**Original Research Article**

**Fine-Tuning Deepspeech Speech-To-Text Model For Nigerian English And Yoruba-English Code-Switched Speech**

# ABSTRACT

|  |
| --- |
| This study focuses on adapting the deepspeech 0.9.3 speech-to-text (STT) model to enhance transcription accuracy for Nigerian English and Yoruba-English Code-Switched (CS) speech, addressing the model's limitations with non-Western accents and multilingual contexts. A custom dataset was gathered, comprising 235 audio clips of Nigerian English (40.52 minutes) and 445 clips of Yoruba-English Code-Switched (77.48 minutes), totalling 680 clips (118 minutes or ~1.97 hours). The dataset was split into 80% training (544 samples, ~94.4 minutes), 10% validation (68 samples, ~11.8 minutes), and 10% testing (68 samples, ~11.8 minutes). Testing the best model checkpoint yielded a Word Error Rate (WER) of 0.760261, a Character Error Rate (CER) of 0.381241, and a loss of 95.734718. The study highlights the challenges of adapting STT models for low-resource languages and provides a foundation for future research into scalable, multilingual speech recognition systems. |

*Keywords: DeepSpeech, Speech-to-Text technology, Code Switching, Accents*

# INTRODUCTION

Speech-to-Text (STT) technology generally covers the tools and systems that have been designed to convert actual spoken words into textual transcriptions. This technology implements models and algorithms designed for speech processing in several ways, including voice assistants and accessibility equipment and other transcription services (Thomas et al., 2024). STT technology has become an indispensable component of various applications such as virtual assistants and accessibility tools, with a good example of an STT model being DeepSpeech: a recent innovation that has improved accuracy and scalability when implemented in other STT applications (Hannun et al., 2014).

Despite the advancements in the technology, however, STT systems still exhibit poor generalisability to diverse accents (Dubey & Shah, 2022). For instance, the Nigerian English accent is transcribed poorly due to lower accuracy in STT models when encountering this accent variation. This happens mainly because the data used to train these STT models tend to favour more Western accents. The various Nigerian accents and Nigerian Pidgin English that make up Nigerian English are under-represented in most STT models, so the models aren't generalised to them (Amuda et al., 2010; Lin et al., 2023).

This transcription accuracy issue is exacerbated in situations where speakers switch between languages during conversations, known as Code-Switching (CS) (Mustafa et al., 2022). Yoruba, one of the three major languages in Nigeria, is one language commonly found in CS scenarios. It is often interspersed with English during conversations by both first and second language speakers across Nigeria, and accurate speech recognition involving the language gets complicated for STT systems not designed to handle its nuances (Amuda, 1994).

This paper attempts to address the problem via methods like fine-tuning and transfer learning. Fine-tuning is a method where you take a pre-trained model and give it some more training with some smaller datasets that focus on specific areas. It is simply about tweaking the features the model learned in its initial training to match the unique traits of the target language (Church et al., 2021). Fine-tuning has been used extensively in a lot of studies, with an excellent example being the work of Ibejih et al. (2022) that involved adapting and fine-tuning QuartzNet15x5 to Nigerian English. Transfer learning, on the other hand, is a training method that involves taking models that have already been trained on large, high-resource datasets and applying them to specific tasks or domains with new data (usually smaller than the original training data). This effectively transfers the knowledge they had gained in their initial training to this new task (Baller et al., 2021). It is a method that is incredibly useful in the development of STT models, particularly in low-resource linguistic contexts. A good example of a study doing this is the work of Dubey & Shah, (2022), which involved creating a model to better recognise Indian-English accents.

Using these methods, this paper presents the process undergone to adapt and fine-tune the pre-trained deepspeech 0.9.3 STT model, from the checkpoint file made available by the Mozilla developers since 2021, to improve its Nigerian English and Yoruba-English CS recognition. The fine-tuned model, trained on a custom dataset featuring various Nigerian English accents and Yoruba-English CS in .wav audio and .txt transcript files, is assessed using Word Error Rate (WER) and Character Error Rate (CER) metrics. Its performance is compared to that of the baseline deepspeech-0.9.3 model’s performance.

The desire to improve on STT models and create ones tailored to handle native accents and speech nuances such as CS motivated this study. These models could have an enormous impact on the various applications built on the technology. In education, for instance, it could improve the automated transcription of lectures in Nigerian English, which can be seen in the work of Ibejih et al. (2022). It can also help with the transcription of Yoruba language courses being taught to beginners who primarily speak English, helping said students learn visually from the transcriptions and through language intermingling characteristic of Yoruba-English CS speech. Furthermore, it could yield applications such as virtual assistants capable of note-taking, journaling, and even speech-based accessibility tools for people with disabilities.

