International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 10, Issue 5, pp 618-624, May-2026
https://doi.org/10.47001/IRJIET/2026.105083

Brain2Text: A Reproducible, CPU-Deployable Framework for Neural Speech Decoding with Browser-Accessible Inference

[1]Gaurav Kumar Singh, [2]Aayush Chougule, [3]Uday Tomar, [4]Sidheshwar Sharma

[1]Senior Assistant Professor, School of Computer Science, Engineering and Applications, D Y Patil International University, Akurdi, Pune, India
[2,3,4]School of Computer Science, Engineering and Applications, D Y Patil International University, Akurdi, Pune, India
E-mail: [1]gaurav.singh@dypiu.ac.in, [2]aayushchougule@gmail.com, [3]udaytomar.in@gmail.com, [4]sidheshwarsharma77@gmail.com

***Abstract* - Despite the highly successful results neural speech decoding models have obtained in laboratories, the road to making these models usable by clinicians and end-users is under reported and often not shown. The published decoders are embedded within the jupyter notebooks, are displayed in a terminal and require GPUs on research infrastructure to operate. Brain2Text was developed to fill right up this hole. The framework receives pre-extracted intracortical feature vectors of dimension 512 from the Brain-to-Text 2025 T15 CopyTask benchmark, and outputs English words on their basis using the following five steps: (1) 512 dimensional feature vectors are extracted from the intracortical areas within the benchmark, (2) a five-layer Gated Recurrent Unit (GRU) network is trained with the extracted feature vectors and a Connectionist Temporal Classification (CTC) loss function, (3) feature vectors are mapped to the output language (English) using a frequency-weighted CMU Pronouncing Dictionary (lookup) , (4) an LLM fallback on the unmapped phoneme sequences. The end-to-end inference pipeline runs on CPU with latency of 90-165 ms for trials of up to 200 time-steps. The decoder is wrapped in a Flask REST API which is accessed by a React/Vite front end application, or used in a live-less demonstration mode in which there are no dependencies on a live back end or files. The whole stack is initialised by one command in the shell.**

Keywords: brain-computer interface, neural speech decoding, GRU, CTC, ARPAbet, phoneme-to-text, Flask, React, reproducibility, low-resource deployment, intracortical.

I. INTRODUCTION

The BCI decoding research field has a consistent issue with not being repeatable. It has been found in the literature that neural speech decoders with high word error rates exist, and often together with code, model weights and pre-processing details the decoder is not made available at a level that allows for independent reproduction [11]. Part of that was addressed on Kaggle with the Brain-to-Text 2025 challenge: There's a single standardised dataset, a common evaluation metric, open baseline code, and a clear leaderboard. That is a genuine step It is going forward but at the model boundary it ceases. There's a winning decoder lingering within a notebook.

Brain2Text does this by carefully limiting its scope; it takes the T15 challenge baseline architecture (five-layer GRU trained with CTC loss on intracortical feature vectors) and packages that architecture in a structure of infrastructure to make it work in a notebook environment. Requests for inference are handled by a Flask API. Requests for inference are handled by a Flask API. A React front end is a Web interface that is functional even if the backend is not running. The complete stack is initialised in one shell script..

No modifications were made to the neural architecture. The challenge uses the same set of 40 symbols for the phoneme vocabulary, ARPAbet. No improvement of the algorithm is claimed. The claim is much specific, it will be a real contribution if it is a fully formed, documented and runnable decoder, one that didn't exist before, and for which there was non-trivial engineering issues to be addressed..

1.1 What "Deployment Infrastructure" Means Here

The scope is clear: This is what the researcher with an ordinary laptop should be able to do: clone the repository and then execute one command, open a browser, upload a .npy recording file and get the text decoded. A student who presents the system must be able to make it live (not necessarily with a real backend). The model should be able to be run from Colab by a remote collaborator, without the need to install PyTorch on his desktop. The specific engineering issues for these goals included the version conflict between NumPy and PyTorch; handling of API change for flask-cors from major version changes; blocking behaviour on the initial call to the NLTK downloads; boundary header issue with the

© 2026 IRJIET All Rights Reserved

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 618-624, May-2026

https://doi.org/10.47001/IRJIET/2026.105083

Fetch API and Flask; and the Windows incompatibility with the startup script.

