**MIND: Motor Interface for Neural Decoding – Proof-of-Concept System**

**Abstract**

The Motor Interface for Neural Decoding (MIND) system is a proof-of-concept platform for translating muscle signals into control commands for prosthetic or robotic devices. It leverages surface electromyography (sEMG) signals from minimally invasive sensors to decode hand and wrist gestures using a six-stage signal processing pipeline. The pipeline encompasses signal analysis, visualization, preprocessing (filtering/windowing), feature standardization, Gaussian mixture model (GMM) based feature extraction, and deep learning classification. We explore state-of-the-art neural network models – including one-dimensional convolutional neural networks (CNN1D), Long Short-Term Memory (LSTM) recurrent networks, and Temporal Convolutional Networks (TCN) – to accurately recognize muscle activation patterns. The decoded outputs drive a simulated robotic arm in Unity3D, demonstrating real-time gesture control in a virtual environment. This white paper details the MIND system architecture and methodology, provides insight into the EMG decoding pipeline, and discusses the future deployment vision for a robust, user-friendly neural interface. Relevant literature in electromyography signal processing, pattern recognition, and neural network-based brain-computer interfaces (BCIs) is cited to contextualize design choices. The focus is on clearly explaining the technical approach and system components, laying the groundwork for future development and real-world implementation.

**Introduction**

Restoring or augmenting motor function through neural decoding is a central goal in brain-computer interface (BCI) and prosthetic control research. Surface electromyography (EMG) offers a practical, non-invasive window into the peripheral nervous system by measuring electrical activity of muscles during contraction ([Processing pipeline for EMG signal classification | Download Scientific Diagram](https://www.researchgate.net/figure/Processing-pipeline-for-EMG-signal-classification_fig1_353943547#:~:text=Electromyography%20,interface%20applications%20%28electrical%20wheelchairs%2C%20virtual)). These signals encode a user’s movement intentions and have been widely used in human-machine interface applications such as powered prosthetic limbs, exoskeletons, and gesture-controlled devices ([Processing pipeline for EMG signal classification | Download Scientific Diagram](https://www.researchgate.net/figure/Processing-pipeline-for-EMG-signal-classification_fig1_353943547#:~:text=Electromyography%20,interface%20applications%20%28electrical%20wheelchairs%2C%20virtual)). However, EMG signals are notoriously noisy, non-stationary, and high-dimensional, posing significant challenges for real-time decoding and classification ([Processing pipeline for EMG signal classification | Download Scientific Diagram](https://www.researchgate.net/figure/Processing-pipeline-for-EMG-signal-classification_fig1_353943547#:~:text=Electromyography%20,computer%20mice%2C%20prosthesis%2C%20robotic%20fin)). Advances in signal processing and deep learning have begun to overcome these challenges, enabling more accurate interpretation of EMG for complex control tasks ( [A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC8117925/#:~:text=Results%3A) ) ([An Affordable 3D-printed Open-Loop Prosthetic Hand Prototype with Neural Network Learning EMG-Based Manipulation for Amputees | SHS Web of Conferences](https://www.shs-conferences.org/articles/shsconf/abs/2022/09/shsconf_etltc2022_03004/shsconf_etltc2022_03004.html#:~:text=the%20signals,performance%20of%20the%20decoding%20hand)).

**MIND System Overview:** The MIND (Motor Interface for Neural Decoding) system is designed as a proof-of-concept platform that translates sEMG signals from a user’s muscles into control commands for a robotic arm. The system focuses on **gesture recognition** – identifying distinct muscle activation patterns corresponding to hand or wrist gestures – and using those decoded gestures to control a virtual robotic arm in real time. The approach prioritizes minimal invasiveness by relying on surface EMG electrodes placed on the skin (for now, simulated or off-the-shelf sensors) rather than implanted neural sensors. By processing EMG signals through a structured pipeline and leveraging modern machine learning, MIND aims to demonstrate that intuitive, high-fidelity control of assistive devices is feasible with non-surgical methods.

**Contributions and Features:** This technical white paper presents:

* A six-step EMG signal decoding pipeline that systematically transforms raw EMG signals into predicted gestures.
* The integration of classical statistical feature extraction (via Gaussian mixture models) with deep learning classifiers (CNN, LSTM, TCN) for robust performance.
* A plan for real-time visualization and testing of decoded outputs using a Unity3D robotic arm simulation, bridging the gap to practical prosthetic control.
* Emphasis on a modular and extensible design, allowing future incorporation of additional sensors or advanced algorithms, and a vision for deploying the system in real-world scenarios (e.g., for amputee prosthetics or human-robot interaction).

By combining insights from biomedical signal processing and artificial intelligence, the MIND system serves as a bridge between raw biosignals and functional movement control. In the following sections, we detail the system architecture and each stage of the processing pipeline, discuss the machine learning models employed, and describe the Unity3D simulation environment. We then outline the future work and deployment vision, including potential improvements and steps toward using MIND in practical settings.

**System Architecture and Minimally Invasive Interface**

( [Figure - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/figure/sensors-24-05631-f001/) ) *Figure 1: Example of a minimally invasive EMG-based interface. Surface EMG electrodes placed on the forearm (red band, left) capture muscle activation signals, and an instrumented glove (right) records finger positions for ground truth in experiments (* [*A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC*](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=As%20shown%20in%20Figure%201%2C,The%20sEMG) *). The MIND system uses a similar non-intrusive setup with electrodes on residual or intact muscles to detect motor intent. Such wearable sensors avoid surgical implantation while providing rich control information from muscle activity.*

The MIND system is built around a wearable EMG sensor interface and a modular processing pipeline that runs on a computing device (e.g., a PC or embedded processor). Figure 1 illustrates a representative EMG capture setup: multiple electrodes are placed around the forearm to measure electrical signals from underlying muscles during different hand movements. In our PoC implementation, we simulate this sensor input using recorded sEMG data or a commercially available armband. The key hardware and software components are:

