Scaling and Distilling Transformer Models for sEMG

Anonymous authors
Paper under double-blind review

Abstract

Surface electromyography (sEMG) signals offer a promising avenue for developing innovative human-computer interfaces by providing insights into muscular activity. However, the limited volume training data and computational constraints during deployment have restricted the investigation of scaling up the model size for solving sEMG tasks. In this paper, we demonstrate that vanilla transformer models can be effectively scaled up on sEMG data and yields improved cross-user performance up to 110M parameters, greatly surpassing the model size regime investigated in other sEMG research (usually <10M parameters). While bigger models might not fit into on-device compute constraint, we show that >100M parameter models can be effectively distilled into models 50x smaller with minimal loss of performance (< 1.5% absolute). This results in efficient and expressive models suitable for complex real-time sEMG tasks in real-world environments.

1 Introduction

Recently, there has been growing interest in using surface electromyography (sEMG) in conjunction with powerful deep learning techniques to decode human muscle activity (e.g. Di Nardo et al., 2022; Gaso et al., 2021; Wimalasena et al., 2022; Buongiorno et al., 2021; Ozdemir et al., 2020). sEMG offers the potential for novel human-computer interfaces (HCIs), where user gestures or movements can serve as direct control input (CTRL-labs at Reality Labs, 2024). Advances in hardware (e.g. Lu et al., 2024; CTRL-labs at Reality Labs, 2024) have now made it feasible to reliably capture sEMG outside of a controlled clinical setting. Supported by these developments, deep learning methods have been applied to a variety of EMG tasks, including muscle activation detection (Di Nardo et al., 2022), (Wimalasena et al., 2022), gesture classification (Atzori et al., 2016), (He et al., 2018), (Zhang et al., 2023b), and speech recognition (Wand & Schmidhuber, 2016).

However, there are a few limitations in previous works. First, many introduce significant complexity such as non-vanilla deep learning architectures (Wanga et al., 2024; Zabihi et al., 2023; Putro et al., 2024; Chen et al., 2023; Zhang et al., 2022; 2023a;b; Liu et al., 2024) or additional training objectives (Dai et al., 2023; Zeng et al., 2022), which can act as a barrier for usability by the practitioners as complex algorithms are error prone. This is especially true for in-the-wild practitioners, typically neuroscience scientists who might not be as familiar with deep learning systems. This limitation may historically be due to the restricted quantity and/or diversity of available training data (Li et al., 2021), as large-scale data is often viewed as a prerequisite for applying contemporary deep-learning methods to complex tasks out-of-the-box without which it is expected that tricks should be used to achieve good performance (e.g., manual featurization techniques). A second consideration commonly unaddressed in previous works is the computational challenges associated with running an HCI in the wild. In particular, there are likely to be substantial constraints on the model size (small enough to run on an edge device) and inference time (fast enough for the system to be responsive). Finally, the ability to effectively generalize to unseen users is critical to successful deployment in real-world applications, yet evaluation on unseen users is often neglected in the existing literature.

In this paper, we take steps towards addressing these challenges:

1. We start by applying a simple convolution and transformer architecture (Schneider et al., 2019) on the emg2qwerty task (Sivakumar et al., 2024) and outperform previous CNN-based SOTA by 20% (absolute). Then, we show that the performance of the transformer can be further improved with

model scale, enabling us to improve over the SOTA performance by an additional 5% (absolute) by increasing the number of parameters from 2.2M to 109M. This differs from other works which mainly focus on small model scales (<10M) parameters(Wanga et al., 2024; Rahimian et al., 2021; Montazerin et al., 2023; Zabihi et al., 2023; Zhang et al., 2023a; Yang et al., 2024) with complex non-standard deep learning methodology (Wanga et al., 2024; Dai et al., 2023; Zabihi et al., 2023; Putro et al., 2024; Chen et al., 2023; Zhang et al., 2022; 2023a;b; Liu et al., 2024; Zeng et al., 2022).

2. We analyze simple logit model distillation (Hinton et al., 2015) across model size reduction ranging from no reduction to 180x parameter reduction. We show that training larger transformer models followed by distillation into smaller models substantially outperforms direct training of the small-sized transformer without distillation across all size reduction investigated, and that we can reduce the parameter count of the transformer model by up to 50x before seeing significant performance degradation ($< 1.5\%$ absolute).

3. The analysis presented in this work focus on performance on cross-user generalization unlike most previous works in sEMG processing which focus heavily on reporting performance on cross-session generalization from the 'seen' (during training) users, often times using the Ninapro databases as evaluation, which we think is not a representative measure of real-world performance (CTRL-labs at Reality Labs, 2024; Saponas et al., 2008; Yang et al., 2024).

Put together, these contributions provides a simple and pragmatic recipe for practitioners to apply in real-world setting. First, vanilla transformers can be scaled up for better cross-user performance up to bigger than previously reported in other works. Second, the simplest form of distillation works and can be used to bring scaled models to fit into specific compute constraints with minimal performance degradation (up to 50x).

We release the code used for training and distilling the models to make it easier for the scientific community to reproduce our results and build on top of this work.

2 Background

This section overviews the works related to our analysis, including sEMG modeling research in Section 2.1 and deep learning knowledge distillation in Section 2.2. We visually summarize the relation of closest related research in the field to our own in Table 6 along the contribution axes described in Section 1, i.e., Modeling, Distillation and Evaluation.

2.1 Surface Electromyography

The human central nervous system initiates muscular activity by transmitting an electrical impulse along a nerve bundle (Plonsey & Barr, 2007; CTRL-labs at Reality Labs, 2024). Surface electromyography (sEMG) uses external electrodes to measure these electrical action potentials as they propagate from the nerve fiber to the motor unit (Mokhlesabadifarahani & Gunjan, 2015; CTRL-labs at Reality Labs, 2024). sEMG data is noisy and non-stationary (Chowdhury et al., 2013; Cochrane-Snyman et al., 2016), making it a difficult signal modality for machine learning tasks. Better modeling sEMG has potential applicability to other electrophysiological data modalities, including EEG and EKG (Brambilla et al., 2021; Li et al., 2023; Yang et al., 2023)

Due to the difficulty of collecting EMG data, most open-source EMG datasets are small in terms of users and total recording time. Furthermore, most focus on capturing isolated movements that are relatively distinct from one another, such as the flexion of different fingers. For example, despite being some of the largest and most popular sEMG datasets, the Ninapro (Atzori et al., 2014) corpus is predominantly focused on recognizing isolated gestures and contains only 77 subjects in its largest dataset. The EPN dataset (Benalcazar et al., 2020) is much larger but still focuses on gesture recognition where relatively simple models (LSTM) seem to saturate performance on it Eddy et al. (2024), reaching 98% on cross-session generalization and 93% on cross-user generalization. Most other datasets, such as Amma et al. (2015) (5

Reference	Task	Number of participants	Model
(She et al., 2010)	Lower-limb movt.	3	SVM
(Alkan & Günay, 2012)	Upper-arm movt	Not Reported	SVM
(Atzori et al., 2016)	Hand gesture	78	CNN
(Wand & Schmidhuber, 2016)	Speech recog.	4 dev, 7 eval	DNN + HMM
(He et al., 2018)	Hand gesture	27	LSTM + MLP
(Cai et al., 2018)	Facial expr.	7	SVM
(Xia et al., 2018)	3D limb motion est.	8	CNN + RNN
(Shioji et al., 2018)	Auth., hand gesture	8	CNN
(Morikawa et al., 2018)	Authentication	6	CNN
(Ozdemir et al., 2020)	Hand gesture	30	CNN (ResNet-50)
(Rahimian et al., 2021)	Hand gesture	40	Transformer
(Gaso et al., 2021)	Myopathy, ALS det.	25	FC-DNN
(Godoy et al., 2022)	Hand gesture	10	VIT
(Di Nardo et al., 2022)	Muscle activation	18 + 30	FC-DNN
(Chen et al., 2023)	Finger joint est.	12	Transformer
(Zabihi et al., 2023)	Hand gesture	40	Transformer
(Zhang et al., 2023b)	Hand gesture	20	LSTM + Transf.
(Liu et al., 2024)	Hand gesture	50	CNN + VIT
(Rani et al., 2024)	Hand gesture	8 + Not Reported + 6	RF, KNN, LDA
(Putro et al., 2024)	Finger joint est.	5	Transformer
(Eddy et al., 2024)	Hand gesture	612	RNN
(Sivakumar et al., 2024)	Typing	108	TDS-ConvNet

Table 1: Prior work: most datasets are relatively small, and often classical ML approaches or older neural network architectures are used. When number of participants is of the format X + Y, different devices or protocols were used so that they cannot be trivially combined into one dataset. In some cases, the number of participants is not reported in the papers and we denote them as 'Not Reported'. Most of these prior works have not evaluated their models on unseen users.

subjects) and Ortiz-Catalan et al. (2013) (17 subjects), are even smaller and still primarily based on single-gesture/movement recognition. In contrast, an effective human-computer interface (HCI) should have the ability to disentangle multiple sequential gestures (that may overlap each other) and generalize across a diverse set of users.

Improving on the above, recently Sivakumar et al. (2024) released a dataset that represents a significant advancement over existing sEMG benchmarks in terms of its scale, task complexity, and real-world applicability. As described in Figure 1, the task is to predict key presses while touch typing on a keyboard using sEMG activity alone. The dataset captures dynamic typing behavior across 108 users and $1,135$ sessions totaling 346 hours of high-resolution wrist-based sEMG recordings. The naturalistic, high-dimensional output space (key pressed on a keyboard) and the larger data scale make it suitable for studying both cross-user zero-shot generalization and personalized finetuning on unseen (during training) users. For these reasons, we focus on this dataset in our experiments.

As a result of both limited data availability and the inherent challenges of the modality (e.g., noise, subject-based variance), prior works have tended to utilize classical approaches for sEMG tasks. In some, simple machine learning approaches such as support vector machines, random forests, K-nearest-neighbors or linear discriminant analysis are used (Rani et al., 2024; Atzori et al., 2014). In others, models such as CNNs (Ozdemir et al., 2020; Atzori et al., 2016) and LSTMs (He et al., 2018) have been applied, primarily for

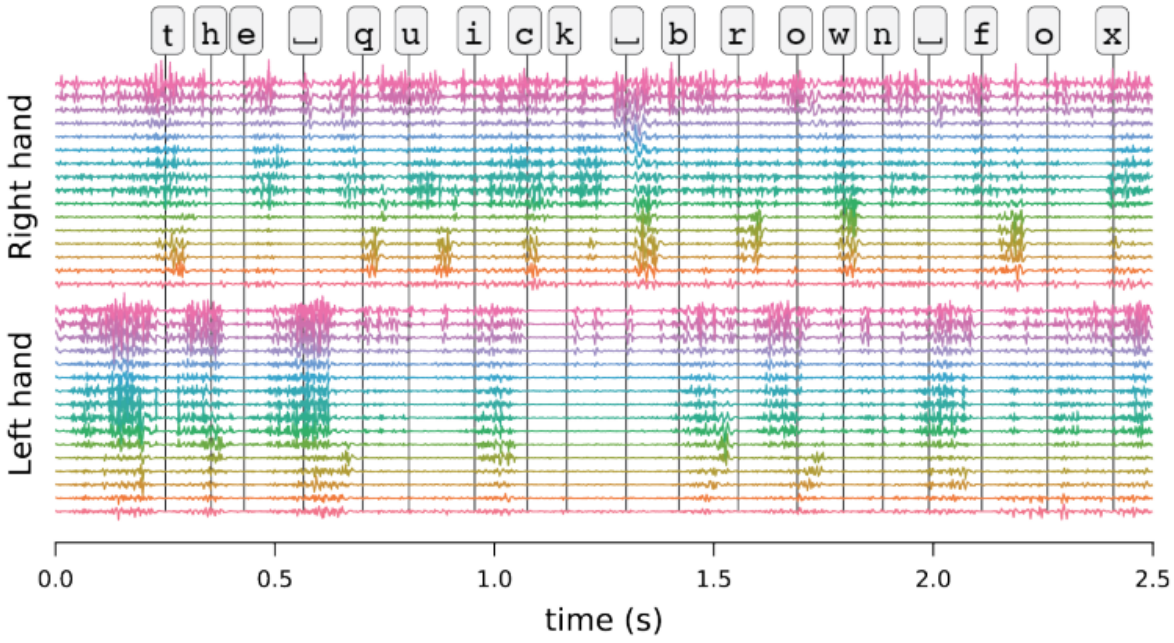


Figure 1: The *emg2qwerty* task: participants type on a keyboard while sEMG activity is recorded from both hands. The goal is to map from sequences of sEMG signals to sequences of characters. Figure cropped from `https://github.com/facebookresearch/emg2qwerty`, licensed CC BY-NC-SA.

