DistillSort: Distilling Autoencoders for Efficient Spike Sorting in Intracortical Brain–Computer Interfaces

Mohammad Masnour
Department of Electrical and Computer Engineering
McMaster University
Hamilton, ON, Canada, L8S 4L8
mansom54@mcmaster.ca

Ameer Abdelhadi
Department of Electrical and Computer Engineering
McMaster University
Hamilton, ON, Canada, L8S 4L8
ameer@mcmaster.ca

***Abstract*—As intracortical brain–computer interfaces (BCIs) scale to thousands of channels, separating overlapping neural spikes in real time becomes increasingly challenging under strict power and memory budgets. Deep autoencoders have recently emerged as powerful feature extractors for spike sorting, yet their large parameter counts and high inference costs preclude deployment on implantable devices. This work introduces DistillSort, a spike-sorting pipeline that compresses autoencoder-based feature extractors through knowledge distillation. A deep teacher autoencoder is trained on neural waveforms, and a shallow student is distilled to match both its latent codes and reconstructions via a weighted combination of self-reconstruction, code-distillation, and reconstruction-distillation losses. The distilled student produces latent representations that closely approximate those of the teacher while containing only half of the teacher's parameters and requiring 2.3× fewer multiply–accumulate operations. On real multichannel neural recordings, the distilled model maintained the deep autoencoder's clustering quality and was consistently superior to a shallow autoencoder trained without distillation. Inference latency is reduced from 97.0 μs to 60.1 μs. These results demonstrate that knowledge distillation enables efficient and deployable neural feature extraction for spike sorting in BCIs.**

***Index Terms*—Spike Sorting, Autoencoder, Intracortical BCI, Model Compression, Knowledge distillation.**

I. Introduction

Spike sorting, the process of separating extracellular action potentials/spikes by their originating neuron, is widely recognized as a critical first step in neural signal processing pipelines [1]. This step is especially vital in intracortical brain–computer interfaces (BCIs), where microelectrode arrays record multi-neuron activity that must be disentangled into single-unit (single neuron) signals based on the similarity of their shapes [2]. Effective spike sorting enables prosthetic devices to interpret neural firing patterns on a per-neuron basis, which is essential for understanding brain function and for high-fidelity BCI control. Conventional spike-sorting pipelines perform four sequential steps: filtering, spike detection, feature extraction, and clustering. They rely heavily on the feature extraction stage to achieve separability of spike clusters [3]. Manual sorting by human operators is often still used because many algorithms fail to achieve satisfactory performance; however, this manual curation is subjective, labour intensive and cannot scale to modern high-channel devices [4].

The need for better algorithms has become acute with the development of high-density multi-electrode arrays that record from hundreds or thousands of channels simultaneously [5]. These platforms generate huge volumes of data and increase the likelihood of overlapping spikes and electrode drifts, making spike sorting more complex. Current spike-sorting methods often saturate at about eight to ten distinguishable units per channel and perform particularly poorly when neurons fire sparsely [1]. Furthermore, wireless or fully implantable intracortical systems cannot afford to stream raw signals off-chip due to power and bandwidth limits [6]. Instead, they must detect and sort spikes *in situ* with minimal delay to enable fast feedback like the closed-loop neuroprosthetics [7], [8]. The hardware environment of an implantable device imposes stringent resource limitations. These systems must operate under strict power, memory and latency budgets. For example, to prevent tissue heating, the power density of spike-sorting processors should remain below 277 $\mu W mm^{-2}$ [8]. Meeting these constraints while maintaining high sorting accuracy is a major challenge for intracortical BCIs.

The performance of spike-sorting pipelines depends critically on the choice of feature-extraction method. Many early approaches used principal component analysis (PCA) or wavelet transforms to extract spike features, but these handcrafted features can struggle to separate similar spike shapes [9]. Deep learning has therefore been applied to learn features directly from waveforms. Autoencoders (AEs) are neural networks that learn low-dimensional representations of their inputs and reconstruct the input from this latent code. Ardelean et al. [3] performed a systematic study of autoencoders for spike sorting and showed that AE-based features improved clustering performance compared with state-of-the-art methods on both synthetic and real datasets. They highlighted that the separability of clusters is driven by the feature extraction technique rather than the clustering algorithm. Li et al. [2] used one-dimensional convolutional neural networks (1D-CNNs) for spike sorting and achieved over 99% clustering

accuracy on simulated data and high robustness to noise and overlapping spikes. These methods have demonstrated superior clustering performance over classical methods, but their large model sizes and high inference cost make them difficult to deploy on energy-constrained devices.

Several studies have investigated hardware-friendly implementations of autoencoder-based spike sorting. Seong et al. [8] proposed a spike-sorting processor that uses an L2-normalized convolutional autoencoder and a cosine-similarity-based K-means algorithm. Their ASIC implementation achieved 95.54% clustering accuracy. However, it consumed 224.75 $\mu W mm^{-2}$, roughly 61$\times$ more power and chip area than traditional DSP-based designs. Choi et al [7]. introduced a Δ-based spike-sorting system-on-chip that incorporates an event-driven binary autoencoder neural network implemented in an analog computing-in-memory architecture. By using Δ-spikes to suppress low-frequency noise, their system achieved 94.54% on-chip classification accuracy and reduced the data transmission rate by 48.8$\times$ Although these works demonstrate that autoencoders can be efficiently implemented, the networks still occupy a significant area and require careful optimization. In this context, Sha et al. [10] developed Marple, a hardware pipeline for real-time spike sorting using template matching. Marple detects spiking events on each channel, verifies locality with neighboring channels, and sends a 9×60 sample window to a matching stage. Marple proposes a centroid-based compression scheme that reduces template storage by 8-11$\times$ without affecting accuracy, as each template is stored as nine 60-sample waveforms. The authors explored alternatives by replacing template matching with a small, medium, or large CNN that accepts the same 9×60 window and channel ID. Using the same 9$\times$60 sample window and channel ID, the CNN variants outperformed template matching with 85.6–91.9% accuracy for populations of up to 30,000 neurons, but at the cost of increased memory usage and more than 1.48 tera-operations per second even for the smallest model. This highlights the need for lightweight feature extraction methods that match deep-network accuracy without the computational and memory costs of large CNNs.