* **Surface EMG Sensors:** MIND employs surface electrodes to pick up muscle electrical potentials. These can be arranged as an electrode array or an armband (such as the 8-channel Thalmic Labs Myo armband ([An Affordable 3D-printed Open-Loop Prosthetic Hand Prototype with Neural Network Learning EMG-Based Manipulation for Amputees | SHS Web of Conferences](https://www.shs-conferences.org/articles/shsconf/abs/2022/09/shsconf_etltc2022_03004/shsconf_etltc2022_03004.html#:~:text=model,With%20the%20use%20of%20Recurrent))). The sensors are **minimally invasive**, requiring only skin contact. This approach drastically reduces risk and complexity compared to implanted electrodes, while still capturing signals ~50–100 ms before actual limb movement (as muscles activate) ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=known%20as%20the%20passive%20mode,Utilizing) ). Signals are sampled at an appropriate rate (often 500–1000 Hz for EMG, though some datasets use ~100 Hz ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=As%20shown%20in%20Figure%201%2C,27%20%2C%2050) ) after preprocessing).
* **Data Acquisition and Transmission:** The analog EMG signals are amplified and digitized via an EMG acquisition system. Many systems also apply initial analog filtering (e.g., to remove DC offset and minimize motion artifacts ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=data%20glove%2C%20respectively,27%20%2C%2050) )). The digitized data stream is then sent to the MIND processing pipeline. In a wearable scenario, this could be wireless (Bluetooth in the Myo armband ([An Affordable 3D-printed Open-Loop Prosthetic Hand Prototype with Neural Network Learning EMG-Based Manipulation for Amputees | SHS Web of Conferences](https://www.shs-conferences.org/articles/shsconf/abs/2022/09/shsconf_etltc2022_03004/shsconf_etltc2022_03004.html#:~:text=to%20control%20the%20prosthetic%20hand,of))) or wired to a PC for prototyping.
* **Processing Pipeline:** The core of MIND is a software pipeline (detailed in the next section) that runs signal processing and decoding algorithms on the incoming EMG data in real time. This pipeline is responsible for extracting meaningful features from noisy signals and inferring the user’s intended gesture.
* **Control Output (Unity3D Simulation):** Instead of directly driving a physical prosthetic (at this PoC stage), the system outputs the decoded commands to a Unity3D-based robotic arm model. This simulation acts as a stand-in for a prosthetic limb or robotic manipulator. It provides visual feedback and validation for the decoding process – for example, if the user attempts a fist clench, the virtual arm’s hand closes correspondingly when the system correctly interprets the EMG pattern as a “grasp” gesture.

All components work in concert: the sensors capture muscle activations, the pipeline decodes them, and the simulation animates the result. Importantly, the architecture is modular – the EMG input could later be replaced or augmented (e.g., higher-density electrodes or even neural signals), and the output could be a real prosthetic hand. The emphasis on surface EMG makes the interface **user-friendly and non-surgical**, aligning with the goal of creating practical assistive technology that users (including amputees or those with neurological injury) can adopt easily ([An Affordable 3D-printed Open-Loop Prosthetic Hand Prototype with Neural Network Learning EMG-Based Manipulation for Amputees | SHS Web of Conferences](https://www.shs-conferences.org/articles/shsconf/abs/2022/09/shsconf_etltc2022_03004/shsconf_etltc2022_03004.html#:~:text=model,With%20the%20use%20of%20Recurrent)).

**EMG Signal Decoding Pipeline**

To reliably translate EMG signals into device commands, MIND utilizes a **six-step processing pipeline**. Each step corresponds to a stage of data transformation or analysis, gradually reducing complexity and enhancing signal features relevant to gesture classification. The six stages are: **(1) Signal Analysis, (2) Signal Visualization, (3) Preprocessing, (4) Feature Standardization, (5) GMM-Based Feature Extraction,** and **(6) Deep Learning Classification**. This section describes each stage in detail, including the rationale and methods used.

**1. Signal Analysis**

The first step is an offline (or initial) **signal analysis** to understand the raw EMG data characteristics. EMG signals are examined in both time and frequency domains to inform subsequent processing choices. Key aspects include signal amplitude ranges, frequency content, noise levels, and intra- and inter-class variability of the gestures:

* **Time-Domain Inspection:** We examine how muscle activation for each gesture manifests in the raw voltage vs. time plots. For instance, a strong hand contraction might produce a burst of high-amplitude, oscillatory EMG activity on flexor muscle channels. By plotting raw signals, we identify artifacts (e.g., motion-induced spikes) and the general timing of muscle firing relative to gesture events. We also determine an appropriate sliding window size for analysis by observing how quickly EMG patterns rise and fall. Prior research suggests using window lengths on the order of 50–150 ms for capturing transient EMG features while maintaining responsiveness ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=signal,was%20used%20in%20the%20experiments) ) (100 ms yielded good performance in one study ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=signal,was%20used%20in%20the%20experiments) )).
* **Frequency-Domain Analysis:** We perform Fourier analysis or spectrogram visualization of the EMG signals to identify dominant frequency bands. EMG from active muscle typically contains energy from roughly 20 Hz up to about 500 Hz, with most power often in 50–150 Hz range (depending on electrode placement and muscle) ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=data%20glove%2C%20respectively,27%20%2C%2050) ). By analyzing power spectral density, we can design digital filters to remove irrelevant frequencies (e.g., high-frequency noise or low-frequency motion artifacts). For example, if significant power is seen around 50/60 Hz (power line interference), a notch filter may be planned in preprocessing.
* **Cross-Channel and Cross-Gesture Analysis:** If using multiple electrodes, we analyze how different channels correlate and which muscles are most informative for each gesture. This may involve computing correlations or plotting muscle activation maps. Such analysis can reveal if some channels are redundant or very noisy, suggesting dimensionality reduction or sensor adjustments. It also helps verify that each gesture produces a distinguishable pattern of activation across the channels.

This analysis stage establishes a baseline understanding of the data. It guides the selection of filter parameters, normalization approach, and feature extraction strategies. For instance, if analysis shows consistent DC offsets, we ensure high-pass filtering; if certain frequency components carry the discriminative information (e.g., a certain muscle’s spike), we ensure the pipeline preserves those. Essentially, signal analysis tailors the pipeline to the data at hand, which is critical given EMG’s variability across users and setups ([Processing pipeline for EMG signal classification | Download Scientific Diagram](https://www.researchgate.net/figure/Processing-pipeline-for-EMG-signal-classification_fig1_353943547#:~:text=Electromyography%20,computer%20mice%2C%20prosthesis%2C%20robotic%20fin)).

**2. Signal Visualization**

In tandem with analysis, **signal visualization** plays a crucial role in developing and debugging the MIND system. This step involves generating real-time or recorded plots of the EMG signals and intermediate processing outputs. Visualization serves several purposes:

* **Understanding Gesture Signatures:** By visualizing EMG waveforms for different known gestures, we build intuition on their “signature.” For example, one might plot the raw EMG for a hand open vs. a hand closed gesture trial. The closed-hand gesture may show higher amplitude activity in flexor muscle channels, whereas the open-hand might show more in extensors. Visual overlays of multiple trials can confirm consistency. Such plots allow human verification that the data contains separable patterns for each class (a prerequisite for any classifier to succeed).
* **Verifying Preprocessing Effects:** Each processing step (filtering, smoothing, etc.) can be visualized to ensure it behaves as expected. For instance, after applying a band-pass filter, we can plot the filtered signal over the raw to confirm noise (like baseline drift or high-frequency interference) is reduced. Visualizing the rectified and/or smoothed signal (if those are used) can show the envelope of muscle activation, which is often easier to align with gesture events.
* **Real-Time Feedback:** During live tests, a simple graphical interface can display EMG traces and the detected gesture in real time. This feedback is invaluable for adjusting electrode placement or asking the user to repeat a gesture with more effort if the signal is weak. It also helps in identifying latency in the system by observing how quickly the visualized classification responds to changes in the EMG.
* **Feature Space Visualization:** In more advanced analysis, we might visualize feature distributions – for example, plotting the projection of EMG feature vectors (post-GMM or other extraction) in 2D using techniques like PCA. This can illustrate how well-separated the gesture clusters are in feature space, guiding tweaks in feature extraction or model complexity.

