Contents lists available at ScienceDirect

Expert Systems With Applications

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

End-to-end automated speech recognition using a character based small scale transformer architecture

Alexander Loubser [*], Pieter De Villiers, Allan De Freitas

Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Private Bag X20, Hatfield 0028, South Africa

ARTICLE INFO

Keywords:
Speech recognition
Transformer
End-to-end
Character based
Connectionist temporal classification

ABSTRACT

This study explores the feasibility of constructing a small-scale speech recognition system capable of competing with larger, modern automated speech recognition (ASR) systems in both performance and word error rate (WER). Our central hypothesis posits that a compact transformer-based ASR model can yield comparable results, specifically in terms of WER, to traditional ASR models while challenging contemporary ASR systems that boast significantly larger computational sizes. The aim is to extend ASR capabilities to under-resourced languages with limited corpora, catering to scenarios where practitioners face constraints in both data availability and computational resources. The model, comprising a compact convolutional neural network (CNN) and transformer architecture with 2.214 million parameters, challenges the conventional wisdom that large-scale transformer-based ASR systems are essential for achieving high accuracy. In comparison, contemporary ASR systems often deploy over 300 million parameters. Trained on a modest dataset of approximately 3000 h – significantly less than the 50,000 h used in larger systems – the proposed model leverages the Common Voice and LibriSpeech datasets. Evaluation on the LibriSpeech test-clean and test-other datasets produced character error rates (CERs) of 6.40% and 16.73% and WERs of 16.03% and 35.51% respectively. Comparisons with existing architectures showcase the efficiency of our model. A gated recurrent unit (GRU) architecture, albeit achieving lower error rates, incurred a computational cost 24 times larger than our proposed model. Large-scale transformer architectures, while achieving marginally lower WERs (2%–4% on LibriSpeech test-clean), require 200 times more parameters and 53,000 additional hours of training data. Modern large language models are used to improve the WERs, but require large computational resources. To further enhance performance, a small 4-g language model was integrated into our end-to-end ASR model, resulting in improved WERs. The overarching goal of this work is to provide a practical solution for practitioners dealing with limited datasets and computational resources, particularly in the context of under-resourced languages.

1. Introduction

Automated speech recognition (ASR) is the process of converting spoken language or a command into text using computer processing techniques (Maas, Xie, Jurafsky, & Ng, 2015). End-to-end automated speech recognition utilizes modern architectures, such as transformers, to directly translate the audio data into text without the need of phoneme or lexicon data. Traditional ASR models employed large acoustic models, such as those discussed in Liao, McDermott, and Senior (2013) and Li, Akita, and Kawahara (2016), integrating deep neural networks (DNNs) with hidden Markov models (HMMs) and feed-forward neural layers. These models, reliant on extensive dictionaries and language models, demanded large-scale architectures with billions of parameters for satisfactory performance. The advent of recurrent neural network (RNN) models and long short-term memory (LSTM) models marked a pivotal moment in ASR, enabling contextual awareness through sequential speech data (Li et al., 2014; Long, Li, Wei, Zhang, & Yang, 2019; Wu et al., 2017).

A language model compares the joint probabilities of sequences of the predicted words in a sentence. The model uses the collection of words in the sentence to predict the most probable form of each word in the sentence. Language models, crucial for predicting word sequences in a sentence, underwent significant improvements with the introduction of RNNs and LSTMs (Chen, Liu, Wang, Gales, & Woodland, 2016; Lippi, Montemurro, Degli Esposti, & Cristadoro, 2019; Sundermeyer, Ney, & Schluter, 2015; Williams, Prasad, Mrva, Ash, & Robinson, 2015). Modern large language models (LLMs) such as GPT-3/4 and BERT improved contextual understanding above and beyond RNNs and LSTMs and the increased contextual awareness offered by these models led to enhanced performance, particularly with larger

[*] Corresponding author.
E-mail addresses: a.loubser@tuks.co.za (A. Loubser), pieter.devilliers@up.ac.za (P. De Villiers), allan.defreitas@up.ac.za (A. De Freitas).

https://doi.org/10.1016/j.eswa.2024.124119
Received 18 December 2023; Accepted 25 April 2024
Available online 1 May 2024

datasets. However, computational demands and training times also escalated drastically and the LLMs also changed words to match better contextually instead of producing the correct words from the speech audio received. This caused a reduction in WER in some cases (Kubo, Karita, & Bacchiani, 2022; Min & Wang, 2023).

The introduction of attention based models such as transformers from Vaswani et al. (2017), allowed ASR models to be more efficient and deliver the same results with less computational power required. The transformer architecture was first developed for machine translation and text based tasks where it improved the results compared to any previous RNN and CNN networks (Luong, Pham, & Manning, 2015). Transformer based models use encoders and decoders linked with a self attention system and improve on RNN and CNN models due to the recurrence and convolution steps falling away. This allows more sequential data to be used with less memory and fewer parameters required. In this paper, a novel approach is presented where a small character based ASR model is implemented, using a transformer architecture. Contrary to the trend of larger transformer-based models, our approach challenges the notion that high accuracy requires a vast number of parameters. The model from this work is compared to larger transformer based models and RNN models. The model consists of a compact convolutional neural network and a transformer network that amounts to 2.214 million parameters, a fraction of the size of contemporary ASR system, to predict characters based on the input audio features. The input audio features and text is aligned by leveraging connectionist temporal classification (CTC) from Graves, Fernández, Gomez, and Schmidhuber (2006).

ASR requires large datasets and different machine learning algorithms to successfully translate audio into text. Feature extraction takes place by separating the audio data, based on audio properties such as pitch and frequency. Standard audio features are extracted from the raw audio data such as the Mel frequency cepstral coefficient (MFCC) and the delta coefficients the MFCC values (Vergin, O'Shaughnessy, & Farhat, 1999). The use of Mel Frequency Cepstral Coefficients (MFCC) as input features not only enriches the feature representation but also reduces dimensionality, contributing to the efficiency of the model. Two publicly available datasets used for the training and testing of the ASR model are known as the Mozilla common voice dataset (Mozilla, 2021) and the LibriSpeech dataset (Panayotov, Chen, Povey, & Khudanpur, 2015). The data is divided into training, development and testing data. The total data consists of about 3000 h of English speech with different dialects, which is limited as larger ASR models are trained with over 52000 h of data. The model is trained using CTC loss and the validation data CTC loss is used to tune hyperparameters during training and to prevent overtraining.

Multiple performance metrics are considered for evaluating the ASR model to ensure the model evaluation is accurate. The word error rate (WER) and character error rate (CER) are performance matrices to determine the percentage of words and characters that were predicted correctly. The model aims to minimize the footprint of ASR models, making it suitable for scenarios with limited access to extensive datasets and computational resources. The architecture comprises a 2-layered convolutional neural network followed by a 2-headed, triple-layer transformer for contextual training. Notably, the output provides character-based predictions, subsequently refined by a small 4-gram language model for improved word error rates. Despite its reduced scale, our architecture demonstrates comparable results to larger ASR models, such as GRU, Wav2Vec2 and other existing transformer models (Baevski, Zhou, Mohamed, & Auli, 2020; Drexler & Glass, 2019; Gulati et al., 2020).

2. Related work

In recent research, there has been notable exploration in enhancing Automatic Speech Recognition (ASR) systems leveraging LLMs and knowledge transfer techniques. In Min and Wang (2023), the integration of LLMs into ASR systems was investigated to improve transcription accuracy. The study, utilizing the Aishell-1 and LibriSpeech datasets, evaluated the potential of employing LLMs' in-context learning capabilities. Despite initial experiments resulting in higher WER, the study shed light on the challenges of effectively leveraging LLMs for ASR applications. Another approach presented in Kubo et al. (2022) addresses the data hunger challenge in training end-to-end ASR systems by transferring knowledge from pretrained language models. The paper proposes a method to transfer semantic knowledge from embedding vectors of large-scale language models to improve ASR decoders' performance without added computational costs during decoding. The research in Kubo et al. (2022) is attempting to improve speech recognition by improving the language context with the use of LLMs.