Other approaches include contractive autoencoders and self-supervised methods. Radmanesh et al. [9] proposed an online spike-sorting algorithm based on a deep contractive autoencoder that achieved over 90% accuracy on unseen waveforms and outperformed existing methods by about 40% in the offline case. Brockhoff et al. [11] introduced PseudoSorter, a self-supervised spike-sorting framework employing density-based pseudo-labelling and iterative fine-tuning, which outperformed other algorithms and enabled investigation of Tau-induced neuronal dysfunction. Meanwhile, Li et al. [4] presented AECuration, which uses an autoencoder trained on synthetic waveforms to automatically classify neural and non-neural events and improved the sensitivity of various sorting pipelines by up to 20%. These approaches reduce manual curation but still depend on deep models.

To address these challenges, we propose DistillSort, a spike-sorting pipeline that leverages knowledge distillation to compress autoencoder-based feature extractors for efficient deployment in intracortical BCIs. Knowledge distillation transfers the knowledge learned by a large teacher model to a smaller student model by training the student to mimic the teacher's outputs or latent representations [12]. This technique has been widely used in natural language processing and computer vision to reduce computational costs without sacrificing accuracy. In the context of autoencoders, distillation can be performed by training a lightweight student encoder–decoder network to match the teacher's latent codes and reconstructions. By aligning the latent space of the student with that of the teacher, the distilled model preserves clustering and reconstruction quality while significantly reducing the number of parameters and multiply-accumulate operations (MACs).

In DistillSort, we first train a deep autoencoder on a real, publicly available dataset from the Transylvanian Institute of Neuroscience. We then distill the knowledge from this deep model into a shallow autoencoder that retains the essential latent structure of the spike waveforms. We compare the performance of three configurations, deep, shallow (without distillation) and distilled autoencoders, using unsupervised clustering metrics. Specifically, we compute the Silhouette Score (SS), Davies–Bouldin Score (DBS) and Calinski–Harabasz Score (CHS) on a per-channel basis to quantify sorting quality. To assess hardware suitability, we also measure the required number of MACs, memory footprint and latency. Although our experiments are conducted offline to provide a feasibility analysis, the distilled model is designed for eventual online deployment in intracortical BCIs.

II. Methodology

A multi-stage pipeline is designed to transform raw intracortical recordings into neuron-specific spike trains as shown in Fig. 1. The process starts with preprocessing, in which continuous voltage traces from multiple channels are conditioned and spike events are extracted. Feature extraction then reduces each spike waveform to a low-dimensional representation using knowledge-distilled (KD) AEs. Finally, the resulting feature vectors are clustered to assign spikes to putative neurons and produce sorted spike trains. Each stage operates on a different signal type, from raw broadband voltage to normalized spike snippets to latent codes. The following subsections describe each stage in turn.

A. Dataset

A dataset from the Transylvanian Institute of Neuroscience for intracortical electrophysiological recordings was used for training and evaluation [3]. The dataset comprises multi-channel extracellular waveforms acquired from the primary visual cortex of adult mice using 32-channel A32-tet probes (NeuroNexus Technologies, Inc.) sampled at 32 kHz. During the original experiment, full-field drifting gratings (0.11 cycles/deg at 1.75 cycles/s with contrasts between 25% and 100%) were presented in eight directions on a 60 Hz monitor to evoke neural responses. All recordings were performed