The paper is organized as follows: Section 2 reviews related work on STT models and CS recognition. Section 3 details the methodology, including data collection, preprocessing, model fine-tuning, and evaluation metrics. Section 4 presents the results and discusses the model’s performance. Section 5 concludes with findings and recommendations for future work.

# LITERATURE REVIEW

Yılmaz et al. (2018) worked on CS detection in Frisian-Dutch speech using acoustic and language models with augmented data. The focus was on improving CS ASR performance for under-resourced Frisian by working with monolingual Dutch data and synthetic CS text generation. They used acoustic models trained on semi-supervised data and (generated) CS text for Language Model (LM) augmentation. Results showed a 27.6% WER with language tags and an 8.6% EER for CS detection, highlighting the effectiveness of data augmentation in low-resource scenarios.

Agarwal and Zesch (2019) used DeepSpeech for German speech recognition. They wanted to increase the number of available speech recognition models for languages other than English, so they worked on training and evaluating a German DeepSpeech model. They preprocessed speech data, fine-tuned DeepSpeech’s architecture, and then compared the model’s performance to that of commercial ASR services. They showed that open-source systems can do as well as commercial ones, making them a good option for languages with fewer resources. They also noted that having more varied data for training is important for the system to work well in noisy environments.

Dubey and Shah (2022) focused on adapting DeepSpeech to create a model that performed better at recognising Indian-English accents, addressing the model's limitation of being largely trained on American-English datasets. They used transfer learning techniques, fine-tuning DeepSpeech using an Indic Text-to-Speech (TTS) dataset. After working on improving the model, they achieved a WER equal to 15.7895%, showing that it was better at recognising Indian-English accents than the original DeepSpeech model and other existing ASR systems. Their work showed that transfer learning and data augmentation are very useful in ASR systems that perform better when dealing with regional accents.

Ibejih et al. (2022) developed EDUSTT, an STT model fine-tuned for recognising Nigerian English accents in educational contexts. They started off with the pre-trained QuartzNet15x5 model and trained the model with a custom dataset of educational speech featuring various Nigerian English accents, and covering subjects like Mathematics and Biology. For EDUSTT, they increased the amount of training data from 3.82 hours to 7.79 hours in order to improve the model's training. This resulted in a reduced WER, from 33% to 27%. Despite this success, they noted that the model faced challenges with subjects not included in the training data and with very strong Nigerian accents. Their work showed the importance of using domain-specific datasets to improve STTmodels' performance, especially where resources are limited.

Hassan et al. (2022) used transfer learning to train the DeepSpeech2 model to better recognise South Asian English accents. The model was adapted using a South Asian accented speech dataset. A WER of 18.08% was achieved by the fine-tuned model, which showed a significant improvement in the model's recognition of non-native English accents in comparison with the base model. The study did not mention potential limitations such as variations in training data or the size of the training data. The study also did not compare the model's performance across different South Asian accents, or its robustness in noisy environments.

Zhou et al. (2024) aimed to improve the performance of STT models in CS speech recognition. Their research explored the challenges posed by language boundary ambiguity during CS, which affects language recognition and transcription accuracy, with a focus on Chinese-English CS. The authors employed a methodology that combined transfer learning with fine-tuning on a synthesised dataset of code-switched speech. This approach allowed them to adapt pre-trained STT models, enhancing their ability to handle linguistic shifts within conversations. Their results showed improvements in transcription accuracy for Mandarin-English CS speech, using the WER metric for English, CER for Chinese and a total MER, proving the efficacy of tailored model adaptation.

# RESEARCH GAP

There exist several state-of-the-art STT models that have performed exceptionally well for speech recognition. These models are implemented in systems, such as Mozilla’s DeepSpeech, that have been developed on deep learning techniques and trained on large speech datasets (Dubey & Shah, 2022). This training mainly focuses on English with Western accents, which makes for poor generalisability when applied to other English accents. Some studies have addressed this limitation. Dubey and Shah (2022), in their study, used transfer learning to adapt DeepSpeech for various Indian-English accents. Ibejih et al. (2022), in their study, explored fine-tuning a different STT model, QuartzNet15x5, for Nigerian English accents, focusing on educational contexts. Both studies showed that transfer learning is effective in improving a model’s generalisability to other English accents. However, no work has been done to create a model that accurately transcribes speech in Nigerian English accents and that recognises and transcribes speech that features CS with the Yoruba language.