gesture classification on a restricted set of users. Some recent work have begun to explore the application of transformer-based models (Lai et al., 2023; Wanga et al., 2024; Dai et al., 2023; Zabihi et al., 2023; Putro et al., 2024; Chen et al., 2023; Zhang et al., 2022; 2023a;b), vision transformers (ViTs) to sEMG data (Scheck & Schultz, 2023; Rahimian et al., 2021; Yang et al., 2024; Godoy et al., 2022; Montazerin et al., 2023; Liu et al., 2024), but these efforts are also mostly limited to small-scale datasets (<50 participants, many of them on a subset of Ninapro databases) and model sizes (<10M parameters). Additionally, many introduce significant complexity such as non-vanilla deep learning architectures (Wanga et al., 2024; Zabihi et al., 2023; Putro et al., 2024; Chen et al., 2023; Zhang et al., 2022; 2023a;b; Liu et al., 2024) or use manual feature extraction techniques on sliding windows of the sEMG signals, e.g., spectrograms (Sivakumar et al., 2024), multivariate power frequency features (CTRL-labs at Reality Labs, 2024), Hjorth parameters (Rani et al., 2024) and others (Zhang et al., 2022; 2023a; Putro et al., 2024). We instead focus on scaling analysis on learning vanilla transformer-based models that uses learned sEMG featurization directly from the raw data, as is popular in other modalities.

Finally, the existing literature often falls short in addressing a critical aspect of training data on sEMG data: cross-user generalization. It is well-established that sEMG signals, like other bio-signals, exhibit high inter-individual variability (Chowdhury et al., 2013; CTRL-labs at Reality Labs, 2024), making it essential to evaluate the robustness of models on "unseen" (heldout during training) users. However, many prior works neglect this crucial consideration, instead opting to test their models solely on held-out *trials* from the same individuals contained within the training set. This does not provide a meaningful assessment of the model's ability to generalize across diverse users. We focus our evaluation on new participants unseen in the training set.

Table 1 reviews recent work in the domain of sEMG decoding, showing that for the most part, datasets used are small (<100 participants, or even <10) and modeling techniques used are often classical and data-efficient.

2.2 Knowledge distillation

Knowledge distillation was popularized by (Hinton et al., 2015), who proposed training a small 'student' model on the outputs (logits) of a larger pretrained 'teacher' model, along with the ground-truth labels from the training data. This improved the performance of the smaller 'student' model compared to the case where the smaller model was trained from scratch. Subsequent works have hypothesized that distillation helps because the 'teacher' model's logits provide information about interclass relationships (Tang et al., 2020) as well as sample difficulty (Zhao et al., 2022).

Some works go beyond using outputs or logits alone for distillation, especially for deeper models. These approaches drive the 'student' model intermediate layer representations 'close' to the intermediate layer representations in the 'teacher' model (Romero et al., 2015). In practice, this is achieved by regularizing the 'distance' (e.g. ℓ_1, ℓ_2) between the activations of the 'teacher' and the 'student' model for pre-determined pairs of layers. In case of a mismatch between the shape of the layers, linear regression can be used to align the dimensions. These 'feature-based' distillation methods have been successfully used to distill large foundation models like HuBERT with minimal performance degradation (Lee et al., 2022; Peng et al., 2023; Wang et al., 2023). Komodakis & Zagoruyko (2017) proposed applying a function that maps hidden representation of a CNN to a 2-D attention map and training the 'student' model to imitate the attention map from the 'teacher' instead of the features from the 'teacher'. Chen et al. (2021) eliminates the need for ad-hoc mapping functions by learning the optimal 'student-to-teacher' mapping layer. In contrast to directly minimizing some form of distance measure, Xu et al. (2018) proposed training an adversarial discriminator network to distinguish between the representations from the 'teacher' and the 'student'.

Even more sophisticated approaches encode 'teacher' knowledge in higher-order relationships between multiple samples. For example, Tung & Mori (2019) proposed computing an inner-product-based similarity matrix between a batch of samples for both the 'teacher' and 'student' models and then training the student to match the teacher's matrix. A similar approach was presented by Park et al. (2019b), which explored both Euclidean and angular similarity metrics. However, the benefits of these relational distillation methods are marginal compared to feature-based approaches, especially considering the added computational complexity.

Some past studies began to explore the application of distillation techniques for models trained on sEMG datasets. For instance, Lai et al. (2023) distilled a ResNet model ensemble into a single model instance with good success, however their evaluation is limited to intra-user and intra-session generalization across only five users. In contrast, our work: (i) focuses on cross-user evaluation, (ii) trains on a significantly larger dataset (346 hours compared to 10 hours) and model (∼130M parameters compared to ∼6x6M parameter ensemble), (iii) distills the model extensively (up to 180x size reduction range compared to up to 10x range) which contributes to a better understanding of different regimes where the distillation works effectively and where it yields diminishing returns. Another example is work from Wanga et al. (2024), which modifies the transformer attention with depthwise-followed-by-pointwise convolutional feature projections, replaces the MLP part of the teacher model with a variational information bottleneck layer (with removed skip connection compared to vanilla MLP block of transformers) and performs simultaneous logit and feature distillation. This complexity hinders the broad applicability of the approach, which furthermore does not investigate (a) how the teacher models perform at larger scales or (b) how the distillation procedure behaves when the teacher to student parameter reduction is larger. In contrast, we show that the *simplest* distillation approach works and scales, making it pragmatically usable for in-the-wild sEMG practitioners.

Other research works have investigated cross-modal knowledge distillation in context of sEMG tasks. Notably, Dai et al. (2023) examined distillation between sEMG and hand gestures, while Zeng et al. (2022) explored distillation between sEMG and ultrasound. In contrast, our work concentrates on distilling knowledge from models trained exclusively on sEMG signals.

3 Experiments

We design experiments to demonstrate that: **i)** Transformer models can be effective for sEMG tasks even when considering datasets and models that are small by modern deep learning standards; **ii)** The performance of the transformer models further improves with scale, resulting in improvements to SOTA performance; **iii)**

Transformer models can be distilled into smaller-sized models, recovering most of the large-model performance with 50x fewer parameters. We primarily focus on zero-shot performance on held-out users, and additionally report personalization performance (where a model is fine-tuned on a small amount of a held-out single-user's data, then evaluated on held-out sessions from that user). Both cross-user generation and personalization are key challenges for real-world sEMG tasks (CTRL-labs at Reality Labs, 2024; Saponas et al., 2008; Yang et al., 2024).

3.1 Dataset

We use the *emg2qwerty* dataset (Sivakumar et al., 2024) in our experiments. The dataset consists of two-handed sEMG recordings from users typing on a computer keyboard. The data is labeled with the corresponding keystrokes, and the task is to map from sEMG sequences to character sequences. Figure 1 shows a representative example. In total, the dataset contains 346 hours of sEMG recordings across 108 unique users. The dataset is split into 100 users for training and validation and 8 held-out users for testing. For each user, we hold out 2 validation sessions and 2 testing sessions, then use the rest for training. In the generic setting, we train on the 100 user training set, validate on the 100 user validation set and evaluate on the 8 user testing set. In the personalization setting, for each of the 8 users, we train on their individual training set, then validate and test on their respective validation and testing set. Sessions are windowed to form 4 second samples, padded with an additional 900 ms of past context and 100 ms of future context.

3.2 Models

Our baseline is the Time-Depth Separable Convolutional Network (TDS-ConvNet) model introduced in Hannun et al. (2019) and used by Sivakumar et al. (2024), which reports that the parameter-efficiency of TDS-ConvNet allows for wider receptive fields which have proven important in *emg2qwerty* modeling. We use the same train, validation, and test splits as used in Sivakumar et al. (2024) and report the performance of the baseline models from that paper.

Our model architecture consists of a convolutional featurizer followed by a transformer encoder and a linear decoder. The featurizer always uses 3 convolutional layers, with instance norm applied along the time axis after the first convolutional layer, and downsamples the input sEMG data (which is sampled at 2kHz) to a sequence of features (sampled at 100 Hz). The encoder consists of a series of Transformer blocks (Vaswani et al., 2017) whose number and width we manipulate to create larger or smaller models. The transformer blocks use causal attention so that they can be used in an online streaming setup. The linear decoder converts the transformer's output to a sequence of logits. Note that unlike Sivakumar et al. (2024) which uses log-spectrograms as input features, we train our model end-to-end using raw sEMG data directly, without using any hand-designed feature engineering pipeline. Following Sivakumar et al. (2024), during training we apply channel rotation, which randomly shifts the sEMG channels by ± 1 as a data augmentation technique to simulate different spatial orientations of the device.

We have trained 20 different architectures, generated by permuting [2, 4, 6, 8, 10] layers and inner dimension of [128, 256, 512, 1024]. The ratio of the transformer inner dimension and the transformer feed-forward dimension is fixed at four. While we report on performance of all models in the supplement, for ease of exposition in the main text, we are concerned with three 'reference' architectures: the Tiny architecture consisting of 10 layers of inner dimension 128 (about 2.2M parameters); the Small architecture consisting of 6 layers of inner dimension 256 (about 5.4M parameters, close to the 5.3M TDS-ConvNet baseline); and the Large architecture consisting of 8 layers of inner dimension 1024 and about 109M parameters. We use the AdamW optimizer(Loshchilov, 2017) for training all models. In the figures we additionally include other models along the size-performance Pareto frontier (i.e. ones which perform better than any model of the same or lesser parameter count). For all the experiments, we report standard deviation across multiple seeds (6 seeds for supervised training of transformer models, 3 seeds for personalization experiments, and 3 seeds for distillation experiments).

3.2.1 Supervised training of transformer models

Following Sivakumar et al. (2024), we use the connectionist temporal classification (CTC) loss (Graves et al., 2006) to train the transformer models on the *emg2qwerty* task. We train the transformer models on a single V100 node (8 32Gb GPUs) using cosine learning-rate scheduling (Loshchilov & Hutter, 2017) and linear learning rate warmup for first 5% of updates. We document all the hyperparameters in the Appendix in Section B.

3.2.2 Model distillation

We use logit distillation to distill the pretrained LARGE teacher model into a set of smaller (but differently sized) student models. For the student models, we include all 20 architectures discussed above (including ones of the same size as the teacher). This enables us to demonstrate the benefits of distillation at varying student model sizes. The process is depicted in Figure 2.