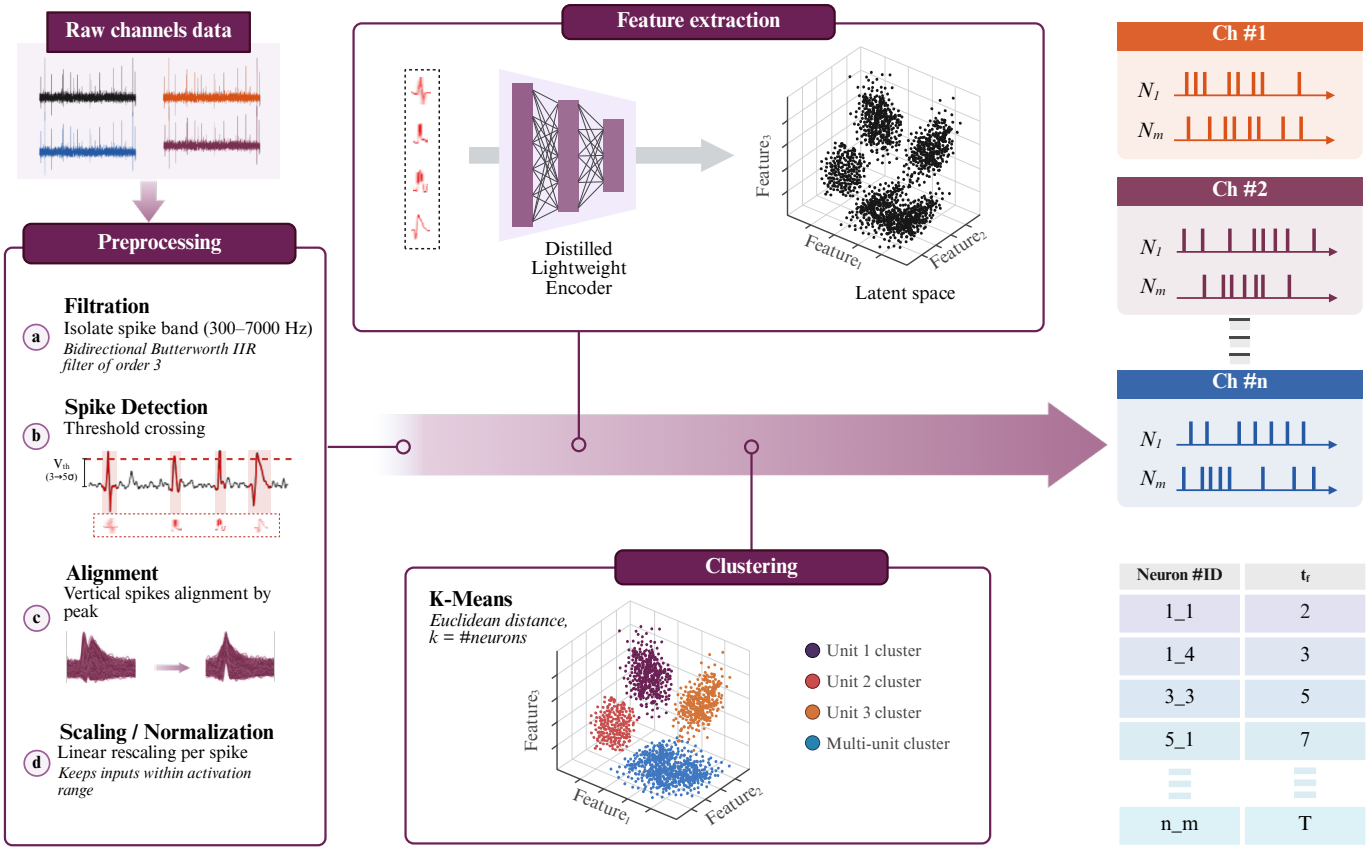

Fig. 1. **Proposed spike sorting pipeline.** The processing stages from raw extracellular data to the generation of neuron-specific spike trains.

under anesthesia, and the resulting extracellular traces are provided as raw voltage samples without ground-truth spike labels.

B. Pre-processing

In the first stage of the pipeline, raw extracellular waveforms are conditioned to yield uniform spike snippets ready for feature extraction (see Fig. 1). Each channel is band-pass filtered between 300 Hz and 7 kHz using a bidirectional third-order Butterworth IIR filter to isolate the spike band and suppress low-frequency drift and high-frequency noise. Spikes are then detected by threshold crossing, with the threshold chosen between three and five times the standard deviation of the filtered signal. Once detected, each spike waveform is temporally aligned by shifting it so that its peak amplitude occurs at a consistent reference sample. Finally, every aligned snippet is linearly rescaled to lie in [0,1], keeping amplitudes within the activation range of the autoencoders.

C. Autoencoders Architecture and Distilling Process

The feature extraction stage comprises a deep autoencoder *teacher* network and a lightweight *student* network trained in a teacher-student configuration. Both networks are fully connected and operate on one-dimensional spike snippets. The teacher autoencoder is a symmetric network that compresses each spike waveform into a low-dimensional latent vector z_T through a sequence of dense layers with gradually decreasing dimensions from 70 down to 2 samples. Each block applies batch normalization and a rectified linear unit (ReLU). The decoder mirrors this structure in reverse order to reconstruct the input (see Fig. 2). This depth allows the teacher to capture subtle, non-linear structure in spike shapes and to produce discriminative latent representations.

Formally, let $x \in \mathbb{R}^L$ denote a pre-processed spike snippet of length L. The teacher encoder f_T applies a series of n fully connected layers to produce a latent code $z_T = f_T(x)$. For layer $l \in \{1, \ldots, n\}$, the transformation is expressed as

$$h^{(l)} = \phi\big(W^{(l)} h^{(l-1)} + b^{(l)}\big), \tag{1}$$

where $h^{(0)} = x$, $W^{(l)}$ and $b^{(l)}$ are the layer weights and biases, and ϕ is the activation function (ReLU for hidden layers and $\tanh$ for the output layer). The encoder output $h^{(n)}$ corresponds to the latent code z_T. The decoder g_T reconstructs the waveform by reversing the layer dimensions:

$$\hat{h}^{(l-1)} = \phi\big((W^{(l)})^\top \hat{h}^{(l)} + \tilde{b}^{(l)}\big), \tag{2}$$

starting from $\hat{h}^{(n)} = z_T$ and producing the reconstruction $\hat{x}_T = \hat{h}^{(0)}$. The student encoder f_S and decoder g_S follow the same formulation but with only three fully connected layers per side (60, 40, 20) until the latent code of size 2 is reached, as illustrated in Fig. 2. The student generates its latent code $z_S = f_S(x)$ and reconstruction $\hat{x}_S = g_S(z_S)$.