As such, this study focuses on developing an STT model that can handle the Nigerian English accent and Yoruba-English CS. It involves the use of transfer learning and fine-tuning in order to successfully adapt DeepSpeech to this speech processing task.

# METHODOLOGY

This section describes the approach used to fine-tune the deepSpeech-0.9.3 model to improve transcription accuracy for Nigerian English and Yoruba-English (CS) speech. The rationale behind selecting the DeepSpeech model, the creation of the custom dataset, environment setup, data preprocessing, hyperparameter-based fine-tuning of the model, and the metrics for model evaluation are all discussed. Mozilla deepspeech-0.9.3

## Mozilla deepspeech-0.9.3

DeepSpeech is an open-source ASR system implemented by Mozilla in 2017. It is based on the Deep Speech research paper published by a research team at Baidu Research, Silicon Valley in 2014 (Hannun et al., 2014). It has been chosen for this work because of its simple setup process and its pre-trained model checkpoints that can be applied directly to STT tasks. Despite its inherent limitations when handling complex accents, especially non-native English speech or CS, the model is easily applied to recognise patterns in Nigerian English and Yoruba-English CS with relatively manageable data. This meant fine-tuning with smaller, annotated datasets of Nigerian English and Yoruba-English CS speech still yielded some improvement in transcription accuracy.

While DeepSpeech offers some support for non-English languages, it is not specifically trained to handle CS between languages. Analysis of the Proposed Model

## Analysis of the Proposed Model

The proposed model attempts to address the limitations of deepspeech-0.9.3 in accurately transcribing Nigerian English and Yoruba-English CS, yielding a better model for STT tasks. This was achieved with the creation and utilisation of a custom training dataset that exposed the pre-trained DeepSpeech model to Yoruba-English CS in order to integrate CS recognition into the fine-tuned model. Figure 1 shows the sequence of steps undergone in adapting and fine-tuning the proposed model.

The fine-tuned model incorporates a custom training dataset that includes monologues featuring Nigerian English accents and Yoruba-English CS; conversational dialogues in which speakers fluidly alternate between Yoruba and English; and a limited proportion of Nigerian Pidgin to enhance the representation of authentic, real-world conversational patterns.



**Fig. 1. Sequence Diagram of the Proposed Model**

## Data Collection and Preprocessing

The data collection process involved coming up with simulated monologues and dialogues featuring the Nigerian English accent and Yoruba-English CS and then gathering participants (speakers with varying accents and dialects) to perform audio recordings. As such, the transcriptions were ready before the audio was created, requiring only a few adjustments in cases where the speaker added or left out certain parts of the original transcript they were given.

The dataset gathered to train the deepspeech 0.9.3 model was a custom one that had a total of 680 clips (118 minutes or ~1.97 hours), including 235 audio clips of Nigerian English (40.52 minutes) and 445 clips of Yoruba-English CS (77.48 minutes). It was then split according to the ratio 80:10:10. The model was trained on 80% of the dataset, i.e. 544 samples (~94.4 minutes), and model validation and testing used 10% each, i.e. 68 samples (~11.8 minutes) each. Table 1 gives detail on how the dataset was split for training the model.

**Table 1. Data Splits**

|  |  |  |
| --- | --- | --- |
| **Data Splits** | **Duration (Hours)** | **Number of samples** |
| Train (80%) | ~1.573333 | 544 |
| Validation (10%) | ~0.196667 | 68 |
| Test (10%) | ~0.196667 | 68 |

The transcripts initially featured manual annotations to mark CS points. ‘|’ was used to mark the boundary between English and Yoruba in CS transcripts, e.g., “| ori mi n fọ́ mí jọ́ọ̀ | reduce the volume of that your music." Eventually though the annotations had to be removed in preprocessing because the character used to mark the CS boundary was not part of the tokens/alphabets recognised by the model. Also, Yoruba diacritics—since DeepSpeech’s alphabet file (alphabet.txt) does not readily contain the accented and tonal letters—were cleaned out of the transcripts given to the model during training.