In Gulati et al. (2020), a Conformer is introduced, an architecture that combines CNNs and Transformers for ASR tasks. This model achieves state-of-the-art accuracies on the LibriSpeech benchmark, outperforming previous Transformer and CNN-based models by effectively modelling both local and global dependencies in audio sequences. Our approach is similar but uses a CNN architecture before the transformer architecture, while Gulati et al. (2020) builds the CNN inside of the transformer and the architecture is noticeably bigger in parameter size. Our approach uses MFCC audio features while the approach in Gulati et al. (2020) uses raw audio. In Winata, Cahyawijaya, Lin, Liu, and Fung (2020), a low rank transformer was used for an end-to-end speech recognition system with reduced parameters and increased training speed and inference. The model was based on mandarin speech and produced a CER of 13.6% for an 8.7 million parameter model without an added language model on the mandarin AiShell-1 dataset. Maas et al. (2015) and Pham et al. (2019) produce character based ASR models with an added language model. The models vary and have up to 48 layer transformers. All the ASR models consist of CTC based transformer architectures.

Synnaeve et al. (2019) compares multiple end-to-end transformer models with the LibriSpeech (Panayotov et al., 2015) dataset. The paper shows that transformer models are superior with little speech data, but as the data is increased all models converge to the same results and rely less on language models for accurate results. The transformer model with 322 million parameters and using CTC training, achieved a WER of 2.99% and 7.31% on the dev-clean and dev-other data respectively. The transformer also achieved a WER of 3.09% and 7.40% on the test-clean and test-other data respectively. Adding a 4-gram language model increased the model accuracy slightly to a WER of 2.86% and 6.72% on the test-clean and test-other data respectively.

The model in Drexler and Glass (2019) uses CTC based training for end-to-end ASR and is tested on the LibriSpeech dataset. The model consists of 2 convolutional layers and 5 GRU layers. The model produces a WER of 11.9 and 31.1 on the test-clean and test-other dataset respectively. An added language model produces a WER of 8.3 and 24.4 on the test-clean and test-other dataset respectively. Hwang and Sung (2016) also uses an older LSTM based architecture to create a character based ASR model.

Modern very large ASR systems such as Baevski, Hsu, Conneau, and Auli (2021), produce lower word error rates, but are very large and take a lot of computational power to train and run. A WER of 3.8% and 6.5% on the LibriSpeech test-clean and test-other data is achieved by Baevski et al. (2021), that uses a very large transformer based ASR model of over 300 million parameters called the Wav2Vec architecture and a 4 gram language model. Baevski et al. (2020) achieves a WER of 1.8% and 3.3% on the LibriSpeech clean and other test set. The model is trained using 53.2k hours of audio. The model contains 24 transformer blocks of dimension 1024 and inner dimensions of 4096 and 16 attention heads. The model is accompanied by a transformer based language model. The large model has 317 million parameters without the language model. Including the language model, it consists of over a billion parameters. In Conneau, Baevski, Collobert, Mohamed, and Auli (2020), a cross-lingual speech representation model called XLSR

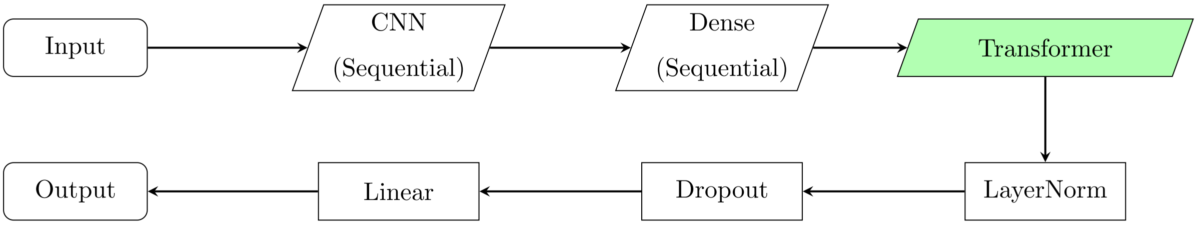

Fig. 1. A flow diagram describing the basic layout of the architecture used to train the small scale ASR system. The model consists of a 1 dimensional CNN layer and a dense layer to restructure and normalize the audio data. A transformer layer is used to train the model according to the sequential correlation of the data. The normalization and linear layers after the transformer are used to convert the data into 30 classes of the output characters of the model.

is created from the wav2vec2.0 model from Baevski et al. (2020). The existing model is fine-tuned with common voice data (Mozilla, 2021) consisting of 53 languages and other multilingual datasets. The larger amount of training data reduces the WER drastically when compared to other models. The model produces an average WER of English common voice testing data of 7.6% over 53 languages with a 4 gram language model. Using the Common Voice en 7.0 dataset in Grosman (2021) the Wav2Vec2 XLSR model trained on 53k hours of data produced a WER of 27.72 and a CER of 11.65 for the testing dataset. Adding a language model produced a WER of 20.85 and a CER of 11.01. A new transformer model named XLS-R, Babu et al. (2021), produces a 2 billion parameters, transformer based, ASR system that is trained on 436k hours of data from over 128 languages. The model outperforms all previous single language models for languages that have a small amount of training data available. For English the Wav2Vec2 model outperforms the new XLS-R model when using the LibriSpeech data for comparison by a WER of 5.6% compared to 5.9% using the test-clean dataset.

3. Data PreProcessing

3.1. Audio data

The audio input to the model is preprocessed in order to get sufficient information from the audio to predict text. Generally the audio data is transformed into a spectrogram to extract as much frequency and time based data from the audio files as possible. The Mel spectrogram is converted from the spectrogram based on the perceived frequency or pitch from human hearing. The equation,

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

where f is the frequency and $M(f)$ is the Mel frequency from Vergin et al. (1999), explains the transition from the spectrogram to the Mel spectrogram. To reduce the input features, while containing the necessary data, and create a smaller model, the Mel frequency cepstral coefficients (MFCC) are used. The Mel spectrogram is log-scaled and the discrete cosine transform (DCT) is computed to get the given MFCC values (Vergin et al., 1999). A total of 81 Mel spectrogram features were selected for each time segment. The 81 Mel features were decreased to 16 MFCCs. The deltas of the MFCC values were also calculated to add to the features of the audio and take into account change in frequency to improve the model for multiple different speakers. The MFCC features are more decorrelated than the Mel spectrogram which is beneficial to the linear parts of the model.

Different preprocessing methods such as “SpecAugment” and time stretching was implemented to change random selected audio data slightly during training. This virtually increased training data and made the ASR model more generalized for all different speech types. The first preprocessing method was completed by augmenting the audio features during training using the “SpecAugment” method. This method removes a small percentage of randomly selected data features. The method removes some frequency and time components of the audio features to create a larger variety of training data and a more robust model. The second method was completed by applying time stretching of 10% randomly to the data during training, to ensure that the speed of the audio is taken into account.

3.2. Text data

The correlating text data was preprocessed into lower-case, English alphabetic letters only. A few added classes were also included such as a space and an apostrophe. These characters could change the way the audio is perceived and were therefore added. The text or label data was tokenized into numerical values for class vector creation to train the model. An extra class was added for unknown characters or letters not previously seen by the model. The label data and the preprocessed audio data are not the same size and therefore each input value does not have a given class. To fix this for training, connectionist temporal classification (CTC) is used to align the audio data to the label data. Each input X is given an output distribution over all possible labels Y.