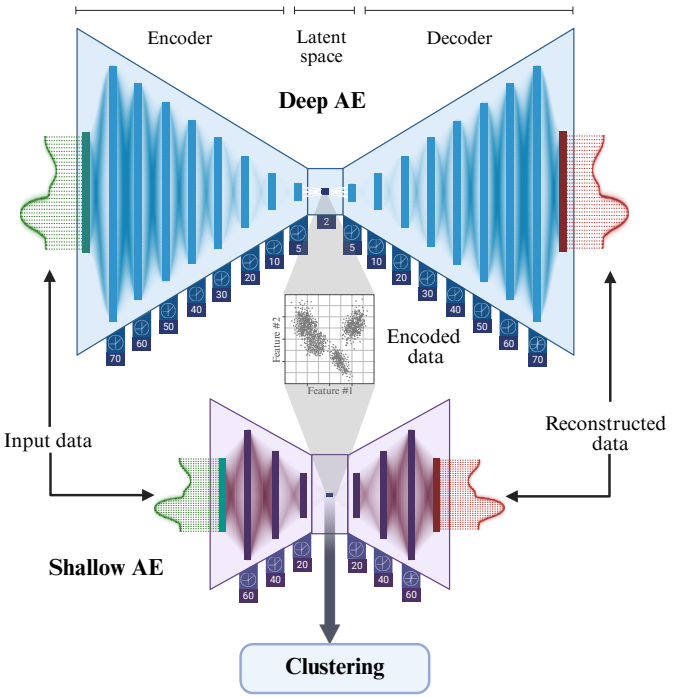

Fig. 2. **Deep and shallow autoencoder architectures.** Both models follow a symmetric, fully connected design composed of encoder and decoder branches with contracting and expanding layer dimensions. The deep variant uses multiple dense layers to capture complex spike waveform features, whereas the shallow variant employs a reduced configuration optimized for low-latency, hardware-efficient deployment.

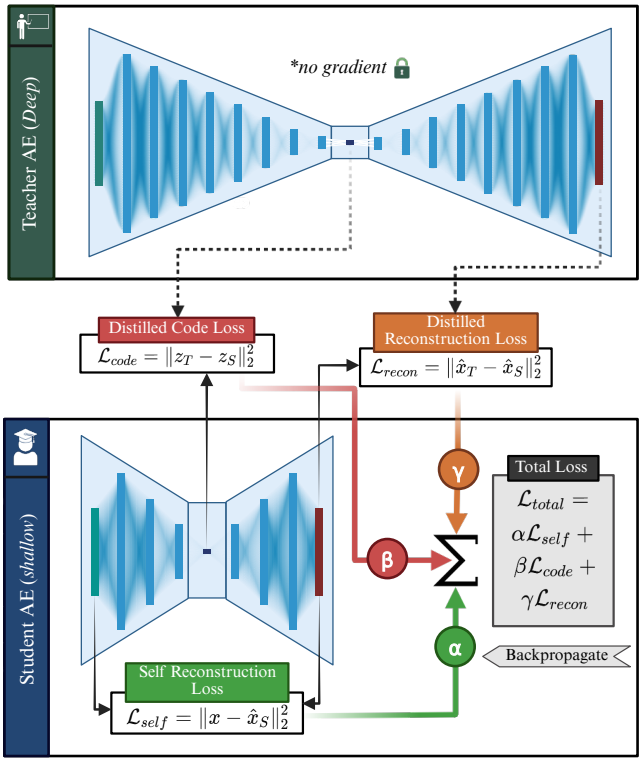

Fig. 3. **Knowledge distillation process.** During training, the teacher network remains frozen while the student learns to approximate its latent codes and reconstructions. A weighted combination of self-reconstruction, distilled code, and distilled reconstruction losses guides the student to preserve the teacher's representational fidelity in a compact, hardware-efficient form. Only the student's encoder is deployed during inference to generate compact feature embeddings for clustering.

To enable deployment on resource-constrained hardware, the shallow student autoencoder is distilled from the deep teacher. Despite its reduced capacity, the student is trained to approximate both the teacher's latent codes and its reconstructions, as depicted in Fig. 3. Let x denote a normalized spike snippet, $\hat{x}_T$ and $\hat{x}_S$ the reconstructions produced by the teacher and student, and z_T and z_S their latent codes. Three loss terms are combined to train the student: (i) a *self-reconstruction loss*

$$\mathcal{L}_{\text{self}}(x) = \|x - \hat{x}_S\|_2^2, \tag{3}$$

which encourages the student to reconstruct the input; (ii) a *distilled code loss*

$$\mathcal{L}_{\text{code}}(x) = \|z_T - z_S\|_2^2, \tag{4}$$

which aligns the student's latent representation with the teacher's; and (iii) a *distilled reconstruction loss*

$$\mathcal{L}_{\text{recon}}(x) = \|\hat{x}_T - \hat{x}_S\|_2^2, \tag{5}$$

which matches the student's reconstruction to the teacher's output. The total objective is a weighted sum:

$$\mathcal{L}_{\text{total}} = \alpha\,\mathcal{L}_{\text{self}} + \beta\,\mathcal{L}_{\text{code}} + \gamma\,\mathcal{L}_{\text{recon}}, \tag{6}$$

where α, β, and γ balance reconstruction fidelity against distillation. During training, the teacher's parameters are frozen, and only the student is updated using mini-batch Adam optimization (learning rate $= 0.001$, batch size $= 128$, epochs $=$ 100). The teacher network participates only during distillation, providing the reference latent codes z_T and reconstructions $\hat{x}_T$ that guide the student. During inference, the teacher is discarded, and only the student encoder f_S is executed in feed-forward mode to map each spike snippet into a latent representation for clustering.

D. Clustering

After feature extraction, each spike is represented by the student's latent code z_S. To group spikes originating from the same neuron, the latent codes within each channel are clustered using the *k-means* algorithm with Euclidean distance. The number of clusters k corresponds to the estimated number of neurons recorded on that channel. For extracellular recordings, spikes generated by neurons located close to the electrode tip exhibit considerably higher amplitudes than the background noise and are therefore easily detected and sorted. Neurons represented in gray in Fig. 4 correspond to cells that fire very sparsely and are thus undetected. The neurons within the light gray region emit spikes that are higher than the background but not sufficiently large to be individually classified; their activity is typically grouped into a multi-unit cluster. Neurons beyond this area contribute primarily to the background noise.

