

Appendix Z: Literature review and the gap

Before looking at this project, I looked at work on silent speech and on eye-controlled systems. Most people tried to perfect silent speech or to find the best electrode layout. My learning was different: people who are sick or in pain often only need essential words. They do not need long conversations. So, my question became: What is the smallest number of electrodes and the smallest useful word list that still lets a person say the important things? When the confidence is high, muscle activity alone can confirm the word but when the confidence is weak, can I add one simple eye blink to confirm the word? That blink is not used for all typing. It is used only when the system is unsure. I hypothesized that this could keep the load on the eyes very low and make the whole system more reliable in urgent care.

Z1: Key EMG Studies reviewed

Study (title)	Authors	Year	Objective	Material / device used	Sample (participants and data collected)	Results	Important notes
All-weather, natural silent speech recognition via machine-learning-assisted tattoo-like electronics	Youhua Wang, Tianyi Tang, Yin Xu, Yunzhao Bai, Lang Yin, Guang Li, Hongmiao Zhang, Huicong Liu, YongAn Huang	2021	Build a practical, “all-weather” silent speech system with flexible skin-like electrodes	Ultra-thin tattoo-like electrodes on facial muscles; four channels; ear-mounted wireless data unit; cloud linear discriminant analysis classifier	The article’s summary describes 110 words trained with 10 repeats per user	92.64 percent average accuracy on a one hundred and ten-word list; robust to about forty-five percent facial deformation; usable in noisy or dark places	Showed real-world demos and reports that accuracy stays high with 4 channels; also analyzed what happened when channels were lost
A novel silent speech recognition approach based on parallel inception convolutional neural network and Mel-frequency spectral coefficients	Jinghan Wu, Yakun Zhang, Liang Xie, Ye Yan, Xu Zhang, Shuang Liu, Xingwei An, Erwei Yin, Dong Ming	2022	Improve silent speech recognition using a deep parallel inception model with spectral features	6 channel facial surface electromyography and Mel-frequency spectral coefficients; deep convolutional neural network	28 subjects; 100 classes of short Chinese phrases; plus, transfer tests on 4 new subjects	Best accuracy around 90.76 within subjects; transfer learning helped across subjects	First large study to use Mel-frequency spectral coefficients with surface electromyography for silent speech.

Decoding silent speech from high-density surface electromyography using transformer	Rui Song, Xu Zhang, Xi Chen, Xiang Chen, Xun Chen, Shuang Yang, Erwei Yin	2023	Do syllable-level sequential decoding with a Transformer and a language model	264-channel high-density arrays on face and neck	Eight subjects silently reading 33 phrases (82 syllables), 20 repeats each	Phrase accuracy about 96.37 percent; character error rate about 5.14 percent	Moved beyond word lists to sequence decoding; showed the strength of high-density sensors with a language model.
Decoding Silent Speech Based on High-Density Surface Electromyogram Using Spatiotemporal Neural Network	Xi Chen, Xu Zhang, Xiang Chen, Xun Chen	2023	Syllable-level end-to-end decoding from high-density signals	464-channel arrays over facial and laryngeal muscles	15 subjects; 33 Chinese phrases (82 syllables)	Phrase accuracy about 97.17 percent; character error rate about 3.11 percent	Careful coverage of face and larynx; very high performance with high-density arrays.
Knowledge-Distilled Ensemble Model for surface electromyography-based Silent Speech Interface	Wenqiang Lai, Qihan Yang, Ye Mao, Endong Sun, Jiangnan Ye	2023 (preprint)	Make a light-weight ensemble model for portable silent speech	Standard facial surface electromyography (three facial muscles in one version); deep learning with knowledge distillation	Dataset of the NATO 26 alphabet; 3900 samples	85.9 percent test accuracy on alphabet classification	Spelling by letters enables open-vocabulary text, but letter-by-letter entry is slower for urgent messages.
Sequence-to-Sequence Voice Reconstruction for Silent Speech in a Tonal Language	Huiyan Li, Haohong Lin, You Wang, Hengyang Wang, Ming Zhang, Han Gao, Qing Ai, Zhiyuan Luo, Guang Li	2021–2022	Convert surface electromyography of silent Mandarin speech into audible speech	Facial and neck surface electromyography; sequence-to-sequence model and vocoder	6 Mandarin speakers	Character error rate about 6.41 percent for synthesized speech (human evaluation)	Early proof that tone can be recovered from muscle signals in Mandarin.
Diff-ETS: Learning a Diffusion Probabilistic Model for Electromyography-to-Speech Conversion	Zhao Ren, Kevin Scheck, Qinhan Hou, Stefano van Gogh, Michael Wand, Tanja Schultz	2024	Improve naturalness of electromyography-to-speech using a diffusion model	8 channel facial electromyography at 1000 hertz; encoder plus diffusion plus vocoder	Open-vocabulary, single-speaker database with silent and audible sessions	Clear gains in naturalness by listening tests and objective measures; supports a trade-off	First diffusion-based approach for electromyography-to-speech.

						between latency and quality	
Electrode Setup for Electromyography-Based Silent Speech Interfaces: A Pilot Study	Inge Salomons, Eder del Blanco, Eva Navas, Inma Hernáez	2025	Find a practical 8 channel electrode layout for future databases for laryngectomized speakers	Compared paired versus concentric electrodes; chose 8 bipolar pairs over key facial and lower-face muscles; Quattrocento amplifier at 2048 hertz	One Spanish male speaker; three sessions; 250 balanced sentences per session	Reports phone-classification results that guided the final eight-channel setup	A clear, physiology-guided recipe that others can copy for data collection.
SVIT-SSR: a surface electromyography-based Vision Transformer approach for Silent Speech Recognition	Zhao Li, Bin Ma, Weifan Mao, Yizhou Lu (and co-authors)	2024	Apply a vision transformer style network to surface electromyography	Surface electromyography with time–frequency features	Article summary reports a 33-command dataset	Reported accuracy about 96.67 percent	Short communication; shows transformer-style gains; details in full text.
A Silent Speech Decoding System from electroencephalography and electromyography with heterogeneous electrode configurations	Masakazu Inoue, Motoshige Sato, Kenichi Tomeoka, Nathania Nah, Eri Hatakeyama, Kai Arulkumaran, Ilya Horiguchi, Shuntaro Sasai	2025 (preprint)	Achieve robust decoding across different electroencephalography and electromyography placements	Combined electroencephalography and electromyography; multi-task learning across varied layouts	Healthy participants and one speech-impaired patient	Word classification about 95.3 percent in healthy participants; 54.5 percent in the patient	Suggests promise for clinical users and across-language calibration when sensors are not in perfect positions.
Can large language models understand unvoiced speech? Exploring electromyography-to-text conversion with large language models	Payal Mohapatra, Akash Pandey, Xiaoyuan Zhang, Qi Zhu	2025 (preprint and conference paper)	Map unvoiced facial electromyography into a frozen large language model for closed-vocabulary text	An adaptor network in front of a large language model; unvoiced facial electromyography	Person-specific, very small data setting (as low as six minutes per person)	Word error rate about 0.9 in a closed vocabulary; improved over specialized baselines in low-data settings	Highlights strong person-specific signals and data efficiency, good inspiration for compact systems.

Table 1. What other teams tried for silent speech with surface electromyography.

Z2: Eye Gaze and eye blink Studies reviewed

Study (title)	Authors	Year	What was tested	People and place	Main finding	Why this matters for my design
Fast gaze typing with an adjustable dwell time	Päivi Majaranta, Ulla-Kaija Ahola, Oleg Špakov	2009	Eye typing on a screen, where each letter needs a gaze “dwell”	11 healthy adults	Longer dwell helps accuracy, but long fixes are tiring and slow the speed; even trained users must wait for the dwell time for each key	Eye typing needs many long looks. That is tiring for sick users and slow for urgent needs. I therefore avoid full gaze typing.
Assessing oral comprehension with an eye-tracking based test in the intensive care unit	Lucie Bodet-Contentin and colleagues	2022	Eye tracking to check understanding in the intensive care unit	Patients in intensive care	Tool was usable with training and the right device, but it is a special test, not day-to-day conversation	Even when eye tracking can work, it takes careful setup. My design must work fast with minimal setup.
Decision-making for access to augmentative and alternative communication in late-stage amyotrophic lateral sclerosis	Darlene McLaughlin and colleagues (Oregon Health & Science University)	2021	Guidance for access methods including eye gaze	People with amyotrophic lateral sclerosis	Notes that a typical short dwell time is about 0.5 seconds, but people with reduced eye movement will struggle and may need alternate switches	Even small dwell times can be too hard when eye movement is weak. I therefore use only a single blink to confirm only when needed.
Prevalence and risk factors of exposure keratopathy among critically ill patients: a systematic review and meta-analysis	Ying Chen and colleagues	2023	Eye surface damage in intensive care (drying, lids not closing, etc.)	Critically ill patients in intensive care units	Dry eyes and poor lid closure are common (prevalence around 23 to 34 percent); risk rises with ventilation and sedation	Dry eyes make sustained gaze control harder. I keep eye effort minimal.
Tear Film and Ocular Surface Society Dry Eye Workshop II (tear film report)	Mark D. P. Willcox and colleagues	2017	How dry eye affects seeing and comfort	Global consensus report	Dry eye causes unstable tear film and visual disturbance	Dry eye lowers visual performance over time, so long gaze sessions are not ideal in hospital rooms.
Impact of dry eye on prolonged reading	Optometry and Vision Science study team	2018	Reading for a long time with dry eye	Clinical study	Dry eye hurts long-duration reading performance	Supports my choice to avoid long stretch eye typing for urgent talk.
Electrooculograms for human–computer interaction	Wei-De Chang and colleagues	2019	Using eye signals, including blinks, for selection	Review	Blinks work well for selection; saccades for moving	I use one voluntary blink to confirm only when the word is uncertain. That is low effort and robust.
Vision-based eye-blink detection for selection	Aleksandra Królak, Paweł Strumiłło	2012	Detecting deliberate blinks for commands	49 users (twelve with disabilities)	Blink-to-select is feasible and needs simple hardware	Confirms that a minimal blink can be a reliable confirmation step.

Table 2. Why we did not rely on full eye-gaze typing in patient-like environments

Z3: What these studies told us

1. High-density sensors and big models can be very accurate, but they need many electrodes and careful placement. That is hard to keep working on a busy ward bed (Chen et al., 2023; Song et al., 2023).
2. Small channel counts can still work for daily words. The tattoo-like four-channel system reached above ninety percent accuracy on one hundred and ten daily words and even reported how accuracy falls when channels drop. This encouraged my idea of “minimal channels, minimal words” plan (Wang et al., 2021).
3. Spelling each letter is slower. Alphabet systems are flexible, but not ideal for urgent “I need help now” messages (Lai et al., 2023).
4. Eye-gaze typing can be tiring and slow, especially with long dwell times. In intensive care, dry eyes and poor lid closure are common, which makes long gaze control even harder (Majaranta et al., 2009; Karaskus et al., 2018; Chen et al., 2024).
5. A single blink can be a simple “yes”. Many human–computer interaction studies use blinks for selection. I use exactly that: one blink only when the guess is uncertain (Królak & Strumillo, 2012; Chang, 2019)

Z4: The research gap and my guiding question

The gap: Most research tries to perfect silent speech or to find the best large electrode layout. But patients who are unwell often just need to say a few essential things. They need fast, dependable messages, not perfect full-sentence speech.

My solution:

- Uses the minimum number of electrodes that are easy to place and keep in place.
- Supports a small, essential word list (for example: “pain,” “up” “down,” “hmmm,” “Silence” “yes,” “no,” “help”).
- If the model is confident, speaks the word right away.
- If the model is not confident, asks the person for one blink to confirm. If they do not blink, the system quickly offers the next best word or a short choice list.
- This way, eyes do very little work most of the time, which is safer for dry eyes and tired users, and communication stays fast.