The distillation loss $\mathcal{L}_{\text{distill}}$ is the cross-entropy loss where the student's output probabilities at each time step are expected to match the soft targets provided by the teacher's per-timestep output probabilities. We use a temperature scaling factor of 2 when computing the probabilities. The distillation loss is then combined with the task loss (i.e. CTC) to give the final training loss for the student model:

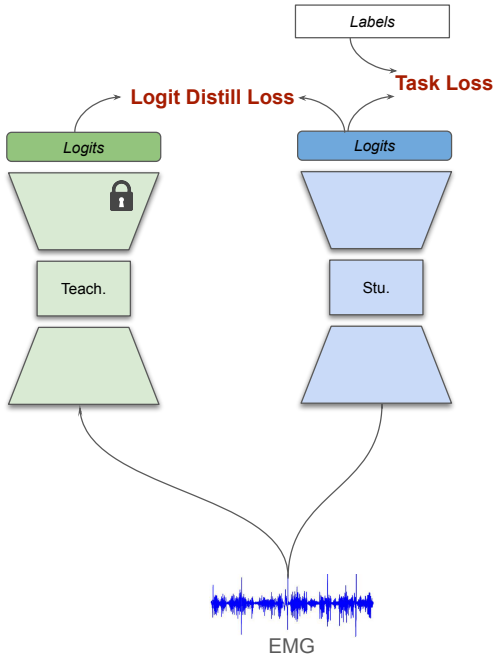


Figure 2: An illustration of the distillation process: the smaller student model receives training signal from the logits of the larger teacher model in addition to the regular supervised loss.

$$\mathcal{L} = \frac{1}{\alpha + \beta}\left(\alpha \mathcal{L}_{\text{distill}} + \beta \mathcal{L}_{\text{task}}\right). \tag{1}$$

Here, α and β are hyperparameters. We set $\beta = 1.0$ and experimented with α values in [0.1, 2] and found the best values to be in the [0.3, 0.5] range, we use $\alpha = 0.5$ throughout our distillation experiments. The rest of the distillation hyperparameters used in our experimentation are documented in Appendix B.3.

Benchmark	Architecture	Parameters	CER (↓%)
Generic	TDS-ConvNet	5.3M	55.57
	TINY TRANSFORMER	2.2M	35.9± 0.9
	SMALL TRANSFORMER	5.4M	35.2± 1.1
	LARGE TRANSFORMER	109M	30.5± 0.6
Personalized	TDS	5.3M	11.39
	TINY TRANSFORMER	2.2M	9.7± 0.13
	SMALL TRANSFORMER	5.4M	7.9± 0.06
	LARGE TRANSFORMER	109M	6.8± 0.07

Table 2: Cross-user performance of transformer models trained on the *emg2qwerty* dataset, showing that [a] even the TINY transformer models substantially outperform the TDS-ConvNet baseline in spite of having fewer parameters, and [b] the performance of the transformer model keeps improving as we increase the number of parameters in the model. For the transformer models, we report standard deviation across 6 seeds for Generic benchmark and across 3 seeds for the Personalized benchmark. The standard deviation for the baseline models is not reported in Sivakumar et al. (2024).

	Model		CER (↓%)		Abs. Gain (↑%)
Benchmark	**Architecture**	**Params**	**Standard**	**Distilled**	
Generic	Tiny	2.2M	35.9± 0.9	31.9± 0.4	4.0
	Small	5.4M	35.2± 1.1	32.7± 0.5	2.5
Personalization	Tiny	2.2M	9.7± 0.1	8.6± 0.04	1.1
	Small	5.4M	7.9± 0.06	7.1± 0.06	0.8

Table 3: Cross-user performance of small student models on the *emg2qwerty* dataset with and without distillation. Performance is measured by character error rate (CER). The 'Abs. Gain' column reflects the absolute improvement in performance from using distillation as opposed to standard supervised training for a given architecture. Personalized models are personalized from the distilled student. All models see a substantial benefit (7-11% relative improvement) from the use of the distillation loss. We report standard deviation across 3 seeds for the distillation results. Here Tiny and Small refer to Tiny Transformer and Small Transformer in Table 2 respectively.

3.3 Metrics

Following (Sivakumar et al., 2024), we evaluate the *emg2qwerty* models using Character Error Rate (CER), defined as the Levenshtein edit-distance between the predicted and the ground-truth sequence. It can be expressed as $\frac{(S+D+I)*100}{N}$ where, given the predicted and the ground-truth sequences, S is the number of character substitutions, D is the number of deletions, and I is the number of insertions between them and N is the total number of characters in the ground-truth sequence.

4 Results

4.1 Parameter-matched comparison of transformer with TDS baseline

In Table 2, we compare the performance of our transformer models with the baseline TDS-ConvNet model. On the generic (zero-shot, cross-user) benchmark, even the Tiny model outperforms the TDS-ConvNet model baseline by a margin of about 20% absolute CER, in spite of having fewer than half the number of parameters as the baseline. While the performance of the Small (which is about the same size of the baseline) is not very different from that of the Tiny model, scaling the model more aggressively to the Large size does yield a further 5% improvement, reducing baseline's CER by nearly a factor of 2 (from about 55% to about 30%).

On the personalized benchmark, where the trained model is fine-tuned on a single heldout user's data, CERs are much lower across the board and therefore the absolute gains of the transformer are more modest (1.7%, 1.8% and 4.5% CER for Tiny, Small and Large model respectively). However, the relative magnitude of improvement is similar to that seen on the generic benchmark, especially for the largest model.

These performance gains over the baseline across all model sizes and benchmark support the usage of transformer architecture for text-transcription based sEMG tasks. We provide comparison of architecture in Table 2 with other common models in the sEMG field (Time-CNN, LSTM and Vision Transformers) in Appendix C.1.

4.2 The performance of transformer models improves with scale

In Table 2, we observe that as we scale the size of the transformer model, the performance of the model continues to improve even with the same amount of data. An alternate view of this is shown in Figure 3 (red line) where we report on the 10 architectures that are on the Pareto front w.r.t. scale and performance, i.e. they perform at least as well as any model of the same or smaller size. These models show a nearly log-linear

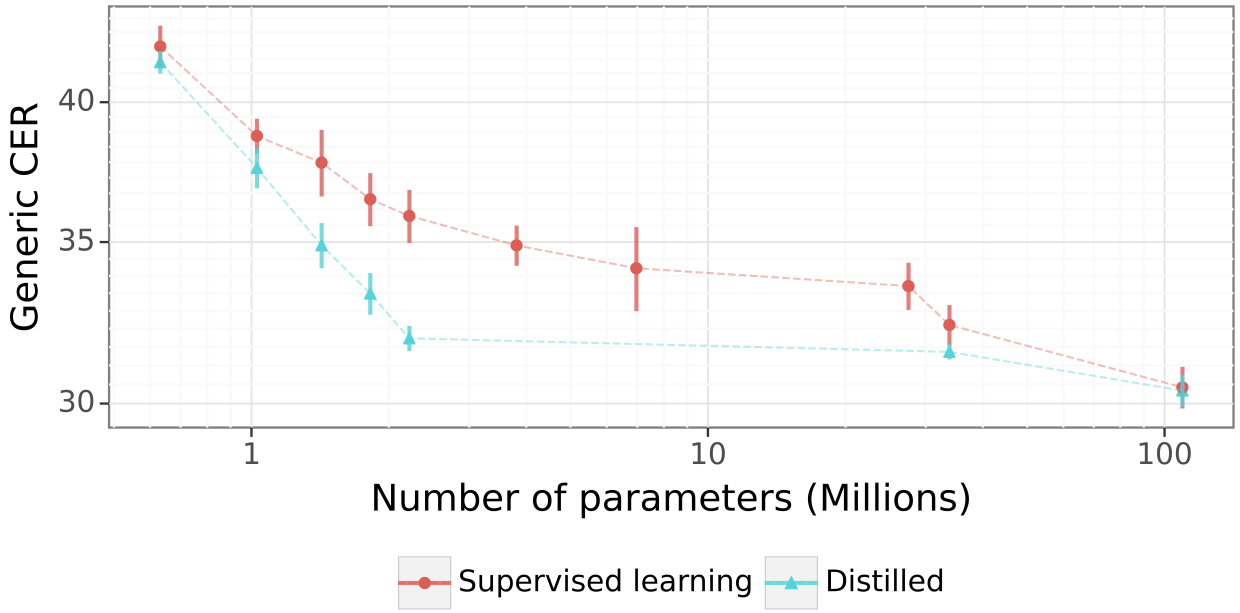


Figure 3: Scaling curve of transformers on the *emg2qwerty* dataset, showing the benefit of model size across 3 orders of magnitude from <1M to over 100M parameters, and the benefit of distillation. A few things are of note: [a] even the smallest transformer we consider here (about 600K parameters) outperforms the 5.3M parameter TDS-ConvNet baseline (55.57 CER), in spite of having almost an order of magnitude fewer parameters; and [b] the majority of distillation benefit is seen for small-but-not-too-small models, with a 2.2M parameter distilled model getting within 1.5% CER (absolute) of the top-performing model in spite of having 50x fewer parameters. Figure includes only models along the pareto front, i.e. ones which outperform others at the same or smaller parameter count. The vertical bars denote standard deviation across 6 seeds.

scaling curve, validating that there are performance benefits to be realized by using large transformer models for sEMG data.

One obvious challenge with the use of large transformer models is that this increases the computational overhead of the models during inference and any potential real-time usage with a HCI (e.g. to control a prosthetic, for computer text entry, or to control a cursor). Distillation techniques mitigate this limitation, which we investigate in the following section.

4.3 Model distillation can be applied effectively to models for sEMG

Table 3 shows the results of distilling small-capacity student models on the *emg2qwerty* dataset, using the best performing Large model as a teacher. An alternate view is in Figure 3 (turquoise line), where we see that the benefit of distillation is largest for small-but-not-too-small models. That is, for the very tiniest models, they seem unable to fully benefit from distillation (perhaps too few layers, or too few parameters to model the teacher's distribution), while for bigger models they have enough capacity to learn the task on their own. Notably, the 2.2M parameter Tiny achieves performance within 1.5% of the Large teacher in spite of having about 50x fewer parameters. We also report that while the Large teacher model takes 26.966 ms for inference over a single sample, the Small model takes 5.7002 ms only, thus providing a 4.7× speedup during inference. These numbers are obtained by running inference over a single sample a 1000 times to get an average runtime and then further repeating this protocol 3 times and reporting the median of the three numbers. Additional details around benchmarking are provided in the Appendix. While the specific size cutoff for a real-time model may vary, distillation provided a benefit across many small model sizes.

		Featurizer type	
		raw sEMG signal+CNN	Spectrogram+MLP
Encoder type	TDS	45.84± 0.06	56.78± 0.17
	Transformer	36.96± 0.95	45.15± 1.17

Table 4: Featurizer and Encoder type ablation on cross-user performance when trained from scratch without distillation signal. Performance is measured by character error rate (CER) on the *emg2qwerty* dataset. We report standard deviation across 3 seeds.

5 Analysis

In Section 4.1, we report considerable improvement of our modeling approach over results reported in Sivakumar et al. (2024). In this section we investigate which differences contributed to this performance improvement. We identify and ablate two core differences between our approach and Sivakumar et al. (2024): first the featurizer and second the encoder type. Additionally, we investigate the contribution of data augmentation techniques to the training procedure. The technical details of the details of these experiments are detailed in Appendix B.5.

Featurizer type We refer to the "featurizer" as all modeling parts between the raw sEMG signal and the core encoder architecture, i.e., the Transformer in our approach or the TDS model in the case of Sivakumar et al. (2024). A featurizer is commonly in the deep learning literature as a way to create meaningful vector representation of the data for the encoder to process. In the case of sEMG, part of this featurization procedure has historically often been handcrafted with transformation of the data that we think is useful for the task, e.g., Mean Absolute Value (MAV), Variance (VAR), Root Mean Square (RMS), spectrograms, etc. In this work we employ a data-driven way of doing featurization through a CNN architecture applied directly to the raw sEMG signal, as popularized by Schneider et al. (2019). We perform the ablation experiment between our raw sEMG signal+CNN→encoder featurization strategy to the spectrogram+MLP→encoder strategy from Sivakumar et al. (2024) with the TDS and Transformer encoders in Table 4. We see that TDS and Transformer encoder benefits 10.94 and 8.19 absolute CER improvement, respectively, from using a data-driven featurization approach.