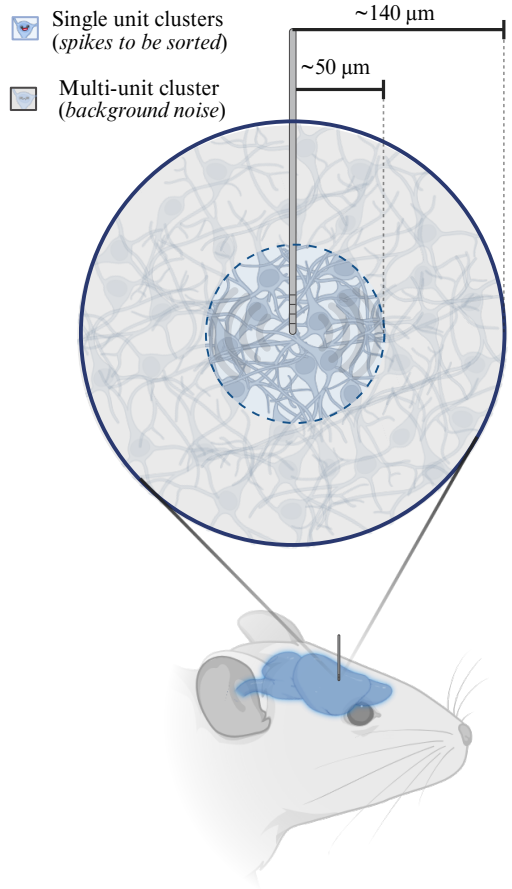

Fig. 4. **Extracellular recordings and spatial clustering behavior.** Neurons located near the electrode tip (inner $\sim$ 50 μm region) generate high-amplitude spikes that are individually resolved as single-unit clusters. Cells farther away emit lower-amplitude spikes that merge into a multi-unit cluster.

The *k-means* algorithm employed here is a partition-based clustering method that divides the feature space into k partitions, assigning each sample to the nearest centroid according to the Euclidean distance. Despite its simplicity, this method requires the number of clusters to be predefined, which is challenging for real neural data. Nevertheless, this step is not the main focus of the present study. The final cluster assignments are projected back to the original spike timestamps to construct neuron-specific spike trains (see Fig. 1).

E. Evaluation Metrics

Because the dataset lacks ground-truth labels, the quality of clustering is assessed using internal evaluation metrics. Three complementary criteria, the Silhouette Score (SS), the Davies-Bouldin Score (DBS), and the Calinski-Harabasz Score (CHS), are computed for each channel.

- **Silhouette Score (SS).** For each data point i, the silhouette coefficient measures the balance between the average intra-cluster distance $a(i)$ and the smallest average inter-cluster distance $b(i)$:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}. \tag{7}$$

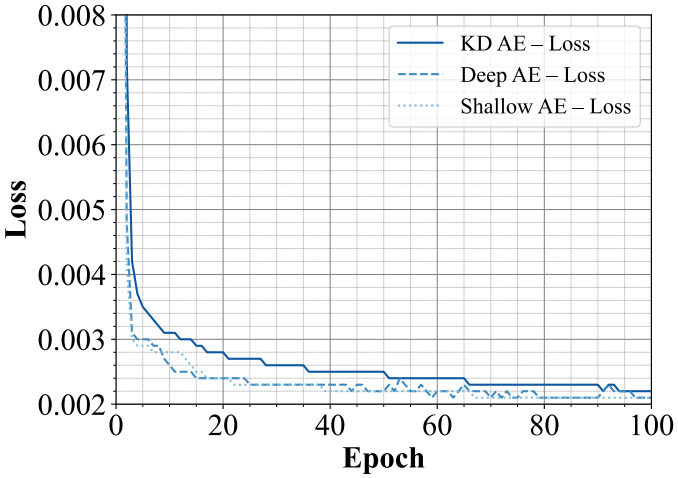

Fig. 5. Training loss trajectories for the deep, shallow, and KD autoencoders over 100 epochs.

Values close to 1 indicate that the point is well matched to its assigned cluster and poorly matched to neighboring clusters; values near 0 suggest an ambiguous assignment, and negative values signal misclassification. The overall Silhouette Score is the mean of $s(i)$ across all spikes.

- **Davies–Bouldin Score (DBS).** The Davies-Bouldin Score quantifies the average similarity between each cluster and its most similar cluster. For k clusters,

$$\mathrm{DBS} = \frac{1}{k}\sum_{i=1}^{k} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right), \tag{8}$$

where σ_i is the mean distance of points in a cluster i to their centroid c_i and $d(c_i, c_j)$ is the distance between centroids c_i and c_j. Lower values indicate tighter, better separated clusters.

- **Calinski-Harabasz Score (CHS).** The Calinski-Harabasz Score measures the ratio of between-cluster dispersion to within-cluster dispersion. Given N samples and K clusters,

$$\mathrm{CHS} = \frac{\mathrm{BCSS}/(K-1)}{\mathrm{WCSS}/(N-K)}, \tag{9}$$

where the between-cluster sum of squares is

$$\mathrm{BCSS} = \sum_{k=1}^{K} n_k \|c_k - c\|^2, \tag{10}$$

with n_k the number of points in cluster k, c_k the cluster centroid and c the global centroid, and the within-cluster sum of squares is

$$\mathrm{WCSS} = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - c_k\|^2. \tag{11}$$

Larger CHS values indicate clusters that are compact and well separated.