References:

- Wang, Y., Tang, T., Xu, Y., Bai, Y., Yin, L., Li, G., Zhang, H., Liu, H., & Huang, Y. (2021). All-weather, natural silent speech recognition via machine-learning-assisted tattoo-like electronics. *npj Flexible Electronics*, 5, Article 20. <https://doi.org/10.1038/s41528-021-00119-7>
- Wu, J., Zhang, Y., Xie, L., Yan, Y., Zhang, X., Liu, S., An, X., Yin, E., & Ming, D. (2022). A novel silent speech recognition approach based on parallel inception convolutional neural network and Mel frequency spectral coefficient. *Frontiers in neurorobotics*, 16, 971446. <https://doi.org/10.3389/fnbot.2022.971446>
- Song, R., Zhang, X., Chen, X., Chen, X., Chen, X., Yang, S., & Yin, E. (2023). Decoding silent speech from high-density surface electromyographic data using transformer. *OpenReview*. 80 (1), <https://doi.org/10.1016/j.bspc.2022.104298>.
- Chen, X., Zhang, X., Chen, X., & Chen, X. (2023). Decoding Silent Speech Based on High-Density Surface Electromyogram Using Spatiotemporal Neural Network. *IEEE transactions on neural systems and rehabilitation engineering: A publication of the IEEE Engineering in Medicine and Biology Society*, 31, 2069–2078. <https://doi.org/10.1109/TNSRE.2023.3266299>
- Lai, W., Yang, Q., Mao, Y., Sun, E., & Ye, J. (2023). Knowledge Distilled Ensemble Model for sEMG-based Silent Speech Interface. *Cornell University*. <https://doi.org/10.48550/arXiv.2308.06533>
- Li, H., Lin, H., Wang, Y., Wang, H., Zhang, M., Gao, H., Ai, Q., Luo, Z., & Li, G. (2021). Sequence-to-Sequence Voice Reconstruction for Silent Speech in a Tonal Language. *Cornell University*. <https://doi.org/10.48550/arXiv.2108.00190>
- Salomons, I., del Blanco, E., Navas, E., & Hernáez, I. (2025). Electrode Setup for Electromyography-Based Silent Speech Interfaces: A Pilot Study. *Sensors*, 25(3), 781. <https://doi.org/10.3390/s25030781>
- Li, Z., Ma, B., Mao, W., Zhang, J., Yu, Z. and Lu, Y. (2024). SVIT-SSR: A sEMG-based vision transformer approach for silent speech recognition. *Electronics Letter*, 60: e13285. <https://doi.org/10.1049/ell2.13285>
- Inoue, M., Sato, M., Tomeoka, K., Nah, N., Hatakeyama, E., Arulkumaran, K., Horiguchi, I., & Sasai, S. (2025). A silent speech decoding system from EEG and EMG with heterogeneous electrode configurations. *Cornell University*. <https://doi.org/10.48550/arXiv.2506.13835>

- Mohapatra, P., Pandey, A., Zhang, X., & Zhu, Q. (2025). Can LLMs understand unvoiced speech? Exploring EMG-to-text conversion with LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 703–712). Vienna, Austria: Association for Computational Linguistics. <https://aclanthology.org/2025.acl-short.56/>
- Majaranta, P., Ahola, U.-K., & Špakov, O. (2009). Fast gaze typing with an adjustable dwell time. In *Proceedings of the 27th ACM Conference on Human Factors in Computing Systems (CHI 2009)* (pp. 357–360). Association for Computing Machinery. <https://doi.org/10.1145/1518701.1518758>
- Bodet-Contentin, L., Messet-Charrière, H., Gissot, V., Renault, A., Muller, G., Aubrey, A., Gadrez, P., Tavernier, E., & Ehrmann, S. (2022). Assessing oral comprehension with an eye tracking based innovative device in critically ill patients and healthy volunteers: a cohort study. *Critical care (London, England)*, 26(1), 288. <https://doi.org/10.1186/s13054-022-04137-3>
- McLaughlin, D., Peters, B., McInturf, K., Eddy, B., Kinsella, M., Mooney, A., Deibert, T., Montgomery, K., & Fried-Oken, M. (2021). Decision-making for access to AAC technologies in late stage ALS. In B. T. Ogletree (Ed.), *AAC Challenges and Solutions: Improving Everyday Service Delivery* (pp. 169–199). Plural Publishing.
- Chen, Y., He, J., Wu, Q., Pu, S., & Song, C. (2024). Prevalence and risk factors of exposure keratopathy among critically ill patients: A systematic review and meta-analysis. *Nursing open*, 11(1), e2061. <https://doi.org/10.1002/nop2.2061>
- Willcox, M. D. P., Argüeso, P., Georgiev, G. A., Holopainen, J. M., Laurie, G. W., Millar, T. J., Papas, E. B., Rolland, J. P., Schmidt, T. A., Stahl, U., Suarez, T., Subbaraman, L. N., Uçakhan, O. Ö., & Jones, L. (2017). TFOS DEWS II Tear Film Report. *The ocular surface*, 15(3), 366–403. <https://doi.org/10.1016/j.jtos.2017.03.006>
- Karakus, S., Mathews, P. M., Agrawal, D., Henrich, C., Ramulu, P. Y., & Akpek, E. K. (2018). Impact of dry eye on prolonged reading. *Optometry and Vision Science*, 95(12), 1105–1113. <https://doi.org/10.1097/OPX.0000000000001303>
- Chang W. D. (2019). Electrooculograms for Human-Computer Interaction: A Review. *Sensors (Basel, Switzerland)*, 19(12), 2690. <https://doi.org/10.3390/s19122690>
- Królak, A., & Strumiłło, P. (2012). Eye-blink detection system for human–computer interaction. *Universal Access in the Information Society*, 11, 409–419. <https://doi.org/10.1007/s10209-011-0256-6>

Appendix A: Standardizing Electrode Placement (MindScribe)

A1. Purpose

My goal was to record repeatable, high-quality facial and submental surface electromyography (sEMG) for silent speech recognition. I used 8 bipolar channels (two electrodes per channel, 16 electrodes total) plus 1 BIAS electrode on the ear lobe, connected to an OpenBCI Cyton. I standardized the exact muscles, landmarks, polarity (P/N), skin preparation, wiring, and verification steps so the signals stay consistent over many sessions with my grandmother (participant).

A2. Brief literature review (why these muscles and why bipolar pairs)

Recent silent-speech work shows the most informative sites are on the cheeks and lip-corner muscles (e.g., zygomaticus major, risorius, depressors), jaw (masseter), and submental/suprahoid region (anterior digastric and stylohyoid). A pilot study by Salomons, del Blanco, Navas, & Hernáez (2025) compared electrode types and screened many muscles, then fixed an 8-muscle set for database recording: ABD, DAO, RIS, LLS, MAS, ZYG, DLI, SLH using paired single-electrode bipolar recording (two small electrodes per muscle aligned with the fibers). This outperformed concentric designs and was more practical near the lips. Prior datasets (e.g., the Berkeley silent-EMG work) highlight the value of consistent landmarks, ear-lobe bias, and careful strain relief to reduce session-to-session drift.

Salomons et al. ran a step-by-step selection (compare electrode types → test 14 muscles individually → confirm reduced set), then published a final eight that match exactly the muscles I need. Their choices also match what I found in my pilot trials, and they explain practical issues near the lip edge that I also observed.

Study (focus)	What they tested	Montage highlights (relevant to me)	Key placement result I used
Salomons et al., 2025 (electrode-setup pilot)	They ran a pilot to find a practical and accurate electrode setup for silent-speech EMG. They compared paired single-electrode bipolar recordings with concentric electrodes and they tested fourteen individual face and neck muscles, then confirmed a smaller set.	They used paired single-electrode bipolar placement. Based on their tests, they finalized eight muscles: ABD, DAO, RIS, LLS, MAS, ZYG, DLI, and SLH.	I followed this study exactly. I used the same eight muscles and I placed two electrodes per muscle along the muscle fibers in a bipolar pair.
Berkeley silent-EMG dataset (single speaker)	They built a silent-speech EMG dataset and showed how to record across many sessions for one person.	Their montage covered the cheeks, chin, and throat/under-chin areas. They also used ear-lobe bias and focused on re-attaching	I kept the ear-lobe BIAS and I used the same re-attachment checks before every recording so my

		electrodes in the same way each time.	signals stayed consistent.
<u>CSL / array corpora</u> (region arrays)	They released dense electrode arrays over the face and neck to study EMG-to-speech.	These arrays confirmed that the cheek and submental regions carry a lot of speech information, but arrays on the face can be rigid and can increase cross-talk on small muscles.	I chose to target specific muscles with small bipolar pairs instead of using arrays, so I could get cleaner signals and easier, repeatable placement.

Table A1. Studies used in Channel Selection.

A3. What I did (final montage and protocol)

A3.1 Muscles by side (your exact layout)

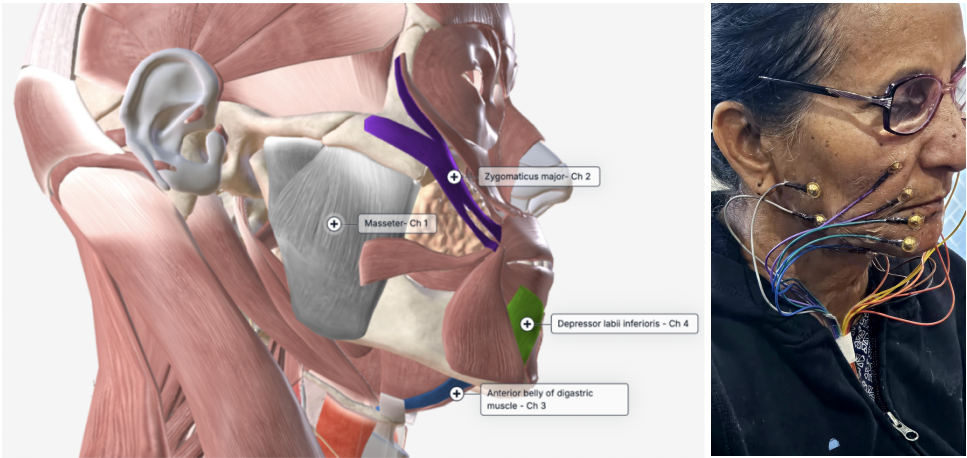


Figure A1. Right Side Muscle Montage. Software Bio digital Human.

Right side (4 channels / 4 muscles): Zygomaticus major (ZYG), Masseter (MAS), Depressor labii inferioris (DLI), Anterior belly of digastric (ABD).

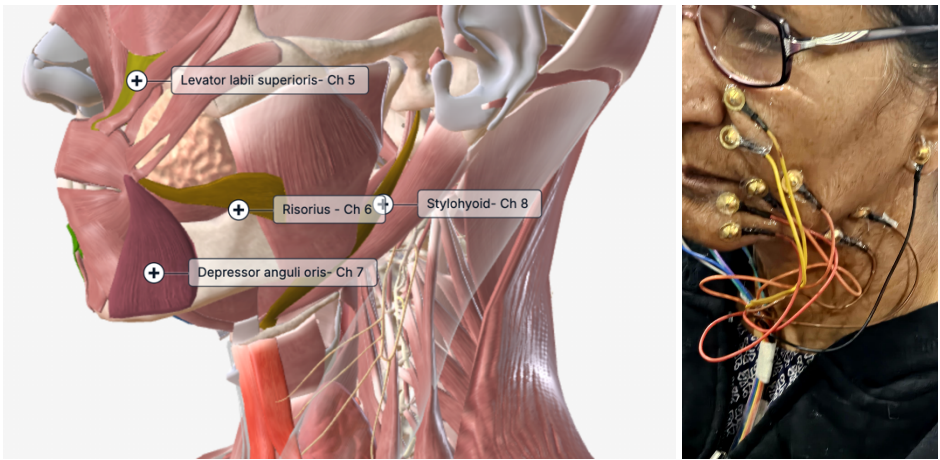


Figure A2. Left Side Muscle Montage. Software Bio digital Human.

Left side (4 channels / 4 muscles): Levator labii superioris (LLS), Risorius (RIS), Depressor anguli oris (DAO), Stylohyoid (SLH).

I used gold OpenBCI cup electrodes with TENS paste. Each muscle had two electrodes (bipolar pair), placed 10–20 mm apart and aligned with the muscle fibers. I kept this same side assignment for every session.

A3.2 Skin preparation, securing, and impedance

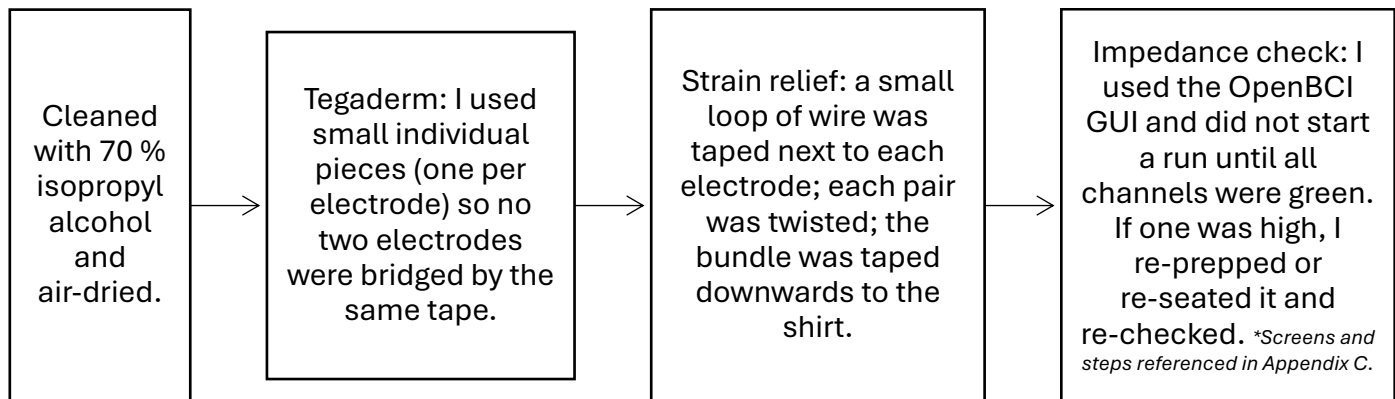
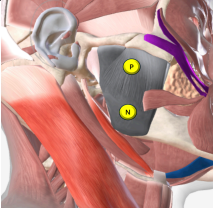

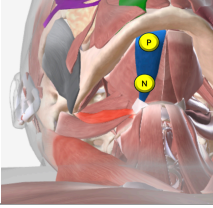


Figure A3: Skin preparation Steps.