Architecture ablation In addition to the featurization strategy, we investigate the contribution of using a transformer encoder instead of a TDS encoder. In Table 4, we show the performance of both these encoder types on both featurizer type, i.e., raw sEMG signal+CNN and Spectrogram+MLP. We see that the transformer outperforms the TDS encoder in both raw sEMG signal+CNN and Spectrogram+MLP featurizers settings by 8.88 and 11.63 CER respectively.

Data augmentation ablation We also investigate contribution of data augmentation techniques used in this work, specifically input masking, also called SpecAugment, Park et al. (2019a) as used by Sivakumar et al. (2024)), and input channel rotation. For both of these augmentations, we evaluate the Transformer with raw sEMG signal+Conv input feature type shown in Table 4 and show the results in Table 5. We see that using input masking improves performance over not using the augment, however there does not seem to be a clear optimal masking length between 7 and 15. Channel rotation augmentation doesn't seem to provide much performance improvement over not using it. We see an exception to that when using input masking with mask lengths of 15 in which case not having rotation augmentation improves performance by approximately 1.7 CER, indicating a potential interaction between the two augmentation techniques.

6 Conclusion

Our work provides a new state-of-the-art on the *emg2qwerty* task without increasing the parameter count over the baseline. We achieved this with a simple and pragmatic recipe that can easily be followed by practitioners in other task settings. First, we do this by scaling up a vanilla Transformer architecture, which

		Input masking		
		None	Length=7	Length=15
Channel rotation	None	36.24± 1.04	33.96± 1.27	33.05± 0.24
	± 1	36.92± 0.82	33.32± 0.97	34.77± 1.59

Table 5: Data augmentation ablation on cross-user performance when trained from scratch without distillation signal. We investigate the cross-product of 3 input masking and 2 channel rotation configurations. Performance is measured by character error rate (CER) on the *emg2qwerty* dataset. We report standard deviation across 3 seeds.

yielded improved performance at all scales investigated in this work (up to 110M parameters). Second, if bigger models cannot be deployed because of on-device compute constraints, we show one can apply the simplest form of knowledge distillation works to bring scaled models back into a usable regime (up to 50x reduction in size) with minimal performance degradation. Thirdly, we provide an empirical limit where this approach seems to yield diminishing returns (>50x size reduction). We perform all of the evaluations in the challenging cross-user generalization setting. We see our work as providing a pragmatic recipe applying standard and battle-tested tools to deep learning for sEMG at scale.

Broader Impact Statement

Beyond this paper, the broader usage of sEMG, and the specific development sEMG-based textual input models, pose novel ethical and societal considerations. There are numerous societal benefits for the development of sEMG models for textual input. sEMG allows one to directly interface a person's neuromotor intent with a computing device, which can be used to, for example, develop adaptive controllers for those who struggle to use existing computer interfaces.

References

Ahmet Alkan and Mücahid Günay. Identification of emg signals using discriminant analysis and svm classifier. *Expert systems with Applications*, 39(1):44–47, 2012.

Christoph Amma, Thomas Krings, Jonas Böer, and Tanja Schultz. Advancing muscle-computer interfaces with high-density electromyography. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pp. 929–938, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450331456. doi: 10.1145/2702123.2702501. URL `https://doi.org/10.1145/2702123.2702501`.

M Atzori, A Gijsberts, C Castellini, B Caputo, AG Hager, S Elsig, G Giatsidis, F Bassetto, and H Müller. Electromyography data for non-invasive naturally-controlled robotic hand prostheses. sci data. 2014; 1: 140053, 2014.

Manfredo Atzori, Matteo Cognolato, and Henning Müller. Deep learning with convolutional neural networks applied to electromyography data: A resource for the classification of movements for prosthetic hands. *Frontiers in Neurorobotics*, 10, 2016. ISSN 1662-5218. doi: 10.3389/fnbot.2016.00009. URL `https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2016.00009`.

Marco E. Benalcazar, Lorena Barona, Leonardo Valdivieso, Xavier Aguas, and Jonathan Zea. Emg-epn-612 dataset, November 2020. URL `https://doi.org/10.5281/zenodo.4421500`.

Cristina Brambilla, Ileana Pirovano, Robert Mihai Mira, Giovanna Rizzo, Alessandro Scano, and Alfonso Mastropietro. Combined use of emg and eeg techniques for neuromotor assessment in rehabilitative applications: A systematic review. *Sensors*, 21(21):7014, 2021.

Domenico Buongiorno, Giacomo Donato Cascarano, Irio De Feudis, Antonio Brunetti, Leonarda Carnimeo, Giovanni Dimauro, and Vitoantonio Bevilacqua. Deep learning for processing electromyographic signals: A taxonomy-based survey. *Neurocomputing*, 452:549–565, 2021. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.06.139. URL https://www.sciencedirect.com/science/article/pii/S0925231220319020.

Yi Cai, Yifan Guo, Haotian Jiang, and Ming-Chun Huang. Machine-learning approaches for recognizing muscle activities involved in facial expressions captured by multi-channels surface electromyogram. *Smart Health*, 5:15–25, 2018.

Defang Chen, Jian-Ping Mei, Yuan Zhang, Can Wang, Zhe Wang, Yan Feng, and Chun Chen. Cross-layer distillation with semantic calibration. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 7028–7036, 2021.

Zhouping Chen, Hong Wang, Haonan Chen, and Tao Wei. Continuous motion finger joint angle estimation utilizing hybrid semg-fmg modality driven transformer-based deep learning model. *Biomedical Signal Processing and Control*, 85:105030, 2023. ISSN 1746-8094. doi: https://doi.org/10.1016/j.bspc.2023.105030. URL https://www.sciencedirect.com/science/article/pii/S1746809423004639.

Reaz H. Chowdhury, Mamun Bin Ibne Reaz, Md. Alauddin Ali, Ahmad Abu Bakar, Kasturi Chellappan, and Teck Guan Chang. Surface electromyography signal processing and classification techniques. *Sensors (Basel)*, 13(9):12431–12466, Sep 17 2013. doi: 10.3390/s130912431.

K. C. Cochrane-Snyman, T. J. Housh, C. M. Smith, E. C. Hill, N. D. Jenkins, R. J. Schmidt, and G. O. Johnson. Inter-individual variability in the patterns of responses for electromyography and mechanomyography during cycle ergometry using an rpe-clamp model. *Eur J Appl Physiol*, 116(9):1639–1649, Sep 2016. doi: 10.1007/s00421-016-3394-y. Epub 2016 Jun 20.

CTRL-labs at Reality Labs. A generic noninvasive neuromotor interface for human-computer interaction. *bioRxiv*, 2024. doi: 10.1101/2024.02.23.581779. URL https://www.biorxiv.org/content/early/2024/07/23/2024.02.23.581779.

Qingfeng Dai, Yongkang Wong, Mohan Kankanhali, Xiangdong Li, and Weidong Geng. Improved network and training scheme for cross-trial surface electromyography (semg)-based gesture recognition. *Bioengineering*, 10(9):1101, 2023.

Francesco Di Nardo, Alessandro Nocera, Alessandro Cucchiarelli, Sandro Fioretti, and Chiara Morbidoni. Machine learning for detection of muscular activity from surface emg signals. *Sensors (Basel)*, 22(9):3393, Apr 2022. doi: 10.3390/s22093393.

Ethan Eddy, Evan Campbell, Scott Bateman, and Erik Scheme. Big data in myoelectric control: large multi-user models enable robust zero-shot emg-based discrete gesture recognition. *Frontiers in Bioengineering and Biotechnology*, 12, 2024. ISSN 2296-4185. doi: 10.3389/fbioe.2024.1463377. URL https://www.frontiersin.org/journals/bioengineering-and-biotechnology/articles/10.3389/fbioe.2024.1463377.

Mekia Shigute Gaso, Selcuk Cankurt, and Abdulhamit Subasi. Electromyography signal classification using deep learning. In *2021 16th International Conference on Electronics Computer and Computation (ICECCO)*, pp. 1–6, 2021. doi: 10.1109/ICECCO53203.2021.9663803.

Ricardo V. Godoy, Gustavo J. G. Lahr, Anany Dwivedi, Tharik J. S. Reis, Paulo H. Polegato, Marcelo Becker, Glauco A. P. Caurin, and Minas Liarokapis. Electromyography-based, robust hand motion classification employing temporal multi-channel vision transformers. *IEEE Robotics and Automation Letters*, 7(4):10200–10207, 2022. doi: 10.1109/LRA.2022.3192623.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 369–376, New York, NY, USA,

2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143891. URL https://doi.org/10.1145/1143844.1143891.

Awni Hannun, Ann Lee, Qiantong Xu, and Ronan Collobert. Sequence-to-sequence speech recognition with time-depth separable convolutions. *arXiv preprint arXiv:1904.02619*, 2019.

Yunan He, Osamu Fukuda, Nan Bu, Hiroshi Okumura, and Nobuhiko Yamaguchi. Surface emg pattern recognition using long short-term memory combined with multilayer perceptron. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 5636–5639, 2018. doi: 10.1109/EMBC.2018.8513595.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015. URL https://api.semanticscholar.org/CorpusID:7200347.

Nikos Komodakis and Sergey Zagoruyko. Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In *ICLR*, Paris, France, June 2017. URL https://enpc.hal.science/hal-01832769.

Wenqiang Lai, Qihan Yang, Ye Mao, Endong Sun, and Jiangnan Ye. Knowledge distilled ensemble model for semg-based silent speech interface. In *IEEE EUROCON 2023-20th International Conference on Smart Technologies*, pp. 117–122. IEEE, 2023.

Yeonghyeon Lee, Kangwook Jang, Jahyun Goo, Youngmoon Jung, and Hoirin Kim. Fithubert: Going thinner and deeper for knowledge distillation of speech self-supervised learning, 2022. URL https://arxiv.org/abs/2207.00555.

Haoyang Li, Hongfei Ji, Jian Yu, Jie Li, Lingjing Jin, Lingyu Liu, Zhongfei Bai, and Chen Ye. A sequential learning model with gnn for eeg-emg-based stroke rehabilitation bci. *Frontiers in Neuroscience*, 17:1125230, 2023.

Wei Li, Ping Shi, and Hongliu Yu. Gesture recognition using surface electromyography and deep learning for prostheses hand: state-of-the-art, challenges, and future. *Frontiers in neuroscience*, 15:621885, 2021.

Xiaoguang Liu, Lijian Hu, Liang Tie, Li Jun, Xiaodong Wang, and Xiuling Liu. Integration of convolutional neural network and vision transformer for gesture recognition using semg. *Biomedical Signal Processing and Control*, 98:106686, 2024. ISSN 1746-8094. doi: https://doi.org/10.1016/j.bspc.2024.106686. URL https://www.sciencedirect.com/science/article/pii/S1746809424007444.

I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

Changjia Lu, Xin Xu, Yingjie Liu, Dan Li, Yue Wang, Wenhao Xian, Changbing Chen, Baichun Wei, and Jin Tian. An embedded electromyogram signal acquisition device. *Sensors*, 24(13), 2024. ISSN 1424-8220. doi: 10.3390/s24134106. URL https://www.mdpi.com/1424-8220/24/13/4106.

Behnam Mokhlesabadifarahani and V. K. Gunjan. *EMG Signals Characterization in Three States of Contraction by Fuzzy Network and Feature Extraction*. Springer Singapore Pte. Limited, 2015th edition, 2015. doi: 10.1007/978-981-287-320-0.