4. Model

The ASR model architecture, as in Fig. 1, consists of a sequential CNN and a transformer network that are connected with normalization layers. The CNN is used to expand the input audio features into multi-layered neurons or data points. The audio features are abstracted into a feature map that can be used by the transformer architecture. The CNN consists of a one dimensional convolution layer, a dropout layer and a normalization layer as seen in Fig. 2. The dense architecture changes the dimensions of the data and normalizes the data for the transformer. The dense architecture consists of a linear and normalization layer as well as a Gaussian error linear unit (GELU) layer and a dropout layer. The 4 layers are repeated twice, and each layer has a size of 128 feature inputs and outputs.

The transformer architecture has a two-headed layout, where each head contains three encoder layers and three decoder layers with a final feedforward layer that has a hidden layer size of 1024 nodes. A basic layout of one head of the transformer can be seen in Fig. 4. The output of the transformer architecture is transferred to a final normalization layer and dropout layer. The output of the dropout layer is reduced to 30 output classes using a final linear feedforward layer.

4.1. Convolutional neural network architecture

The convolutional neural network (CNN) architecture utilizes a convolutional operation to capture spatial context for input data features in multiple dimensions. The MFCC and delta coefficients, representing audio feature data in the frequency and sequential time domains, undergo normalization during pre-processing. The convolutional layer employs a grid of n-dimensional filters to convolve with the input data, capturing local features in time.

4.1.1. Filter design and convolution

The CNN convolutional layer utilizes a filter grid for cross-correlation with sections of the input matrix, shifting horizontally based on a stride parameter l_{stride}. This process generates an output matrix, capturing spatial features in the input. Cross-correlation is described by:

$$\mathrm{R}_{\mathrm{XY}} \triangleq \mathrm{E}[\mathbf{XY}], \tag{2}$$

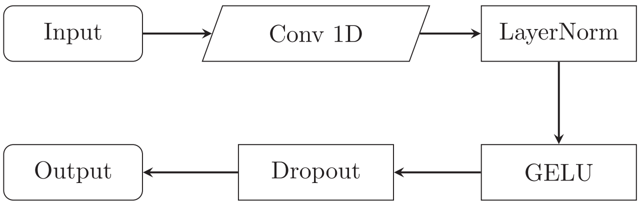

Fig. 2. A diagram describing the basic layout of the CNN architecture used in the small-scale ASR system. The CNN model consists of the main 1 dimensional convolutional layer and additional normalization, GELU and dropout layers.

measuring similarity or displacement between vectors.

The stride distance l_{stride} affects dimensionality reduction (subsampling) in the output matrix. Padding can be added to maintain dimensionality, but reduction aligns with the model's goal of parameter reduction. Output dimensions are given by:

$$l_{out} = \left(\frac{l_{in} - f_{conv} + l_{stride}}{l_{stride}} \right) \tag{3}$$

and

$$w_{out} = \left(\frac{w_{in} - f_{conv} + l_{stride}}{l_{stride}} \right), \tag{4}$$

where $l_{out} \times w_{out}$ represents the output dimension size and $l_{in} \times w_{in}$ represent the input dimension size. In our case l represents the time dimension w represents the Mel frequency dimension. The $l \times w$ image represents the entire spectrogram. The filter size is represented by $f_{conv} \times f_{conv}$ and l_{stride} is the stride length. In the case of audio processing, the 32 audio features are 32 different channels and cross-correlation occurs for the sequential time units over all the channels. This requires only Eq. (3) as the filter and the inputs would both have 32 channels. Adding slight padding does retain some time-based information and ensures that data is not lost between the changes for data over time. Adding padding changes Eq. (3) to:

$$l_{out} = \left(\frac{l_{in} + 2 \times l_{pad} - f_{conv} - 2}{l_{stride}} + 1 \right), \tag{5}$$

where l_{pad} is the padding size.

4.1.2. Model implementation

The chosen CNN applies one-dimensional convolution over the time-based dimension for all 32 audio feature channels. The convolution process, denoted by $\star$, is described by:

$$\left(N_{\text{bat}}, C_{\text{out}}\right) = b(C_{\text{out}}) + \sum_{k=0}^{C_{\text{in}}-1} w(C_{\text{out}}, k) \star \text{in}(N_{\text{bat}}, k), \tag{6}$$

where N_{bat} is the batch size, C_{in} is the number of input channels, and L is the sequential length of the input audio signal.

The CNN layer involves bias values b and weight values w, adjusted during training using backpropagation. These weights and biases are randomly initialized with a Gaussian distribution. The model takes multiple sequential time-based inputs, each having 32 channels, represented as $l_{\text{in}} \times w_{\text{in}}$.

The output, denoted by l_{out}, maintains 32 channels but has half the time-based outputs. The selected filter size is $f = 10$, with a stride length of $l_{\text{stride}} = 2$. A zero padding value $p = 5$ is applied to add 5 zeros on each side. The output size is determined by:

$$l_{\text{out}} = \left(\frac{l_{\text{in}} + 2 \times 5 - 10 - 2}{2} + 1 \right) = \left(\frac{l_{\text{in}}}{2} \right). \tag{7}$$

4.1.3. Model additions

Normalization, GELU Activation, and Dropout Layers: A normalization layer is applied to the CNN's batch output data, producing a normalized output y based on the mean $E[x]$, variance $\text{Var}[x]$, and learnable parameters γ and β. The normalization equation is given by:

$$y = \frac{x - E[x]}{\sqrt{\text{Var}[x] + \epsilon}} \cdot \gamma + \beta, \tag{8}$$

where ϵ is a small value (1×10^{-5}) added to the variance to avoid division by zero. Following normalization, a Gaussian Error Linear Unit (GELU) activation function is applied:

$$x = x \cdot \Phi(x), \tag{9}$$

where $\Phi(x)$ is the cumulative distribution function for a Gaussian distribution. The final layer is a dropout layer, randomly masking or zeroing selected channel features during training. A probability p determines the chance of zeroing each feature:

$$z_i \sim \text{Bernoulli}(p), \quad z_i \in z,$$

where z_i follows a Bernoulli distribution, and z has the same size as the input data. During evaluation, the probability is set to 0, and the module computes an identity matrix during prediction. This sequence of normalization, GELU activation, and dropout layers ensures data transformation, non-linearity, and regularization in the CNN architecture.

4.2. Dense architecture

A dense architecture subsystem is appended to the output of the CNN subsystem, serving to normalize and reshape data for the transformer architecture. This subsystem includes linear layers, normalization layers, activation functions, and dropout layers as illustrated in 3.

4.2.1. Model implementation

The first layer, a linear layer or dense layer, transforms the output from the CNN architecture for compatibility with the main transformer. This layer employs matrix multiplication with a weight matrix and bias parameters, converting the 32 input features from the CNN dropout layer into 128 features for each unit in time:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}, \tag{10}$$

where $\mathbf{y}$ represents output features, $\mathbf{x}$ represents input features, $\mathbf{W}$ is the weight matrix, and $\mathbf{b}$ is the bias vector. Following the linear layer, a normalization layer (similar to that used in the CNN architecture) is applied, ensuring discrimination and stabilizing training. A Gaussian Error Linear Unit (GELU) activation function as defined in Eq. (9), is applied to introduce non-linearity to the data. A dropout layer is included to randomly mask elements during training, preventing overfitting. A dropout probability of 0.1 is used for the Bernoulli distribution.

The above four layers (linear, normalization, activation, dropout) are repeated to maintain model stability during training. This repetition ensures gradual parameter adjustments and enhances prediction accuracy, particularly for voice data with varying accents and input features. The second linear layer has the same number of input and output features (128). The repeated normalization layer, GELU activation function layer, and dropout layer mirror the first set of dense layers, with the weight and bias parameter values adjusting during training. This dense architecture establishes a linear relationship between the CNN output features and the transformer input features, facilitating data feature normalization before reaching the transformer architecture.