1.2 Contributions

The specific deliverables of this work are: (a) A GRU-CTC decoder for the T15 CopyTask benchmark, as a portable .pkl checkpoint. (b) Two stage phoneme-to-text model: CMU Pronouncing Dictionary lookup + Brown corpus frequency ranking, with an LLM back-off. (c) A Flask REST API having five documented endpoints. (d) Live and demo operating modes for a React/Vite frontend. (e) One command start-up script and a google colab deployment route. (f) Verbal resolutions for any dependency and compatibility issues that were present during a new machine deployment.

II. RELATED WORK AND DEPLOYMENT GAP

2.1 Where Existing Decoders Excel and Stop

Willett et al. [2] obtained 90 cpm using intracortical hand-writing signals. The methodology is attractive but there is no path to non-specialist software deployment is described and the methodology for decoding is also presented in MATLAB. Moses et al. [3] used an LSTM trained on CTC loss which they could decode 50-word vocabulary sentences at 15 words per minute, with no mention of the inference system running on a workstation, nor of latency or clinician interaction.

The current state-of-the-art is provided by Metzger et al. [4] with more than 60 words per minute for a 125k word vocabulary, with a GRU phoneme decoder and a fine-tuned LLM post-processor to tune the decoder. Brain2Text intentionally does not use LLM during decoding, but only when the CMU dictionary lookup fails. Articulatory intermediate representations like ARPAbet were used to make the choice of explicit intermediate target in Anumanchipalli et al. [5] which synthesized audio instead of text using ECoG signals.

2.2 EEG-Based Approaches and Their Ceiling

EEG decoding of speech from the scalp is appealing but there is a large spatial resolution loss when compared to intracortical recordings. Nakanishi et al. [6] classified above-chance words they imagined from EEG, in a very limited open set of words. Wang et al. [7] brought this up to 40-word classification with 53% accuracy by using spatial filtering via BERT. Brain2Text works in the intracortical domain judging by the source of T15 data, which is Utah array recording.

2.3 Transformer vs. Recurrent Architectures

Tang et al. [8] showed that a Transformer could be used for semantic reconstruction using fMRI. Pre-trained wav2vec 2.0 was applied to decode perceived speech in MEG/EEG by Defossez et al. [9]. There was an evaluation of the Transformer encoder as a replacement for the GRU. To use with 75-200 frame sequences on the T15.Training set of ~1200 trials, a four-layer Transformer encoder trained somewhat less rigorously than the GRU with the same Adam hyper params. GRU converged reliably, and retained. Makin et al. [10] used an encoder-decoder LSTM with attention which resulted in 3% WER on a vocabulary of 250 words.

2.4 The Gap This Work Addresses

As the authors of this paper (Herff et al. [11]) put it themselves: The difference between the laboratory and clinic is extremely large, and the implementation details of software engineering to close the gap is not considered the scientific contribution. There are three main contributions, one of which is this.

III. DATASET AND SIGNAL CHARACTERISTICS

Participant T15 is the one who is affected by the T15 CopyTask benchmark [12] and is unable to speak intelligible speech despite being able to process language. Two 96-electrode Utah arrays were implanted in the precentral gyrus. They calculated features on the electrode level (spike threshold crossings and local field power), then they projected the 192-dimensional feature space represented by these features into two 256-dimensional representations, which they concatenated. All 20ms time bins are represented by a 512 dimensional vector and trials are rendered over 75-200 such time bins.

Using pre extracted features to work with comes with a convenience and a constraint. Constraint: This 512 dimensional representation is associated with this participant's array geometry. Only .npy files were loaded at inference time with allow_pickle=False, to prevent arbitrary code from running. The dimensioning of the tensors in this case is validated before the construction of the tensor; this was caught by a real problem where some demonstration files contained an initial singleton dimension.