Mansooreh Montazerin, Elahe Rahimian, Farnoosh Naderkhani, S Farokh Atashzar, Svetlana Yanushkevich, and Arash Mohammadi. Transformer-based hand gesture recognition from instantaneous to fused neural decomposition of high-density emg signals. *Scientific reports*, 13(1):11000, 2023.

Shion Morikawa, Shin-ichi Ito, Momoyo Ito, and Minoru Fukumi. Personal authentication by lips emg using dry electrode and cnn. In *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, pp. 180–183, 2018. doi: 10.1109/IOTAIS.2018.8600859.

Max Ortiz-Catalan, Rickard Brånemark, and Bo Håkansson. BioPatRec: A modular research platform for the control of artificial limbs based on pattern recognition algorithms. *Source Code for Biology and Medicine*, 8:11, 2013. doi: 10.1186/1751-0473-8-11.

Mehmet Akif Ozdemir, Deniz Hande Kisa, Onan Guren, Aytug Onan, and Aydin Akan. Emg based hand gesture recognition using deep learning. In *2020 Medical Technologies Congress (TIPTEKNO)*, pp. 1–4, 2020. doi: 10.1109/TIPTEKNO50054.2020.9299264.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019a.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3967–3976, 2019b.

Yifan Peng, Yui Sudo, Shakeel Muhammad, and Shinji Watanabe. Dphubert: Joint distillation and pruning of self-supervised speech models, 2023. URL `https://arxiv.org/abs/2305.17651`.

Robert Plonsey and Robert C. Barr. *Bioelectricity: A quantitative approach*. Springer, 2007.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624, 2023.

Nur Achmad Sulistyo Putro, Cries Avian, Setya Widyawan Prakosa, Muhammad Izzuddin Mahali, and Jenq-Shiou Leu. Estimating finger joint angles by surface emg signal using feature extraction and transformer-based deep learning model. *Biomedical Signal Processing and Control*, 87:105447, 2024. ISSN 1746-8094. doi: https://doi.org/10.1016/j.bspc.2023.105447. URL `https://www.sciencedirect.com/science/article/pii/S1746809423008807`.

Elahe Rahimian, Soheil Zabihi, Amir Asif, Dario Farina, S. Farokh Atashzar, and Arash Mohammadi. Temgnet: Deep transformer-based decoding of upperlimb semg for hand gestures recognition, 2021. URL `https://arxiv.org/abs/2109.12379`.

Parul Rani, Sidharth Pancholi, Vikash Shaw, Manfredo Atzori, and Sanjeev Kumar. Enhancing gesture classification using active emg band and advanced feature extraction technique. *IEEE Sensors Journal*, 24(4):5246–5255, 2024. doi: 10.1109/JSEN.2023.3344700.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *ICLR*, abs/1412.6550, 2015. URL `https://api.semanticscholar.org/CorpusID:2723173`.

T Scott Saponas, Desney S Tan, Dan Morris, and Ravin Balakrishnan. Demonstrating the feasibility of using forearm electromyography for muscle-computer interfaces. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 515–524, 2008.

Kevin Scheck and Tanja Schultz. Multi-speaker speech synthesis from electromyographic signals by soft speech unit prediction. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

Qingshan She, Zhizeng Luo, Ming Meng, and Ping Xu. Multiple kernel learning svm-based emg pattern classification for lower limb control. In *2010 11th International Conference on Control Automation Robotics & Vision*, pp. 2109–2113. IEEE, 2010.

Ryohei Shioji, Shin ichi Ito, Momoyo Ito, and Minoru Fukumi. Personal authentication and hand motion recognition based on wrist emg analysis by a convolutional neural network. *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, pp. 184–188, 2018. URL `https://api.semanticscholar.org/CorpusID:57753769`.

Viswanath Sivakumar, Jeffrey Seely, Alan Du, Sean Bittner, Adam Berenzweig, Anuoluwapo Bolarinwa, Alex Gramfort, and Michael Mandel. EMG2QWERTY: A Large Dataset with Baselines for Touch Typing using Surface Electromyography. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2024.

Jiaxi Tang, Rakesh Shivanna, Zhe Zhao, Dong Lin, Anima Singh, Ed H Chi, and Sagar Jain. Understanding and improving knowledge distillation. *arXiv preprint arXiv:2002.03532*, 2020.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1365–1374, 2019.

A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, and I Polosukhin. Attention is all you need. In *NIPS*, 2017.

Michael Wand and Jürgen Schmidhuber. Deep neural network frontend for continuous emg-based speech recognition. In *Interspeech*, pp. 3032–3036, 2016.

Yujin Wang, Changli Tang, Ziyang Ma, Zhisheng Zheng, Xie Chen, and Wei-Qiang Zhang. Exploring effective distillation of self-supervised speech models for automatic speech recognition, 2023. URL `https://arxiv.org/abs/2210.15631`.

Zefeng Wanga, Bingbing Hub, Junfeng Yaoa, and Jinsong Sua. Lightweight transformer for semg gesture recognition with feature distilled variational information bottleneck. 2024.

L. N. Wimalasena, J. F. Braun, M. R. Keshtkaran, D. Hofmann, J. Á. Gallego, C. Alessandro, M. C. Tresch, L. E. Miller, and C. Pandarinath. Estimating muscle activation from emg using deep learning-based dynamical systems models. *J Neural Eng*, 19(3), May 2022. doi: 10.1088/1741-2552/ac6369.

Peng Xia, Jie Hu, and Yinghong Peng. Emg-based estimation of limb movement using deep learning with recurrent convolutional neural networks. *Artificial Organs*, 42(5):E67–E77, 2018. doi: https://doi.org/10.1111/aor.13004. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/aor.13004`.

Zheng Xu, Yen-Chang Hsu, and Jiawei Huang. Training shallow and thin networks for acceleration via knowledge distillation with conditional adversarial networks, 2018. URL `https://arxiv.org/abs/1709.00513`.

Chia-Yen Yang, Pin-Chen Chen, and Wen-Chen Huang. Cross-domain transfer of eeg to eeg or ecg learning for cnn classification models. *Sensors*, 23(5):2458, 2023.

Jehan Yang, Maxwell Soh, Vivianna Lieu, Douglas Weber, and Zackory Erickson. Emgbench: Benchmarking out-of-distribution generalization and adaptation for electromyography. *Advances in Neural Information Processing Systems*, 37:50313–50342, 2024.

Soheil Zabihi, Elahe Rahimian, Amir Asif, and Arash Mohammadi. Trahgr: Transformer for hand gesture recognition via electromyography. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 31:4211–4224, 2023. doi: 10.1109/TNSRE.2023.3324252.

Jia Zeng, Yixuan Sheng, Yicheng Yang, Ziliang Zhou, and Honghai Liu. Cross modality knowledge distillation between a-mode ultrasound and surface electromyography. *IEEE Transactions on Instrumentation and Measurement*, 71:1–9, 2022.

Jiaxuan Zhang, Yuki Matsuda, Manato Fujimoto, Hirohiko Suwa, and Keiichi Yasumoto. Feasibility analysis of semg recognition via channel-wise transformer. In *2022 IEEE 11th Global Conference on Consumer Electronics (GCCE)*, pp. 105–106. IEEE, 2022.

Jiaxuan Zhang, Yuki Matsuda, Manato Fujimoto, Hirohiko Suwa, and Keiichi Yasumoto. Movement recognition via channel-activation-wise semg attention. *Methods*, 218:39–47, 2023a.

Wenli Zhang, Tingsong Zhao, Jianyi Zhang, and Yufei Wang. Lst-emg-net: Long short-term transformer feature fusion network for semg gesture recognition. *Frontiers in Neurorobotics*, 17, 2023b. ISSN 1662-5218. doi: 10.3389/fnbot.2023.1127338. URL https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2023.1127338.

Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 11953–11962, 2022.

Appendix

The appendix is organized as follows: In Section B we lay out all of the experimental details of the results presented in this work and in Section C we show additional results which were not presented in the main text.

A Extended related works

Table 6: Overview of related works to this paper along our contribution axes. For the model axis, we distinguish 4 subcategories: Architecture, "Vanilla" (whether it the architecture is used as-is or modified specifically for sEMG), Raw features (whether the model processes sEMG signal of a handcrafted representation of it) and the number of parameters. For the distillation axis, we distinguish between 3 subcategories: whether it does distillation at all, "Vanilla" (again, used as-is or modified specifically for sEMG) and the model size reduction achieved. Along the evaluation axis, we distinguish between 2 subcategories: Whether is it evaluating cross-user generalization and whether the core evaluation of the approach was done on Ninapro databases.

	Model				Distillation			Evaluation	
Paper	**Arch**	**Vanilla**	**Raw Features**	**#params**	**Does KD?**	**Vanilla**	**Size reduction**	**Cross-user?**	**Is Ninapro?**
Lai et al. (2023)	Resnet	YES	YES	35M (ensemble)[a]	YES	YES	[4-10]x	NO	NO
Wanga et al. (2024)	Transformer	NO	YES	<1M	YES	NO	∼7x	NO	YES
Dai et al. (2023)	Transformer	NO	YES	Not specified	YES	NO	NA	NO	YES
Zeng et al. (2022)	CNN	NO	YES	Not specified	YES	NO	2x	NO	YES
Rahimian et al. (2021)	ViT	YES	YES	<100k	NO			NO	YES
Yang et al. (2024)	ViT	YES	YES	6M	NO			YES	YES+others
Scheck & Schultz (2023)	Transformer	YES	YES	Not specified	NO			YES	NO
Godoy et al. (2022)	ViT	YES	YES	Not specified	NO			NO	YES
Montazerin et al. (2023)	ViT	YES	YES	<500k	NO			NO	NO
Zabihi et al. (2023)	Transformer	NO	YES	<1M	NO			NO	YES
Putro et al. (2024)	Transformer	NO	NO	Not specified	NO			NO	YES
Chen et al. (2023)	Transformer	NO	YES	Not specified	NO			NO	NO
Zhang et al. (2022)	Transformer	NO	NO	Not specified	NO			NO	YES
Zhang et al. (2023a)	Transformer	NO	NO	<500k	NO			NO	YES
Zhang et al. (2023b)	Transformer	NO	YES	Not specified	NO			NO	YES+others
Liu et al. (2024)	ViT	NO	YES	Not specified	NO			NO	YES+others

[a] The model size reduction is not specified by Lai et al. (2023), we infer an approximation of the size reduction from the model memory footprint (assuming FP32 weights).

B Experimental details

We describe the hyperparameters and model details in the following subsections. We provide constant hyperparameters in one table and hyperparameters that are subject to vary in a separate table. In Section B.2, we list the hyperparameters of the supervised learning models presented in this paper, i.e., without distillation signal; in Section B.3, we list the hyperparameters of the distilled models; and in Section B.4 we describe the hyperparameters of the personalization experiment.

B.1 Dataset details

In Table 7, we detail the aggregated statistics of the *emg2qwerty* dataset.

Table 7: *emg2qwerty* dataset statistics

Total subjects	108
Total sessions	1,135
Avg sessions per subject	10
Max sessions per subject	18
Min sessions per subject	1
Total duration	346.4 hours
Avg duration per subject	3.2 hours
Max duration per subject	6.5 hours
Min duration per subject	15.3 minutes
Avg duration per session	18.0 minutes
Max duration per session	47.5 minutes
Min duration per session	9.5 minutes
Avg typing rate per subject	265 keys/min
Max typing rate per subject	439 keys/min
Min typing rate per subject	130 keys/min
Total keystrokes	5,262,671

B.2 Supervised learning details

In the supervised learning experiments, we train a grid of model with sizes $[128, 256, 512, 1024]$ transformer hidden representation size and $[2, 4, 6, 8, 10]$ layers. For each of the hidden representation sizes, we set the transformer feed-forward dimension such that the feed-forward ratio ($d_{\text{ff}}/d_{\text{hidden}}$) is maintained at 4. For each of these configuration, we launch multiple learning rates ($[3e-3, 1e-3, 3e-4, 1e-4]$) across 6 different seeds. Seeding is used to determine dataloading order and model initialization. We train all models to completion (200 epochs) and evaluate the model on the training and validation at the end of every epoch. For each model, we artificially early-stop the model post-hoc by choosing the epoch with the lowest validation CER and record the test set CER of the model for this epoch. The rest of the training-related hyperparameters (specifically dropout probabilities, weight decay and learning rate schedule) were chosen according to best values in prior experimentation with a fixed model size.