Overall, visualization is an ongoing aid for both developers and end-users. It ensures the pipeline’s transformations are intuitive and correct, and it can build user trust by making the “invisible” signals visible. In the PoC, we have used Python tools (like Matplotlib) to generate plots of EMG waveforms and have plans for a simple dashboard that shows the EMG channels and the recognized gesture in real time as the user performs movements.

**3. Preprocessing**

Raw EMG signals require careful **preprocessing** to enhance signal quality and consistency before any feature extraction or classification. In the MIND pipeline, preprocessing includes filtering, segmentation, and signal conditioning techniques commonly used in EMG analysis ([pyemgpipeline: A Python package for electromyography processing](https://www.theoj.org/joss-papers/joss.04156/10.21105.joss.04156.pdf" \l ":~:text=their%20order%2C%20the%20choice%20of,Drake%20%26%20Callaghan%2C%202006)) ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=Using%20high,window%20affects%20the%20accuracy%20of) ):

* **Filtering:** We apply a band-pass filter to each EMG channel to isolate the relevant frequency band of muscle activity. A typical band-pass might be ~20 Hz to 450 Hz, which removes motion artifacts and drift (below 20 Hz) and high-frequency noise (above 450 Hz). For example, a Butterworth band-pass filter can be used, with parameters guided by literature and our signal analysis findings. If powerline interference (50 or 60 Hz) is present, a notch filter at that frequency is also used. In one optimized EMG pipeline, a 5 Hz high-pass cut-off was used to remove baseline wander ([Optimised EMG pipeline for gesture classification - IEEE Xplore](https://ieeexplore.ieee.org/document/9871089/#:~:text=Optimised%20EMG%20pipeline%20for%20gesture,implements%20a%20sliding%20window%20approach)), highlighting the need to remove low-frequency noise. We ensure phase distortion is minimal (using zero-phase filtering if needed) to not disrupt signal timing.
* **Rectification (Optional):** Full-wave rectification (taking absolute value of the signal) is a common step to make EMG signals unipolar. Rectification can simplify subsequent processing by turning the raw oscillatory EMG into a series of positive pulses proportional to muscle activation intensity. This step is optional in MIND depending on the feature approach – rectification is very useful if we plan to compute an envelope or use simple features, but deep learning models might learn relevant features from the raw waveform without rectification. We keep this as a configurable step.
* **Smoothing (Optional):** Sometimes an envelope of the EMG is extracted by smoothing the rectified signal, for example with a moving average or low-pass filter (around 3–10 Hz) to get the muscle activation profile. In classical myoelectric control, this envelope is used for proportional control or onset detection ([Processing pipeline for the EMG signal. In this figure, 10 seconds of... | Download Scientific Diagram](https://www.researchgate.net/figure/Processing-pipeline-for-the-EMG-signal-In-this-figure-10-seconds-of-the-signal-were_fig3_265292311#:~:text=signal%20,green%2C%20dotted)). In our pipeline, if we were implementing a simple control (e.g., directly mapping EMG amplitude to a robotic arm speed), such smoothing would be crucial. However, for pattern classification with deep networks, we often skip explicit smoothing, letting the model handle feature extraction. Still, we may use a mild smoothing in the visualization step to show a clean activation curve.
* **Segmentation (Windowing):** The continuous EMG stream is segmented into overlapping time windows for feature extraction and classification. This is a critical step for real-time decoding. Each window might be on the order of 100 ms (with, say, 50% overlap) – meaning we take 100 ms of recent data to decide what gesture is occurring, and we update every 50 ms. The choice of window length balances latency and signal content; shorter windows give faster response but less data (risking noisy classification), whereas longer windows provide more data at the cost of delay ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=signal,was%20used%20in%20the%20experiments) ). Based on prior experiments, MIND uses an ~100 ms window as a sweet spot (as also suggested by Table 3 in ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=signal,was%20used%20in%20the%20experiments) ) which showed best performance at 100 ms in their case). This windowing also defines the cadence at which classification outputs are produced.
* **Artifact Handling:** Preprocessing can include artifact removal strategies. Large artifacts from electrode cable movement or transient muscle twitches can confound classifiers. Techniques such as thresholding (clipping extreme values) or more advanced methods (independent component analysis or wavelet denoising) could be applied. In our current PoC, we rely on the assumption that filtering and robust classification will handle most artifacts, but future iterations might incorporate explicit artifact detection. Notably, Riviere *et al.* (2021) explored handling artifacts in EMG classification, noting that robust preprocessing improves cross-subject model performance ( [A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC8117925/#:~:text=In%20an%20attempt%20to%20address,methods%20for%20processing%20of%20artifacts) ).

After preprocessing, the EMG data is in a much cleaner and standardized form: band-limited, (optionally rectified), and segmented into analysis frames. This establishes a consistent input for the next stages, reducing variability that is not related to the user’s intended gesture.

**4. Feature Standardization**

Because EMG signal amplitudes can vary greatly between sessions or subjects (due to electrode placement, skin conductivity, individual muscle strength, etc.), **feature standardization** is employed to normalize the data. Standardization ensures that the subsequent feature extraction and classification operate on a consistent scale, preventing one channel or trial from dominating merely due to higher magnitude.

In MIND, we perform standardization at two levels:

* **Within-Channel Normalization:** For each EMG channel (each electrode), we normalize the signal values to have zero mean and unit variance (z-score normalization) over a baseline recording or the training dataset. This is done after filtering and segmentation, on the features that will go into the classifier. If we are feeding raw time-series into a CNN, we might normalize each channel of the time-series. If we compute feature vectors (like RMS or GMM posterior probabilities), we normalize each feature dimension. This step addresses the issue that one muscle’s EMG might inherently produce larger voltage swings than another’s; after normalization, the classifier can pay equal attention to all channels without bias towards higher-amplitude ones ( [A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC8117925/#:~:text=In%20an%20attempt%20to%20address,methods%20for%20processing%20of%20artifacts) ).
* **Across-Subject or Session Normalization:** If the system is used by multiple users or across multiple sessions for one user, we apply normalization to account for global changes. Riviere *et al.* (2021) specifically noted that a data normalization pipeline was critical to handle classification across different subjects ( [A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC8117925/#:~:text=In%20an%20attempt%20to%20address,methods%20for%20processing%20of%20artifacts) ). In practice, this can involve scaling each channel’s data by a reference contraction value (e.g., maximum voluntary contraction or MVC of that user) or by statistics from a calibration recording. MIND incorporates a brief calibration phase where a user performs a couple of reference gestures; from this, we derive scaling factors. For example, if a user gives a full-strength fist clench, we can use that as 100% activation for scaling. This way, a percentage of maximum contraction becomes the normalized unit, making the features more comparable across users.