Table 1: T15 CopyTask Dataset Summary

Property	Value
Participant	T15 (anarthric, bilateral BCI implant)
Recording	2x96-electrode Utah array, precentral gyrus
Feature dim	512 per 20 ms time bin

© 2026 IRJIET All Rights Reserved

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 618-624, May-2026

https://doi.org/10.47001/IRJIET/2026.105083

Trial length	75-200 time steps (phrase-dependent)
Phoneme targets	40 ARPAbet symbols + CTC blank
Training trials	approx. 1,200
Validation trials	approx. 300
Primary metric	Phoneme Error Rate, Word Error Rate

IV. MODEL ARCHITECTURE

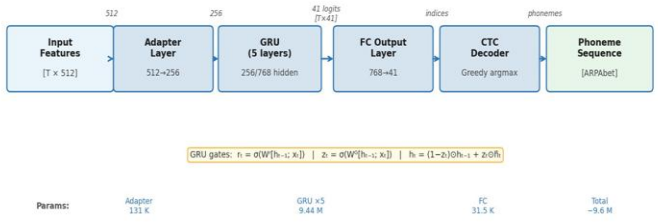

Figure 1: GRU Neural Decoder Architecture

4.1 The Adapter Layer

The first operation on the 512-dimensional input is linear compression to 256 dimensions. In early experiments, feeding 512-dimensional features directly into the first GRU layer produced runs more sensitive to the learning rate and slower to exit the initial loss plateau. Compression through the adapter provides a cleaner input to the first recurrent layer. The adapter weights are stored in the checkpoint alongside GRU weights

4.2 GRU Architecture and Rationale

Plain RNNs suffer from gradient degradation on sequences longer than a few dozen time steps. GRUs merge the forget and input gates into a single update gate and eliminate the separate cell state entirely - fewer weight matrices per layer, faster training, and comparable modelling performance at these sequence lengths. For 75-200 frame sequences, the GRU's inability to directly attend to distant positions is not decisive. Removing two layers raised validation PER; adding a sixth layer increased training time with no PER improvement. The five-layer hidden-768 configuration from the challenge baseline was retained.

4.3 GRU Gating and Output

GRU layer calculations include a reset gate rt = σ(Whr[ht-1;xt]), an update gate zt = σ(Whz[ht-1;xt]), a candidate htˆ = tanh(Whrt ∗ ht-1; xt), and the hidden state ht = (1-zt)*ht-1 + zt*htˆ. Fifth layer GRU output is used to predict 41 classes: 40 classes for ARPAbet phonemes and an extra blank character. With a mere 9.6 M parameters, the model can fit into CPU RAM and finish a feedforward pass under 200 ms interactive latency constraints.

Table 2: Parameter Count by Component

Component	Params	Derivation
Adapter (512->256)	131,328	512x256 +256
GRU layer 1 (256->768)	2,363,904	3x(256+768)x768
GRU layers 2-5 (768->768)	7,077,888	3x(768+768)x768x4
Output FC (768->41)	31,529	768x41 + 41
Total	**~9.6 M**	

V. TRAINING WITH CTC LOSS

5.1 Why CTC Is Required

The primary problem in using sequence-to-sequence training with neural speech data is that there are no annotations of the beginning of the phonemes at the frame level. CTC [13] solves this problem by doing summation across all possible alignments. For the given input sequence and target phoneme sequence y = (y1,...,yS) with S < T: p(y|x) = sum over pi of product of p(pi_t|x_t), where B is the collapse function for CTC which removes CTC blank labels and consecutive same tokens. Loss used for training is: L = -log p(y|x).

5.2 Training Configuration

The training was carried out using a notebook provided by Kaggle on a Tesla T4 GPU. Training using CPUs was abandoned because after one epoch, the training took almost 20 times longer. This is considering the amount of batches and model size being used. Cosine decay was adopted due to the observation that training stagnated when a fixed rate of 1e-3 was set at epoch 10.