A3.3 Trial readiness checks (before data)

For every new session I ran 3 short test trials (smile, pucker, jaw-clench, swallow) while watching the live traces. If a channel did not respond as expected, I moved that pair slightly and repeated the check. Only then did I move to the fixed trial protocol in Appendix C.

A4. Final montage with landmarks (what/where/why)

Muscle (abbr.)	Side	Placement	Landmarks I used (pair is 10–20 mm apart along fibers)	Why this muscle helps speech EMG
Masseter (MAS) (Ch 1)	Right		Ask to clench; place one pad ~1 cm below the cheekbone, the other ~1 cm above the mandibular angle	Jaw closing; large, reliable SNR
Zygomaticus major (ZYG) (Ch 2)	Right		Find the diagonal smile bulge from mouth corner toward cheekbone; place one pad higher along the bulge and the other lower toward the mouth	Lip-corner elevation (smile/spread) contrasts with rounded/pursed shapes
Anterior belly of digastric (ABD) (Ch 3)	Right-midline under chin		In the submental triangle: one pad closer to the chin point, the second 1–2 cm posterior toward the hyoid	Jaw opening / tongue-hyoid movement

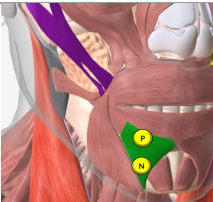
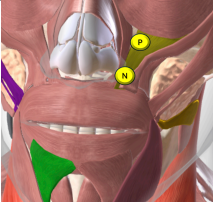

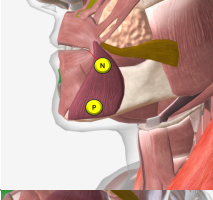
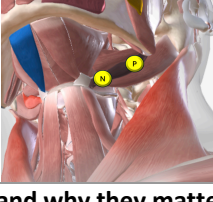
Depressor labii inferioris (DLI) (Ch 4)	Right		On lower chin, halfway between chin center and mouth corner; one just below the vermilion, the other vertically below	Lower-lip depression; helpful for certain consonants/vowels
Levator labii superioris (LLS) (Ch 5)	Left		Beside the nostril on the nasolabial fold and ~2 cm below toward upper lip	Upper-lip raise; practical alternative to LAO near the lip edge
Risorius (RIS) (Ch 6)	Left		From the mouth corner toward the ear along the horizontal smile line; one pad at the smile dimple, the other more posterior	Lateral retraction (“ee”)
Depressor Anguli Oris (DAO) (Ch 7)	Left		N about 1 cm below lip edge at mouth corner; P just above the jaw's lower edge on the same diagonal	Lip-corner depression (frown)
Stylohyoid (SLH) (Ch 8)	Left under jaw		P just below mandibular angle; N 1–2 cm anterior toward the hyoid (Adam's apple)	Hyoid/laryngeal elevation cues

Table A2: Muscles, landmarks, and why they matter.

A5. Polarity (P/N) — exactly how I wired each pair on Cyton

I followed the polarity shown in my figures in the above table (which determines the sign of the signal but not the amplitude). The key is consistency every session.

Cyton Channel	Muscle	P electrode	N electrode
Ch 1	MAS	~1 cm below cheekbone	~1 cm above mandibular angle
Ch 2	ZYG	Higher on the diagonal bulge (toward the cheekbone)	Lower toward the mouth corner
Ch 3	ABD	More anterior (closer to chin point)	1–2 cm posterior toward hyoid
Ch 4	DLI	Just below lower-lip vermilion border	Lower on the same vertical line
Ch 5	LLS	Beside nostril (upper)	Below toward upper lip
Ch 6	RIS	At smile dimple (near mouth corner)	Posterior toward ear
Ch 7	DAO	Just above jaw's lower edge	~1 cm below lip edge at mouth corner
Ch 8	SLH	Just below mandibular angle	1–2 cm toward the hyoid

Table A3: P/N assignment I used.

A5.1. Cyton wiring notes

- Each channel used true differential on the ADS1299: $INxP = P$, $INxN = N$, SRB2 OFF for those channels.
- BIAS went to the ear lobe.
- I disabled any unused channels (Like SRB) so no inputs floated.
(General board and timing settings are documented in Appendix B.)

A6. Repeatability and verification

- Same side, same distances every session; I measured from fixed landmarks (e.g., mandibular angle, cheekbone edge, nostril, mouth corner).
- Photo templates: I kept a front and two oblique photos with labels from a “good” session and matched them each time.
- Strain-relief loops, twisted pairs, taped bundle to the shirt to prevent motion artefact.
- Session gate: no recording until impedance = green on all channels; do 3 test trials and visually confirm that ZYG spikes on smile, MAS on clench, RIS on “ee,” ABD/SLH on swallow, etc. If not, re-seat and re-check.

Appendix B. Cyton configuration and acquisition parameters

B1. Scope

This appendix records the board-level settings I used to collect silent-speech sEMG with an OpenBCI Cyton (8-channel). Electrode positions are in Appendix A. Trial timing, randomization, and the saved 2.0-second window are in Appendix C. The files produced here are the inputs to Appendix D (preprocessing) and Appendix E (training).

B2. Hardware and firmware summary

- **Board:** OpenBCI Cyton (8-channel)
- **Dongle/Port:** COM3
- **Firmware / RFduino:** v3.1.2
- **Power: Battery** (I tried to start each session with a full charge)
- **Host PC / OS:** Windows 10 Home
- **Streaming/SDK:** BrainFlow 5.18 (Python)

B3. Connection and startup

1. I connected the Cyton and created a BrainFlow board object.
2. I called `prepare_session()` and then applied the same channel settings to channels 1–8.
3. I started the stream with `start_stream()` and, if needed, I also enabled a multicast monitor for debugging.

The channel-config command I send per channel is:

`x<ch>03010X` (explained in B5).

In code I kept `SRB2=False` for all channels.

B4. Sampling and saved window

- Saved window per trial: 2.0 s covering mind-preview + speak (from the trial UI). This is explained in Appendix C.
- Actual sampling rate: 250 Hz (recorded at session start from Cyton/BrainFlow). Each trial file therefore contains 500 samples per channel (250 Hz × 2.0 s).
- In the collection UI, a constant like `SR=1000` was used only for timers/plotting; it was not the acquisition rate.

B5. Channel configuration codes (what “x<ch>03010X” means)

The six digits after the channel number follow the OpenBCI ADS1299 mapping. For each channel 1–8, I send:

Digit	Value	Meaning (plain language)	What I used
-------	-------	--------------------------	-------------

1	0	Channel power state	ON
2	3	PGA gain code	6× gain
3	0	Input type	Normal (differential)
4	1	BIAS include	Included
5	0	SRB2 include	OFF
6	0	SRB1 include	OFF

This kept every channel in true differential mode with BIAS engaged and SRB disabled, which matched how I wired P/N pairs in Appendix A.

B6. Reference, bias, and grounding

- **SRB (shared reference bus):** Disabled for all channels.
- **BIAS:** On.
- **Reference / ground sites:** Same as Appendix A (ear-lobe BIAS; true differential P/N on each muscle). This matched my wiring note “SRB2 OFF” in Appendix A.

B7. Channel map used in this project (for clarity)

I kept the Cyton channel numbers aligned with the eight muscles I described in **Appendix A**.

- **Ch 1:** Masseter (MAS)
- **Ch 2:** Zygomaticus major (ZYG)
- **Ch 3:** Anterior belly of digastric (ABD)
- **Ch 4:** Depressor labii inferioris (DLI)
- **Ch 5:** Levator labii superioris (LLS)
- **Ch 6:** Risorius (RIS)
- **Ch 7:** Depressor anguli oris (DAO)
- **Ch 8:** Stylohyoid (SLH)

The **polarity (P/N)** for each pair is exactly as listed in **Appendix A Table A3**; I keep it the same every session. New channel numbers were tested during ablation for electrode reduction.

B8. Files I save during acquisition

- **Channels saved:** I used BrainFlow’s EEG channel list for Cyton and treated them as sEMG channels in the rest of the pipeline.
- **One file per trial:** data/<SubjectID>/silent/<WORD>_<rep>.npz
- **Array shape:** (samples × channels) with 2.0 s of data. These files were the inputs to Appendix D (preprocessing), which read them, filtered them (20–120 Hz with 60/120 Hz notches), cropped to a fixed length, and exported FIF and NPZ for training (Appendix E).

B9. Quick pre-run checklist

1. **Battery** charged and cables strain-relieved.

2. **Board connects** on COM3; streaming starts cleanly.
3. **Sampling rate recorded:** 250 Hz in the session log.
4. **SRB disabled** and **BIAS noted** in the GUI.
5. **Channel config string verified:** x1–8 03010X (SRB2 = 0).
6. **10-second baseline** looks stable on the live plot (no saturation).
7. **Impedance green** on all channels; if not, I re-seat and re-check.
8. **Three test trials** run and saved; overlays look as expected (MAS on clench, ZYG on smile, etc.), then I proceed to the fixed trial protocol in Appendix C.

Appendix C. Data Collection Protocol (Silent-Speech sEMG)

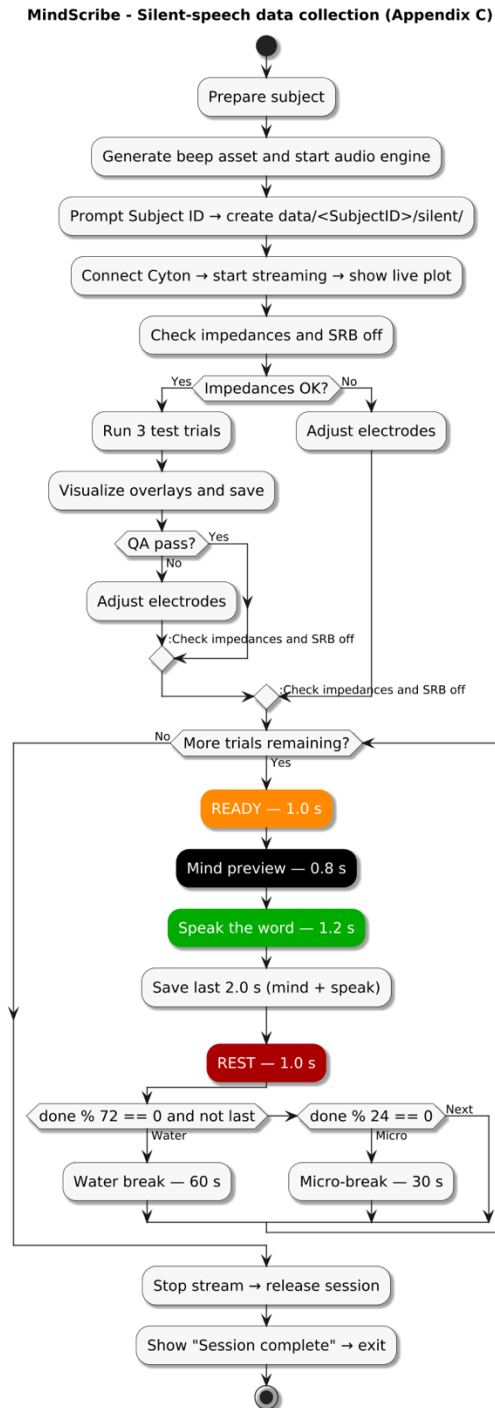


Figure C1. Session flowchart.

End-to-end pipeline for silent-speech data collection: prepare subject → initialize session → per-trial loop (READY → mind preview → speak → save → REST) with scheduled breaks → finish. Colored boxes mark the cue screens: READY (orange), mind preview (black), speak (green), and REST (red).

1. Scope and rationale.

I collected surface electrical activity from facial and jaw muscles while a participant silently mouthed short words. I moved to silent-only trials after pilot runs showed that including vocal or whisper modes could let a model rely on sound rather than muscle patterns. The vocabulary contained eight items—Yes, No, Help, Pain, Up, Down, Hmmm, Blank - with 75 repetitions per word (600 trials). “Blank” served as a silence/control class. The full session sequence is summarized in Figure C1.