We aggregate the results by first taking the validation and test CER averages across seeds, then pick the best aggregated validation learning rate value and report the average and standard deviation of the test results. The chosen hyperparameters are reported in Table 9 and the test CER reported in Table 21.

Table 8: Supervised learning task details and hyperparameters held constant for models from Table 21.

Data	Input sEMG channels	32
	Window length	8000
	Padding	$[1800, 200]$
Architecture	Featurizer channels	$[128, 64, 64]$
	Featurizer kernels	$[11, 3, 3]$
	Featurizer strides	$[5, 2, 2]$
	Encoder feed-forward ratio	4
	Encoder attentions heads	16
	Tokenizer	Character-level
	Vocab size	99
	Encoder hidden, attention and activation dropout	0.2
	Encoder feature projection dropout	0.2
	Encoder final layer dropout	0.2
Training	Epochs	200.0
	Effective batch size	640
	Encoder causal attention	True
Optimizer	Learning rate schedule	linear warmup + cosine decay
	Learning rate warmup ratio	0.05
	Weight decay	0.2
	CTC zero infinity	True
	Gradient clipping	0.1
Software	Torch version	2.3.1+cu121
	Transformers version	4.36.2

Table 9: Supervised learning task hyperparameters that vary for models from Table 21.

Architecture	Encoder hidden size	Encoder layers	Optimizer learning rate
	128	2	1e-03
	128	4	1e-03
	128	6	1e-03
	128	8	1e-03
Tiny	**128**	**10**	**1e-03**
	256	2	1e-03
	256	4	1e-03
Small	**256**	**6**	**1e-03**
	256	8	1e-03
	256	10	3e-04
	512	2	1e-03
	512	4	3e-04
	512	6	3e-04
	512	8	3e-04
	512	10	3e-04
	1024	2	1e-03
	1024	4	3e-04
	1024	6	3e-04
Large	**1024**	**8**	**3e-04**
	1024	10	3e-04

B.3 Distillation details

In the distillation experiments, we train the same grid of model width and depth as in the supervised learning experiments (Section B.2) to use as the student model. The teacher model is chosen by picking the best performing model from the best model configuration from the supervised learning experiments in Table 21. For each of the hidden representation sizes of the student model, we set the transformer feed-forward dimension such that the feed-forward ratio ($d_{\text{ff}}/d_{\text{hidden}}$) is maintained at 4. For each of these configurations, we launch multiple learning rates ($[3e-3, 1e-3, 3e-4]$) across 3 different seeds. Seeding is used to determine dataloading order and model weight initialization. We train all models to completion (200 epochs) and evaluate the model on the training and validation at the end of every epoch. For each model, we select the checkpoint with the lowest validation CER and record the test set CER of this checkpoint. The rest of the training-related hyperparameters (specifically distillation penalty weight, student dropout probabilities, weight decay and learning rate schedule) were chosen according to best validation metrics in prior experimentation with a fixed model size.

We aggregate the results by first taking the validation and test CER averages across seeds, then pick the best aggregated validation learning rate value and report the average and standard deviation on the test results. The chosen hyperparameters are reported in Table 11 and the test CER reported in Table 22.

Table 10: Distillation task details and hyperparameters held constant for models from Table 22.

Data	Input sEMG channels	32
	Window length	8000
	Padding	[1800, 200]
Student arch.	Featurizer channels	[128, 64, 64]
	Featurizer kernels	[11, 3, 3]
	Featurizer strides	[5, 2, 2]
	Encoder attentions heads	16
	Text Tokenizer	Character-level
	Vocab size	99
	Encoder hidden, attention and activation dropout	0.2
	Encoder feature projection dropout	0.2
	Encoder final layer dropout	0.2
Teacher arch.	Featurizer channels	[128, 64, 64]
	Featurizer kernels	[11, 3, 3]
	Featurizer strides	[5, 2, 2]
	Encoder hidden size	1024
	Encoder feed-forward ratio	4
	Encoder layers	8
	Encoder convolutional dim	[64]
	Encoder attentions heads	16
	Text Tokenizer	Character-level
	Vocab size	99
	Encoder hidden, attention and activation dropout	0.0
	Encoder feature projection dropout	0.0
	Encoder final layer dropout	0.0
Training	Epochs	200
	Effective batch size	640
	Encoder causal attention	True
Optimizer	Learning rate schedule	linear warmup + cosine decay
	Learning rate warmup ratio	0.05
	Weight decay	0.1
	CTC zero infinity	True
	Gradient clipping	0.1
	Distillation loss weight	0.5
	Distillation loss (logits)	Cross Entropy
Software	Torch version	2.3.1+cu121
	Transformers version	4.36.2

Table 11: Distillation hyperparameters that vary for models from Table 22.

Student arch.	Student encoder hidden size	Student encoder layers	Optimizer learning rate
	128	2	1e-03
	128	4	1e-03
	128	6	1e-03
	128	8	1e-03
Tiny	128	10	1e-03
	256	2	1e-03
	256	4	1e-03
Small	256	6	1e-03
	256	8	1e-03
	256	10	1e-03
	512	2	3e-03
	512	4	1e-03
	512	6	1e-03
	512	8	1e-03
	512	10	3e-04
	1024	2	1e-03
	1024	4	1e-03
	1024	6	3e-04
Large	1024	8	3e-04
	1024	10	3e-04

B.4 Personalization details

In the personalization experiments, we focus on our three highlighted architecture configurations, i.e., TINY, SMALL and LARGE. For each architecture, we pick three models from the supervised learning experiments, irrespective of hyperparameters and seeds, according to the best validation performance. We refer to these three models as seed A, B and C. For the TINY and SMALL architecture, we repeat this procedure with the distillation set of experiments. We refer to these models as being from the "Distillation" origin, as opposed to the "Supervised" origin for the models taken from the supervised learning experiments. For each of those models, we initialize the personalization models from the chosen checkpoint and we launch multiple learning rates ($[3e-3, 1e-3, 3e-4]$) across 3 different seeds for each of the 8 personalization users. Seeding is used to determine dataloading order and model weight initialization. We train all models to completion (100 epochs) and evaluate the model on the training and validation sets at the end of every epoch. For each model, we select the checkpoint with the lowest validation CER and record the test set CER of this checkpoint. The rest of the training-related hyperparameters (specifically student dropout probabilities, weight decay and learning rate schedule) were chosen according to best validation metrics in prior experimentation with a fixed model size.

We aggregate the results by first taking the validation and test CER averages across the personalized users and seeds. We then pick the best aggregated validation learning rate value and report the average and standard deviation on the test results. The chosen hyperparameters are reported in Table 13 and the test CER reported in Table 23. Table 13 also reports the supervised or distillation learning rate used by the seed generic model the personalized was initialized from.

Table 12: Personalization task details and hyperparameters held constant for models from Table 23.

Data	Input sEMG channels	32
	Window length	8000
	Padding	[1800, 200]
Architecture	Featurizer channels	[128, 64, 64]
	Featurizer kernels	[11, 3, 3]
	Featurizer strides	[5, 2, 2]
	Encoder feed-forward ratio	4
	Encoder attentions heads	16
	Tokenizer	Character-level
	Vocab size	99
	Encoder hidden, attention and activation dropout	0.2
	Encoder feature projection dropout	0.2
	Encoder final layer dropout	0.2
Training	Epochs	100.0
	Effective batch size	640
	Encoder causal attention	True
Optimizer	Learning rate schedule	linear warmup + cosine decay
	Learning rate warmup ratio	0.05
	Weight decay	0.0
	Training warmup ratio	0.05
	CTC zero infinity	True
	Gradient clipping	0.1
Software	Torch version	2.3.1+cu121
	Transformers version	4.36.2

Table 13: Personalization hyperparameters that vary for models from Table 23.

Architecture	Init. origin	Init. seed	Generic training lr	Personalization training lr
Tiny	Distillation	seed A	1e-03	3e-04
Tiny	Distillation	seed B	1e-03	3e-04
Tiny	Distillation	seed C	1e-03	3e-04
Tiny	Supervised	seed A	1e-03	3e-04
Tiny	Supervised	seed B	1e-03	3e-04
Tiny	Supervised	seed C	1e-03	3e-04
Small	Distillation	seed A	1e-03	3e-04
Small	Distillation	seed B	1e-03	3e-04
Small	Distillation	seed C	1e-03	3e-04
Small	Supervised	seed A	1e-03	3e-04
Small	Supervised	seed B	3e-04	3e-04
Small	Supervised	seed C	3e-04	3e-04
Large	Supervised	seed A	3e-04	3e-05
Large	Supervised	seed B	3e-04	1e-04
Large	Supervised	seed C	3e-04	1e-04

B.5 Ablation experiments

In the ablation experiments, we focus on a single architecture configurations for each investigated variations, i.e., Raw sEMG/spectrogram, Transformer/TDS, With/without channel rotation and with/without input

masking. All of the configuration for these experiments can be found in Table 14 for the architecture ablation and Table 16 for augmentation. For each of these configuration, we launch two learning rates ($[1e-3, 3e-4]$) across 3 different seeds. Seeding is used to determine dataloading order and model weight initialization. We train all models to completion (number of epochs mentioned in the tables) and evaluate the model on the training and validation sets at the end of every epoch. For each model, we select the checkpoint with the lowest validation CER and record the test set CER of this checkpoint. The rest of the training-related hyperparameters (specifically student dropout probabilities, weight decay and learning rate schedule) were chosen according to best validation metrics in prior experimentation with a fixed model size.

We aggregate the results by first taking the validation and test CER averages across seeds, then pick the best aggregated validation learning rate value and report the average and standard deviation of the test results. The chosen hyperparameters are reported in Table 15 and 17, while the test CERs are reported in Table 4 and 5.

Table 14: Constant hyperpameter for the featurizer and encoder architecture ablation experiment in Table 4.

Data	Input sEMG channels	32
	Window length	8000
	Padding	$[1800, 200]$
Raw sEMG+CNN featurizer	CNN channels	$[128, 256, 384]$
	CNN kernels	$[11, 3, 3]$
	CNN strides	$[5, 2, 2]$
Spectrogram+MLP featurizer	MLP dims	$[384]$
Transformer encoder	feed-forward ratio	4
	Num. layers	8
	hidden size	384
	attentions heads	12
	hidden, attention and activation dropout	0.2
	feature projection dropout	0.2
	final layer dropout	0.2
TDS encoder	Encoder block channels	$[24, 24, 24, 24]$
	Encoder kernel width	32
Task	Tokenizer	Character-level
	Vocab size	99
Training	Epochs	200.0
	Effective batch size	640
	Encoder causal attention	True
Optimizer	Learning rate schedule	linear warmup + cosine decay
	Learning rate warmup ratio	0.05
	Weight decay	0.2
	CTC zero infinity	True
	Gradient clipping	0.1
Software	Torch version	2.3.1+cu121
	Transformers version	4.36.2

Table 15: Varying hyperpameter for the featurizer and encoder architecture ablation experiment in Table 4.

Featurizer	Encoder	Optimizer learning rate
Raw sEMG+CNN featurizer	Transformer	0.001
Spectrogram+MLP featurizer	Transformer	0.001
Raw sEMG+CNN featurizer	TDS	0.0003
Spectrogram+MLP featurizer	TDS	0.001

Table 16: Hyperpameter for data augmentation ablation experiment in Table 5.

