 (one), which tends toward a smaller y value. Because of its formulation, GELU showed better results than ReLU.

This study also examined the model performance in terms of different values of p, which is defined as a measure of the likelihood of a component being zeroed within the dropout layers, as can be seen in Fig. 13. The model performance seems to decrease gradually with the increase in p values. In terms of statistics, for $Train\_Clean\_100$ and $Test\_Clean$ as the train and test datasets, the model achieved 7.98 and 12.2% of improvements in terms of CER with a p of 0.1 compared to its value of 0.15 and 0.20, respectively.

The current study also trained the proposed model on the LibriTTS dataset (Zen et al., 2019), and the performance was less impressive compared to the LibriSpeech and LJ Speech datasets, which can be seen as a drawback of the proposed model. The LibriTTS contains 49 different characters instead of 29 and 31 of the used datasets. Text mapping is one of the crucial steps for effective speech recognition. With the increase in the number of characters in datasets, the proposed model tends to become less effective in terms of accuracy. For example, the special characters ", [, ;, ?, } and / were difficult to correctly predict. So, the exhibited poor performance of the model, in this case, was due to the increased number of characters in the data samples.

Compared with some already published state-of-the-art models, the proposed model requires fewer parameters while maintaining excellent CER values on the LibriSpeech and LJ Speech datasets. However, the application of depthwise separable convolution layers could also reduce the computational costs by a more significant margin (Han et al., 2020 and Kriman et al., 2020). The authors argue that this approach is likely to fail to learn properly during training if the network size is small. Also, a tremendous amount of training steps (around 1200 epochs) are required to acquire state-of-the-art accuracy. The currently proposed model achieved good accuracy with only 100 epochs of training and is suitable for any network regardless of its size. Table 3 depicts a comprehensive comparison of the proposed model with some already published research based on the LibriSpeech corpus. The comparisons were made regarding the number of parameters, CER and WER values. It exhibited the best CER value among the mentioned works. However, the obtained WER values are less competitive than other baselines under consideration, which is the future scope of this research.

7. Conclusion

This study aimed to develop an effective ASR model without the requirement of any external language model to provide good results while relying on fewer computational resources. ResNets's suitability for smoothing model training and giving excellent performances was validated. In addition, layer normalization instead of batch normalization was shown to be more effective in ASR tasks in improving the model's performance. For optimizer, AdamW, instead of traditional Adam, demonstrated better results. Also, the number of ResNets and Bi-GRU layers used in the model played a crucial role in acquiring competitive performance. The number of training datasets greatly influenced the achievement of the best performance. For example, the model achieved an improvement of CER by 43.37 and 26.67% while using 500 hours of audio data relative to the other used datasets. The proposed model demonstrated superior performance using less computational costs in CER by surpassing previous state-of-the-art models on the LibriSpeech datasets. However, the proposed model was less effective than other baselines regarding WER. A word is already incorrectly recognized if just one letter in it is not correct. Hence, with the tremendous results on CER (4.7% means 95.3% characters, i.e., letters, punctuation and spaces, were correctly recognized), it should be enough to justify the efficiency of the model. Henceforth, the future emphasis would be given to employing the model effectively on other datasets, for example, on the MLS dataset suggested in Pratap, Xu, Sriram, Synnaeve, and Collobert (2020), and the introduction of a 1D CNN network to further reduce the computational costs.

Table 3
Performance comparison of different published models on LibriSpeech $Test\_Clean$ dataset.

Models	Parameters (M)	WER (%)	CER (%)
Likhomanenkoet al. (2021)	270	2.8*	N/A
Park et al. (2019)	360	2.8*	N/A
Synnaeve et al. (2019)	270	2.89	N/A
Karita et al. (2019)	130	18.0	8.4
Moritz et al. (2019)	70	5.4	6.6
Wang et al. (2018)	35	24.55	7.06
Proposed model	33*	15.6	4.7*

The values with* represent the best performance and N/A means not available, i.e., the authors did not use CER for model evaluation.

CRediT authorship contribution statement

Selim Reza: Investigation, Data collection, Code implementation, Formal analysis, Writing – original draft. **Marta Campos Ferreira:** Writing – review & editing. **J.J.M. Machado:** Writing – review & editing. **João Manuel R.S. Tavares:** Conceptualization, Funding acquisition, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data is public available.

Acknowledgements

This article results from the project Safe Cities– "Inovação para Construir Cidades Seguras", with reference POCI-01-0247-FEDER-041435, co-funded by the European Regional Development Fund (ERDF), through the Operational Programme for Competitiveness and Internationalization (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement.

References

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning* (pp. 173–182). PMLR.

Benzeghiba, M., De Mori, R., Deroo, O., Dupont, S., Erbes, T., Jouvet, D., et al. (2007). Automatic speech recognition and speech variability: A review. *Speech Communication*, *49*(10–11), 763–786.

Chan, W., Jaitly, N., Le, Q., & Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing* (pp. 4960–4964). IEEE.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Fan, R., Chu, W., Chang, P., Xiao, J., & Alwan, A. (2021). An improved single step non-autoregressive transformer for automatic speech recognition. *vol. 2*, In *Proceedings of the annual conference of the international speech communication association, INTERSPEECH* (pp. 1406–1410). http://dx.doi.org/10.21437/Interspeech.2021-1955.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on machine learning* (pp. 369–376).