These metrics provide complementary assessments of cluster compactness and separation and guide the selection of

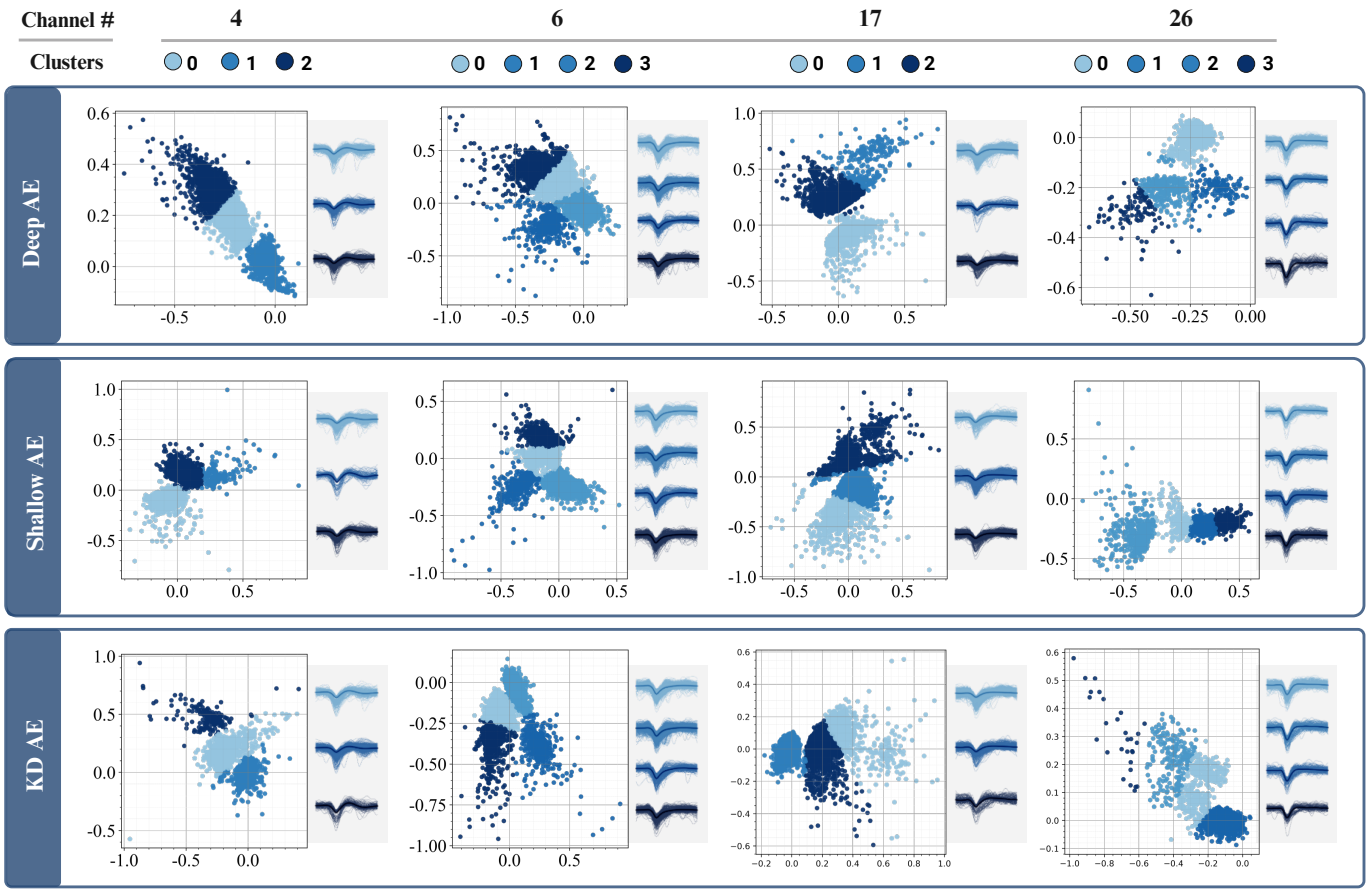

Fig. 6. **Latent-space distributions and corresponding cluster waveforms for representative channels obtained using the deep, shallow, and KD autoencoders.** Each scatter plot shows the two-dimensional latent embeddings of detected spikes, color-coded by cluster assignment

hyper-parameters such as the number of clusters and the distillation weights.

TABLE I
TRAINING HYPERPARAMETERS USED FOR DISTILLSORT.

Parameter	**Value**
Optimizer	Adam
Learning Rate	0.001
Batch Size	128
Activation (Hidden Layers)	ReLU
Activation (Output Layer)	tanh
Loss Function	Mean Squared Error (MSE)
Epochs	100
KD Parameters	
KD Loss Function	Weighted sum of L_{self}, L_{code}, L_{recon}
α (Self-reconstruction weight)	1.0
β (Code-distillation weight)	0.5
γ (Reconstruction-distillation weight)	0.1

III. RESULTS & DISCUSSION

In this section the learning behavior of the teacher and student autoencoders is examined, followed by an analysis of the resulting sorting performance and a discussion of computational efficiency.

A. Learning dynamics

The autoencoders were trained for 100 epochs using the Adam optimizer with a learning rate 1×10^{-3} and a batch size of 128 (Table I). Hidden layers employed rectified linear activations, while the output layer used a hyperbolic tangent to match the normalized spike range. For the KD AE, the student's loss was a weighted sum of self-reconstruction, code-distillation and reconstruction-distillation terms, with weights of $\alpha = 1.0$, $\beta = 0.5$ and $\gamma = 0.1$, respectively. These settings encourage the student to match both the teacher's latent code and its reconstructions while still learning to reconstruct spikes on its own.

Figure 5 compares the reconstruction loss curves for the deep and shallow autoencoders and the weighted loss for the KD autoencoder. All models exhibit rapid reduction in mean squared error during the first 10-20 epochs, reflecting the efficiency of AEs to capture the spikes' waveform structure. The deep AE attains the lowest loss early in training and converges to a final error of approximately 2.5×10^{-3}. The shallow AE, trained without distillation, converges slightly faster but plateaus at a higher error of approximately 2.7×10^{-3}. The KD student initially follows the trajectory of the deep AE and reaches a final loss indistinguishable from the teacher's,

TABLE II
COMPARISON OF CLUSTERING METRICS FOR NORMAL, SHALLOW, AND AUTOENCODERS.

