NeuroCalc: Invariant Dual-Hand Arithmetic Recognition via Spatio-Temporal Graph Convolutional Networks

Antoine Besson
Telecom Paris
antoine.besson@telecom-paris.fr

January 22, 2026

Abstract

We present NeuroCalc, a computer vision system capable of recognizing dynamic arithmetic operations performed by human hands in real-time. Unlike traditional CNN-based approaches that rely on pixel-level features, our method leverages a geometric modeling approach using Spatio-Temporal Graph Convolutional Networks (ST-GCN). We introduce a novel "Unified Scene Projection" technique that canonicalizes dual-hand topology into a rotation-invariant manifold, effectively decoupling the semantic gesture from the user's global orientation. The system achieves $> 93\%$ accuracy on a custom dataset of mathematical operators and digits, operating with low latency on standard CPU hardware through an asynchronous strided inference engine.

1 Introduction

Hand gesture recognition is a fundamental problem in Human-Computer Interaction (HCI). While static sign language recognition is well-studied, the recognition of mathematical operations adds a temporal complexity: signs like "Plus" (+) and "Times" ($\times$) are geometrically similar but topologically distinct, often requiring the coordination of two hands.

Standard Convolutional Neural Networks (CNNs) struggle with this task due to the high variance in hand orientation and camera distance. Training a CNN to be invariant to these transformations requires massive datasets. We propose a geometric approach that solves invariance analytically rather than statistically. By treating the hands as a dynamic graph of 42 vertices, we project the input data into a canonical reference frame before it reaches the neural network, drastically reducing the complexity of the learning manifold.

2 Methodology

2.1 Architectural Overview

The NeuroCalc pipeline consists of four modular stages:

1. **Sensor Abstraction:** Extraction of raw 3D landmarks via MediaPipe.
2. **Geometric Kernel:** Canonicalization of the dual-hand scene.
3. **Neural Inference:** Feature extraction via ST-GCN.
4. **Logic Solver:** Temporal smoothing and Reverse Polish Notation (RPN) evaluation.

2.2 The Dual-Hand Geometric Kernel

A critical contribution of this work is the handling of dual-hand inputs. Raw coordinates $P \in \mathbb{R}^{42\times 3}$ are sensitive to the user's position relative to the camera. We implement a *Unified Scene Projection* that anchors the coordinate system to the dominant hand's wrist.

Let H_L and H_R be the sets of landmarks for the left and right hands. We define the anchor A as H_R^{wrist} if the

right hand is present, otherwise H_L^{wrist}. We construct a local basis $\mathcal{B} = \{\vec{x}, \vec{y}, \vec{z}\}$ using Gram-Schmidt orthogonalization:

$$\vec{v}_{primary} = H^{middle\_mcp} - A \tag{1}$$

$$\vec{y} = \frac{\vec{v}_{primary}}{||\vec{v}_{primary}||} \tag{2}$$

$$\vec{z} = \frac{\vec{y} \times \vec{v}_{secondary}}{||\vec{y} \times \vec{v}_{secondary}||} \tag{3}$$

$$\vec{x} = \vec{y} \times \vec{z} \tag{4}$$

The entire scene is then projected via the rotation matrix $R = [\vec{x}, \vec{y}, \vec{z}]^T$ and scaled by the hand size σ:

$$P_{canonical} = \frac{(P_{raw} - A) \cdot R^T}{\sigma} \tag{5}$$

This transformation guarantees that a gesture performed at 45° or at 2 meters depth yields the exact same numerical input to the network.

2.3 Spatio-Temporal Graph Convolution

We employ a Spatio-Temporal Graph Convolutional Network (ST-GCN). The skeleton is represented as a graph $G = (V, E)$ where $V = 42$. The adjacency matrix A is initialized with the natural biological connections of the human hand.

2.3.1 Adaptive Topology

To capture semantic relationships between unconnected joints (e.g., thumb tip touching index tip), we use an Adaptive Graph Convolution layer. The network learns a residual mask B that modifies the physical adjacency A:

$$H_{l+1} = \sigma\left(\sum_k (A_k + B_k) H_l W_k\right) \tag{6}$$

This allows the model to "invent" new edges that maximize information flow for specific mathematical operators.

2.3.2 Temporal Modeling

Temporal evolution is handled by interleaved TCN blocks. We use dilated convolutions with kernel size 9×1 along the temporal axis. This provides a large receptive field, enabling the model to distinguish the start and end phases of dynamic signs like "Minus" (a sweeping motion) versus static digits.

3 Implementation Details

3.1 Data Acquisition

We constructed a custom dataset containing 15 classes: digits $0 - 9$ and operators $\{+, -, \times, \div, =\}$.

- **Format:** Raw '.npy' sequences of shape $(T, 2, 21, 3)$.
- **Sampling:** 64 frames per sample (approx. 2 seconds).
- **Protocol:** To ensure robustness, data was recorded with variations in pitch, yaw, and camera distance.

3.2 Training Strategy

The model was implemented in PyTorch 2.0. We utilized the `AdamW` optimizer with a learning rate of $1e-3$ and Cosine Annealing scheduling. To prevent overfitting on the graph structure, we applied:

- **DropGraph:** Randomly removing edges during training.
- **Label Smoothing:** Set to 0.1 to penalize overconfident predictions on ambiguous transitions.
- **Gradient Clipping:** Capped at 1.0 to stabilize the training of recurrent geometric features.

3.3 Real-Time Optimization

Running a deep GCN on every video frame is computationally expensive. We implemented an *Asynchronous Strided Inference* engine. The heavy neural network inference runs only every $K = 4$ frames. Between inference steps, the visualization engine renders the result of the last valid prediction. This decoupling allows the UI to maintain a smooth 30 FPS while reducing CPU load by approximately 75%.

Algorithm 1 Strided Inference Logic

$Buffer \leftarrow \emptyset$
while Camera is Active **do**
  $Frame \leftarrow Capture()$
  $Skel \leftarrow MediaPipe(Frame)$
  $Graph \leftarrow Canonicalize(Skel)$
  $Buffer.push(Graph)$
  if $Counter \pmod 4 == 0$ **then**
    $Pred \leftarrow Model(Buffer)$
    $State \leftarrow Solver(Pred)$
  end if
  $Render(Frame, State)$
end while

4 Results and Discussion

The model achieved a validation accuracy of **93.65%**. Analysis of the confusion matrix reveals that errors are concentrated in topologically similar pairs, such as "Two" vs. "Three" (when the ring finger is partially occluded). The "Unified Scene Projection" successfully resolved the ambiguity between single-hand and dual-hand gestures, allowing robust detection of the "Equal" sign (parallel hands) invariant of the user's torso rotation.

5 Conclusion

NeuroCalc demonstrates that geometric priors can significantly outperform brute-force data augmentation for 3D action recognition. By mathematically normalizing the input space, we reduced the learning problem to a topological one, enabling high-accuracy arithmetic recognition on standard hardware. Future work will focus on expanding the vocabulary to include complex algebraic functions.