2. Participant preparation.

Before any software steps, I prepared the participant: obtained informed consent, seated them comfortably, cleaned and dried the skin, placed eight electrode pairs plus bias according to the montage in Appendix A, secured leads for strain relief, and performed a quick signal check.

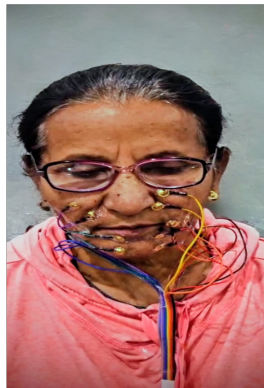


Figure C2. Subject preparation (example).

Example showing eight facial electrodes plus Bias prior to recording. Electrode locations follow the bio registered montage in Appendix 1. Photo published with consent.

3. Session initialization.

I generated a 0.15-second 880 Hz beep asset, started the audio engine (pygame.mixer at 44.1 kHz, 16-bit mono, buffer = 512), and loaded the beep. The program then prompted for a Subject ID and created the folder data/<SubjectID>/silent/. I connected the Cyton (OpenBCI), began streaming, and opened a live, multi-channel scrolling plot to monitor signal quality (board configuration is in Appendix B).

3. Pre-flight quality check (impedance + three test trials)

For every session, after connecting the Cyton I verified channel quality before initiating the data collection program. I first checked impedance on all eight channels in the OpenBCI GUI and set SRB off as in Appendix B. If any channel was out of range, I adjusted the electrode and re-checked impedance until all channels were acceptable.



Figure C3. Impedance check (example).

OpenBCI GUI display used to verify channel impedances before data collection.

I then collected three short test trials and used a small visualizer to confirm that the expected activity appeared in the saved 2.0-second window. The visualizer loaded the .npy files for a chosen word, applied the same filters as the OpenBCI GUI (using the exported JSON settings), added a notch and band-pass, then rectified the signal and overlaid all repetitions by channel to show consistency. I saved these overlays for the session record. If the overlay suggested poor contact or drift, I adjusted electrodes and repeated the three test trials until the overlay looked stable.

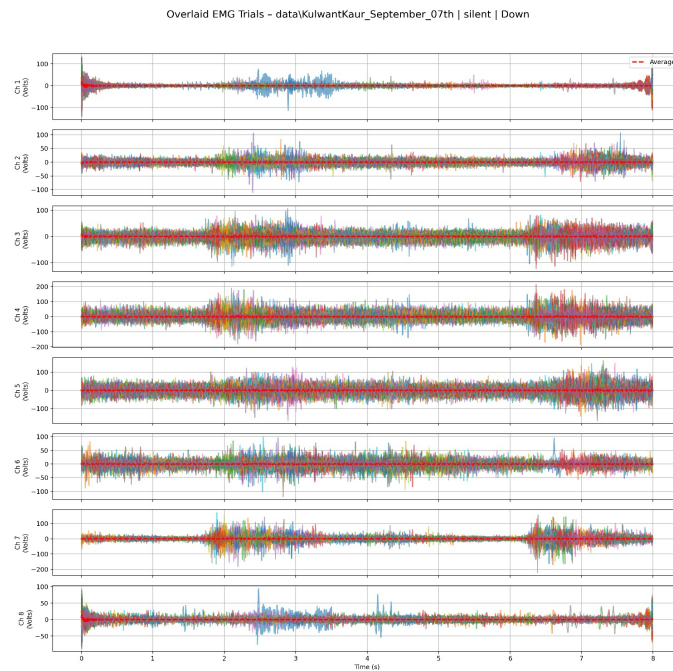


Figure C4. Three-trial overlay (example).

Overlaid traces across channels for a chosen word; used to confirm that activity is captured in the 2.0-second save window.

4. Trial procedure and timing (fixed for every trial).



Figure C5. Cue screens and dashboard (example).

Representative UI states for one trial: READY (beep + countdown), mind preview, speak (silent mouthed), and REST. The right panel displays a live multi-channel plot for signal quality.

1. READY – 1.0 s (orange). A short beep plays at the start; a large countdown is visible.
2. Mind preview – 0.8 s (black). The screen shows “Here’s the word. read in your mind” and the word.
3. Speak the word – 1.2 s (green). The participant silently mouths the word.
4. Save window. Immediately after the green period, the program saves the last 2.0 seconds, which span mind preview + speak (step 2 and step 3)
5. REST – 1.0 s (red).

4.1 Why was this window saved.

The mind preview contains preparation and small pre-movement activity; the green period contains the silent articulation. Saving only this 2.0 s window avoids unrelated activity from READY and REST and gives a clean, same-length example for every trial.

4.2 Randomization and balance.

For each repetition block, I shuffled the eight words so the participant could not learn a fixed order. Across 75 blocks, this yielded 600 balanced trials. Final counts across runs are reported in Acquisition yield, interruption, and targeted follow-ups on the following page.

4.3 Breaks and comfort.

To reduce fatigue and maintain stable signals, I scheduled micro-breaks of 30 s every 24 trials and water breaks of 60 s every 72 trials. Water breaks took priority when both coincided. A large Pause/Resume button allowed pausing countdowns at any point and resuming from the exact remaining time. This strategy allowed the participant to take a break (whether that be to go to the washroom or drink water) if needed.

4.4 Data saved per trial.

For each trial the program wrote one NumPy file to data/<SubjectID>/silent/<WORD>_<rep>.npz. Files contain the 2.0-second segment as samples \times channels. A “Recent” list in the UI showed the last few trials; if movement or noise was visible on the live plot, I could double-click to redo the trial, which re-queued it and marked it in the list.

5. Acquisition yield, interruption, and targeted follow-ups

For clarity, I use trial to mean one word presentation (a 2.0-second saved segment) and run to mean a pass that attempts 75 trials per word across all eight words.

- **Planned acquisition:** As mentioned above, I planned 8 runs (75 trials/word/run), for a target of 600 trials per word.
- **Interruption:** During run 8, the Cyton battery discharged. I completed 7 full runs ($75 \times 7 = 525$ trials per word) and salvaged 43 trials per word from run 8 before shutdown, giving an initial yield of 568 trials per word for all eight words.
- **Targeted remediation:** After training for a first pass, a confusion-matrix review showed the five most confused classes were Pain, Help, No, Down, Up. I scheduled two targeted runs, each adding 75 trials per selected word. These follow-ups increased those five classes by 150 trials each (from 568 to 718), while the remaining three classes stayed at 568. The flowchart in Figure C1 is unchanged because the targeted runs used the same trial procedure and save window.

Word	Trials
Pain	718
Help	718
No	718
Down	718
Up	718
Yes	568
Hmm	568
Blank (Silence)	568

Table C1. Final per-class counts (trials).

- **Total dataset size:** 5,294 trials ($718 \times 5 + 568 \times 3$). Because five classes are larger, class weighting or sampling can be used during model training if needed.

6. Reproducibility notes.

The program enforces identical cue durations and a consistent save window; the beep timing and the break rules are fixed; the word order is randomized every repetition block. Configuration details for electrode placement and the Cyton (including sampling rate and channel map) are referenced in Appendix A and Appendix B.

Appendix D: Preprocessing and Export (Step 1)

D1: Purpose

My first goal was to filter the signals, crop each trial to a fixed length, package each session as an MNE epochs file (-epo.fif), and also export one combined compressed file (.npz) that I could train on later. The recording protocol and save window came from Appendix C.

D2. Inputs and outputs

- **Input folders:** a list of session directories, each with a silent/ sub-folder full of per-trial .npy files, one file per trial.
- **Input arrays:** each .npy contained an $8 \times T$ array (8 channels, T samples). If an array was $T \times 8$, I transposed it; anything not 8-channel was skipped.
- **Outputs I wrote:**
 1. **One -epo.fif per session** in processed_step1/ (if that session had at least one valid trial).
 2. **One combined training file:** processed_step1/single_subject_{number_of_words}words.npz containing:
X (shape = $n_epochs \times 8 \times target_len$), y (0..K-1), session_id, class_names, session_names, sfreq, target_len.
 3. **One summary:** processed_step1/summary.json with counts and metadata.

The scripts that implement these exports are saved with the project.

D2. Classes were dynamic

I chose classes per run. At the top of Step 1 I set ALLOWED_WORDS to the subset I wanted to keep for that run (order matters). The code normalized the list (lowercased, de-duplicated, blanks became "silence"), validated it against the full vocabulary, and built CLASS_NAMES and CLASS_TO_IDX from it. Here is a small excerpt:

```
ALL_WORDS = ["pain", "help", "yes", "no", "up", "down", "silence", "hmmm"]

# I changed this per experiment (4, 5, 6, 7, or 8 classes)
ALLOWED_WORDS = ["pain", "yes", "no", "silence", "hmmm"] # example

CLASS_NAMES = _prepare_allowed_words(ALLOWED_WORDS)
CLASS_TO_IDX = {name: i for i, name in enumerate(CLASS_NAMES)}
```

So, in different preprocessing runs I included different class sets (4–8 words), depending on the ablation or the evaluation I was running next. The session-wise FIF files and the combined NPZ only contained the trials for the classes I allowed in that pass.

D3. Target length I used

Before I processed any single session, I scanned all candidate trials across the listed sessions (only those whose labels were in ALLOWED_WORDS) and measured their lengths. I then chose one fixed window length as the 90th percentile of those lengths, with a minimum of 256 samples. 90% of trials were at or below target_len, so after cropping/padding every trial had a consistent length used by the model later.

In code:

```
q90 = int(np.quantile(lengths, 0.90))
target_len = max(256, q90)
```

D4. What I did to each trial

For every .npy file in a session's silent/ sub-folder I:

1. Read the label from the filename (e.g., "yes_023.npy" → "yes"). If the label was not in my ALLOWED_WORDS for that run, I skipped it.

```
def stem_from_filename(path):
    base = os.path.splitext(os.path.basename(path))[0]
    stem = base.rsplit("_", 1)[0] if "_" in base else base
    return stem.strip().lower() or "silence"
```

2. Enforced shape to $8 \times T$ (transpose if needed; anything not 8-channel was skipped).

```
def ensure_8xT(a):
    a = np.asarray(a)
    if a.ndim != 2: raise ValueError
    if a.shape[0] == 8: return a
    if a.shape[1] == 8: return a.T
    raise ValueError
```

3. Filtered the signal with a 20–120 Hz band-pass and notches at 60 and 120 Hz (North American mains + harmonic):

```
def bandpass_notch(x, sfreq):
    x = mne.filter.filter_data(x, sfreq, l_freq=20, h_freq=120,
    verbose=False)
    x = mne.filter.notch_filter(x, sfreq, freqs=[60, 120], verbose=False)
```

```
return x
```

4. Cropped to the fixed window:

- If the label was silence, I used a center crop.
- For words, I found the highest-energy segment of length `target_len` and cut there.

The “energy-peak” crop was simple math in plain words:

- I took the absolute value of each channel and averaged across channels to form a single envelope over time:

$$\text{envelope}(t) = \frac{1}{C} \sum_{c=1}^C |x_c(t)|$$

- I smoothed that envelope with a short moving average (~40 ms).
- I slid a window of length `target_len` along the smoothed envelope, summed the values in each window, and took the start with the largest sum.

In Code:

```
def energy_peak_crop(x, target_len, sfreq):
    env = np.mean(np.abs(x), axis=0)
    env = uniform_filter1d(env, size=max(3, int(0.04*sfreq)),
mode="reflect")
    score = np.convolve(env, np.ones(target_len, np.float32),
mode="valid")
    start = int(np.argmax(score))
    return x[:, start:start+target_len]
```

D5. Packaging each session and the combined dataset

- For each session with at least one valid trial, I built an MNE EpochsArray with 8 channels named EMG1..EMG8, stored events with a small 100 ms gap between epochs, and saved it as <SessionName>-epo.fif in `processed_step1/`. The per-session `event_id` included only the classes present in that session; the integer codes in the events still aligned with my global label indices (`y + 1`).
- After I processed all sessions, I concatenated all trials into arrays `X` (`n_epochs × 8 × target_len`), `y` (`0..K-1`), and `session_id` (the index of the session in the list I passed in). I saved these to one compressed NPZ along with `class_names`, `session_names`, `sfreq`, and `target_len`, and I wrote a `summary.json` with per-session and total class counts.

All of this is implemented in the Step 1 exporter.