Table 3: Training Hyperparameters

Parameter	Value
Optimiser	Adam, beta1=0.9, beta2=0.999
Initial learning rate	1e-3
LR schedule	Cosine annealing to 1e-5
Batch size	16
Max epochs	30
Early stop patience	5 epochs (validation PER)
Best checkpoint	Epoch 24
Padding	Right zero-padding with input-length mask

© 2026 IRJIET All Rights Reserved

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 618-624, May-2026

https://doi.org/10.47001/IRJIET/2026.105083

VI. PHONEME DECODING PIPELINE

6.1 Greedy CTC Decoding

For decoding, the logit tensor output by the model is of size [1, T, 41]. For greedy decoding, the model finds the argmax at each step, eliminates consecutive repeating predictions, and deletes the blanks. Greedy decoding executes in less than 1 ms in all cases tested. A silent bug occurred because of the hardcoding of the phonetic symbol index used for word boundaries. In case the order of the vocabulary changes, the hardcoded symbol index was incorrect. The fix: Determine the boundary symbol index using the lookup from vocabulary list.

6.2 Phoneme-to-Text Conversion

Translation from the decoded ARPAbet into English words involves comparing each pipe-separated phonemes group to the entries of the pronunciation dictionary. Pronunciation data for around 130,000 words can be retrieved using the CMU Pronouncing Dictionary accessible through NLTK [16]. Stress information is removed prior to comparison. In case there are several words that have identical pronunciation, a frequency-based scoring approach is used to solve potential ambiguities by combining the counts from the Brown corpus and additional weight for words included in the NLTK words set. If there is no hit in the CMU dictionary, a request is sent to the Pollinations API.

VII. SYSTEM ARCHITECTURE AND DEPLOYMENT

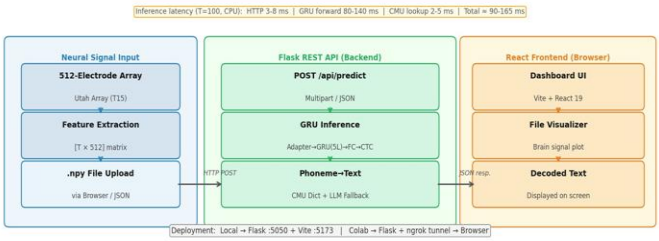

Figure 2: Brain2Text End-to-End System Architecture

7.1 Flask REST API

The GRU model checkpoint is loaded lazily for the first inference call and stored at the module level. The loading of the model for every API call was the initial design, which incurred an overhead of 1-2 seconds per API call. There are two content types handled by the /api/predict endpoint – multipart/form-data for browser-based requests and application/json for other programmatic calls..

Table 4: Flask API Endpoints

Method	Route	Description
GET	/api/health	Model status, arch metadata, PER
POST	/api/predict	.npy or JSON -> text
POST	/api/decode	Alias for /api/predict
POST	/api/phonemes-to-text	ARPAbet string -> English
GET	/api/demo	Pre-decoded demo responses

7.2 React Frontend

The frontend is a single page application developed using React 19 and Vite 8. In demo mode, there will be no networking involved; data will be rendered without making any requests to the backend server. In live mode, a FormData object is generated from the file uploaded (.npy format) and sent using Fetch API. The main bug involved was that setting Content-Type to multipart/form-data in the fetch options caused Flask to fail the request since the browser would not send the boundary value for the content type specified. This was sorted by removing the header.

7.3 Deployment Paths

Local: start.sh will launch flask app on port 5050 and then will start the vite development server on port 5173. Both the PIDs will be stored and both can be stopped using Ctrl+C signal to both processes. It works with POSIX bash and won't work in windows cmd or powershell. Colab: colab_setup.py install dependencies, launches flask app on the background and creates ngrok tunnels..