Channel	DBS (↓)			CHS (↑)			SS (↑)		
	Deep AE [3]	Shallow AE [3]	KD AE	Deep AE [3]	Shallow AE [3]	KD AE	Deep AE [3]	Shallow AE [3]	KD AE
4	**0.363**	0.541	0.376	22,566	5,602	**29,377**	0.812	0.644	**0.851**
6	0.429	0.528	**0.423**	33,088	16,356	**42,328**	0.659	0.632	**0.700**
17	0.317	0.493	**0.204**	59,064	16,597	**94,965**	0.868	0.711	**0.895**
26	**0.417**	0.571	0.496	**10,326**	2,967	10,007	**0.809**	0.591	0.763

despite the additional added losses for matching the latent and the reconstructed representations. This indicates that the added distillation objectives do not impede convergence. The close alignment between the KD and deep curves demonstrates that the student effectively learns the salient representations of the teacher despite having far fewer parameters.

B. Sorting performance

The quality of the learned features was assessed by clustering the latent codes produced by each autoencoder. Because ground-truth neuron identities are unavailable, internal validation metrics were used: DBS, CHS, and SS. Figure 6 visualizes the two-dimensional latent spaces of four representative channels (4, 6, 17 and 26) produced by the deep AE, the shallow AE and the KD AE. Each point corresponds to a spike and is colored by its cluster label obtained via density-based clustering. The side panels show the spikes of each cluster, highlighting the cluster centroid spike.

Across channels, the deep AE produces well-separated, compact clusters with distinct waveform morphologies. The shallow AE often yields more diffuse clusters. For example, channel 17 shows a prominent spread of the clusters 0 and 1. This is reflected in the internal metrics (Table II). the shallow AE has the highest DBS values (0.493–0.571) and the lowest CHS (2,967–16,597), indicating poorer cluster compactness and separation. It also yields lower silhouette scores (0.591–0.711) than the deep AE on all channels.

The KD AE improves cluster quality relative to the shallow model and closely matches or surpasses the teacher. In the latent spaces of channels 4 and 6 (see Fig. 6), the KD clusters align with those of the deep AE, preserving the separation between neuron groups. For channel 17, the KD representation tightens the clusters compared with both other models. Quantitatively, the KD AE achieves the lowest DBS on channels 6 (0.423) and 17 (0.204) and ties the deep AE on channels 4 and 26 (Table I). Its CHS is the highest on three of the four channels, with particularly large gains relative to the shallow AE (29,377 versus 5,602 on channel 4 and 94,965 versus 16,597 on channel 17), indicating an improvement of roughly fivefold in between-cluster dispersion. The silhouette scores of the KD AE are also the highest on channels 4 (0.851) and 17 (0.895) and remain comparable to the deep AE on the remaining channels. These results suggest that the KD student retains the teacher's discriminative power despite its reduced capacity, while the shallow student loses some of the fine morphological distinctions required for accurate sorting.

TABLE III
COMPARISON OF COMPUTATIONAL EFFICIENCY.

Metric	**Deep AE** [3]	**Shallow & KD AEs**
Total Parameters (↓)	31,270	**13,740**
Model Size (bytes) (↓)	125,080	**54,960**
MACs (↓)	30,640	**13,440**
Latency (µs) (↓)	97.0	**60.1**

C. Model computational efficiency

The motivation for introducing knowledge distillation was not only to maintain sorting performance but also to reduce the computational burden of the feature extractor so that real-time on-chip deployment becomes feasible. Table III summarizes the resource requirements of the deep AE and the distilled student. The KD AE contains 13,740 parameters (about 56% of the deep network's 31,270) and its memory footprint is reduced from 125 kB to 54.9 kB. Multiply-accumulate operations (MACs) per inference decrease from 30,640 to 13,440, and measured inference latency drops from 97.0 μs to 60.1 μs. These reductions translate to a 2.3$\times$ decrease in compute and a 38% reduction in latency while preserving clustering quality. Such savings directly benefit battery-powered BCIs by lowering energy consumption and heat dissipation. While the shallow AE has a similar architectural size to the KD AE, its inferior sorting performance underscores the importance of distillation. Thus, knowledge distillation offers an effective compromise between accuracy and efficiency, enabling autoencoder-based spike sorting to meet the stringent resource constraints of implantable devices without sacrificing cluster fidelity.

IV. CONCLUSIONS

The proposed DistillSort framework qualitatively demonstrates that sophisticated neural feature extraction can be reconciled with the stringent resource and latency constraints of implantable brain–computer interfaces. The proposed model maintained clustering quality and waveform reconstruction fidelity close to the deep autoencoder's performance, while achieving substantial reductions in model size and computational load. This retention of performance, coupled with a much lower memory and power profile, suggests that DistillSort could enable accurate spike sorting in real-time, low-power BCI systems where conventional deep networks would be infeasible. By distilling advanced neural network

techniques into a deployable form, the proposed framework takes a step toward efficient and scalable spike sorting. Future development of DistillSort will focus on three main fronts. First, broader validation will be pursued using paired datasets containing ground-truth neuron identities for more rigorous performance assessment. Second, the pipeline will be extended to exploit cross-channel information. High-density microelectrode arrays can detect each neuron on multiple electrodes, and geometric features derived from multi-channel recordings have been shown to aid localization and clustering. Finally, the distilled autoencoder and clustering stages will be implemented on custom hardware to further optimize power and area efficiency, enabling portable and implantable applications. We plan to exploit three complementary optimization strategies: (1) bit-serial computations, which have been previously shown as an efficient solution for neural decoding [13]; (2) MSB-first arithmetic, leveraging the serial MSB-first digitization inherent to Successive Approximation Register Analog-to-Digital Converters (SAR ADCs) commonly used in neural data acquisition, which enables early termination of less significant bits and further enhances computation efficiency and reduces latency [14]; and (3) hardware-efficient data compression methods, optimized for deep learning workloads, to minimize data transfers [15]–[18].