Standardization also helps the training of our machine learning models. Neural networks converge faster and perform better when inputs are normalized, preventing issues like one feature gradient overshadowing others. In summary, feature standardization in our pipeline acts to homogenize the data, ensuring that differences between gestures are due to actual signal pattern differences and not trivial amplitude shifts or sensor gains. This greatly aids the generalization of the classifier, especially important if MIND is to be deployed to new users or devices without extensive retraining.

**5. GMM-Based Feature Extraction**

As an intermediate step before final classification, we explore **Gaussian Mixture Model (GMM)-based feature extraction**. The idea is to employ an unsupervised learning technique (GMM clustering) to model the underlying structure of the EMG feature space, and then use the parameters or outputs of that model as inputs to the classifier. There are a few motivations for this step:

* **Dimensionality Reduction and Feature Discovery:** A GMM can capture the distribution of multi-channel EMG data by representing it as a mixture of Gaussian components. For example, suppose we transform each EMG window into a basic feature vector (such as root-mean-square value on each channel, or a set of time-domain statistics). A GMM trained on these feature vectors would effectively cluster the EMG data into a certain number of components. These components might correspond to different muscle activation patterns. We could then use the posterior probabilities of each GMM component (for a given input window) as a new feature vector. This drastically reduces dimensionality (e.g., from many sensor features down to, say, 5–10 mixture activation features) while retaining a nonlinear combination of original features. The GMM is thus acting as a feature transformer.
* **Robustness and Smoothing:** GMMs provide a probabilistic encoding of the signal features. If a particular gesture’s EMG pattern has some variability, the mixture model can accommodate it by having broad or multiple components representing that class’s feature distribution. GMM-based classification has historically been noted for its robustness to signal variability and noise ([A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses - PubMed](https://pubmed.ncbi.nlm.nih.gov/16285383/#:~:text=segmentation%20of%20the%20data%2C%20and,classification%20with%20low%20computational%20load)). For instance, Huang *et al.* (2005) demonstrated that a GMM-based limb motion classifier achieved high accuracy and was computationally efficient compared to neural networks, partly because it effectively modeled the feature distributions with limited data ([A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses - PubMed](https://pubmed.ncbi.nlm.nih.gov/16285383/#:~:text=This%20paper%20introduces%20and%20evaluates,performance%20of%20the%20GMM%20is)) ([A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses - PubMed](https://pubmed.ncbi.nlm.nih.gov/16285383/#:~:text=segmentation%20of%20the%20data%2C%20and,classification%20with%20low%20computational%20load)). By extracting GMM features, we hope to inherit some of that robustness.
* **Hybrid Classification Approach:** The outputs of the GMM (either cluster IDs or posterior probabilities) can be fed into the deep learning classifier alongside or instead of raw features. This creates a hybrid model where the deep network doesn’t have to learn the cluster structure from scratch – it’s given a head start by the GMM. We optionally experiment with using GMM as a standalone classifier as well: each Gaussian component can be associated with a gesture class (via training labels), forming a generative classifier. While our primary path is deep learning, having a GMM classifier in the loop provides a lightweight alternative that could run on low-power hardware with lower latency, useful for comparisons.

**Implementation:** During training, we take all the preprocessed, standardized feature vectors from the training set and fit a GMM (using, for example, the Expectation-Maximization algorithm) with a certain number of components. The number of Gaussian components can be tuned; it might be set to the number of classes \* k (to allow multiple clusters per gesture) or determined by cross-validation. Once fitted, for each input window we compute the posterior probability for each Gaussian (i.e., given the feature vector, the probability it belongs to component j for all j). This probability vector (summing to 1 across components) is then used as the feature representation of that window. If the GMM is class-specific (e.g., a separate GMM trained for each class), then the likelihood under each class’s GMM could serve as features.

One special use of GMM in EMG is detection of muscle activation onsets in continuous signals via sequential probability modeling ([Robust Muscle Activity Onset Detection Using an Unsupervised ...](https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0127990#:~:text=Robust%20Muscle%20Activity%20Onset%20Detection,to%20detect%20muscle%20activity%20onsets)). While MIND currently focuses on segmented classification, future enhancements might use a sequential GMM to detect when a new gesture begins (i.e., unsupervised onset detection ([Robust Muscle Activity Onset Detection Using an Unsupervised ...](https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0127990#:~:text=Robust%20Muscle%20Activity%20Onset%20Detection,to%20detect%20muscle%20activity%20onsets))) and trigger classification accordingly.

In summary, the GMM feature extraction step provides an **optional bridge** between raw signal features and deep learning. It embeds domain knowledge about EMG feature distributions into the pipeline. We will evaluate whether this extra layer of processing improves overall accuracy and stability. If it does, it can be kept as part of the pipeline; if the deep learning alone proves sufficient, the GMM step can be bypassed, highlighting the modular nature of MIND’s design.

**6. Deep Learning Classification**

The final stage of the pipeline is **deep learning classification**, where the processed EMG data (possibly augmented with GMM features) is fed into a trained neural network that outputs the recognized gesture or motion intent. We investigate several network architectures to decode EMG, namely **1D Convolutional Neural Networks (CNN1D)**, **Long Short-Term Memory (LSTM) networks**, and **Temporal Convolutional Networks (TCN)**. Each of these has shown promise in EMG and neural signal decoding literature, and we also explore hybrid models combining their strengths.

**Convolutional Neural Network (1D CNN):** A 1D CNN applies learnable convolution filters across the time dimension of the EMG signal (and across channels). This allows the network to automatically learn features such as specific waveforms or frequency patterns indicative of each gesture. For example, a CNN might learn a filter that detects a burst of 150 Hz activity on an extensor muscle channel – a feature that could correspond to a particular finger extension gesture. CNNs are great at local feature extraction and are invariant to slight shifts in the timing of patterns. In MIND, a CNN1D model would take as input the multi-channel EMG window (e.g., shape [channels × time\_samples]) and pass it through a series of convolutional layers and nonlinear activations. The final layers are typically fully connected neurons or global pooling leading to a softmax output over gesture classes. CNNs have been successfully used in EMG pattern recognition tasks; for instance, studies have used CNNs to classify hand gestures from multi-channel EMG with high accuracy ( [A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC8117925/#:~:text=Results%3A) ). CNNs can also be combined with other networks as feature extractors. Our pipeline can use a pure CNN or a CNN as the first stage (to extract a feature map per window), feeding into a temporal model like an LSTM.