4.3. Transformer architecture

The transformer architecture, a modern improvement on RNNs, incorporates attention mechanisms for enhanced performance. Unlike RNNs, transformers leverage parallel processing, eliminating sequential constraints (Luong et al., 2015). Comprising encoder and decoder layers, transformers use self-attention units and positional encodings.

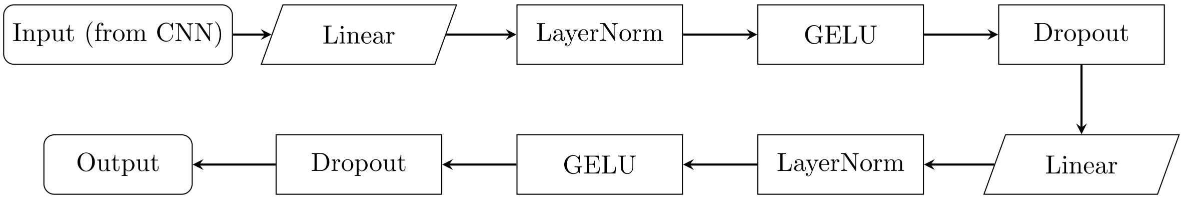

Fig. 3. A diagram describing the dense architecture. The dense architecture consists of two sets of linear layers as the main layers. The architecture also has additional normalizations, GELU and dropout layers.

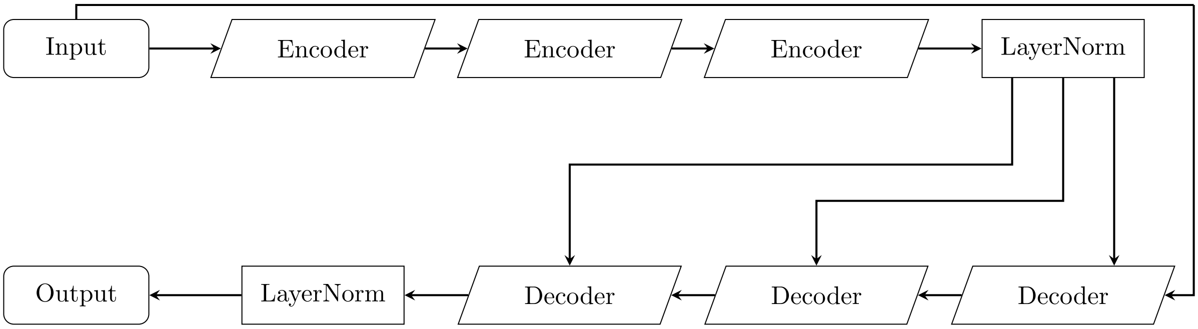

Fig. 4. The flow diagram describes the 2-headed transformer architecture. The diagram is a representation of one of the heads. Each head contains three encoder layers and three decoder layers with normalization layers after the encoder layers and decoder layers.

The encoder iteratively processes input data, producing output values for subsequent layers. Similarly, decoder layers include self-attention units and an additional mechanism drawing information from encoder outputs. Each layer incorporates a feedforward neural network (FFNN) for residual connections and layer normalization. The last decoder layer employs a linear transformation and softmax layer for output probabilities (Vaswani et al., 2017).

Transformer networks use scaled dot product attention units in the encoders and decoders. The attention weights are calculated between every token simultaneously to produce embeddings for every token in context. This context contains information about the token and a weighted combination from other relevant tokens, weighted by attenuation weights. Each attention unit consists of a query weight W_q, a key weight W_k and a value weight W_v matrix. Each token i consists of the input word embedding x_i, multiplied with each of the three weight matrices. This produces a query vector $q_i = x_i W_q$, a key vector $k_i = x_i W_k$ and a value vector $v_i = x_i W_v$. The attention weights are calculated from the query and key vectors where a_{ij} is the dot product of q_i and k_j from i to j. The attention weights are also divided by $\sqrt{d_k}$, where d_k is the dimension of the key vector, to stabilize the gradient during training. Lastly the attention weights are passed through a softmax to normalize the data to a sum of 1. The output of an attention unit for a token i is the weighted sum of the value vectors for all tokens weighted by a_{ij}. The attention calculation for all tokens is therefore a large matrix calculation for matrices Q, K and V representing all the tokens q_i, k_i and v_i, such that $Q = \{q_0, q_1, \dots, q_n\}$, where n represents the total number of tokens. This produces the attention equation from Vaswani et al. (2017),

$$Att(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{11}$$

One set of attention units (W_q, W_k and W_v) is known as an attention head. Each layer in a transformer model consists of multiple attention heads. One attention head attends to tokens that are relevant to each individual token. Multiple attention heads help the model to create different definitions of relevance between tokens. Multi-head attentions are processed in parallel and concatenated to be passed through a final FFNN layer for further processing.

4.3.1. Model implementation

The main transformer architecture for this model includes two heads, each composed of three encoder layers and three decoder layers. The encoder processes a $(t \times b \times x)$ dimensional input, where t is the number of time-based inputs, b is the batch size, and x represents the input features from the dense layer. The positional encodings are utilized in the attention function to generate an output of dimension $(t \times b \times x)$. The encoder output is sequentially passed through subsequent encoder layers to form the encoder system. The decoder, receiving inputs from both the dense layer and the encoder system, produces outputs of the same dimension. The decoder uses masking to prevent reverse flow information and repeats this process for the specified number of decoder layers, resulting in the decoder system. An example of a three-encoder, three-decoder layer transformer is depicted in Fig. 4. The encoder layer comprises an attention layer, normalization layers, and linear layers. The decoder layer involves attention layers for both inputs, normalization layers, and additional linear layers. Following the transformer architecture, the model incorporates a final normalization layer, dropout layer, and linear layer. The normalization layer, as defined in Eq. (8), produces a normalized output by applying learnable parameters γ and β to the batch output data. The dropout layer randomly masks or zeros selected elements during training using a Bernoulli distribution, with a chosen probability p of 0.1 for zeroing items.

A final linear layer is introduced to reshape the data dimensions from $(t \times b \times 128)$ to $(t \times b \times 30)$, where t represents the time-based dimension size, and b represents the batch size. The resulting 30 features correspond to the possible character classes of the model. Subsequently, a softmax function is applied in the final layer to enable a cost function, facilitating the comparison of predicted output classes to actual target classes during training.

4.4. Language models

Language models play a crucial role in augmenting end-to-end character-based Automatic Speech Recognition (ASR) systems by enhancing word understanding from predicted characters. The n-gram language model is a statistical approach that estimates the probability of word sequences based on the occurrence of previous words. Specifically, for a sequence of words $W = (w_1, w_2, \ldots, w_n)$, the probability $P(W)$ is calculated as:

$$P(W) = \prod_{i=1}^{n} P\left(w_k | \phi(W_{k-1})\right), \tag{12}$$

where $\phi(W_{k-1})$ represents the context of the preceding words. To measure the performance of the n-gram language model, quality measures such as perplexity and Word Error Rate (WER) are employed. Perplexity $(PP(W))$ is defined as:

$$PP(W) = -\frac{1}{m} \sum_{k=1}^{m} \log_n \left[P(W_i | W_{i-1}) \right], \tag{13}$$

while WER provides a more practical evaluation for the ASR model. In the context of ASR, the n-gram language model is applied to correct and refine the output sequences generated by the ASR model. By utilizing a 4-gram model, the language model leverages relationships between words in a dataset to enhance the accuracy of predictions.

4.4.1. Rationale for using N-gram model

The decision to employ an n-gram language model in tandem with the ASR model stems from its practicality and effectiveness for a small-scale ASR system. While larger transformer-based models like BERT offer extensive contextual understanding, the n-gram model proves more feasible, striking a balance between size, efficiency, and performance. This choice ensures effective speech correction without overwhelming computational requirements, aligning with the specific needs of the ASR system under consideration. In summary, the n-gram language model, with its simplicity and adaptability, serves as a valuable component in refining ASR predictions, contributing to the overall efficacy of the end-to-end system.