Data	Input sEMG channels	32
	Window length	8000
	Padding	$[1800, 200]$
Architecture	Featurizer channels	$[128, 256, 256]$
	Featurizer kernels	$[11, 3, 3]$
	Featurizer strides	$[5, 2, 2]$
	Encoder feed-forward ratio	4
	Num. Encoder layers	8
	Encoder hidden size	256
	Encoder attentions heads	8
	Tokenizer	Character-level
	Vocab size	99
	Encoder hidden, attention and activation dropout	0.2
	Encoder feature projection dropout	0.2
	Encoder final layer dropout	0.2
Training	Epochs	200.0
	Effective batch size	640
	Encoder causal attention	True
Optimizer	Learning rate schedule	linear warmup + cosine decay
	Learning rate warmup ratio	0.05
	Weight decay	0.2
	CTC zero infinity	True
	Gradient clipping	0.1
Software	Torch version	2.3.1+cu121
	Transformers version	4.36.2

Table 17: Varying hyperpameter for the featurizer and encoder architecture ablation experiment in Table 5.

Channel rotation	Input masking	Optimizer learning rate
None	None	0.001
$\pm$ 1	None	0.001
None	Length=15	0.001
$\pm$ 1	Length=15	0.001
None	Length=7	0.001
$\pm$ 1	Length=7	0.001

C Additional results

In Section C.1, we show a comparison with the transformer architecture used in this work with other common architectures in the sEMG field. In Section C.2, we show training curve samples from our experiments. In Section C.3, we show the results for the full model scaling grid we explored. In Section C.4, we present measures of the inference speed of our models.

C.1 Additional baselines

In this section, we compare the TRANSFORMER architecture used throughout this work other architecture common in the sEMG literature. We investigate three models: a Time Convolutional Neural Network (CNN), a Long Short-Term Memory (LSTM) network, and a Visual Transformer (ViT). All architectures are constructed as to have about the same number of parameters as the TDS-ConvNet baseline from Sivakumar et al. (2024). We provide high level descriptions of each in the following paragraphs, to get more details see Table 19 and Table 20.

Time-CNN : This model uses a raw sEMG+CNN featurizer (same as our TRANSFORMER architecture) and time-wise convolutional layers as encoder, each with a kernel size of 5 and a stride of 1 with channel dimensions of [512,512,512,256]. Decoding is done with a linear layer to make predictions over the dictionary.

LSTM : This model uses a raw sEMG+CNN featurizer (same as our TRANSFORMER architecture) and a uni-directional LSTM network as encoder with 10 layers and dimension of 256. Decoding is done with a linear layer to make predictions over the dictionary.

ViT : This model employs a transformer encoder architecture (same as our TRANSFORMER architecture). Similar to the TRANSFORMER architecture, the featurization is CNN-based applied on raw sEMG signal, but it uses non-overlapping 10ms patches that include all EMG channels. We use a 8 layer encoder, and we do a simple sweep across the dimension of the transformer to peek into the scalability of the model. Decoding is done with a linear layer to make predictions over the dictionary.

We show the results of these experiments in Table 1.

Architecture	**Parameters**	**CER (↓%)**
TDS-ConvNet	5.3M	55.57
Time-CNN	5.9M	70.14 ±0.35
LSTM	5.3M	45.96 ±1.02
ViT (d=128)	1.8M	37.29 ±1.57
ViT (d=256)	6.9M	35.29 ±0.66
ViT (d=512)	27M	34.73 ±0.99
TINY TRANSFORMER	2.2M	35.9 ±0.9
SMALL TRANSFORMER	5.4M	35.2 ±1.1
LARGE TRANSFORMER	109M	30.5 ±0.6

Table 18: Cross-user generic performance of common architectural baselines on the *emg2qwerty* dataset. The baselines are compared to the TDS model from Sivakumar et al. (2024) and the TRANSFORMER model from the rest of our analysis. We report the standard deviation across 3 seeds for the Simple CNN, LSTM and ViT baselines, and 6 seeds for the TRANSFORMER architecture. The standard deviation for the baseline models is not reported in Sivakumar et al. (2024).

In Table 1, we see that the transformer model demonstrates similar performance to Vision Transformer (ViT) architecture. In contrast, the Time-CNN model exhibits significantly poorer performance compared to the other architectures. We attribute this to the Time-CNN's finite (and short) receptive field, which

limits its ability to utilize past context from the sequence for predictions. Although TDS-ConvNet is also based on CNN layers, it is less affected by this limitation due to its design, which is tailored to maintain larger receptive fields (Hannun et al., 2019).

The LSTM (with Raw sEMG+CNN featurizer) model performs reasonably well, surpassing the previous TDS-ConvNet baseline (with Spectrogram+MLP featurizer) from Sivakumar et al. (2024). However, this improvement may largely be due to the enhanced featurizer (Raw sEMG+CNN vs Spectrogram+MLP). From the ablation experiments in Table 4 of the Section 5, we see that a Raw sEMG+CNN featurizer with TDS encoder performs about the same as the Raw sEMG+CNN featurizer with LSTM encoder model investigated here (i.e., about 45 CER).

Table 19: Hyperpameter for baseline experiment in Table 18.

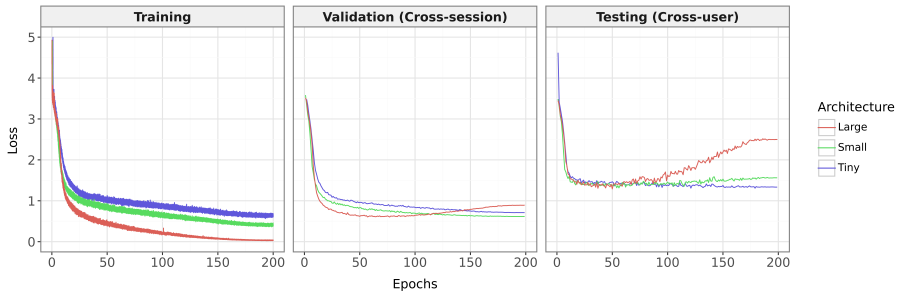
Data	Input sEMG channels	32
	Window length	8000
	Padding	[1800, 200]
Time-CNN	Featurizer channels	[384, 384, 384]
	Featurizer kernels	[11, 5, 5]
	Featurizer strides	[5, 2, 2]
	CNN Encoder input channels	384
	CNN Encoder channels	[512, 512, 512, 256]
	CNN Encoder kernels	[5, 5, 5, 5]
	CNN Encoder strides	[1, 1, 1, 1]
	Linear Decoder input size	256
LSTM	Featurizer channels	[128, 128, 128]
	Featurizer kernels	[11, 3, 3]
	Featurizer strides	[5, 2, 2]
	LSTM Encoder input channels	128
	LSTM Encoder hidden dim	128
	LSTM Encoder num. layers	10
	Linear Decoder input size	256
ViT (d=*)	Featurizer channels	[128, 64, 64]
	Featurizer kernels	[5, 2, 2]
	Featurizer strides	[5, 2, 2]
	Encoder feed-forward ratio	4
	ViT Encoder hidden size	depends (see name)
	Num. Encoder layers	8
	ViT Encoder attentions heads	16
	ViT Encoder hidden, attention and activation dropout	0.2
	ViT Encoder feature projection dropout	0.2
	ViT Encoder final layer dropout	0.2
	ViT Encoder causal attention	True
	Linear Decoder input size	256
Task	Tokenizer	Character-level
	Vocab size	99
Training	Epochs	200.0
	Effective batch size	640
Optimizer	Learning rate schedule	linear warmup + cosine decay
	Learning rate warmup ratio	0.05
	CTC zero infinity	True
	Gradient clipping	0.1
Software	Torch version	2.3.1+cu121
	Transformers version	4.36.2

Table 20: Varying hyperpameter for the baseline experiment in Table 18.

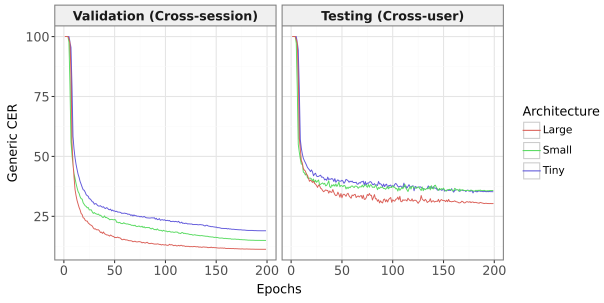
Architecture	Optimizer weight decay	Optimizer learning rate
Time-CNN	0.0	0.003
LSTM	0.2	0.0003
ViT (d=128)	0.2	0.001
ViT (d=256)	0.2	0.001
ViT (d=512)	0.2	0.0003

C.2 Sample training metrics

We show training curves from some of our experiments assist others to more easily reproduce our results. In Figure 4 and 5, we show training loss along with validation (cross-session) and test (cross-user) generic CER for the supervised and distillation tasks respectively. In Figure 6, we show personalization training loss and personalized CER for three of the eight personalization users.

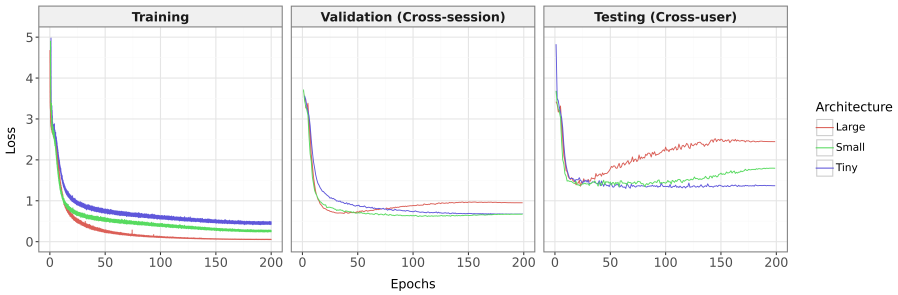


(a) Loss

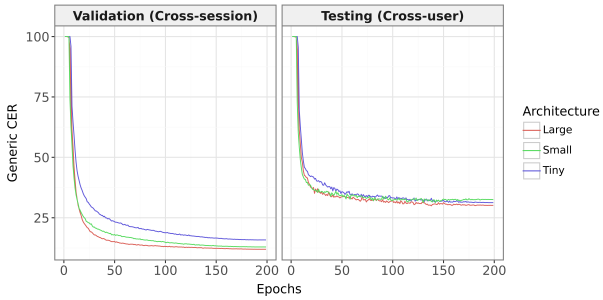


(b) Generic CER

Figure 4: Supervised training sample training curves for the Tiny, Small, Large architecture. Validation is done with unseen sessions from the training users while testing is done across unseen sessions from the unseen users.