**Recurrent Neural Network – LSTM:** LSTM networks are a type of recurrent neural network (RNN) well-suited for sequence data because they maintain an internal memory state. In EMG decoding, LSTMs can capture the temporal evolution of muscle signals over multiple windows or continuous time, learning dependencies such as the progression of a gesture. For example, a complex gesture might involve a sequence of muscle activations; a pure CNN that looks only at one window might miss context, whereas an LSTM can learn that after a fist clench often comes a release, etc., if modeling time sequences. In our use, we employ LSTM layers to process either raw time-series or sequences of feature vectors across windows. An LSTM cell, with its input, output, and forget gates, can decide how much of past information to carry forward ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=LSTM%20is%20a%20kind%20of,The%20update%20of%20the) ) ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=where%20%20represents%20the%20sigmoid,activation%20threshold%20of%20the%20gate) ). This is helpful in smoothing classifications over time and handling minor timing variations. LSTMs have shown strong performance in EMG classification; a hybrid CNN-LSTM model in one study achieved ~89.5% accuracy in cross-subject EMG pattern classification, outperforming standalone models ( [A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC8117925/#:~:text=Results%3A) ). We anticipate that an LSTM-based classifier in MIND will help generalize across time and reduce jitter in predictions. For real-time operation, a unidirectional LSTM (processing past to present) is used, though a bidirectional LSTM can be trained offline for evaluation purposes when sequence future context is available.

**Temporal Convolutional Network (TCN):** TCNs are a relatively newer class of models that use convolutional layers with dilation (skipped connections) to achieve long receptive fields, effectively capturing long-range temporal patterns with convolution rather than recurrence. A TCN can be thought of as an 1D CNN that can see far back in time due to exponentially dilated kernels, and it often includes residual connections to ease training of deep networks ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=When%20dealing%20with%20complex%20prediction,is%20shown%20in%20Figure%205) ) ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=parameters%2C%20leading%20to%20an%20increase,is%20shown%20in%20Figure%205) ). TCNs have gained attention because they can outperform RNNs like LSTMs on certain sequence tasks while being easier to parallelize (since they don’t have recurrent dependencies). For EMG, a TCN could learn multi-scale temporal features – e.g., a small dilation filter might capture fine spikes, while a large dilation captures broader activation patterns. Zhang *et al.* (2024) proposed a hybrid TCN-LSTM model for continuous wrist angle estimation from sEMG, noting that the TCN module excels at extracting local deep features and the LSTM captures longer-term dependencies ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=continuous%20estimation%20of%20wrist%20motion,WUD%29%2C%20and) ). Inspired by that, we include TCN in our model exploration. A standalone TCN can be used for classification by feeding the EMG window (or sequence of windows) into a stack of dilated convolution layers and outputting class probabilities. The advantage is that TCN, with sufficient layers and dilation, can have a receptive field covering several hundred milliseconds or more, similar to an LSTM’s memory, but with potentially less risk of vanishing gradients and simpler training dynamics ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=processing,of%20which%20are%20discussed%20below) ).

**Hybrid Models:** The ultimate performance may come from combining these approaches. As mentioned, CNNs can serve as front-end feature extractors for either an LSTM or TCN. LSTMs and TCNs can even be combined – e.g., a TCN feeding into an LSTM (TCN-LSTM) as in ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=continuous%20estimation%20of%20wrist%20motion,WUD%29%2C%20and) ), where the TCN handles immediate feature extraction from raw signals and the LSTM refines the sequence prediction with memory. This hybrid achieved better accuracy in continuous regression than either alone ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=wrist,improvement%20over%20the%20LSTM%20model) ). In classification context, one could also envision an ensemble where a CNN, an LSTM, and a TCN each vote on the gesture, combining their predictions to improve robustness.

( [Figure - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/figure/sensors-24-05631-f007/) ) *Figure 2: An example deep learning architecture for EMG decoding, combining a Temporal Convolutional Network and an LSTM (* [*A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC*](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=Next%2C%20the%20extracted%20deep%20features,after%2010%20rounds%20of%20training) *). In this hybrid TCN-LSTM model, the TCN layers (blue) extract local temporal features from input EMG (possibly alongside manual features, pink), which are then concatenated (feature stitching) and fed into stacked LSTM layers (green) for learning long-term dependencies. A final linear layer produces the output (e.g., predicted class or continuous value). Such architectures leverage the strengths of convolutional feature extraction and recurrent temporal modeling (* [*A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC*](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=The%20integration%20of%20a%20TCN,LSTM%20model%20is%20anticipated) *), yielding improved decoding performance over single models.*

For the MIND prototype, we are implementing and testing the following configurations: a pure CNN1D classifier, a pure LSTM classifier, a CNN-LSTM hybrid (with one or two convolutional layers feeding into one or two LSTM layers), and a TCN (with residual blocks). These networks are trained on a labeled dataset of EMG signals where the user performs a set of known gestures. Training uses supervised learning, minimizing a cross-entropy loss between the predicted gesture probabilities and the true labels. We utilize techniques like early stopping and data augmentation (e.g., adding slight noise or random time shifts to EMG windows) to improve generalization.

Notably, deep learning can **automatically learn features** that previously required manual design. For instance, older methods rely on time-domain statistics (mean absolute value, zero-crossing rate, waveform length, etc.) or frequency coefficients as features ([A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses - PubMed](https://pubmed.ncbi.nlm.nih.gov/16285383/#:~:text=conducted%20on%20a%2012%20subject,of%20motion%20classification%20with%20low)). In contrast, a CNN might learn a combination of those implicitly. Our pipeline’s allowance for GMM or manual features is complementary – it provides a fallback or an additional input, but the deep networks can, in principle, learn directly from preprocessed EMG streams.

In real-time operation, once the network outputs a gesture class for each window, that output can be smoothed or debounced (to avoid rapid flipping between classes due to noise). We implement a majority vote or a short-term memory on the output: e.g., requiring 2 out of 3 consecutive windows to agree before changing the detected gesture. This is analogous to a majority vote post-processing which has been shown to improve stability in EMG classification ([A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses - PubMed](https://pubmed.ncbi.nlm.nih.gov/16285383/#:~:text=conducted%20on%20a%2012%20subject,of%20motion%20classification%20with%20low)). However, the LSTM inherently provides some smoothing due to its memory.

By the end of this stage, the system produces a high-level command: e.g., “Gesture 1 (fist)” or “Gesture 2 (open hand)” at each time step, or possibly a continuous estimate of movement if we were doing regression. These commands are then used to drive the application (the Unity3D arm). We emphasize that no performance results are included here (as our focus is the methodology), but the design is informed by literature where deep learning models have significantly improved EMG decoding accuracy and robustness ( [A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC8117925/#:~:text=Results%3A) ) ([An Affordable 3D-printed Open-Loop Prosthetic Hand Prototype with Neural Network Learning EMG-Based Manipulation for Amputees | SHS Web of Conferences](https://www.shs-conferences.org/articles/shsconf/abs/2022/09/shsconf_etltc2022_03004/shsconf_etltc2022_03004.html#:~:text=the%20signals,performance%20of%20the%20decoding%20hand)).