D6. Features I used later during training (added here for transparency)

Although Step 1 saved filtered, cropped raw EMG, I produced three streams per channel at training time (Step 2 and Step 3): raw, rectified-smoothed envelope, and RMS, then I stacked them along the channel dimension. That turned 8 channels into 24 channels (3×8). The function I used was:

```
def stack_features(arr, sfreq, env_ms=20, rms_ms=20):
    # arr: (C, T) → return (3*C, T): [raw, envelope, RMS]
    win_env = max(3, int((env_ms/1000.0)*sfreq))
    env = uniform_filter1d(np.abs(arr), size=win_env, axis=1,
mode="reflect")
    win_rms = max(3, int((rms_ms/1000.0)*sfreq))
    ms = uniform_filter1d(arr*arr, size=win_rms, axis=1, mode="reflect")
    rms = np.sqrt(ms + 1e-12)
    return np.concatenate([arr, env, rms], axis=0).astype(np.float32)
```

Math for the two derived signals:

- Envelope: a moving average of $|x|$ over a short window (≈ 20 ms).
- RMS: a moving root-mean-square over the same window, $\text{RMS}(t) = \sqrt{\text{avg}_{\text{window}}(x^2(t))}$.

These features were computed on-the-fly in Step 2 right before the model saw the data (and again in Step 3 for channel ablations). I included the snippet here because it is a key part of the pipeline even though it runs after Step 1.

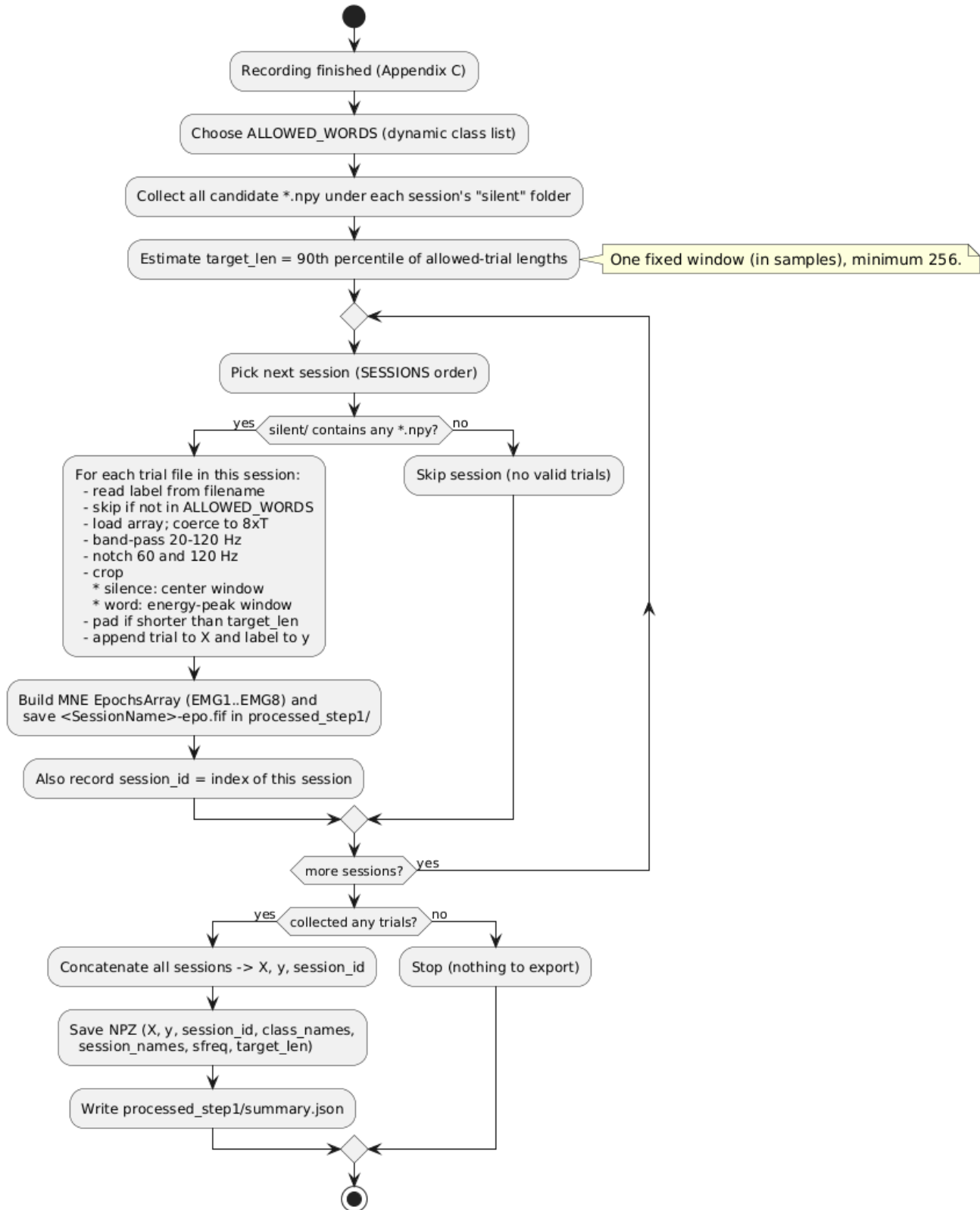
Normalization: I applied per-session mean/std normalization (z-scoring) in Step 2 before building these features.

D7. Guard-rails and edge cases I handled

- **Channel count:** I strictly required **8 channels** at preprocessing time. Any trial not 8-channel was skipped; any session with zero valid trials simply did not contribute to the NPZ (but its name still appeared in session_names so the indexing stayed stable).
- **Power frequency:** I used 60/120 Hz notches.
- **Padding:** Trials shorter than target_len were zero-padded symmetrically.

D8. End-to-end flow

Step 1 - Preprocess & Export (single participant)



Appendix E: Training & Re-training After Each Session (Step 2)

E1. What I loaded and why

After preprocessing (Appendix D / Step 1), I loaded a single NPZ that contains everything needed to train:

- X (trials \times 8 channels \times time), y (labels), session_id, session_names, class_names, sfreq, target_len.
- Step 1 keeps the order of sessions stable and sets session_id equal to each session's index in session_names. This makes my by-session splits reproducible.

I pointed my trainer at the NPZ and print the session map so I could pick splits by index:

```
data = np.load(npz_path, allow_pickle=True)
X, y = data["X"], data["y"].astype(int)
session_id = data["session_id"].astype(int)
session_names = [str(s) for s in data["session_names"]]
print({i:n for i,n in enumerate(session_names)})
```

Here, I read the arrays and confirmed which numbered session is which named folder so the split I chose was clear and repeatable. I trained many times. Each time I recorded a new session (Appendix C) I rebuilt the NPZ and re-trained. I also re-trained after ablation (Step 3) when testing 4-channel and 3-channel variants.

E2. How I split by session (and why the split changed across runs)

I evaluated generalization across days, so I hold out full sessions. In the sample run shown later (for the sample metric), I used:

- Train: sessions 0, 1, 2, 3, 5, 7, 8
- Validation: session 6
- Test: session 4

I saved this mapping to split.json for each run. When I was checking something different (a fresh recording, or a top-4 channel set), I changed the held-out session and re-trained with the same recipe.

```
TRAIN_SESS=[0,1,2,3,5,7,8]; VAL_SESS=[6]; TEST_SESS=[4]
train_idx = np.where(np.isin(session_id, TRAIN_SESS))[0]
val_idx    = np.where(np.isin(session_id, VAL_SESS))[0]
test_idx   = np.where(np.isin(session_id, TEST_SESS))[0]
```

Example, in the above code I picked the whole sessions for train/val/test and gathered all their trial indices. This mimics “train on some days, test on a new day.”

E3. Normalization I apply before training (per-session, per-channel)

I standardized signals inside each session so slow day-to-day shifts don't confuse the model.

```
def per_session_normalize_inplace(X, session_id, eps=1e-6):
    for s in np.unique(session_id):
        Xs = X[session_id==s]          # all trials from day s
        mu = Xs.mean(axis=(0,2), keepdims=True)
        sd = Xs.std(axis=(0,2), keepdims=True)
        X[session_id==s] = (Xs - mu) / (sd + eps)
```

Therefore, for each day s and channel c , I computed a mean and a standard deviation over all trials and time from that day and used:

$$z = (value - mean) / (std + tiny_number)$$

so, each day lands on a comparable scale.

E4. Features I build (same logic for 8-ch, 4-ch, 3-ch)

My inputs from Step 1 are already filtered and cropped. For each trial, I made three views of every channel and stacked them:

1. **Raw** EMG
2. **Envelope** (absolute value then short smoothing)
3. **RMS** (square → smooth → square-root)

That gave 24 feature channels for 8 electrodes (8×3), 12 for 4 electrodes, or 9 for 3 electrodes.

```
def stack_features(arr, sfreq, env_ms=20, rms_ms=20):
    env = uniform_filter1d(np.abs(arr), size=int((env_ms/1000)*sfreq),
axis=1, mode="reflect")
    ms = uniform_filter1d(arr*arr, size=int((rms_ms/1000)*sfreq),
axis=1, mode="reflect")
    rms = np.sqrt(ms + 1e-12)
    return np.concatenate([arr, env, rms], axis=0).astype(np.float32)
```

I kept the raw signal and added two smooth “energy” signals so the model sees both fast detail and slower muscle activation. The same function worked unchanged for 8-, 4-, or 3-channel inputs; only the width changed.

E5. The model and the training loop I used

Model. I used a compact ATCNet because it handles temporal patterns well and stays fast. For 8 channels it sees $24 \times T$ inputs; for 4 channels, $12 \times T$; for 3 channels, $9 \times T$.

Loss and targets. I used cross-entropy with small label smoothing (0.05), which means the target for the true class is slightly less than 1 and the rest get a tiny share; this reduces over-confidence without changing the label.

Optimizer & schedule. AdamW with a short warm-up followed by a cosine learning-rate decay. I also enabled SWA near the end for stability.

Light augmentation. I apply a tiny time-shift, a little Gaussian noise proportional to each channel's own variation, and a small cut-out segment. This improves robustness without changing the meaning of the EMG.

Class balance while training. Because some words had extra targeted follow-ups (Appendix C, Table C1), I let the loss re-weight itself using the training accuracy per class from the previous epoch: if one class is doing worse, its weight rises a bit. I smooth these weights with an EMA, so they change gently.

```
# after each epoch:
y_true_tr = np.concatenate(train_targets_all)
y_pred_tr = np.concatenate(train_preds_all)
new_w, ema = dyn_weights_from_acc(y_true_tr, y_pred_tr, n_classes,
                                   prev_ema=dyn_err_ema, ema_beta=0.7,
                                   min_w=0.5, max_w=4.0)
criterion.weight = torch.tensor(new_w, device=device)
```

I computed per-class training accuracy. The weight for a class is based on

```
"error = 1 - accuracy" (smoothed over time),
```

then normalized so weights average ≈ 1 and clipped to a safe range. Harder classes got a nudge.

I saved the best model by validation loss, an SWA model, learning-curve plots, confusion matrices, and a metrics_summary.json with mean \pm std across seeds. I repeated this for the top-4 / top-3 retrains after ablation.

E6. Re-training when data changed

- **After each new session:** I ran Step 1 again, so the NPZ included the new recording (the session order stays consistent), then I ran Step 2 with the same recipe. Sometimes I kept the same test session to track progress; other times I changed the held-out session to probe generalization. Every run writes a fresh split.json and metrics_summary.json.
- **After ablation:** Step 3 found a greedy ordering of channels on the test split, then I retrained Step 2 on the top-4 (and sometimes top-3) channels using the exact same feature stack (raw + envelope + RMS).