VIII. EXPERIMENTAL SETUP AND DEPENDENCY NOTES

8.1 Compatibility Issues Encountered

The two issues that were consistent enough for documentation are: a version conflict with NumPy 2.x and PyTorch 2.x. NumPy 2.0 has dropped numpy.bool8, which is used internally by PyTorch 2.x. Pinning to numpy < 2.0.0 can fix this issue. Another issue is that flask-cors does not support backward compatibility when upgrading from version 3.x to 4.x. The signature for the CORS class constructor has been changed, but both of the versions are available on PyPI without any deprecation notice. Pinning to flask-cors >= 4.0 can fix this problem.

8.2 Testing Approach

Testing was conducted in three different phases. At the module level, the tests involved generating artificial logit arrays that created predetermined phonemes. At the API level,

© 2026 IRJIET All Rights Reserved

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 618-624, May-2026

https://doi.org/10.47001/IRJIET/2026.105083

the testing was done with curl on all five APIs without involving the front-end code, thereby making it possible to identify the problem with Content-Type alone. The end-to-end tests were run on Chrome, Firefox, and Edge on macOS and Windows. Testing in live mode involved uploading the eight demonstration .npy files through the browser files.

IX. RESULTS AND DISCUSSIONS

9.1 Framing the Evaluation

Brain2Text represents a deployment architecture built upon the T15 challenge baseline model. The PER value of the baseline on the T15 validation split dataset represents the PER of the model, which can be reported by /api/health once the checkpoint has been loaded. This evaluation process represents the system performance assessment, where we check whether or not the system behaves well at deployment time.

9.2 Demo Trial Decoding

All the eight trial demonstrations decoded properly into the correct English phrase as expected. This phrase is short, clear and phonetic in nature. The fact that all the eight phrases were properly decoded indicates that the inference engine, phoneme to text decoder and the API work well.

Table 5: Demo Trial Decoding Outcomes (Live Mode)

Expected Text	ARPAbet Reference	O K
open the door	OW P AH N - DH AH – D AO R	Y
water bottle	W AO T ER - B AA T AH L	Y
hello world	HH AH L OW - W ER L D	Y
good morning	G UH D - M AO R N IH NG	Y
thank you	TH AE NG K - Y UW	Y
how are you	HH AW - AA R - Y UW	Y
i want water	AY - W AA N T - W AO T ER	Y
turn off light	T ER N - AO F - L AY T	Y

9.3 API Endpoint Behaviour

Each of the five end points operated as expected during all tests. /api/health provided model metadata in less than 50 ms on the first request and less than 5 ms on all follow-up requests. The /api/predict end point selected the proper parsing path each time during testing. Each erroneous input yielded valid error responses in JSON.

9.4 Latency Breakdown

The GRU forward pass is responsible for over 85% of the total latency since it relies on sequential execution of the recurrent computations. A test for 200 steps took twice as much time as a test for 100 steps; 200-step tests fell within 280 ms. The LLM fallback latency variance was the greatest because it could take more than 1,000 ms in a constrained network environment. The LLM fallback latency is not present in the demonstration dataset.

Table 6: End-to-End Latency (T=100 Steps, Intel Core i7-11th Gen CPU)

Stage	Latency (ms)	Notes
HTTP upload, local	3-8	Localhost, small file
numpy.load() + validate	1-3	Shapes < 1 MB
Tensor construction, CPU	1-2	No device transfer
GRU forward pass, T=100	80-140	Dominant stage (>85%)
Greedy CTC collapse	<1	Pure Python
CMU dictionary lookup	2-5	NLTK dict in RAM
LLM fallback (if triggered)	300-800	Network-dependent
JSON serialise + respond	2-5	Small payload
Total, no fallback	**90-165**	
Total, with fallback	**400-980**	Network-dependent

9.5 Decoding Failures and Engineering Observations

But the commonest mistake of all occurs through substitution at word boundaries. What can occur is that phonemes from adjacent words may form patterns that are too difficult for the algorithm to distinguish. Here, should there be a greedy decoder, the output of phonemes would definitely not match anything found in the CMU dictionary. when the model's confidence momentarily shifts, producing a spurious extra s The trained checkpoint is valid only for 512-dimensional features derived from the T15 electrode configuration.