**Unity3D Robotic Arm Simulation for Output Visualization**

A key component of the MIND proof-of-concept is the **Unity3D robotic arm simulation**, which serves as a visualization and testing environment for the decoded EMG commands. Rather than immediately interfacing with a physical prosthetic (which could be costly or risk hardware damage during development), the simulation provides a safe and flexible platform to observe how well the decoded gestures can control a device.

**Simulation Setup:** We have created a virtual robotic arm model in Unity3D, a popular game engine that is increasingly used for robotics simulation due to its powerful 3D rendering and physics capabilities ([Robotics simulation in Unity is as easy as 1, 2, 3](https://unity.com/blog/engine-platform/robotics-simulation-is-easy-as-1-2-3#:~:text=Robotics%20simulation%20in%20Unity%20is,effective%2C%20and%20easier%20than%20ever)). The virtual arm is modeled with realistic joints (shoulder, elbow, wrist, and a simple hand or gripper). It can be either an anthropomorphic arm or a simplified industrial manipulator, depending on demonstration needs. The Unity environment runs in parallel with the EMG decoding pipeline and receives commands through a communication interface (such as a TCP socket or Unity’s networking). Whenever the MIND system classifies a gesture, it sends a command to Unity, which then triggers an animation or movement of the robot arm.

**Gesture-to-Motion Mapping:** We define a mapping from each decoded gesture to a specific arm movement or pose in Unity. For example, if our system distinguishes gestures like **“Fist clench”**, **“Hand open”**, **“Wrist flexion”**, **“Wrist extension”**, etc., these can be mapped as:

* *Fist Clench* – close the robotic hand (gripper closes).
* *Hand Open* – open the robotic hand.
* *Wrist Flexion* – rotate the wrist joint downward.
* *Wrist Extension* – rotate the wrist upward.
* *Forearm Supination/Pronation* (if included) – rotate the forearm segment accordingly. These mappings are configurable. In a demo for a prosthetic hand, one might use more fine-grained finger-level gestures (if the classifier can support them) to control individual fingers of a robotic hand model. In an industrial robot context, gestures might map to predefined motions (e.g., point left, point right).

**Visualization and Feedback:** When the Unity3D simulation receives a command, the corresponding movement is visualized instantly. This allows observers (and the user) to see the outcome of their muscle activations. If the system is working correctly, the virtual arm should mimic the user’s intended motion with minimal delay. For example, as the user closes their real hand (generating EMG pattern for fist), they will see the robotic hand on screen closing. This visual feedback can help the user calibrate their effort and also helps developers spot misclassifications (if the arm does something unexpected, we know the decoding was wrong at that moment).

**Simulation Benefits:** Using Unity3D offers several benefits:

* *Safety:* No physical harm can occur from a misinterpreted signal (important when testing new decoding algorithms that might send erratic commands initially).
* *Rapid Iteration:* We can modify the robot model, add new control logic, or change gesture mappings easily in software without hardware changes.
* *Analytics:* Unity can log the received commands and ground-truth movements, which helps in evaluating the decoding performance qualitatively (e.g., how smooth or jittery the arm motion is, how often wrong gestures are displayed).
* *Immersion:* For eventual users, seeing a responsive virtual limb can be motivating and provides a glimpse of the end goal of a prosthetic device under neural control.

There is precedent for this approach in research – for instance, Ganiev *et al.* (2016) created a virtual robotic arm control system driven by Myo armband EMG for an amputee training scenario ([Applications based on electromyography sensors | ITM Web of Conferences](https://www.itm-conferences.org/articles/itmconf/ref/2019/06/itmconf_iccmae2018_02007/itmconf_iccmae2018_02007.html#:~:text=7.%20A.%20Ganiev%2C%20H.,beyond)). Their system allowed an amputee to practice controlling a prosthetic hand in a virtual environment by simply moving their arm and contracting muscles, demonstrating the effectiveness of Unity3D as a training and visualization tool for myoelectric control. Similarly, our Unity3D interface can be seen as a testing ground that could transition into a training simulator for end-users: users could learn how to contract the correct muscles to achieve desired movements by watching the virtual arm, before moving on to a real prosthetic.

**Unity3D Implementation Details:** We use Unity’s built-in **Inverse Kinematics (IK)** for positioning the arm when needed. For example, if a gesture is supposed to move the arm to a certain position, Unity’s IK solver can handle joint angles. For simpler gesture->pose mappings, we directly set joint rotations. The system is designed to run in real time, with Unity rendering at 60+ FPS and the EMG pipeline providing updates ~10–20 times per second (depending on window length). This ensures a smooth visual experience.

The Unity simulation is not only a visual aid but also a stepping stone to real hardware control. Once we validate that the MIND system can consistently drive the virtual arm correctly, the same high-level commands can be issued to a real robotic arm or prosthetic hand (with appropriate hardware API). Unity simply mimics what a real arm would do, so any logic developed for command timing, smoothing, or switching between gestures will apply to physical devices as well.

In summary, the Unity3D robotic arm interface is an integral part of the PoC, demonstrating the **closed-loop functionality** of MIND – from user’s muscles to virtual action. It showcases the potential impact of the system: e.g., an amputee could control a prosthetic arm naturally, or a surgeon could tele-operate a robot via muscle signals. This aligns well with the goals of a tech community or VC fellowship demo, as it makes the abstract decoding technology tangible and compelling.

**Future Work and Deployment Vision**

The current MIND system is a foundational step towards a practical neural interface for motor control. While the proof-of-concept validates the core ideas (signal pipeline, decoding algorithms, and virtual control), several advancements are envisioned to transition this system from the lab to real-world use. Below we outline the future work and our vision for deployment:

* **Integrating Physical Prosthetic Hardware:** The ultimate goal is to control real prosthetic or robotic limbs. As a next step, we will interface MIND with a physical robotic arm or hand. Many modern prosthetic hands (e.g., open-source 3D-printed hands or commercial devices) accept control signals either as high-level commands (grip types) or low-level motor signals. We plan to use an adapter module to translate the MIND classifier’s output into the format required by a prosthetic hand. A successful integration will demonstrate that our decoded gestures can perform useful tasks (grasping objects, pointing, etc.) in the physical world.
* **Real-Time Performance Tuning:** While performance metrics were not the focus of this paper, they will be critical for deployment. We will rigorously evaluate and improve the true positive rate, false activation rate, and especially latency of the system. Our target is a control latency under ~200 ms from muscle activation to device response, which is in line with human reaction times and would feel natural. This may involve optimizing the code (e.g., using C++ or embedded implementations for portions of the pipeline, or deploying the neural network on a GPU or specialized accelerator) and possibly simplifying the models if needed for speed. Fortunately, methods like TCNs and optimized CNNs can be very fast, and GMM is computationally light ([A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses - PubMed](https://pubmed.ncbi.nlm.nih.gov/16285383/#:~:text=linear%20perceptron%20network%2C%20and%20a,classification%20with%20low%20computational%20load)), so we expect to meet real-time requirements on modest hardware.
* **Robustness Across Users and Conditions:** To be broadly useful, the system should work for different users without extensive retraining. We plan to expand our training dataset to include multiple participants and a variety of usage conditions (different days, electrode placements, levels of muscle fatigue, etc.). Techniques like transfer learning or few-shot adaptation will be explored so that a new user can calibrate the system with minimal data (perhaps a one-minute calibration routine). The standardized pipeline and the GMM feature layer could assist in this, by normalizing differences and providing a common feature space ( [A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC8117925/#:~:text=In%20an%20attempt%20to%20address,methods%20for%20processing%20of%20artifacts) ). In terms of environmental robustness, we will test against interference (other electronics, movement artifacts when the user is mobile) and incorporate adaptive filtering if necessary (for example, adjusting notch filters or using spectral subtraction if noise is detected).
* **Additional Sensor Modalities:** Although MIND is centered on EMG, the architecture could integrate other inputs to improve decoding. Inertial measurement unit (IMU) data from the arm, for instance, could help distinguish certain gestures or provide context (if the arm is moving vs. static). Some myoelectric control systems combine EMG with accelerometers or gyroscopes for better accuracy ([Applications based on electromyography sensors](https://www.itm-conferences.org/articles/itmconf/ref/2019/06/itmconf_iccmae2018_02007/itmconf_iccmae2018_02007.html#:~:text=myo%20armband%20for%20the%20selfmanipulation,Eng)). Another modality is EEG or peripheral nerve signals, if available, to complement muscle signals for a more direct neural interface. The pipeline could be extended to fuse such data, perhaps adding separate preprocessing for each and combining features before classification.
* **Improved Deep Learning Models:** Ongoing advances in AI could further boost performance. For example, *transformer*-based models have revolutionized sequence processing in other domains and could be applied to EMG decoding, potentially capturing even longer-term dependencies than LSTM/TCN. We will monitor research in deep learning for BCI and adopt promising techniques. Also, techniques like *data augmentation* (e.g., using generative models to create synthetic EMG data ([Parkinson's Disease EMG Data Augmentation and Simulation with ...](https://pmc.ncbi.nlm.nih.gov/articles/PMC7248755/#:~:text=,DCGANs%29%20and%20Style%20Transfer))) or *unsupervised pre-training* on unlabeled EMG could strengthen the model. The GMM step itself could be replaced or augmented by more powerful autoencoders or clustering algorithms that better capture non-Gaussian structure.
* **User Interface and Feedback:** For a deployable product, the system should include a user-friendly interface. This means not only the visual feedback (which we have via Unity or eventually AR/VR visualization of a phantom limb), but also calibration routines, error feedback (if the system isn’t confident, it should inform the user rather than perform a wrong action), and possibly haptic feedback. For instance, a wearable could vibrate to signal which gesture was recognized, giving the user a feedback loop to correct their muscle signals. Our vision includes a training program for users to practice and adapt to the system, similar to how one would learn to use a new prosthetic; the Unity simulation can be extended into a training game, for example, where users try to hit targets by performing the correct muscle gesture.
* **Miniaturization and Wearability:** Currently, the PoC might rely on a PC for processing, but eventually the goal is to embed this into a wearable unit. Advances in microcontrollers and edge AI devices (like NVIDIA Jetson Nano, Google Coral, etc.) make it feasible to run complex models on small form factors. We plan to port the pipeline to an embedded platform that can be worn on the arm alongside the sensors, making the system self-contained. A possible deployment vision is an arm sleeve that contains dry electrodes and an on-board processor that streams high-level commands to any paired device (be it a prosthetic, computer or AR/VR system). Minimizing power consumption and ensuring wireless connectivity (Bluetooth Low Energy or similar) will be part of this engineering effort.
* **Validation in Real-World Tasks:** Finally, deployment means testing in real use-cases. We envision collaborating with prosthetics researchers or clinicians to test MIND with amputee subjects for tasks like object manipulation, or with healthy subjects using it as an input for AR/VR (imagine controlling virtual objects or a drone with hand gestures decoded from EMG). Success in these scenarios – measured by task completion, intuitiveness, and user satisfaction – will be the true validation of the system. Notably, previous works have shown that multi-gesture myoelectric control can significantly improve prosthesis functionality for amputees ([An Affordable 3D-printed Open-Loop Prosthetic Hand Prototype with Neural Network Learning EMG-Based Manipulation for Amputees | SHS Web of Conferences](https://www.shs-conferences.org/articles/shsconf/abs/2022/09/shsconf_etltc2022_03004/shsconf_etltc2022_03004.html#:~:text=Despite%20the%20advancement%20of%20prosthetic,hand%20to%20operate%20from%20a)) ([An Affordable 3D-printed Open-Loop Prosthetic Hand Prototype with Neural Network Learning EMG-Based Manipulation for Amputees | SHS Web of Conferences](https://www.shs-conferences.org/articles/shsconf/abs/2022/09/shsconf_etltc2022_03004/shsconf_etltc2022_03004.html#:~:text=to%20control%20the%20prosthetic%20hand,LSTM)). Our system aims to contribute to that goal with a modern approach that could potentially offer higher accuracy and ease-of-use by virtue of deep learning and careful design.

**Conclusion**

In this white paper, we presented the design of MIND: a Motor Interface for Neural Decoding, focusing on a proof-of-concept implementation that decodes surface EMG signals into control commands for a robotic arm. We detailed a six-step signal processing pipeline – from initial analysis and visualization of EMG signals to preprocessing, normalization, GMM-based feature extraction, and final classification via advanced deep learning models (CNN, LSTM, TCN). By incorporating both classical and modern techniques, the system is able to handle the complexities of EMG signals and translate them into meaningful actions. We described how the decoded outputs can be visualized in real time through a Unity3D simulation, effectively demonstrating closed-loop control of a virtual prosthetic limb. This not only serves as a powerful demo for stakeholders (researchers, engineers, and investors alike) but also lays the groundwork for user training and hardware integration.

The MIND system represents a convergence of biomedical engineering and AI: it takes the rich but challenging biosignals from muscle, uses signal processing to accentuate the latent information, and applies learning algorithms to interpret the user’s intent. The use of minimally invasive surface sensors aligns with the practical need for non-surgical solutions in assistive technology ([An Affordable 3D-printed Open-Loop Prosthetic Hand Prototype with Neural Network Learning EMG-Based Manipulation for Amputees | SHS Web of Conferences](https://www.shs-conferences.org/articles/shsconf/abs/2022/09/shsconf_etltc2022_03004/shsconf_etltc2022_03004.html#:~:text=model,With%20the%20use%20of%20Recurrent)). Our extensive references to existing literature underscore that the approach is built on proven concepts – from the robustness of GMM classifiers in earlier myoelectric control research ([A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses - PubMed](https://pubmed.ncbi.nlm.nih.gov/16285383/#:~:text=segmentation%20of%20the%20data%2C%20and,classification%20with%20low%20computational%20load)) to the cutting-edge performance of hybrid deep networks in recent EMG studies ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=wrist,improvement%20over%20the%20LSTM%20model) ) ( [A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC8117925/#:~:text=Results%3A) ). What is novel is the particular combination and application of these tools in the MIND system, and the emphasis on creating an end-to-end pipeline suitable for real-world deployment.