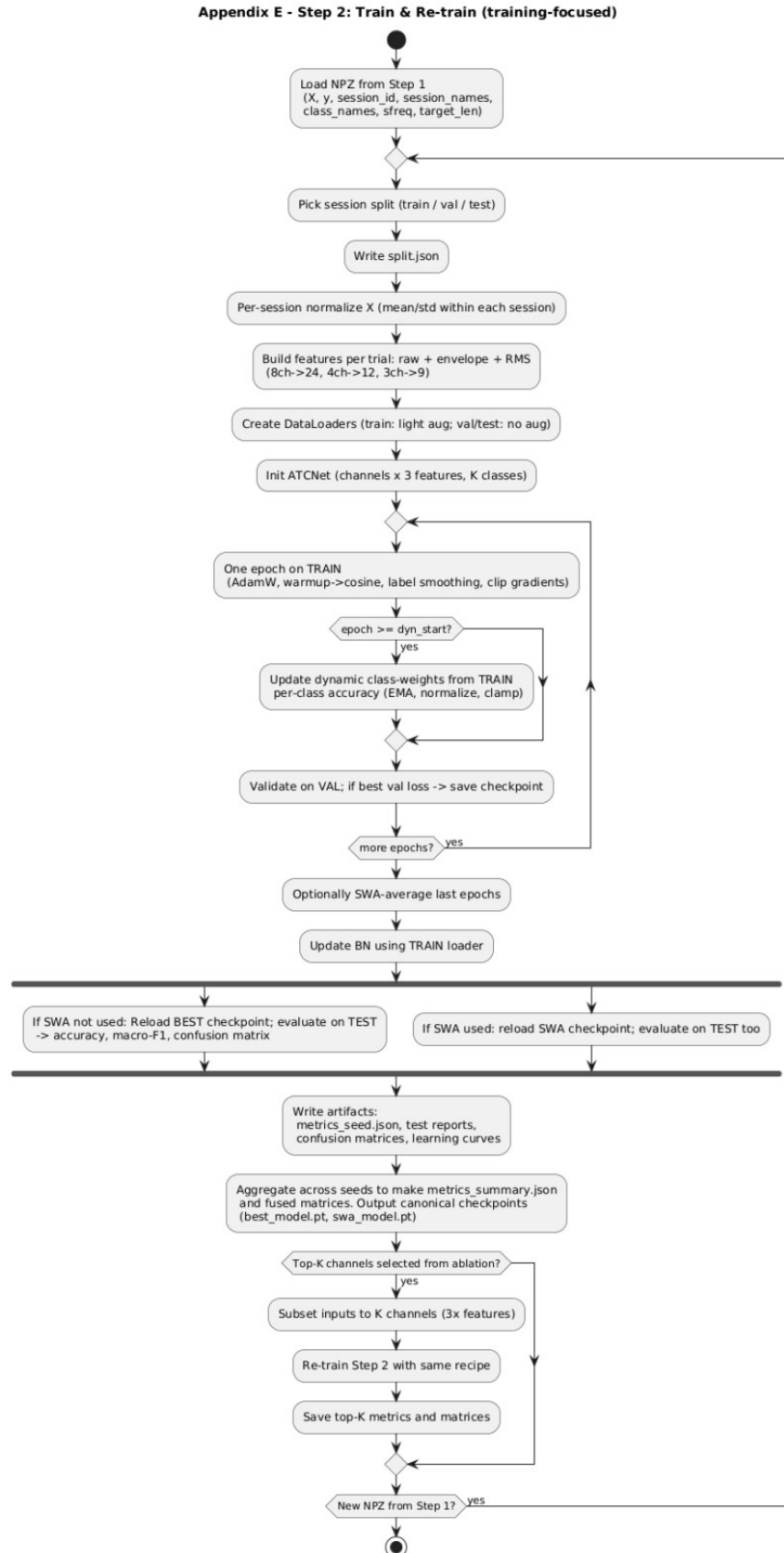
```
# for top-4 retrain
top_channels = [2,5,7,0] # example indices
ds = FeatureDataset(X[idxs], y[idxs], sfreq,
                    channels=top_channels, mask_full=False,
                    augment_train=True)
a = stack_features(a, sfreq) # still raw+env+RMS → 12×T
```

I sliced to the selected electrodes *before* feature stacking, so the model really learned from fewer inputs; nothing else in the recipe changed.

E7. One sample summary metric (many more are in the results appendix)

Because I ran this pipeline many times with different splits and channel counts, I show just one concrete example here and move the full table of runs (including {yes, no, pain, silence, hmmm} tests) to a separate appendix.

```
Classes in this specific sample run (K=5): help, yes, no, silence, hmmm
(Step1's ALLOWED_WORDS chose this set for the NPZ used in this run.)
Split: train=[0,1,2,3,5,7,8], val=[6], test=[4].
Result (mean across 3 seeds): Accuracy≈0.870 (±0.005), Macro-F1≈0.870
(±0.005).
(These numbers and the exact session names are recorded in
metrics_summary.json.)
```

E8. Flowchart of my *train* → *re-train* loop (PlantUML)

Appendix F: Channel Ablation (Step 3): Finding the Best Electrodes

F1. What I wanted

I wanted to rank the 8 EMG channels and see how much accuracy I could keep if I used only the best 3–5 channels. I kept everything else the same. For example, I used the same per-session normalization, and same features (raw + envelope + RMS). (Those are in Appendices D and E.)

F2. Inputs I used

- The NPZ I exported in Step 1 (trials \times 8 \times time, labels, session IDs, class names, sampling rate, target length).
- The trained Step 2 checkpoint (BEST or SWA) and split.json so I could reuse the same train/val/test sessions.
- The 8-muscle montage from my placement appendix (ZYG, MAS, DLI, ABD, LLS, RIS, DAO, SLH).

F3. What the ablation produced for me (files)

When I ran the ablation script, it wrote:

- ablation/ordering.json – the best \rightarrow worst channel order on the test sessions.
- ablation/ablation_deltas.csv and ablation_deltas_matrix.csv – for every round, accuracy and Delta accuracy for each candidate channel.
- Two quick figures: cumulative accuracy vs #channels and marginal Delta per added channel. I used these files to choose Top-3/4/5 sets for retraining.

F4. How the ablation worked (plain words)

1. Load the same NPZ (Produced by Step 1) and the trained model and split.json file (Produced by step 2)
2. Normalize per session exactly as in training.
3. Mask-and-test on the held-out test sessions:
 - Round 0 (baseline): zero all channels \rightarrow accuracy near chance.
 - Round 1 (single-channel scan): test each channel alone; pick the best.
 - Next rounds: with the current set of channels S , try adding every remaining channel; add the one that gives the largest Delta accuracy on the test set.
4. Save the order and the round tables, plus the plots.
5. Use the order to pick Top-K ($K=3, 4$, or 5) for retraining (reported later).

Mask evaluation on TEST (no retraining, just masking):

```
ds = FeatureDataset(X_norm[test_idx], y[test_idx], sfreq,
                    channels=S, mask_full=True, augment_train=False)
# model is the Step-2 checkpoint
probs = model(ds) # softmax inside helper
acc    = (probs.argmax(1) == y[test_idx]).mean()
```

Greedy round:

```
best_c, best_delta = None, -1.0
for c in remaining:
    acc = Acc(S | {c})      # evaluate with mask on TEST
    delta = acc - Acc(S)
    if delta > best_delta:
        best_c, best_delta = c, delta
S.add(best_c)              # grow the set one channel at a time
```

(These mirror `eval_with_mask` and `greedy_channel_ordering` in my Step 3 script.)

F5. The math I used

- Per-session z-score (same as Step 2): for day s , channel c ,

$$z_{s,c}(t) = \frac{x_{s,c}(t) - \mu_{s,c}}{\sigma_{s,c} + \epsilon}.$$

- Greedy choice at round k :

$$c^* = \arg \max_{c \notin S_{k-1}} [\text{Acc}(S_{k-1} \cup \{c\}) - \text{Acc}(S_{k-1})], S_k = S_{k-1} \cup \{c^*\}.$$

- Features per channel before the model saw them (as in Step 2): raw, envelope (smoothed $|x|$), RMS $\sqrt{\text{MA}(x^2)}$. I stacked these so the model saw $3 \times K$ streams for K channels.

F6. What I actually ran (iterations I did)

I ran this code many times:

- Different vocabularies (3-, 5-, 6-, 7-, 8-word sets).
- Different K (Top-3, Top-4, Top-5).
- Across seeds (1, 2, 3) and the same test session split I used for Step 2. The Step-2 baseline for the 5-class run was $\sim 0.870 \pm 0.005$ accuracy, and the ablation started from that trained checkpoint.

F7. What I saw

When I tallied the Top-2/Top-3/Top-4/Top-5 appearances across vocabularies, Zygomaticus (ZYG) and Risorius (RIS) showed up most often; Depressor labii inferioris (DLI) and Levator labii superioris (LLS) were strong second-tier. Masseter (MAS), Anterior belly of digastric (ABD), Depressor anguli oris (DAO), and Stylohyoid (SLH) helped in fewer cases. This matched what the greedy rounds showed and guided my Top-3/Top-4 picks. (Muscle names/locations are in Appendix A.)

How I used this: For each K (3, 4, 5), I took the first K channels in the greedy order as my Top-K set. Then I retrained new models on only those channels and compared actual Top-K accuracy to the predicted accuracy from the mask-only curve. I also recorded more sessions with those Top-K channels and checked if results stayed stable. I put the retrain + extra-session checks in Appendix G.

F8. Guardrails I kept

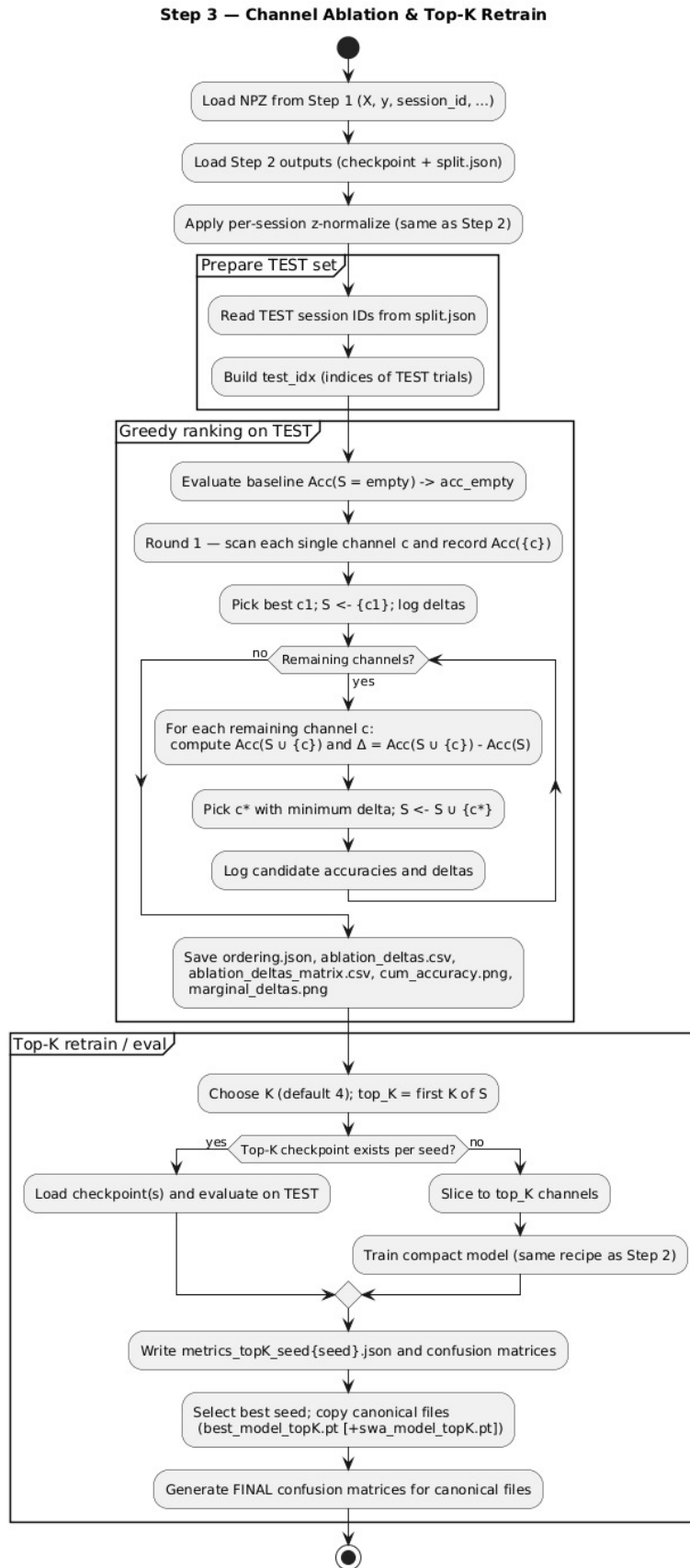
- I never changed the train/val/test sessions during ablation; I always used the Step-2 split.json.
- I always applied the same per-session normalization before any evaluation.
- I used masking to score candidate sets (so the model stayed the same), and only after choosing a Top-K did I re-train a K-channel model (Step-2 recipe) to confirm. (Those retrain results are summarized in Appendix G.)

F9. How to run what I ran (example command)

```
python step3_channel_finder.py \  
  --npz processed_step1/single_subject_5sessions.npz \  
  --model_dir outputs_step2_single_subject_5sessions \  
  --ckpt auto \  
  --out_dir outputs_step3_channels_single_subject_5sessions
```

This command created the ordering/CSV/plots. I repeated the run with different allowed words (from Step 1), and then I retrained Top-3/4/5 models (details in Appendix G).