5. Experiment

5.1. Environment and tools

The proposed end-to-end character-based ASR model is implemented and trained using the following environment and tools:

- **Hardware:** The experiments were conducted on a machine equipped with an AMD RYZEN 5 3600 CPU, 16 GB DDR4 RAM and a Nvidia GeForce RTX3060 GPU with 12Gb of VRAM.
- **Software:** The model is implemented using the PyTorch deep learning framework in Python, and the training process is facilitated by the CUDA parallel computing platform.
- **Dataset:** Two primary datasets are employed for training and evaluation. The Mozilla Common Voice English 7.0 dataset (Mozilla, 2021), consisting of 2000 h of English spoken voice data, and the LibriSpeech English dataset (Panayotov et al., 2015), comprising approximately 1000 h of English read speech from audiobooks.

5.2. Pre-processing comparison

The first model was trained using only the common voice training dataset. The validation dataset was used for parameter tuning and final testing for different sampling frequencies and preprocessing of the audio data. In Table 1, the different CERs and WERs are displayed for different preprocessing techniques and different model output methods. The testing data used was a small segment of the common voice test dataset consisting of 500 audio segments that was not used for either training or validation to avoid data contamination. Each original model was tested using three different output methods. The greedy decoding method utilized the highest probability output character for each input audio segment and combined the characters using the CTC compression to produce output characters. The beam search method used the most likely sequence of characters based on the top 500 output sequences. The final output method was a beam search with an added 4-gram language model for word recognition. The 4-gram language model was selected as it contains a large dictionary of connected words. The model uses a statistical probability of words occurring based on previous and following words. Statistical language modelling estimates the prior probability $P(W)$ for a sequence or string of words W in a total vocabulary V. The vocabulary V can be thousands to hundreds of thousands of words depending on the selected language model. The sequence of words W is usually broken into small sequences or sentences that are conditionally independent so that Eq. (12) can be implemented.

The surrounding words $W_{k-1} = \left(w_1, w_2, \ldots, w_{i-1}\right)$ are used in a function $\phi(W_{k-1})$ to determine the predicted word w_k. Therefore, Eq. (12) can be rewritten as: $P(W) \simeq \prod_{i=1}^{n} P\left(w_k | \phi(W_{k-1})\right)$. The 4-gram language model uses $\phi(W_{k-1}) = w_{k-4+1}, w_{k-4+2}, w_{k-4+3}$, so that the 3 words before the selected word is taken as $\phi(W_{k-1})$. Larger transformer based language models could be used, but they consist of a model with billions of parameters and would defeat the purpose of building a small scale ASR model. In Table 1, the beam search method with an added 4-gram language model produces the best results for all data pre-processing types.

The first model was created by augmenting the audio features during training using the "SpecAugment" method. This model was trained with a sampling rate of 8 kHz and 16 kHz respectively. The results show that the audio data features sampled at 16 kHz produced a CER of 36.59% and a WER of 67.4% compared to a CER of 41.82% and a WER of 74.27% using the 8 kHz sampled audio features. Removing the data augmentation improved the CER and WER slightly. This is most likely due to the audio features being MFCC values and their deltas and not a full spectrogram. Therefore, the "SpecAugment" method removes too much necessary data from the audio features. Time stretching was introduced instead of the "SpecAugment" method. The time stretching method randomly stretched or shrank the length of audio segments by 10% while keeping all the audio data constant. The time stretching improved the CER to 29.98% and the WER to 59.83% as seen in Table 1.

5.3. Language model comparison on initial data

To enhance the understanding of words from predicted characters, two types of language models are incorporated: the n-gram language model and the BERT transformer-based language model. The n-gram language model is chosen for its practicality in small-scale ASR systems. By estimating the probability of word occurrences based on previous and following words, the n-gram model provides a balance between model size and correlation between words. The implementation involves creating a 4-gram language model using the kenLM toolkit. Different 4-gram models are tested, with the selected model based on the LibriSpeech dataset. Beam search and weighted probabilities ensure accurate predictions, and the model output is converted from tokens to characters.

In contrast to the limited contextual understanding of n-gram models, the BERT (Bidirectional Encoder Representations from Transformers) language model utilizes the entire sequence to predict or correct words. The BERT model is fine-tuned for sentence correction, aligning with the goals of speech recognition. The BERT model is implemented with output data from the beam search method, enhancing predictions based on contextual information. By iteratively masking words and predicting the 50 most probable words for each masked word, the BERT model contributes to refining the ASR model's output.

The CERs and WERs in Table 2 are from a small testing sample selected to compare the different language models and select the best performing language model. The CERs and WERs are not reflective of the final model as it was trained on a very small sample for testing purposes only. The BERT language model in Table 2 produces worse

Table 1
Character error rates (CERs) and word error rates (WERs) for different sampling frequencies and preprocessing methods of the Common Voice 7.0 validation data.

Pre-processing performance	Greedy decoding		Beam search 500		Beam search 500 and 4-gram lm	
Performance metrics	CER %	WER %	CER %	WER %	CER %	WER %
Normal MFCC 8 kHz	43.00	90.43	41.99	91.04	41.82	74.27
MFCC with Spec augment 16 kHz	41.31	86.79	39.62	86.91	36.59	67.40
Normal MFCC 16 kHz	40.63	86.66	38.45	85.50	35.15	65.22
MFCC with time stretching 16 kHz	35.84	82.01	34.17	80.08	29.98	59.83

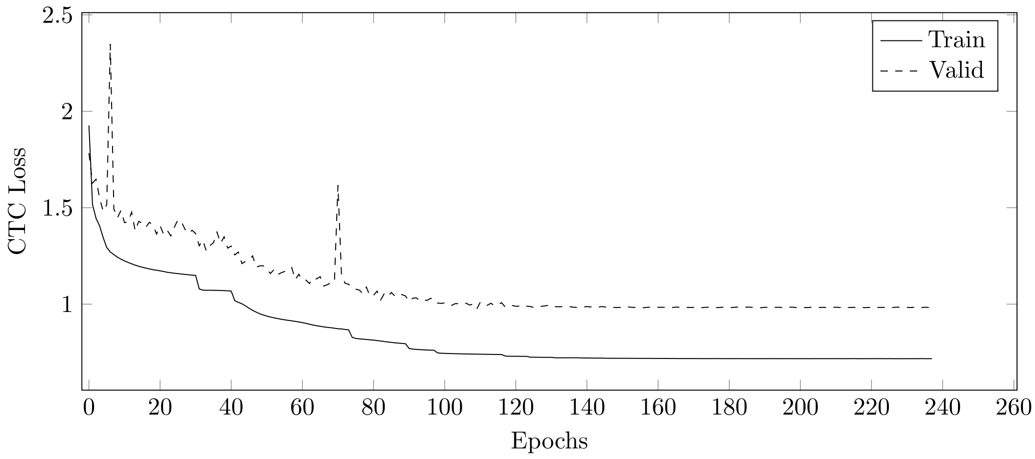

Fig. 5. Average Training and Validation CTC loss per Epoch for the transformer model, using the common voice 7.0 English training and validation dataset. The graph indicates that the average CTC loss value decreased from 1.9268 to 0.7174 for the training dataset and decreased from 1.7829 to 0.9818 for the validation dataset. The audio data was sampled at 16 kHz and converted to 16 MFCC values and 16 delta values. Time stretching of 10% was applied to the training data to simulate more data. 30 possible character classes were selected as possible outputs of the model. The validation data was used to optimize the model by changing the learning rate of the model based on the output of the average validation CTC loss per epoch.