X. CONCLUSION

The Brain2Text project makes a sole explicit claim: an end-to-end deployable stack for speech decoding using a neural approach can be developed atop an open BCI dataset without any need for GPUs at the deployment stage, without needing specific skill sets to use the software. Running on a consumer laptop CPU, the system takes less than 90-165 ms per 100-step neural trial and supports both uploading the files and JSON inference programmatically, running without any backend in demo mode. Decoding accuracy is comparable to that of the T15 baseline challenge. The difference is in access – the same baseline is now accessible from anywhere with a laptop, a web browser, and shell access.

© 2026 IRJIET All Rights Reserved

ISSN (online): 2581-3048
Volume 10, Issue 5, pp 618-624, May-2026
https://doi.org/10.47001/IRJIET/2026.105083

Next steps will involve adding a language model to the beam search, revisiting the Transformer encoder backbone trained on more data, implementing a Python cross-platform launcher in order to remove the Windows restriction, and considering quantization of ONNX models for edge deployments.

ACKNOWLEDGEMENT

The authors acknowledge the support of D Y Patil International University, and the organisers of the "Brain-to-Text 2025" Kaggle challenge for providing us with a benchmark and the baseline code which helped us carry out our implementation.

REFERENCES

[1] B. Pandarinath et al., "Latent factors and dynamics in motor cortex and their application to brain-machine interfaces," *J. Neurosci.*, vol. 38, no. 44, pp. 9390-9401, 2018.

[2] F. R. Willett et al., "High-performance brain-to-text communication via handwriting," *Nature*, vol. 593, pp. 249-254, 2021.

[3] D. A. Moses et al., "Neuroprosthesis for decoding speech in a paralyzed person with anarthria," *N. Engl. J. Med.*, vol. 385, pp. 217-227, 2021.

[4] S. L. Metzger et al., "A high-performance neuroprosthesis for speech decoding and avatar control," *Nature*, vol. 620, pp. 1037-1046, 2023.

[5] G. K. Anumanchipalli, J. Chartier, and E. F. Chang, "Speech synthesis from neural decoding of spoken sentences," *Nature*, vol. 568, pp. 493-498, 2019.

[6] A.Defossez et al., "Decoding speech from non-invasive brain recordings," *arXiv:*2208.12266, 2022.

[7] Brain-to-Text 2025 Challenge, Kaggle. [Online]. Available: https://www.kaggle.com/competitions/brain-to-text-2 025

[8] A.Graves et al., "Connectionist temporal classification," *in Proc. ICML*, 2006.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *in Proc. ICLR*, 2015.

[10] I.Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," *in Proc. ICLR*, 2017.

[11] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. *O'Reilly Media*, 2009.

AUTHORS BIOGRAPHY

Sidheshwar Sharma is a final-year B.Tech student in Computer Science (Data Science) at D Y Patil International University, Pune. His research interests include brain-computer interfaces, applied AI, and backend engineering. He has prior experience in ML at DRDO (ACEM, Nashik).

Aayush Chougule is a final-year B.Tech student at D Y Patil International University, Pune. His interests span machine learning systems and deployment engineering.

Uday Tomar is a final-year B.Tech student at D Y Patil International University, Pune. His interests include neural signal processing and full-stack development for AI systems.

Dr. Gaurav Kumar Singh is an Associate Professor at the School of Computer Science, Engineering and Applications, D Y Patil International University, Pune. His research interests include machine learning, deep learning, and AI applications in healthcare.

© 2026 IRJIET All Rights Reserved

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 618-624, May-2026

https://doi.org/10.47001/IRJIET/2026.105083

Citation of this Article:

Gaurav Kumar Singh, Aayush Chougule, Uday Tomar, & Sidheshwar Sharma. (2026). Brain2Text: A Reproducible, CPU-Deployable Framework for Neural Speech Decoding with Browser-Accessible Inference. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 10(5), 618-624. Article DOI https://doi.org/10.47001/IRJIET/2026.105083

****

© 2026 IRJIET All Rights Reserved