F10. Step-by-Step Flowchart



Appendix G: Key Results

G1. Purpose:

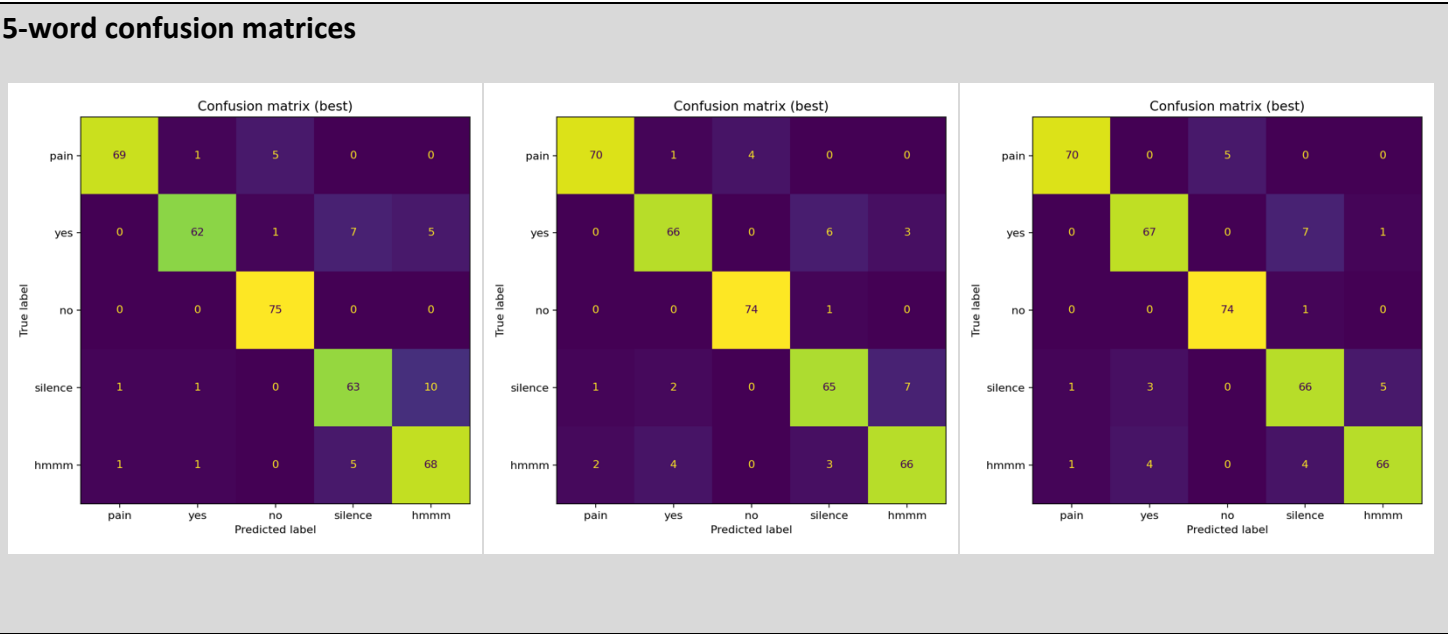
In this appendix I present the results of my silent-speech EMG model across four vocabularies (5, 6, 7, and 8 words). I trained and tested across different days, exactly the way I described in Appendix E, and I kept the same preprocessing and model, so the comparisons were fair. Then I highlighted which words the model heard most clearly and which ones it confused. Finally, I tested how accuracy changed when I kept only the best 2–5 electrodes.

Where the channels came from. I used eight bipolar pairs on the face and under the chin:
1 MAS, 2 ZYG, 3 ABD, 4 DLI, 5 LLS, 6 RIS, 7 DAO, 8 SLH (same map as Appendix A/B).

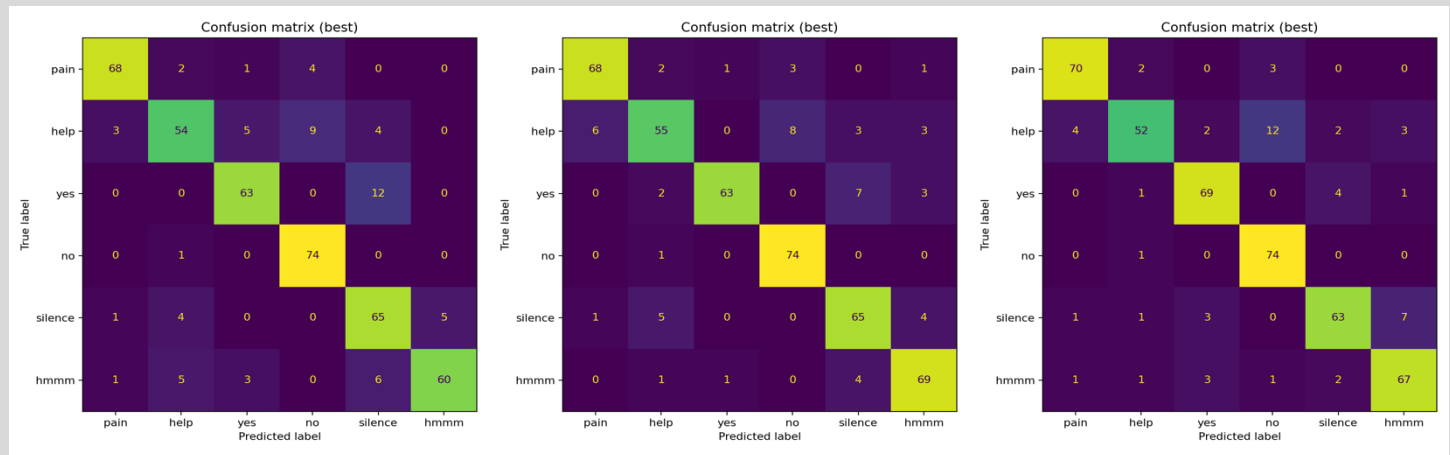
G2. Accuracy at full 8-channel input.

As I increased the vocabulary, the test accuracy decreased in a smooth way. With five words the model reached **90.0%**, and with the full eight-word set it reached **74.6%**. The seven- and six-word sets landed in between. These were cross-session test results using the same split recipe I used in Step 2.

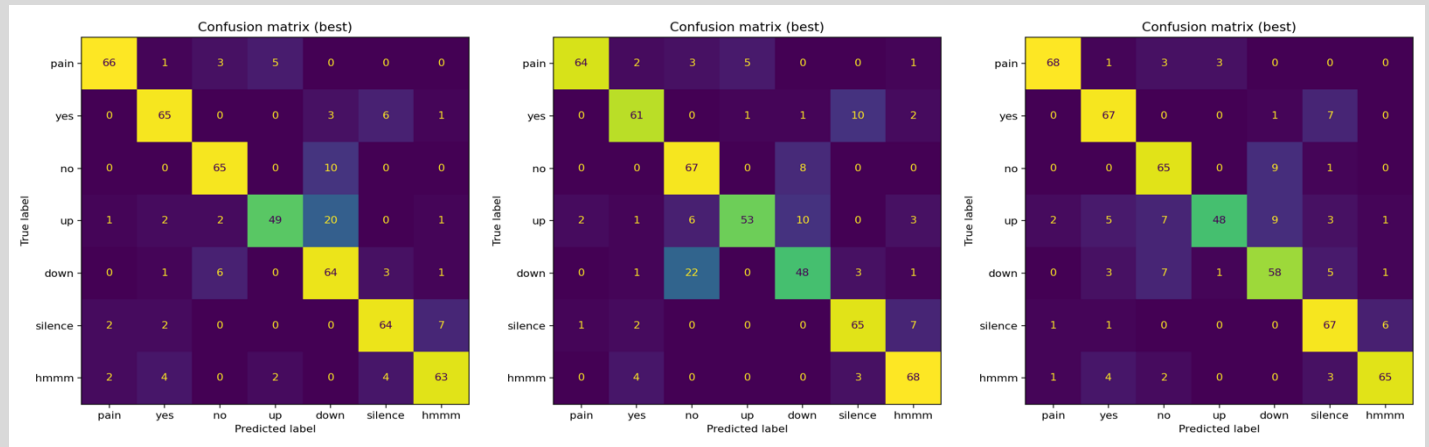
- **5 words:** 90.0%
- **6 words:** 86.0%
- **7 words:** 82.0%
- **8 words:** 74.6%



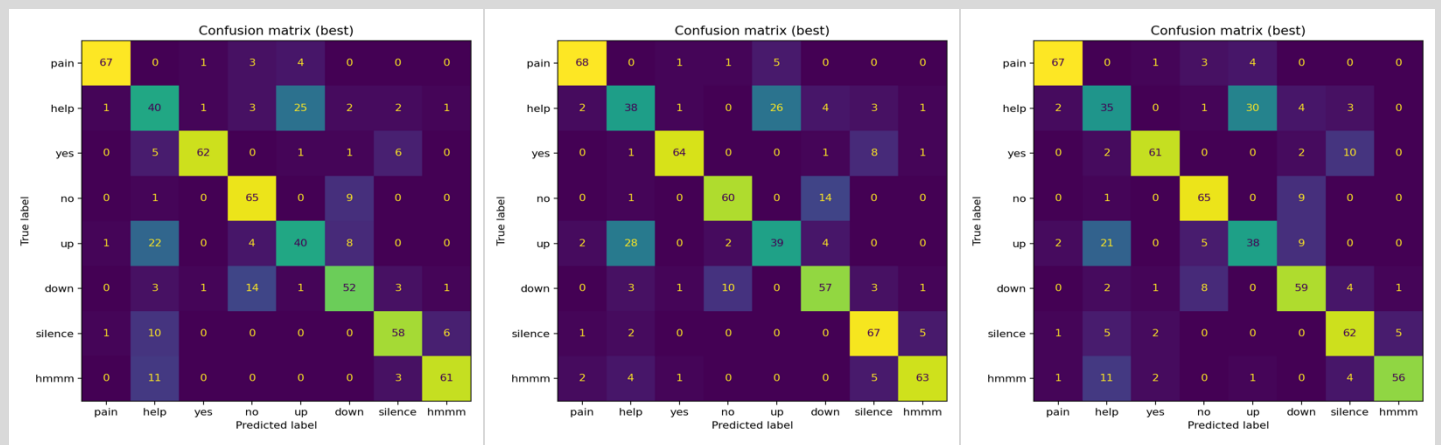
6-word confusion matrices



7-word confusion matrices



8-word confusion matrices



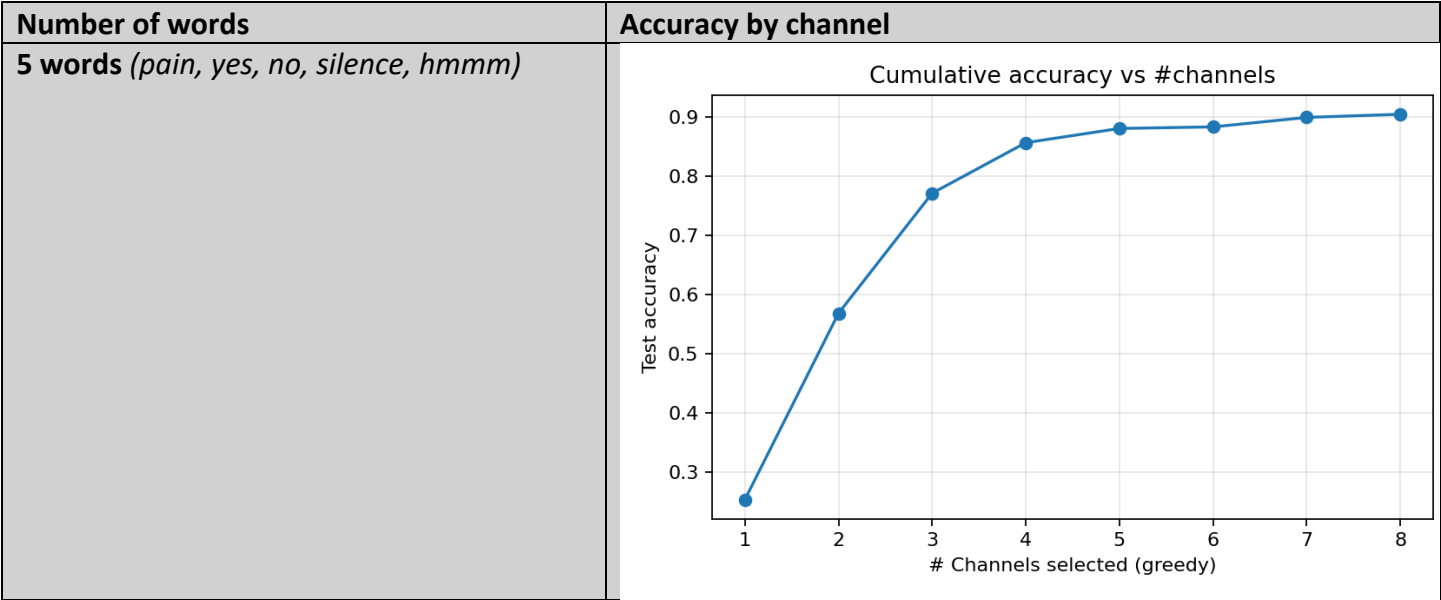
G3. Which electrodes mattered most (greedy order).

I ranked channels with a simple, repeatable greedy test: I kept adding the next channel that helped the most on the held-out test day, while everything else stayed the same. I used the same features and normalization as training (raw + envelope + RMS; per-session z-score). This “mask-only” pass did not retrain the model; it only measured which inputs carried the information. I used this order to pick Top-k sets later.