As we move forward, the vision for MIND is to evolve into a deployable interface that can empower users with intuitive control over assistive devices, be it a prosthetic arm restoring lost function or a remote robot extending human capability. The ongoing and future work will focus on refining the system’s accuracy, speed, and generalization, as well as packaging it into a user-friendly form factor. Success in these endeavors could contribute significantly to the field of rehabilitation and human-computer interaction, reducing the gap between intent and action for individuals through the power of neural decoding.

In conclusion, the MIND proof-of-concept has demonstrated the feasibility of decoding muscle signals for complex gesture control using a structured pipeline and modern AI methods. With further development, it holds promise for real-world impact, translating laboratory insights in EMG signal processing and deep learning into tangible improvements in people’s lives. We aim to carry this work through to a functional system that showcases how merging neuroscience, engineering, and computer science can create a next-generation motor interface – one that is precise, responsive, and above all, accessible.

**References (Key Literature and Tools)**

1. R. Merletti and P. Parker, *Electromyography: Physiology, Engineering, and Noninvasive Applications*. John Wiley & Sons, 2004. (Fundamental text on EMG signal properties and processing) ([Applications based on electromyography sensors | ITM Web of Conferences](https://www.itm-conferences.org/articles/itmconf/ref/2019/06/itmconf_iccmae2018_02007/itmconf_iccmae2018_02007.html#:~:text=Proceedings%20in%20Manufacturing%20Systems%2C%2013%2C,En%2C%2019%2C%20260%E2%80%93270%20%282011))
2. Y. Huang *et al*., “A Gaussian mixture model based classification scheme for myoelectric control of powered prostheses,” *IEEE Trans. Biomed. Eng.*, vol. 52, no. 11, pp. 1801-1811, 2005. (Demonstrated high accuracy and low computational load of GMM for EMG pattern classification) ([A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses - PubMed](https://pubmed.ncbi.nlm.nih.gov/16285383/#:~:text=This%20paper%20introduces%20and%20evaluates,performance%20of%20the%20GMM%20is)) ([A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses - PubMed](https://pubmed.ncbi.nlm.nih.gov/16285383/#:~:text=segmentation%20of%20the%20data%2C%20and,classification%20with%20low%20computational%20load))
3. C. Cipriani *et al*., “Online Myoelectric Control of a Dexterous Hand Prosthesis by Transradial Amputees,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 19, no. 3, pp. 260–270, 2011. (Early demonstration of real-time EMG-based prosthetic hand control in amputees) ([Applications based on electromyography sensors | ITM Web of Conferences](https://www.itm-conferences.org/articles/itmconf/ref/2019/06/itmconf_iccmae2018_02007/itmconf_iccmae2018_02007.html#:~:text=2,Google))
4. M. Ganiev *et al*., “Study on virtual control of a robotic arm via a Myo armband for the self-manipulation of a hand amputee,” *Int. J. Appl. Eng. Res.*, vol. 11, 2016. (Used the Myo EMG armband and Unity3D to let an amputee control a virtual arm) ([Applications based on electromyography sensors | ITM Web of Conferences](https://www.itm-conferences.org/articles/itmconf/ref/2019/06/itmconf_iccmae2018_02007/itmconf_iccmae2018_02007.html#:~:text=7.%20A.%20Ganiev%2C%20H.,beyond))
5. C. N. Riviere *et al*., “A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG,” *IEEE Trans. Med. Robot. Bionics*, vol. 3, no. 1, pp. 37–47, 2021. (Hybrid CNN-LSTM model for EMG classification; highlights need for normalization across subjects) ( [A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC8117925/#:~:text=In%20an%20attempt%20to%20address,methods%20for%20processing%20of%20artifacts) ) ( [A Deep Learning Model for Automated Classification of Intraoperative Continuous EMG - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC8117925/#:~:text=Results%3A) )
6. Z. Zhang *et al*., “A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles,” *Sensors*, vol. 24, no. 17, 5631, 2024. (Combined Temporal Convolutional Network and LSTM to predict continuous joint angles from EMG, inspiring our use of TCN+LSTM) ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=continuous%20estimation%20of%20wrist%20motion,WUD%29%2C%20and) ) ( [A Novel TCN-LSTM Hybrid Model for sEMG-Based Continuous Estimation of Wrist Joint Angles - PMC](https://pmc.ncbi.nlm.nih.gov/articles/PMC11397992/#:~:text=wrist,improvement%20over%20the%20LSTM%20model) )
7. *pyemgpipeline* (Devaprakash *et al*., 2022) – Python package for EMG processing following recommended conventions ([pyemgpipeline: A Python package for electromyography processing](https://www.theoj.org/joss-papers/joss.04156/10.21105.joss.04156.pdf" \l ":~:text=As%20stated%20by%20Stegeman%20%26,decades%20of%20practical%20experience%2C%20we)) ([pyemgpipeline: A Python package for electromyography processing](https://www.theoj.org/joss-papers/joss.04156/10.21105.joss.04156.pdf" \l ":~:text=EMG%20signal%20processing%20conventions%20into,within%20the%20bandpass%20filtering%20step)). (Guided our pipeline design for filtering, etc., though we extended beyond with GMM and deep learning)
8. Unity Technologies, “Robotics simulation in Unity is as easy as 1, 2, 3,” Unity Blog, 2020. (Overview of Unity3D capabilities for robotics simulations) ([Robotics simulation in Unity is as easy as 1, 2, 3](https://unity.com/blog/engine-platform/robotics-simulation-is-easy-as-1-2-3#:~:text=Robotics%20simulation%20in%20Unity%20is,effective%2C%20and%20easier%20than%20ever))
9. A. G. Oskoei and H. Hu, “Support vector machine-based classification scheme for myoelectric control,” *IEEE Trans. Biomed. Eng.*, vol. 55, no. 8, pp. 1956–65, 2008. (Classic approach to EMG gesture classification with SVM, provided baseline concepts that deep learning now often surpasses)
10. D. Farina *et al*., “Myoelectric Control for Upper Limb Prostheses: Current Status and Future Challenges,” *IEEE Trans. Biomed. Eng.*, vol. 65, no. 11, pp. 2757–2765, 2018. (Review paper on myoelectric prosthetic control, covering advanced interface technologies and highlighting the potential of enhanced signal processing and learning algorithms in the field)