Haase, D., & Amthor, M. (2020). Rethinking depthwise separable convolutions: How intra-kernel correlations lead to improved mobilenets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 14600–14609).

Han, W., Zhang, Z., Zhang, Y., Yu, J., Chiu, C.-C., Qin, J., et al. (2020). ContextNet: Improving convolutional neural networks for automatic speech recognition with global context. In *Proc. interspeech 2020* (pp. 3610–3614).

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., et al. (2014). Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567.

Hwang, Y., Cho, H., Yang, H., Won, D.-O., Oh, I., & Lee, S.-W. (2020). Mel-spectrogram augmentation for sequence to sequence voice conversion. arXiv preprint arXiv:2001.01401.

Ito, K., & Johnson, L. (2017). The LJ speech dataset. https://keithito.com/LJ-Speech-Dataset/.

Junqua, J.-C., & Haton, J.-P. (2012). *vol. 341*, *Robustness in automatic speech recognition: Fundamentals and applications*. Springer Science & Business Media.

Karita, S., Watanabe, S., Iwata, T., Delcroix, M., Ogawa, A., & Nakatani, T. (2019). Semi-supervised end-to-end speech recognition using text-to-speech and autoencoders. In *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing* (pp. 6166–6170). IEEE.

Kriman, S., Beliaev, S., Ginsburg, B., Huang, J., Kuchaiev, O., Lavrukhin, V., et al. (2020). Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions. In *ICASSP 2020-2020 IEEE International conference on acoustics, speech and signal processing* (pp. 6124–6128). IEEE.

Lee, J., & Watanabe, S. (2021). Intermediate loss regularization for ctc-based speech recognition. In *ICASSP 2021-2021 IEEE international conference on acoustics, speech and signal processing* (pp. 6224–6228). IEEE.

Li, P., Luo, A., Liu, J., Wang, Y., Zhu, J., Deng, Y., et al. (2020). Bidirectional gated recurrent unit neural network for Chinese address element segmentation. *ISPRS International Journal of Geo-Information*, *9*(11), 635.

Li, S., Xu, M., & Zhang, X.-L. (2021). Efficient conformer-based speech recognition with linear attention. In *2021 Asia-Pacific signal and information processing association annual summit and conference* (pp. 448–453).

Likhomanenko, T., Xu, Q., Pratap, V., Tomasello, P., Kahn, J., Avidov, G., et al. (2021). Rethinking evaluation in ASR: Are our models robust enough? In *Proc. Interspeech 2021* (pp. 311–315). http://dx.doi.org/10.21437/Interspeech.2021-1758.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., et al. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Majumdar, S., Balam, J., Hrinchuk, O., Lavrukhin, V., Noroozi, V., & Ginsburg, B. (2021). Citrinet: Closing the gap between non-autoregressive and autoregressive end-to-end models for automatic speech recognition. arXiv preprint arXiv:2104.01721.

Moritz, N., Hori, T., & Le Roux, J. (2019). Triggered attention for end-to-end speech recognition. In *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing* (pp. 5666–5670). IEEE.

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing* (pp. 5206–5210). IEEE.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., et al. (2019). SpecAugment: A simple data augmentation method for automatic speech recognition. In *Proc. interspeech 2019* (pp. 2613–2617). http://dx.doi.org/10.21437/Interspeech.2019-2680.

Pratap, V., Xu, Q., Sriram, A., Synnaeve, G., & Collobert, R. (2020). Mls: A large-scale multilingual dataset for speech research. arXiv preprint arXiv:2012.03411.

Qasim, S. E., Fried, I., & Jacobs, J. (2021). Phase precession in the human hippocampus and entorhinal cortex. *Cell*, *184*(12), 3242–3255.

Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization? In *Proceedings of the 32nd international conference on neural information processing systems* (pp. 2488–2498).

Synnaeve, G., Xu, Q., Kahn, J., Likhomanenko, T., Grave, E., Pratap, V., et al. (2019). End-to-end asr: from supervised to semi-supervised learning with modern architectures. arXiv preprint arXiv:1911.08460.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*.

Thys, T. M., Treviño, J., & Nadkarni, N. M. (2021). Perceptual–acoustic comparisons of natural sonic environments: Applications for nature-deprived populations. *Ecopsychology*, *13*(3), 151–164.

Wang, K., Guan, D., & Li, B. (2018). Deep group residual convolutional CTC networks for speech recognition. In *International conference on advanced data mining and applications* (pp. 318–328). Springer.

Yu, D., & Deng, L. (2016). *Automatic speech recognition*. Springer.

Yujian, L., & Bo, L. (2007). A normalized levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(6), 1091–1095.

Zen, H., Dang, V., Clark, R., Zhang, Y., Weiss, R. J., Jia, Y., et al. (2019). LibriTTS: A corpus derived from LibriSpeech for text-to-speech. arXiv preprint arXiv:1904.02882.

Zeyer, A., Irie, K., Schlüter, R., & Ney, H. (2018). Improved training of end-to-end attention models for speech recognition. In *Proc. interspeech*. International Speech Communication Association, http://dx.doi.org/10.21437/interspeech.2018-1616.

Zhao, Y., Yang, X., Wang, J., Gao, Y., Yan, C., & Zhou, Y. (2021). BART based semantic correction for mandarin automatic speech recognition system. In *Proc. interspeech 2021* (pp. 2017–2021). http://dx.doi.org/10.21437/Interspeech.2021-739.