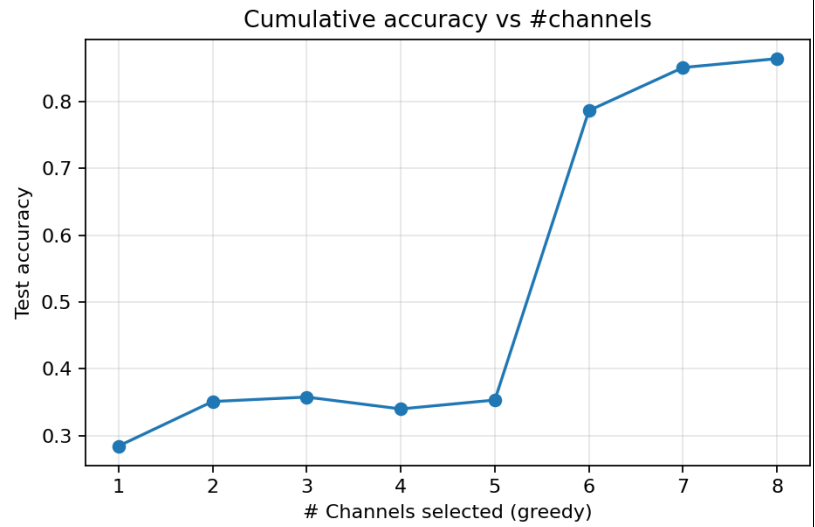
Vocab (classes)	Test accuracy (8ch)	Greedy order (first = most important)	Mask-only Top-k accuracy	
5 words (pain, yes, no, silence, hmmm)	90.0%	2-ZYG, 6-RIS, 5-LLS, 7-DAO, 8-SLH, 3-ABD, 4-DLI, 1-MAS	k=2: 55.7% k=4: 81.6%	k=3: 70.1%
6 words (pain, help, yes, no, silence, hmmm)	86.0%	4-DLI, 7-DAO, 8-SLH, 5-LLS, 6-RIS, 2-ZYG, 3-ABD, 1-MAS	k=2: 48.8% k=4: 70.6%	k=3: 58.9%
7 words (pain, yes, no, up, down, silence, hmmm)	82.0%	7-DAO, 5-LLS, 6-RIS, 2-ZYG, 1-MAS, 4-DLI, 3-ABD, 8-SLH	k=2: 37.7% k=4: 53.3%	k=3: 44.3% k=5: 58.6%
8 words (pain, help, yes, no, up, down, silence, hmmm)	74.6%	2-ZYG, 6-RIS, 5-LLS, 4-DLI, 3-ABD, 1-MAS, 7-DAO, 8-SLH	k=2: 50.0% k=4: 52.5%	k=3: 54.1% k=5: 53.3%

Table G1. Summary by vocabulary (test day). All numbers and greedy orders came from my ablation runs and results package.

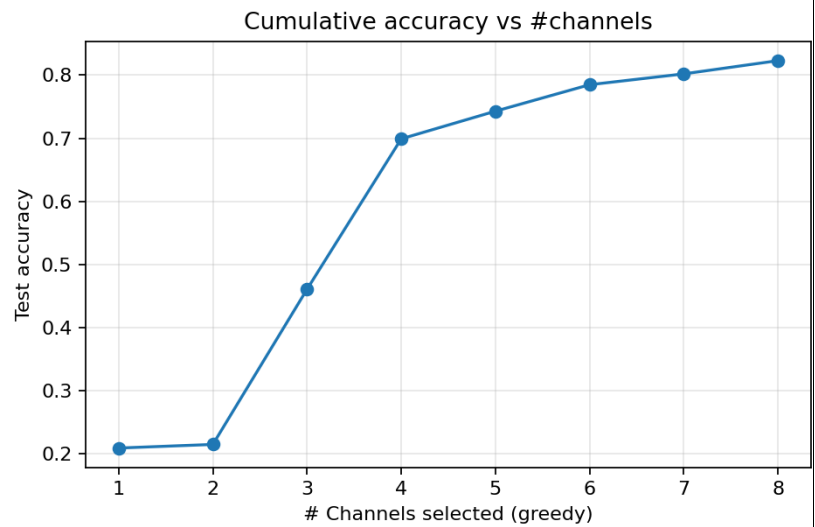
Below is a table depicting the predicted accuracy of the model based on the number of words and channels.



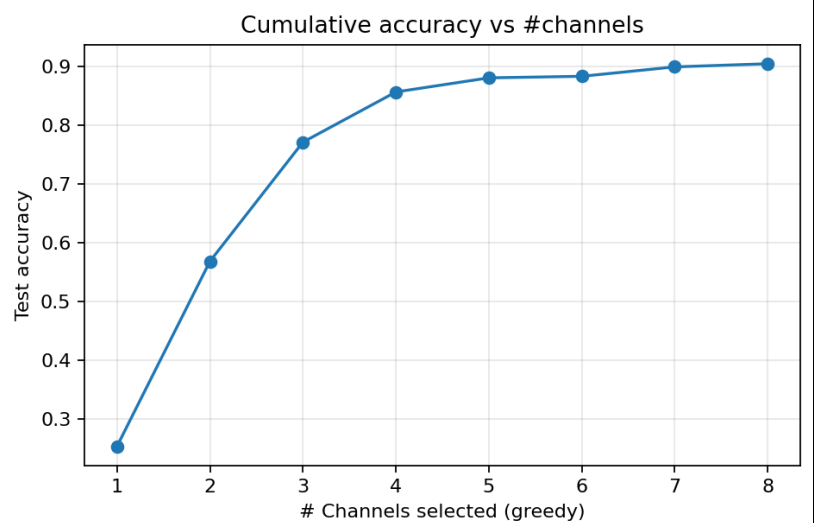
6 words (*pain, help, yes, no, silence, hmmm*)



7 words (*pain, yes, no, up, down, silence, hmmm*)



8 words (*pain, help, yes, no, up, down, silence, hmmm*)



G4. What these orders meant for practical packs.

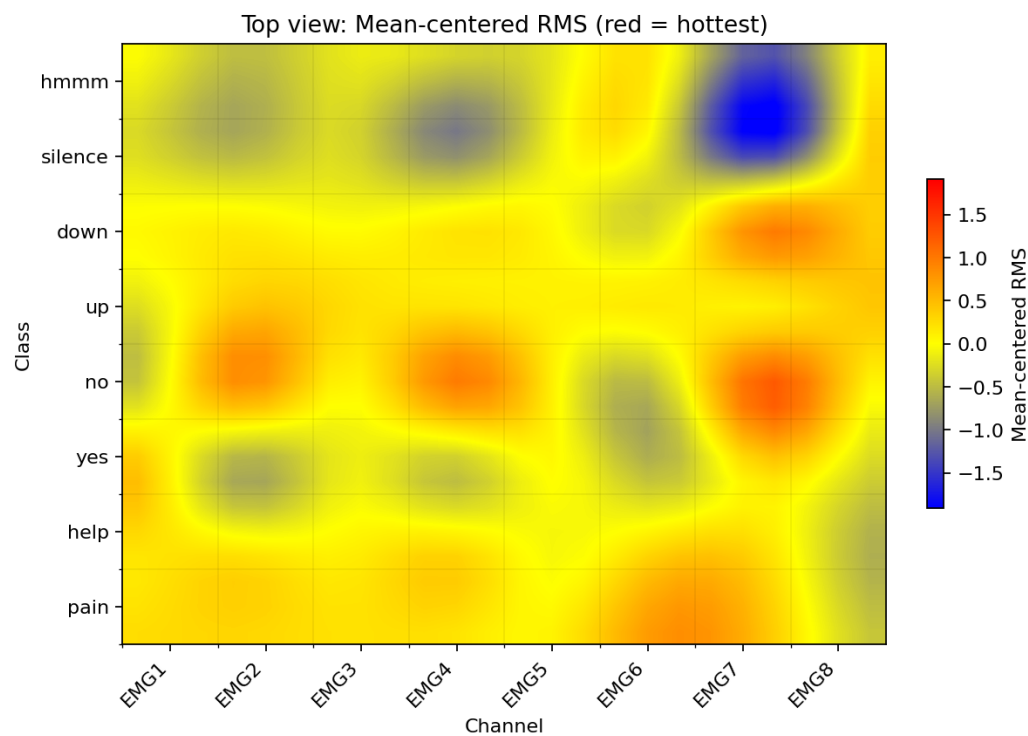
I wanted a small, portable set that still worked across vocabularies. I counted how often each muscle showed up near the top and I kept seeing the same pattern: Levator labii superioris (Ch 5) showed up the most. Zygomaticus (Ch 2), Risorius (Ch 6) and Depressor Anguli Oris (Ch 4) formed a strong second tier.

K	Proposed pack (muscles)	Why I chose it
3	LLS, ZYG, RIS	LLS was the most reliable across runs. RIS and ZYG were the next most common channels. I chose RIS and ZYG above the DAO because the DAO appeared more often as a top-4 or top-5 muscle, while RIS and ZYG almost always were top-2 or top-3
4	LLS, ZYG, RIS, DAO	The analysis from the ablation showed that these were the top 4 channels across all vocabulary sizes.
5	LLS, ZYG, RIS, DAO, DLI	The analysis showed that these are the top 5 channels across all vocabularies. However, it is important to note that the gains become much smaller after 4 channels.

Table G2. Small packs I recommended (from the greedy order).

G5. Which words were the clearest (and which were hard).

I read the confusion matrices by row (true class) and checked recall. Across all sets, no, pain, and hmmm were the clearest. When present, help and up were the hardest. The figure below shows the activation level of each channel at each word. Red indicates above average activation, while blue indicates below average activation.



G6. How I ran everything

I used the same 2.0 s window, the same filtering (20–120 Hz with 60/120 Hz notches), the same feature stack (raw + envelope + RMS), and the same cross-day split that I wrote down in my training appendix. I normalized

per session, I used a compact temporal CNN (ATCNet), and I saved the split and metrics for every run. I kept the hardware and wiring the same for every session (true differential; BIAS on the earlobe; SRB off).

G7. Conclusion

I showed that the model stayed strong as I grew the vocabulary (90% → 86% → 82% → 74.6%). I also showed that I could keep most of the performance with 3–4 electrodes. The LLS muscle was the most informative, and RIS and ZYG completed a very good Top-3 pack. The clearest words were pain, no, and yes; the hardest were help and up.

Vocab	8-ch accuracy	Greedy order (EMG# → muscle)	Mask-only Top-k accuracy	Clearest words (average recall)
5-word	90.0%	2-ZYG, 6-RIS, 5-LLS, 7-DAO, 8-SLH, 3-ABD, 4-DLI, 1-MAS	k=2: 55.7% · k=3: 70.1% · k=4: 81.6%	No 99.1%, pain 92.9%, hmmm 88.9%
6-word	86.0%	4-DLI, 7-DAO, 8-SLH, 5-LLS, 6-RIS, 2-ZYG, 3-ABD, 1-MAS	k=2: 48.8% · k=3: 58.9% · k=4: 70.6%	no 98.7%, pain 91.6%, hmmm 87.1%
7-word	82.0%	7-DAO, 5-LLS, 6-RIS, 2-ZYG, 1-MAS, 4-DLI, 3-ABD, 8-SLH	k=2: 37.7% · k=3: 44.3% · k=4: 53.3% · k=5: 58.6%	pain 88.0%, no 87.6%, silence 87.1%
8-word	74.6%	2-ZYG, 6-RIS, 5-LLS, 4-DLI, 3-ABD, 1-MAS, 7-DAO, 8-SLH	k=2: 50.0% · k=3: 54.1% · k=4: 52.5% · k=5: 53.3%	pain 89.8%, no 84.4%, yes 83.1%

Table G3. Full summary Numbers and orders from the results package and ablation script.

Vocab	Top-2	Top-3	Top-4	Top-5
5-word	2 ZYG, 6 RIS	2 ZYG, 6 RIS, 5 LLS	2 ZYG, 6 RIS, 5 LLS, 7 DAO	N/A
6-word	4 DLI, 7 DAO	4 DLI, 7 DAO, 8 SLH	4 DLI, 7 DAO, 8 SLH, 5 LLS	N/A
7-word	7 DAO, 5 LLS	7 DAO, 5 LLS, 6 RIS	7 DAO, 5 LLS, 6 RIS, 2 ZYG	7 DAO, 5 LLS, 6 RIS, 2 ZYG, 1 MAS
8-word	2 ZYG, 6 RIS	2 ZYG, 6 RIS, 5 LLS	2 ZYG, 6 RIS, 5 LLS, 4 DLI	2 ZYG, 6 RIS, 5 LLS, 4 DLI, 3 ABD

Table G4. Planned Top-k packs for retraining