Table 2
CERs and WERs for different output methods of the Common Voice 7.0 validation data.

Output method	Performance metrics	
Performance metrics	CER %	WER %
Greedy decoding	43.00	90.43
Beam search 500	41.99	91.04
4-gram lm (LibriSpeech)	41.91	86.43
4-gram lm (CV + LibriSpeech)	41.82	74.27
BERT	42.84	89.56
Textblob spell checker	42.92	87.64

results than the smaller n-gram language model, and therefore it was decided that the 4-gram language model would be used with the ASR model. The 4-gram language model is also smaller than the BERT language model and therefore reduces the system complexity with respect to the number of parameters.

5.4. Model training

The ASR model is trained using the training subset of the common voice English dataset and the three training subsets of the LibriSpeech dataset. The CTC algorithm is employed to determine the maximum likelihood label for each input value. A blank token is introduced to the character classes to ensure proper handling of repeating characters.

The model is trained to minimize the CTC loss, which is calculated as the mean difference between the predicted and actual labels. The label error rate (LER) is defined as the normalized edit distance between the output classifications of the model and the target classifications. In CTC loss, the label error rate (LER) can be defined as the normalized edit distance between the output classifications of a classifier model h and the classification targets such that:

$$LER(h, S') = \frac{1}{Y} \sum_{(x,y)\in S'} ED\left(h(x)\right), \tag{14}$$

where S' is a dataset with dimensions $(x \times y)$ and Y is the total number of target labels. The edit distance, $ED\,(p,q)$ is the edit distance between two sequences p and q which would be the predicted and target sequences. The edit distance is the minimum number of insertions, deletions or substitutions required to change sequence p into sequence q (Graves et al., 2006).

The audio data is processed in batch sizes of 64, allowing for efficient training with the available hardware. The model is initialized with random weights and biases, and a standard learning rate of 1e−3 is chosen. A dropout value of 0.1 is applied to the dropout layers during training.

The evaluation subset of the common voice English dataset and LibriSpeech dataset is used to evaluate the model after every epoch and adjust the learning rate if necessary, using the “AdamW” optimizer. If the model parameters become trapped in a local minimum of the CTC loss function, within a few iterations, the optimizer increases the learning rate to change the weight and bias training values more rapidly and to improve training efficiency.

In Fig. 5, the average CTC loss of the common voice training data is provided per epoch. The model was trained using the MFCC and delta audio features, sampled at 16 kHz, and time stretching random audio segments during training. The results indicate that the ASR model trained to minimize the CTC loss value until a minimum value was reached. Fig. 5 also provides the average CTC loss for the common voice validation data per epoch. The validation data is used to optimize the training of the model. When the validation CTC loss value did not drop for 5 epochs, the “AdamW” optimizer changed the learning rate of

Table 3
Comparison of the small scale transformer from this paper, to other large transformer based ASR models using the Common Voice validation dataset and testing dataset.

CommonVoice testing			Validation data		Testing data	
	Parameters	Hours Trained	CER %	WER %	CER %	WER %
End-to-end small scale transformer	2 214 141	2000 CV	22.54	48.17	27.89	54.50
Wav2Vec2 XLS-R 300 (Babu et al., 2021)	300 million	436000 Pre-Train + 2000 CV	7.36	30.71	N/A	N/A
Wav2Vec2 XLSR53 (Conneau et al., 2020; Grosman, 2021)	317 million	56000 Pre-Train + 2000 CV	7.69	19.06	11.65	27.72
Wav2Vec2 XLSR53 With LM (Conneau et al., 2020; Grosman, 2021)	317 million	56000 Pre-Train + 2000 CV	6.84	14.81	11.01	20.85

the model to allow a larger jump in weight values. In Fig. 5, there are sudden drops in CTC loss at selected epochs, this is due to the optimizer changing the learning rate of the model.

6. Results and discussion

6.1. Common voice results

Table 3 compares the final trained small scale transformer model results to existing large transformer based models on the common voice validation dataset and testing dataset. In the table, it is noticeable that the small scale transformer ASR model does not perform as well as the larger Wav2Vec2 models from Babu et al. (2021) and Conneau et al. (2020). The ASR models from previous work consist of architectures that have over 300 million parameters, while the model in this paper only consists of 2.21 million parameters. That is a parameter reduction by a factor of about 150. The other models are also pre-trained with over 50 000 h of speech data, before being fine-tuned with the Common Voice dataset. The small scale transformer is only trained on the 2000 h of Common Voice data.

The end-to-end small scale transformer, with an added 4-gram language model produced a CER of 24.87% and a WER of 52.51% on the validation dataset and a CER of 29.98% and a WER of 58.57% on the testing dataset. The random chance of a single character being guessed correctly by the model is 3.33% and a sequence of characters reduces that probability exponentially. Therefore, a CER of 29.98% is a positive result for a small scaled speech recognition model, but is still far off from the large speech recognition model CER results in Table 3. There were multiple reasons for the poor performance and it was decided to add the LibriSpeech data as an additional 1000 h of training data to test the models performance.

6.2. LibriSpeech results

Table 4 compares the results of the small scale transformer model trained on only 2000 h of Common Voice data and the same model trained on an additional 1000 h of LibriSpeech data. The models are tested on the LibriSpeech dev and test datasets. The fine-tuned model, with the additional training data from LibriSpeech, outperforms the original Common Voice trained model drastically. The test-clean dataset CER is reduced from 13.89% to 6.40% and the WER is reduced from 30.93% to 16.03%. The overall results for the model on the LibriSpeech data significantly improved compared to the Common Voice data, as the LibriSpeech data consists of more clear speech and only in an American accent. The model also performs better with the added LibriSpeech training data due to an additional 1000 h of audio data used to train the model. The LibriSpeech training data is also more similar to the LibriSpeech dev and test data.

Table 5 compares the final small scale transformer model, trained on common voice and LibriSpeech training data, to existing large transformer based models using the LibriSpeech dev dataset and test dataset. In the table, it is noticeable that the small scale transformer ASR model does not perform as well as the larger Wav2Vec2 and large transformer models, but is comparable to older GRU based models.

The transformer based ASR models from previous work consist of architectures that have over 300 million parameters, while the model in this paper only consists of 2.21 million parameters. That is a parameter reduction by a factor of over 150. The other models are also pre-trained with over 53000 h of speech data, before being fine-tuned with the LibriSpeech dataset. The small scale transformer is only trained on the 3000 h of common voice and LibriSpeech data.

6.3. Discussion

The end-to-end small scale transformer, with an added 4-gram language model produced a WER of 16.03% and 35.51% on the test-clean and test-other data respectively on the LibriSpeech dataset. The model also produced a WER of 15.45% and 33.47% on the dev-clean and dev-other data respectively. The small scaled end-to-end ASR model with an added 4-gram language model produced the given CER and WER using a model that was trained and tested using less than 12Gb of video RAM, which was the limitation for this model. The other large transformer models require large clusters of graphics cards to train models that require over 120Gb of video RAM. The result from an older GRU based model in Drexler and Glass (2019), produced a WER of 31.1% for the test other dataset, which is relatively close to the WER achieved by the small scale transformer, but at a significantly larger computational cost, as the architecture had 52.5 million parameters, which is a factor of 24 times larger than the architecture in this paper.

The small scale transformer trains a lot faster computationally, as it optimizes parallel processing with the use of transformer architecture layers. The larger modern transformer based ASR models outperformed with WERs of under 5% for the test clean dataset and WERs of under 10% for the test other datasets. This is due to the models being trained with about 50000 h more speech data. The models also have over 300 million parameters with multiple deep layers to create different mathematical paths, with different weights, between the input data and the output classes.