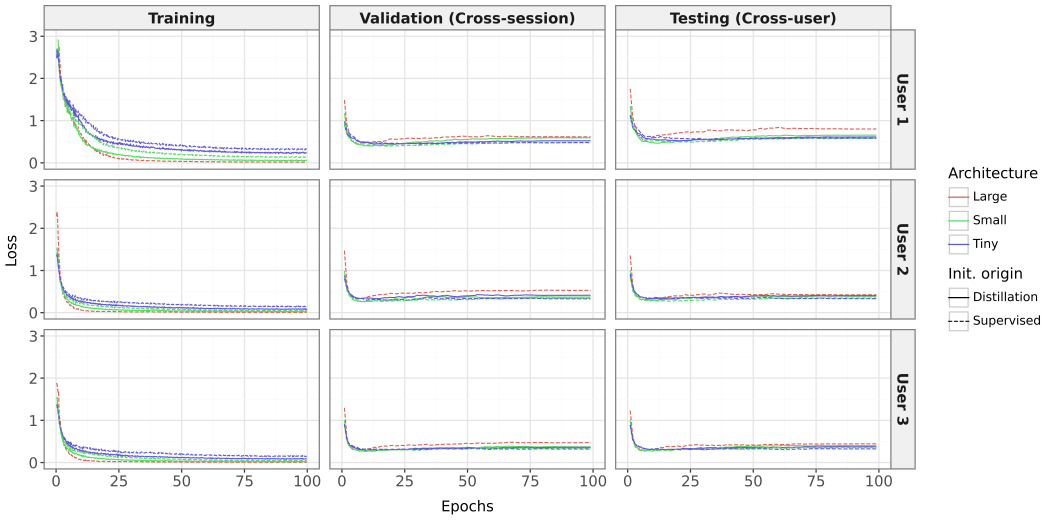


(a) Loss

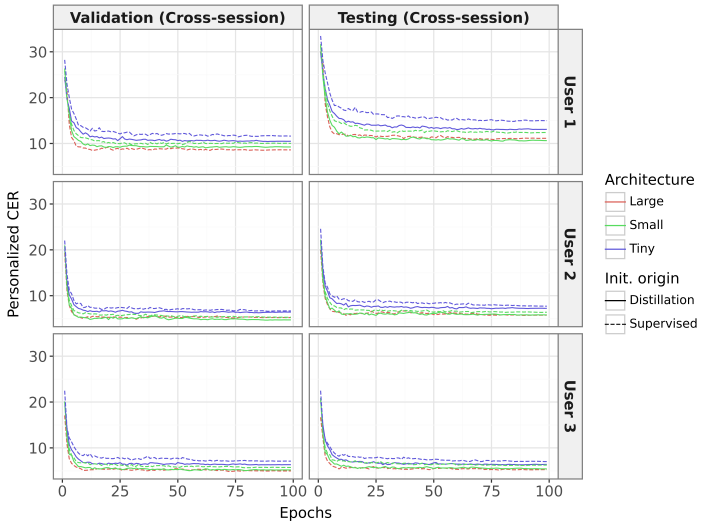


(b) Generic CER

Figure 5: Distillation training sample training curves for the TINY, SMALL, LARGE architecture. Validation is done with unseen sessions from the training users while testing is done across unseen sessions from the unseen users.



(a) Loss



(b) Generic CER

Figure 6: Personalization training sample training curves for the TINY, SMALL, LARGE architecture. For the TINY and SMALL architectures we show the separate curves for initializing the model coming from supervised training and distillation training. We show the training curves for three of the eight personalization users. Both validation and testing is done with unseen sessions from the (single) training user.

C.3 Full model grid scaling results

Model architecture exploration In Figure 7, we show the results for all model shapes investigated in this work. We use this grid of shapes to compute the pareto front, i.e. ones which outperform others at the same or smaller parameter count, presented in Figure 3. Our grid extends from an transformer hidden representation size of 128 to 1024 and a number of transformer layer from 2 to 10. The featurization module which first converts raw EMG into features to feed into the transformer is kept fixed for all models (see Table 8 and Table 10 for the exact configuration). The CER results for generic models (from Figure 7) along with their standard deviation can be found in Table 21 for supervised learning and Table 22 for distilled models.

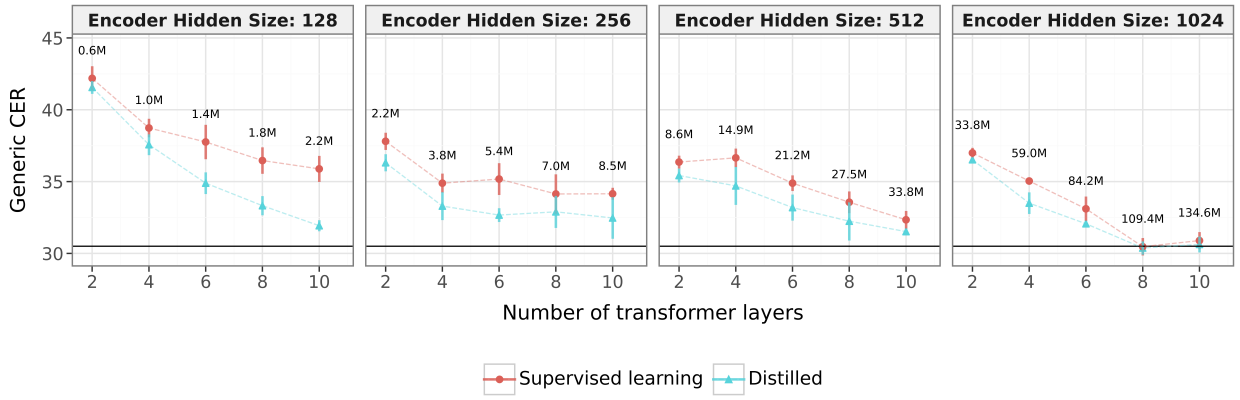


Figure 7: Supervised learning and distilled results on the *emg2qwerty* dataset across multiple model hidden representation size and number of layers. The total number of parameters of the networks are annotated for each configuration. The performance of the teacher model (LARGE architecture) for the distillation training is highlighted by the horizontal line. The vertical bars denote standard deviation across 6 seeds.

Distillation improvement In Figure 8, we show the generic CER improvement observed through distillation across transformer model width and depth.

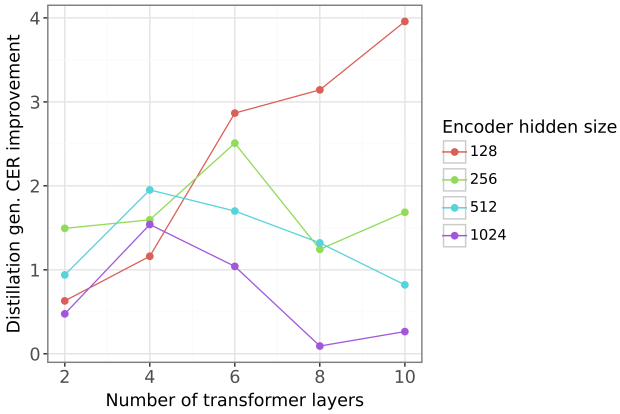


Figure 8: Performance improvement from distillation over supervised learning on the *emg2qwerty* dataset across multiple model hidden representation size and number of layers.

Numerical results Following is the full set of numerical performance reported in this paper. In Table 21 and 22 we annotate the TINY, SMALL and LARGE architecture which corresponds to the architecture highlighted in Table 3 of the main paper.

Table 21: Numerical generic CER and their standard deviation for supervised learning analysis. Hyperparameters provided in Section B.2. The specific named architectures from the main text are highlighted in bold.

Architecture	Encoder hidden size	Encoder layers	Model params	Generic CER
	128	2	631523	42.19 ±0.85
	128	4	1028067	38.73 ±0.64
	128	6	1424611	37.76 ±1.20
	128	8	1821155	36.46 ±0.92
Tiny	**128**	**10**	**2217699**	**35.88 ±0.91**
	256	2	2229091	37.81 ±0.60
	256	4	3808611	34.89 ±0.67
Small	**256**	**6**	**5388131**	**35.18 ±1.11**
	256	8	6967651	34.14 ±1.37
	256	10	8547171	34.15 ±0.42
	512	2	8569955	36.36 ±0.44
	512	4	14874723	36.65 ±0.65
	512	6	21179491	34.89 ±0.54
	512	8	27484259	33.56 ±0.75
	512	10	33789027	32.34 ±0.62
	1024	2	33834595	37.00 ±0.36
	1024	4	59027043	35.04 ±0.19
	1024	6	84219491	33.10 ±0.85
Large	**1024**	**8**	**109411939**	**30.47 ±0.61**
	1024	10	134604387	30.89 ±0.60

Table 22: Numerical generic CER and their standard deviation for distillation analysis. Encoder hidden sizes and layers represents the student encoder sizes. The teacher follows the LARGE architecture. Hyperparameters are provided in Section B.3

.

Student arch.	Student encoder hidden size	Student encoder layers	Model params	Generic CER
	128	2	631523	41.56 ±0.45
	128	4	1028067	37.57 ±0.72
	128	6	1424611	34.89 ±0.76
	128	8	1821155	33.32 ±0.67
Tiny	**128**	**10**	**2217699**	**31.93 ±0.39**
	256	2	2229091	36.31 ±0.60
	256	4	3808611	33.29 ±0.98
Small	**256**	**6**	**5388131**	**32.67 ±0.48**
	256	8	6967651	32.89 ±1.11
	256	10	8547171	32.47 ±1.45
	512	2	8569955	35.42 ±0.48
	512	4	14874723	34.70 ±1.32
	512	6	21179491	33.19 ±0.91
	512	8	27484259	32.24 ±1.34
	512	10	33789027	31.52 ±0.23
	1024	2	33834595	36.52 ±0.19
	1024	4	59027043	33.50 ±0.75
	1024	6	84219491	32.06 ±0.22
Large	**1024**	**8**	**109411939**	**30.38 ±0.46**
	1024	10	134604387	30.63 ±0.56

Table 24: Inference speed of highlighted model sizes (per 4 second window).

Architecture	Encoder hidden size	Encoder layers	Params	Inference speed (ms)
Tiny Transformer	128	10	2.2M	6.1
Small Transformer	256	6	5.4M	5.7
Large Transformer	1024	8	109M	27.0

Table 23: Numerical generic CER and their standard deviation for the personalization results with varying model initialization. The initialization origin column represents whether the generic model used as initialization for the personalized model has been trained through *Supervised* learning (i.e., without distillation loss) or through *Distillation*. The initialization seed column represents distinction between different models with the same architecture, but trained on different seeds or different hyperparameters, selected by choosing the best validation performance among the all the models trained. Hyperparameters provided in Section B.4.

Architecture	Init. origin	Init. seed	Model params	Generic CER
Tiny	Distillation	seed A	2217699	8.64 ±0.04
Tiny	Distillation	seed B	2217699	8.65 ±0.02
Tiny	Distillation	seed C	2217699	8.91 ±0.06
Tiny	Supervised	seed A	2217699	9.72 ±0.13
Tiny	Supervised	seed B	2217699	9.91 ±0.13
Tiny	Supervised	seed C	2217699	10.36 ±0.02
Small	Distillation	seed A	5388131	7.07 ±0.06
Small	Distillation	seed B	5388131	7.02 ±0.03
Small	Distillation	seed C	5388131	7.16 ±0.01
Small	Supervised	seed A	5388131	7.94 ±0.06
Small	Supervised	seed B	5388131	9.78 ±0.07
Small	Supervised	seed C	5388131	9.20 ±0.06
Large	Supervised	seed A	109411939	6.81 ±0.07
Large	Supervised	seed B	109411939	6.48 ±0.02
Large	Supervised	seed C	109411939	6.54 ±0.08

C.4 Inference speed

We measured the inference speed of the highlighted Tiny, Small and Large in Table 2 model architecture to assess their viability to be run in real time and to see how much model scale influences inference speeds. We present the inference speeds in Table 24.

Naive streaming inference In the most naive case of streaming inference, we pass a full window length of data (4 second in emg2qwerty) to the model and use exclusively the final prediction, then repeat this process for the next prediction.

Applying this naive streaming inference paradigm to single-window inference speed in Table 24 tells us that the Large architecture could not be inferable in real-time due to the maximum inference frequency being approximately $f = \frac{1}{27\text{ms}} = 37\text{Hz}$.

Note that under this setting one can trade off latency with inference speed, for example by using the last n predictions from a window as predictions, which alleviates the requirements on inference speed by nx at the cost of increasing latency of the output by nx.

Accelerated streaming inference In a more sophisticated implementation of streaming inference, one could further improve the performance of the models by using optimizations like KV Caching (Pope et al., 2023) or designing custom kernels. These techniques are outside the scope of our work.