In keeping with DeepSpeech’s specifications, the audio data had to be in WAV format, at an audio sample rate of 16 kHz with mono audio channel and a bit depth of 16-bit PCM (Pulse Code Modulation). Following that, the train.csv, dev.csv, and test.csv files were made based on the aforementioned 80:10:10 split ratio, each including the following columns: wav\_filename, wav\_filesize, and transcript, as required by DeepSpeech.

## Evaluation Metrics

To assess the system’s performance, metrics focusing on the system’s transcription accuracy, robustness, and adaptability were employed. The primary metric to evaluate the transcription accuracy for words, calculated separately for Nigerian English and Yoruba-English CS speech, is the Word Error Rate (WER). Mathematically, it is given as:

$WER = \frac{I + D + S}{N}$ (1)

Where $I$ is the number of insertions,

$D$ is the number of deletions,

$S$ is the number of substitutions, and

$N$ is the number of words in the reference (i.e., the transcriptions in the dataset). [(Iakushkin et al., 2018)](https://www.zotero.org/google-docs/?mULmB4).

The second evaluation metric used is the Character Error Rate (CER) metric given by:

$CER = \frac{I + D + S}{N} ✕ 100$ (2)

Where $I$ is the number of insertions,

$D$ is the number of deletions,

$S$ is the number of substitutions, and

$N$ is the number of characters in the reference [(MacKenzie & Soukoreff, 2002)](https://www.zotero.org/google-docs/?fuQGon)

The fine-tuned model was then compared with a baseline model of DeepSpeech, that is, before pre-training by Mozilla Developers, to highlight the improvements achieved.

## Training

To begin work on deepspeech-0.9.3, a virtual environment was set up on a CPU-Based personal computer to provide a sandbox within which the model could be trained and fine-tuned. The Mozilla DeepSpeech github repository was then cloned into that environment. Initial training was carried out on the baseline deepspeech-0.9.3. From that point forward however, the checkpoint file for the English pretrained model, offered in the DeepSpeech 0.9.3 release repository, was adapted for fine-tuning.

The initial training session ran for 2 epochs, which took about 14 hours. While this was unusual when compared to the duration of training sessions reported in other studies, it was somewhat expected due to the CPU-based setup. Eventually, however, adjustments were made to certain parameters, such as --train\_batch\_size, --dev\_batch\_size, --test\_batch\_size, and --learning\_rate, which improved the time taken per epoch.

The fine-tuning process involved tweaking the model’s hyperparameters, assigning them different values over various training sessions. Although some of the hyperparameters used to train the model were assigned the same values given to them in Mozilla’s training regimen, per the DeepSpeech 0.9.3 documentation, the following were iteratively tweaked: --train\_batch\_size, --dev\_batch\_size, --test\_batch\_size, --epochs, --learning\_rate, --n\_hidden, and --dropout\_rate. These hyperparameters were also altered to account for data size, hardware capabilities, and memory usage—with the added benefit of reducing the duration of each epoch during training.

In addition to these hyperparameters, DeepSpeech also offers checkpoint management flags to help with regulating checkpoint creation and storage. These were particularly useful for storing iteratively improved versions of the model at the end of each epoch during training. These checkpoints were stored in a directory from which they could be loaded if/when required.

# RESULTS AND DISCUSSION

The model, after being tested on 10% of the dataset, yielded an average WER equal to 0.760261, CER equal to 0.381241, with a loss equal to 95.734718 over 55 epochs. Table 2 shows the test result averages of the WER, CER and loss values for 5 of the most significant training sessions.

**Table 2. Results of Training Flag Tweaks over 5 Significant Iterations**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iterations** | **WER** | **CER** | **Loss** | **Training Flag Tweaks** |
| Initial | 0.836754 | 0.434092 | 105.857437 | --epochs 5, --train\_batch\_size 48, --dev\_batch\_size 48, --test\_batch\_size 48, --learning\_rate 0.00015 |
| Second | 0.823694 | 0.416963 | 101.721634 | --epochs 10, --train\_batch\_size 48, --dev\_batch\_size 48, --test\_batch\_size 48 |
| Third | 0.801306 | 0.405682 | 99.934967 | --epochs 10, --train\_batch\_size 48, --dev\_batch\_size 48, --dropout\_rate 0.2 |
| Fourth | 0.822761 | 0.413411 | 100.288620 | --epochs 20, --train\_batch\_size 42, --dev\_batch\_size 42, --test\_batch\_size 42, --dropout\_rate 0.2, --learning\_rate 0.0001 |
| Best | 0.760261 | 0.381241 | 95.734718 | --epochs 55, --train\_batch\_size 42, --dev\_batch\_size 42, --test\_batch\_size 42, --dropout\_rate 0.2, --learning\_rate 0.0001 |

The slight but gradual reduction in the model’s error rates, as can be observed in the metric values in Table 2, are visualised in Figure 2, with a screenshot of the performance of the best validating model checkpoint in Figure 3.