The final end-to-end transformer based ASR model designed in the research consisted of a CNN layer and a 2-headed transformer with 3 encoder layers and 3 decoder layers. The data used to train the model was the 2000 h of Mozilla common voice 7.0 training data and the 1000 h of LibriSpeech training data. The audio data was transformed to 32 MFCC and delta features as the input to the model. The output of the model was 30 character based classes. The output of the model was processed by a small 4-gram language model to improve word understanding as it is a character based ASR model. The model produced a CER and WER of 27.89% and 54.5% respectively for the very noisy common voice testing dataset. The model also produced a CER and WER of 6.40% and 16.03% respectively for the

Table 4
LibriSpeech Dev and Test results based on a small scale transformer based ASR model, with 3 encoder layers and 3 decoder layers, trained on the Common Voice dataset and the same model being fine-tuned with the LibriSpeech training dataset.

LibriSpeech testing	Dev clean		Dev other		Test clean		Test other	
End-to-end small scale transformer (2 head: 3 enc, 3 dec)	CER %	WER %	CER %	WER %	CER %	WER %	CER %	WER %
Trained on Common Voice	13.57	30.61	26.29	51.39	13.89	30.93	27.49	53.58
Trained on LibriSpeech	12.08	28.91	25.54	51.85	12.55	30.14	26.56	53.75
Trained on Common Voice and LibriSpeech	**6.04**	15.45	15.90	33.47	**6.40**	16.03	16.73	35.51

Table 5
LibriSpeech dataset WER results for the small scale transformer models compared to very large ASR models using mainly transformer based architectures. The small scaled transformer model has 2.21 million parameters, with an added 4-gram language model. All the other large scale transformer models contain over 52.5 million parameters without their added language models.

LibriSpeech testing	Parameters	Hours trained	Dev clean WER %	Dev other WER %	Test clean WER %	Test other WER %
End-to-end small scale transformer (2 head: 3 enc, 3 dec)	2 214 141 + lang model	1000 LibriSpeech	28.91	51.85	30.14	53.75
End-to-end small scale transformer (2 head: 3 enc, 3 dec)	2 214 141 + lang model	2000 Pre-Train + 1000 LibriSpeech	**15.45**	**33.47**	**16.03**	**35.51**
CTC based GRU model (Drexler & Glass, 2019)	52.5 million	Pre-Train + 1000 LibriSpeech	N/A	N/A	11.9	31.1
CTC based GRU model with 4gram LM (Drexler & Glass, 2019)	52.5 million + lang mod	Pre-Train + 1000 LibriSpeech	N/A	N/A	8.3	24.4
CTC based transformer (Synnaeve et al., 2019)	322 million	53800 Pre-Train + 1000 LibriSpeech	2.99	7.31	3.09	7.40
CTC based transformer with transformer LM (Synnaeve et al., 2019)	322 million + lang mod	53800 Pre-Train + 1000 LibriSpeech	2.63	6.20	2.86	6.72
Conformer model (Gulati et al., 2020)	118.8 million + lang mod	800M words + 1000 LibriSpeech	N/A	N/A	1.9	3.9
Wav2Vec2 transformer (Baevski et al., 2021)	300 million + lang mod	53200 Pre-Train + 1000 LibriSpeech	3.4	6.0	3.8	6.5
Wav2Vec2.0 transformer (Baevski et al., 2020)	317 million + transformer lang mod	53200 Pre-Train + 1000 LibriSpeech	1.6	3.0	1.8	3.3
Crowd-sourced human level performance (Amodei et al., 2015)	N/A	N/A	N/A	N/A	5.83	12.69

LibriSpeech test clean dataset. The exceptional CERs of 6.04% and 6.4% achieved on the Dev and Test Clean datasets, respectively, underscore the robustness of the speech recognition component within our model. These impressively low CER values indicate the effectiveness of the speech recognition mechanism. Notably, leveraging a larger language model could substantially enhance the Word Error Rate (WER) given the strong foundation of accurate character recognition.

Furthermore, the advent of sophisticated, modern large language models, such as GPT-4, presents an opportunity to significantly boost the accuracy of this small-scale character-based speech recognition model. The integration of such advanced language models holds the potential to markedly reduce WER, capitalizing on the already exemplary performance of the speech recognition aspect. This model represents a viable approach for under-resourced languages and scenarios with limited computational power. Our research provides compelling evidence that small-scale speech recognition models can effectively establish speech-to-text systems for under resourced languages with limited access to large corpora, without the need for extensive computational resources or expensive cloud environments. Our approach is designed to be open-source, providing an accessible and straightforward method that does not rely on proprietary models unavailable as open-source resources. By showcasing the feasibility of this approach, we aim to encourage the development of speech recognition systems for a wider array of languages and contexts, thereby contributing to democratizing access to speech recognition technology.

7. Conclusion

A character based small scale transformer architecture introduced an end-to-end automated speech recognition model showcasing performance comparable to large scale character based ASR models. The reduction in performance is not nearly commensurate with the reduction in complexity that has been achieved when compared to the results of large modern character based ASR models. The developed model in the paper has over 150 times fewer parameters and was trained with up to 53000 h less voice data than the larger modern ASR systems. The model produced, was trained on the common voice and LibriSpeech training datasets, and produced a CER and WER of 27.89% and 54.5% respectively on the common voice testing data as well as a CER and WER of 6.40% and 16.03% respectively on the LibriSpeech test-clean dataset. The best current large ASR model produced a WER of 1.8% on the LibriSpeech test-clean dataset, but with a 317 million parameter model and a transformer based language model consisting of over 400 million parameters. The model in this paper has 2.214 million parameters and a small 4-gram language model. The model was comparable to an older GRU based model that produced a WER of 11.9% and 31.1% respectively on the same LibriSpeech dataset, but at a significantly larger computational cost, as the architecture had 52.5 million parameters, which is a factor of 24 times larger than the architecture in this paper.

The performance of the small-scale transformer architecture is noteworthy, particularly evident in the CER results for the LibriSpeech dataset. Notably, future work should entail finding the break-even point where only adding a modest number of parameters may result in performance comparable with the larger models considered in the paper. The observed low CER values for the LibriSpeech dataset, such as the 6.40% on test-clean, underscore the remarkable efficacy of the speech recognition component within the model. This points to the possibility of achieving significantly improved WER figures with the incorporation of a more robust language model. A small 4-gram language model decreased the WER by over 20% in the common voice English 7.0 testing data. Adding more training data should increase the performance of the small scale model significantly as well. Notably, all character-based end-to-end ASR models require a small language model to translate predicted characters into coherent words. Our study primarily aimed to forge an ASR model viable for under-resourced languages devoid of expansive datasets and computing resources. Additionally, it is designed to function as a portable standalone ASR device, ideal for scenarios lacking internet connectivity, where larger recognition tools are inaccessible. This work signifies a step towards democratizing speech recognition technology for resource-constrained languages and environments. It underscores the potential for compact, accessible ASR solutions, eliminating reliance on proprietary models or extensive computational infrastructures.

Code

The Python code for the end-to-end small scale ASR model is available on github at: SmallScale-SpeechRecog.git.

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

Data will be made available on request.

Acknowledgements

We thank the MultiChoice Chair of Machine Learning for sponsoring the funding for the research of this paper. The University of Pretoria EECE department provided the facilities and resources to complete the paper. A thanks also to Mozilla for the publicly available Common Voice dataset and to OpenSLR and Vassil Panayotov for the publicly available LibriSpeech dataset.

References

Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., et al. (2015). Deep speech 2: End-to-end speech recognition in english and mandarin. *vol. 1*, In *33rd international conference on machine learning, ICML 2016* (December), (pp. 312–321). URL http://arxiv.org/abs/1512.02595, arXiv:1512.02595.

Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., et al. (2021). XLS-R: Self-supervised cross-lingual speech representation learning at scale. (pp. 1–23). URL http://arxiv.org/abs/2111.09296, arXiv preprint arXiv:2111.09296.