Acknowledgements

Many thanks to the anonymous reviewers for their valuable feedback and suggestions. This research would not have been possible without access to design tools and libraries provided by the Canadian Microelectronics Corporation (CMC) and access to the Compute Canada Database (CCDB) through the Digital Research Alliance of Canada. This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant program.

References

[1] C. Pedreira, J. Martinez, M. J. Ison, and R. Q. Quiroga, "How many neurons can we see with current spike sorting algorithms?" *Journal of Neuroscience Methods*, vol. 211, pp. 58–65, 10 2012.

[2] Z. Li, Y. Wang, N. Zhang, and X. Li, "An accurate and robust method for spike sorting based on convolutional neural networks," *Brain Sciences*, vol. 10, pp. 1–16, 11 2020.

[3] E. R. Ardelean, A. Coporîie, A. M. Ichim, M. Dînşoreanu, and R. C. Mureşan, "A study of autoencoders as a feature extraction technique for spike sorting," *PLoS ONE*, vol. 18, 3 2023.

[4] X. Li, J. W. Reddy, V. Jain, M. Forssell, Z. Ahmed, and M. Chamanzar, "Aecuration: automated event curation for spike sorting," *Journal of Neural Engineering*, vol. 22, 4 2025.

[5] A. P. Buccino, S. Garcia, and P. Yger, "Spike sorting: new trends and challenges of the era of high-density probes," *Progress in Biomedical Engineering*, vol. 4, 4 2022.

[6] M. Mansour, A. Gamal, A. I. Ahmed, L. A. Said, A. Elbaz, N. Herencsar, and A. Soltan, "Internet of things: A comprehensive overview on protocols, architectures, technologies, simulation tools, and future directions," *Energies*, vol. 16, no. 8, 2023. [Online]. Available: https://www.mdpi.com/1996-1073/16/8/3465

[7] E. Choi, V. Lukito, I. Choi, S. Lee, I. J. Chang, S. Ha, and M. Je, "A Δ-Based Spike Sorting SoC with End-to-End Implementation of Event-Driven Binary Autoencoder Neural Network in Analog CIM Achieving 94.54% Accuracy and 3.11μW/ch," in *Digest of Technical Papers - Symposium on VLSI Technology*. Institute of Electrical and Electronics Engineers Inc., 2024.

[8] C. Seong, W. Lee, and D. Jeon, "A multi-channel spike sorting processor with accurate clustering algorithm using convolutional autoencoder," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 15, pp. 1441–1453, 12 2021.

[9] M. Radmanesh, A. A. Rezaei, M. Jalili, A. Hashemi, and M. M. Goudarzi, "Online spike sorting via deep contractive autoencoder," *Neural Networks*, vol. 155, pp. 39–49, 11 2022.

[10] E. Sha, A. Liu, K. Ibrahim, M. Mahmoud, C. Giannoula, A. Abdelhadi, and A. Moshovos, "Marple: Scalable spike sorting for untethered brain-machine interfacing," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. ACM, 4 2024, pp. 666–682.

[11] M. Brockhoff, J. Träuble, S. Middya, T. Fuchsberger, A. Fernandez-Villegas, A. Stephens, M. Robbins, W. Dai, B. Haider, S. Vora, N. F. Läubli, C. F. Kaminski, G. G. Malliaras, O. Paulsen, and G. S. K. Schierle, "Pseudosorter: A self-supervised spike sorting approach applied to reveal tau-induced reductions in neuronal activity," *Science Advances*, vol. 11, no. 11, p. eadr4155, 2025. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.adr4155

[12] P. V. Dantas, W. S. da Silva, L. C. Cordeiro, and C. B. Carvalho, "A comprehensive review of model compression techniques in machine learning," *Applied Intelligence*, vol. 54, pp. 11 804–11 844, 11 2024.

[13] A. M. S. Abdelhadi, E. Sha, C. Bannon, H. Steenland, and A. Moshovos, "Noema: Hardware-efficient template matching for neural population pattern detection," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 522–534.

[14] A. M. Abdelhadi and L. Shannon, "Revisiting Deep Learning Parallelism: Fine-Grained Inference Engine Utilizing Online Arithmetic," in *Proceedings of the International Conference on Field-Programmable Technology (ICFPT)*, 2019, pp. 383–386.

[15] A. D. Lascorz, M. Mahmoud, A. H. Zadeh, M. Nikolic, K. Ibrahim, C. Giannoula, A. Abdelhadi, and A. Moshovos, "Atalanta: A Bit is Worth a "Thousand" Tensor Values," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, New York, NY, USA, 2024, p. 85–102.

[16] A. H. Zadeh, M. Mahmoud, A. Abdelhadi, A. Moshovos, M. Mahmoud, A. Abdelhadi, and A. Moshovos, "Mokey: Enabling Narrow Fixed-Point Inference for Out-of-the-Box Floating-Point Transformer Models," in *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA)*, New York, NY, USA, 2022, p. 888–901.

[17] I. Edo Vivancos, S. Sharify, D. Ly-Ma, A. Abdelhadi, C. Bannon, M. Nikolic, M. Mahmoud, A. Delmas Lascorz, G. Pekhimenko, and A. Moshovos, "Boveda: Building an On-Chip Deep Learning Memory Hierarchy Brick by Brick," in *Proceedings of Machine Learning and Systems (MLSys)*, vol. 3, 2021, pp. 1–20. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2021/file/12a304a31e42dfefa21c82431e849124-Paper.pdf

[18] A. Brant, A. Abdelhadi, A. Severance, and G. G. Lemieux, "Pipeline frequency boosting: Hiding dual-ported block ram latency using intentional clock skew," in *2012 International Conference on Field-Programmable Technology*, 2012, pp. 235–238.