**Fig. 2. Bar Chart of Model Improvement over 5 Significant Iterations**



**Fig. 3. Screenshot of Test Average Results of the Best Validating Model Checkpoint**

Figure 4 contrasts the performance of the baseline deepspeech-0.9.3 model with that of the best validating model checkpoint.



**Fig. 3. WER and CER Comparison of the Baseline and the Fine-tuned Model**

Table 3 below presents three representative transcription errors from the test set used on the best validating model, focusing on the Yoruba-English CS scenarios. It includes the source transcript, predicted transcription, error metrics, and the associated audio file.

**Table 3. Transcription Errors from Test Set**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Example** | **Source Transcript** | **Predicted Transcription** | **WER** | **CER** | **Loss** | **Audio File** |
| Best | my cousin got engaged a ti start planning | my cousin go engaged at start planning | 0.375000 | 0.073171 | 23.145693 | yor-en-cs-0353.wav |
| Median | mo ti try to call her but o n ring ko si response | won they try to call ar bu alwiy co sive es phone | 0.769231 | 0.448980 | 64.088600 | yor-en-cs-0245.wav |
| Worst | ko si time to iron clothes i will wear this | coitin o so i an clote a readys | 1.000000 | 0.558140 | 74.498535 | yor-en-cs-0335.wav |

The overall WER of 0.760261 and CER of 0.381241 indicate potential overfitting, as it seems that the model struggles with generalising to the test set. The high loss (95.734718) suggests insufficient convergence after even 55 epochs, limited by CPU constraints and the small dataset used (a mere 118 minutes of data). This indicates potential underfitting as the model may not have fully captured training patterns. Overfitting seemed to be more prominent though, which is not too surprising given the size of the dataset. Further tweaks were attempted to try and account for these issues while remaining efficient with the setup for this training but no significant improvement was obtained and sometimes, training got abruptly halted due to memory usage issues.

While this issue might have been solved with a better environment setup, the lack of access to a GPU for this study necessitated the usage of a CPU-based machine. Although remote environments like Google Colab and Kaggle were considered, issues relating to the environment setup arose. In Colab, for instance, this issue could not be resolved due to the incompatible CUDA versions of deepspeech-0.9.3 and Colab’s environment. As such, using Colab offered no practical advantage, especially as the only other option available on the platform was the CPU. This reliance on and utilisation of a CPU meant the time taken for each epoch grew more and more inconvenient as the number of desired epochs increased. As such, the initial maximum number of epochs attempted for this study was 5. It was constrained to this maximum due to the amount of power supply required to keep the computer going for ~7 hours per epoch on the local CPU and the impracticality of securing power for that long.

In the local setup, tweaking certain hyperparameters proved effective in reducing the time taken per epoch. After adjustments, each epoch took ~22 minutes, increasing the feasible number of epochs to 55. Running a larger number of epochs (e.g., 75 or 100), to reduce loss and improve convergence, was still relatively infeasible considering the power supply available as well as the limited battery capacity of the computers relied on during the gaps between power supply.

The results achieved despite these limitations, however, show that there is massive potential for improvement, and demonstrates the viability of this approach to creating models that perform better than contemporary models in recognising various Nigerian English accents and that even extend Yoruba-English CS scenarios and other native CS scenarios as well.

# CONCLUSION

The study successfully adapted DeepSpeech to recognise Nigerian English and Yoruba-English CS speech, achieving modest improvements in transcription accuracy despite significant constraints. The methodology, including data collection from simulated real-world scenarios and preprocessing techniques, proved effective in principle but requires scaling to achieve practical utility. Overall, while the fine-tuned model takes a step toward inclusive STT technology, its current performance suggests that further optimisation is necessary in order for it to be applicable to and practical in the real-world.