Baevski, A., Hsu, W.-N., Conneau, A., & Auli, M. (2021). Unsupervised speech recognition. *Advances in Neural Information Processing Systems*, *34*, 27826–27839. http://dx.doi.org/10.48550/arXiv.2105.11084, https://github.com/pytorch/fairseq/tree/, http://arxiv.org/abs/2105.11084, arXiv:2105.11084.

Baevski, A., Zhou, H., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, *33*, 12449—-12460, arXiv:2006.11477, URL http://arxiv.org/abs/2006.11477.

Chen, X., Liu, X., Wang, Y., Gales, M. J., & Woodland, P. C. (2016). Efficient training and evaluation of recurrent neural network language models for automatic speech recognition. *IEEE/ACM Transactions on Audio Speech and Language Processing*, *24*(11), 2146–2157. http://dx.doi.org/10.1109/TASLP.2016.2598304.

Conneau, A., Baevski, A., Collobert, R., Mohamed, A., & Auli, M. (2020). Unsupervised cross-lingual representation learning for speech recognition. arXiv preprint arXiv:2006.13979, URL http://arxiv.org/abs/2006.13979.

Drexler, J., & Glass, J. (2019). Subword regularization and beam search decoding for end-to-end automatic speech recognition. In *ICASSP 2019 - 2019 IEEE international conference on acoustics, speech and signal processing* (pp. 6266–6270). IEEE, http://dx.doi.org/10.1109/ICASSP.2019.8683531, URL https://ieeexplore.ieee.org/document/8683531/.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification. *vol. 148*, In *Proceedings of the 23rd international conference on machine learning - ICML '06* (pp. 369–376). New York, New York, USA: ACM Press, http://dx.doi.org/10.1145/1143844.1143891, URL http://portal.acm.org/citation.cfm?doid=1143844.1143891.

Grosman, J. (2021). XLSR Wav2Vec2 English by Jonatas Grosman. In *Hugging face unpublished results*. Unpublished, URL https://huggingface.co/jonatasgrosman/wav2vec2-large-xlsr-53-english.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., et al. (2020). Conformer: Convolution-augmented transformer for speech recognition. In *INTERSpEECH, ISCA* (pp. 5036–5040). http://dx.doi.org/10.48550/arXiv.2005.08100.

Hwang, K., & Sung, W. (2016). Character-level incremental speech recognition with recurrent neural networks. *2016-May*, In *2016 IEEE international conference on acoustics, speech and signal processing* (pp. 5335–5339). IEEE, http://dx.doi.org/10.1109/ICASSP.2016.7472696, arXiv:1601.06581m URL http://ieeexplore.ieee.org/document/7472696/.

Kubo, Y., Karita, S., & Bacchiani, M. (2022). Knowledge transfer from large-scale pretrained language models to end-to-end speech recognizers. In *ICASSP 2022 - 2022 IEEE international conference on acoustics, speech and signal processing* (pp. 8512–8516). http://dx.doi.org/10.1109/ICASSP43922.2022.9746801.

Li, S., Akita, Y., & Kawahara, T. (2016). Semi-supervised acoustic model training by discriminative data selection from multiple ASR systems' hypotheses. *IEEE/ACM Transactions on Audio Speech and Language Processing*, *24*(9), 1520–1530. http://dx.doi.org/10.1109/TASLP.2016.2562505.

Li, B., Zhou, E., Huang, B., Duan, J., Wang, Y., Xu, N., et al. (2014). Large scale recurrent neural network on GPU. In *Proceedings of the international joint conference on neural networks* (pp. 4062–4069). IEEE, http://dx.doi.org/10.1109/IJCNN.2014.6889433.

Liao, H., McDermott, E., & Senior, A. (2013). Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription. In *2013 IEEE workshop on automatic speech recognition and understanding, ASRU 2013 - proceedings* (pp. 368–373). IEEE, http://dx.doi.org/10.1109/ASRU.2013.6707758.

Lippi, M., Montemurro, M. A., Degli Esposti, M., & Cristadoro, G. (2019). Natural language statistical features of LSTM-generated texts. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3326–3337. http://dx.doi.org/10.1109/TNNLS.2019.2890970, arXiv:1804.04087.

Long, Y., Li, Y., Wei, S., Zhang, Q., & Yang, C. (2019). Large-scale semi-supervised training in deep learning acoustic model for ASR. *IEEE Access*, *7*, 133615–133627. http://dx.doi.org/10.1109/ACCESS.2019.2940961.

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Conference proceedings - EMNLP 2015: conference on empirical methods in natural language processing* (pp. 1412–1421). http://dx.doi.org/10.18653/v1/d15-1166, arXiv:1508.04025, https://re-work.co/blog/deep-learning-ilya-sutskever-google-openai-, http://arxiv.org/abs/1508.04025.

Maas, A. L., Xie, Z., Jurafsky, D., & Ng, A. Y. (2015). Lexicon-free conversational speech recognition with neural networks. In *NAACL HLT 2015 - 2015 conference of the North American chapter of the association for computational linguistics: human language technologies, proceedings of the conference* (pp. 345–354). Stroudsburg, PA, USA: Association for Computational Linguistics, http://dx.doi.org/10.3115/v1/n15-1038, URL http://aclweb.org/anthology/N15-1038.

Min, Z., & Wang, J. (2023). Exploring the integration of large language models into automatic speech recognition systems: An empirical study. arXiv preprint arXiv:2307.06530.

Mozilla (2021). *Common voice corpus 7.0*, https://commonvoice.mozilla.org/en/datasets.

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). Librispeech: An ASR corpus based on public domain audio books. *2015-Augus*, In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (pp. 5206–5210). http://dx.doi.org/10.1109/ICASSP.2015.7178964.

Pham, N.-Q., Nguyen, T.-S., Niehues, J., Müller, M., Stüker, S., & Waibel, A. (2019). Very deep self-attention networks for end-to-end speech recognition. arXiv preprint arXiv:1904.13377, URL http://arxiv.org/abs/1904.13377.

Sundermeyer, M., Ney, H., & Schluter, R. (2015). From feedforward to recurrent LSTM neural networks for language modeling. *IEEE Transactions on Audio, Speech and Language Processing*, *23*(3), 517–529. http://dx.doi.org/10.1109/TASLP.2015.2400218.

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, URL http://arxiv.org/abs/1911.08460.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*, 5999–6009, arXiv:1706.03762.

Vergin, R., O'Shaughnessy, D., & Farhat, A. (1999). Generalized mel frequency cepstral coefficients for large-vocabulary speaker-independent continuous-speech recognition. *IEEE Transactions on Speech and Audio Processing*, *7*(5), 525–532. http://dx.doi.org/10.1109/89.784104.

Williams, W., Prasad, N., Mrva, D., Ash, T., & Robinson, T. (2015). Scaling recurrent neural network language models. *2015-Augus*, In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (pp. 5391–5395). http://dx.doi.org/10.1109/ICASSP.2015.7179001, arXiv:arXiv:1502.00512v1.

Winata, G. I., Cahyawijaya, S., Lin, Z., Liu, Z., & Fung, P. (2020). Lightweight and efficient end-to-end speech recognition using low-rank transformer. In *ICASSP 2020 - 2020 IEEE international conference on acoustics, speech and signal processing* (pp. 6144–6148). IEEE, http://dx.doi.org/10.1109/ICASSP40776.2020.9053878, URL https://ieeexplore.ieee.org/document/9053878/.

Wu, B., Li, K., Ge, F., Huang, Z., Yang, M., Siniscalchi, S. M., et al. (2017). An end-to-end deep learning approach to simultaneous speech dereverberation and acoustic modeling for robust speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, *11*(8), 1289–1300. http://dx.doi.org/10.1109/JSTSP.2017.2756439.