To enhance the fine-tuned DeepSpeech model, future research should focus on expanding the custom dataset by incorporating a larger and more diverse collection of Nigerian English and Yoruba-English CS audio recordings, potentially sourced from real-world conversations to better capture natural variability. Access to GPU resources, either locally or through compatible cloud platforms (especially if Mozilla developers or others go back to work on DeepSpeech and provide a release that is entirely compatible with the contemporary cloud computing platforms’ machine setup), is recommended to support longer training epochs and improve model convergence. Additionally, adapting the alphabet file to include Yoruba diacritics may improve CS recognition as well as promote the utilisation of accurate Yoruba alphabets.

# REFERENCES

1. Thomas, A., Edanad, R. J., Raju, S. J., Nt, S., & Thampy, S. M. (2024). A Comprehensive Review of Speech-ToText Technologies: Incorporating Translation, Summarization, And Beyond. International Journal of Advances in Engineering and Management, 06(12), 164–168. https://doi.org/10.35629/5252-0612164168
2. Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., & Ng, A. Y. (2014). Deep Speech: Scaling up end-to-end speech recognition (Version 2). arXiv. https://doi.org/10.48550/ARXIV.1412.5567
3. Dubey, P., & Shah, B. (2022). Deep Speech Based End-to-End Automated Speech Recognition (ASR) for Indian-English Accents (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2204.00977
4. Amuda, S., Boril, H., Sangwan, A., & Hansen, J. H. L. (2010). Limited resource speech recognition for Nigerian English. 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, 5090–5093. https://doi.org/10.1109/ICASSP.2010.5495036
5. Lin, P.-J., Saeed, M., Chang, E., & Scholman, M. (2023). Low-Resource Cross-Lingual Adaptive Training for Nigerian Pidgin (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2307.00382
6. Mustafa, M. B., Yusoof, M. A., Khalaf, H. K., Rahman Mahmoud Abushariah, A. A., Kiah, M. L. M., Ting, H. N., & Muthaiyah, S. (2022). Code-Switching in Automatic Speech Recognition: The Issues and Future Directions. Applied Sciences, 12(19), 9541. https://doi.org/10.3390/app12199541
7. Amuda, A. A. (1994). Yoruba/english conversational code‐switching as a conversational strategy. African Languages and Cultures, 7(2), 121–131. https://doi.org/10.1080/09544169408717781
8. Church, K. W., Chen, Z., & Ma, Y. (2021). Emerging trends: A gentle introduction to fine-tuning. Natural Language Engineering, 27(6), 763–778. https://doi.org/10.1017/S1351324921000322
9. Ibejih, S., Oyewusi, W. F., Adekanmbi, O., & Osakuade, O. (2022). EDUSTT: IN-DOMAIN SPEECH RECOGNITION FOR NIGERIAN ACCENTED EDUCATIONAL CONTENTS IN ENGLISH.
10. Baller, T., Bennett, K., & Hamilton, H. J. (2021). Transfer Learning and Language Model Adaption for Low Resource Speech Recognition. Proceedings of the Canadian Conference on Artificial Intelligence. https://doi.org/10.21428/594757db.d3394351
11. Yılmaz, E., Heuvel, H. van den, & Leeuwen, D. A. van. (2018). Code-Switching Detection with Data-Augmented Acoustic and Language Models (No. arXiv:1808.00521). arXiv. https://doi.org/10.48550/arXiv.1808.00521
12. Agarwal, A., & Zesch, T. (2019). German End-to-end Speech Recognition based on DeepSpeech.
13. Hassan, M. A., Rehmat, A., Ghani Khan, M. U., & Yousaf, M. H. (2022). Improvement in Automatic Speech Recognition of South Asian Accent Using Transfer Learning of DeepSpeech2. Mathematical Problems in Engineering, 2022, 1–12. https://doi.org/10.1155/2022/6825555
14. Zhou, J., Zhao, S., Wang, H., Zhang, T.-H., Sun, H., Wang, X., & Qin, Y. (2024). Improving Zero-Shot Chinese-English Code-Switching ASR with kNN-CTC and Gated Monolingual Datastores (Version 5). arXiv. https://doi.org/10.48550/ARXIV.2406.03814
15. Iakushkin, O. O., Fedoseev, G. A., Shaleva, A. S., Degtyarev, A. B., & Sedova, O. S. (2018). RUSSIAN-LANGUAGE SPEECH RECOGNITION SYSTEM BASED ON DEEPSPEECH.
16. MacKenzie, I. S., & Soukoreff, R. W. (2002). A character-level error analysis technique for evaluating text entry methods. Proceedings of the Second Nordic Conference on Human-Computer Interaction, 243–246. https://doi.org/10.1145/572020.572056