OXFORD JOURNAL
OF STUDENT
SCHOLARSHIP

Exploring a New Frontier in Electroencephalographic (EEG) Spike Detection: A Controlled Synthetic Comparison

William Zhang

willpzhang@gmail.com

ABSTRACT

Reliable spike detection requires identifying brief events without repeatedly mistaking noise for spikes. This EEG-motivated proof of concept compared a compact convolutional neural network (CNN) with methods derived from the Smoothed Teager Energy Operator, Bayes optimal template matching (BOTM), and Multi-Template Matching. The signals were simplified mathematical waveforms rather than clinical EEG. The dataset contained 230 parent scenes, each created at -3 dB and 0 dB, yielding 460 arrays divided into training, validation, calibration, and held-out testing sets. All four methods received the same 100 test arrays from 50 parents, including 10 parents without spikes. Detections were matched one-to-one to known spike times within ±15 ms. Across the 40 spike-containing scenes per condition, CNN mean F1 was 95.90% at -3 dB and 96.53% at 0 dB. BOTM-derived achieved 99.22% and 100.00%, respectively. A separate extension tested three independently trained residual CNNs on 500 new parents (1,000 arrays), with each model fixed before testing. They averaged 51.46% fewer false detections than the BOTM-derived method, with a pooled recall of 99.486% versus 99.952%. However, the CNN and BOTM thresholds were chosen using different objectives, and the residual CNN missed more events. The primary comparison favored BOTM-derived by mean F1. The extension showed a promising reduction in false detections, but further work must examine whether this can be retained without losing recall and whether it carries over to real EEG.

INTRODUCTION

EEG motivation and scope

Electroencephalography (EEG) records electrical activity using electrodes placed on the scalp. However, extracting useful information requires separating the activity of interest from background variation and recording artifacts (1). Automated spike detection addresses part of this problem by identifying brief events for further review. To be useful, a detector must find these events without repeatedly mistaking noise for a spike.

This study approaches that problem through a controlled synthetic comparison. The signals contain mathematical spikes with known event times, allowing different detection methods to be tested under the same conditions. These spikes are simplified transients, not recorded action potentials or clinically

identified epileptiform discharges. The experiment is therefore an EEG-motivated proof of concept rather than a physiologically validated EEG simulation. Some of the detection methods used here were originally developed for extracellular recordings, which measure electrical activity near individual neurons (2, 3). This study borrows their approaches to detecting brief waveforms, but does not attempt to reproduce the physiology of those recordings or scalp EEG.

The main advantage of synthetic signals is that their components can be examined separately. The inserted spikes, background, measurement noise, and artifacts are all known. Each method receives the same combined signal, and its detections can be compared with the known spike times. This provides a clear way to examine performance before moving to the less controlled conditions of real recordings.

The goal is to provide a reproducible comparison to guide further testing on annotated real EEG data. Electrode placement, activity across multiple channels, and the full range of physiological waveforms are outside the scope of this experiment. A later study would need to measure clinical accuracy, diagnostic benefit, and reductions in expert review time.

Detection challenges

Noise and artifacts can hide a spike or create a waveform that resembles one. To examine both problems, the signals combined background variation, measurement noise, blink-like pulses, and short bursts of interference. The background and measurement noise had more power at lower frequencies, a property known as colored noise. The artifact shapes were simple, controlled examples of interference. Their similarity to actual clinical artifacts was not validated.

Detection cannot be evaluated solely by the number of spikes found. Recall measures the fraction of true events detected, while precision measures the fraction of reported detections that are correct. The primary comparison used F1 to balance the two. The separate CNN extension imposed an explicit limit on recall loss while aiming for fewer false detections. Both analyses report missed events and false detections because a gain in one measure can come at the expense of the other. The acceptable balance in clinical use would require separate evaluation.

Conventional detection approaches

Threshold-based detection begins with a simple rule: report a candidate event when a signal measure exceeds a cutoff. Its effectiveness depends on how well that cutoff separates spikes from background variation. Energy-based approaches apply the threshold to a measure that emphasizes rapid signal changes, rather than directly to the voltage trace (2).

Semmaoui et al. combined the Smoothed Teager Energy Operator (STEO) with robust statistical threshold estimation for extracellular spike detection (2). Their work provides the basis for the energy-based approach examined here, although extracellular detection differs from scalp EEG analysis. The present implementation uses the median and median absolute deviation of the smoothed energy trace, rather than the authors' Q_n-based threshold. The Methods section describes the calculation and calibration procedures.

Template matching compares short sections of a signal with reference waveforms, called templates. In this study, each detector template contained 100 amplitude values covering 100 milliseconds, expressed in arbitrary units. For BOTM and the CNN's timing step, each template was an average spike shape estimated from noisy training examples. Any detected candidate is more likely to be accepted when its shape resembles a template. The choice of templates, the similarity measure, and how background noise is handled all affect the result. Here, templates are estimated from noisy training observations, independently of the clean held-out event waveforms.

Franke et al. developed Bayes optimal template matching (BOTM) for extracellular spike sorting, combining waveform templates with information about noise covariance (3). Covariance describes how noise at different samples or channels varies together. Incorporating it changes the weight given to differences between a candidate and a template. This study uses a single-channel adaptation of that approach; the theoretical optimality of the full published method is not established for the structured artifacts used here.

These approaches emphasize different aspects of the signal. An energy-based detector looks for rapid changes, whereas template matching also considers waveform shape. Comparing both on identical arrays allows examination of how these differences affect the events found and the false detections produced.

Jiang et al. combined candidate detection, template formation, and additional waveform checks in a multi-template matching approach (4). The MTM-derived implementation used here follows that sequence but changes the signal-window duration, clustering choices, and final checks. These changes are described in Methods so that the tested implementation can be distinguished from the published method and its clinical results.

Together, these methods provide three conventional approaches for comparison with a neural network. Each is evaluated as a complete detection pipeline, including its preparation and post-processing steps. This allows a comparison of the specific implementations used in the study, rather than a general ranking of entire algorithm families.

Learned detection and the research question

Machine learning offers another way to distinguish spikes from noise. Instead of relying only on a predefined threshold or similarity rule, a convolutional neural network (CNN) learns filters from labeled examples. CNNs have been investigated across several EEG analysis tasks (5), making them a promising approach to explore in this controlled detection experiment.

The operation of a one-dimensional CNN can be divided into several stages. First, filters slide across a short signal window, producing feature maps that indicate where learned patterns occur. Further layers combine these features, and the output assigns probabilities to the available classes (6). The CNN used in the main experiment has two convolutional layers, a 128-unit fully connected layer, and a four-class output. Separate alignment and filtering steps then convert candidate windows into detected event times.

Learning these features may be useful when waveform differences are difficult to capture with a small set of fixed rules. However, a network can also learn patterns that are specific to its training data. Separate evaluation data are needed to determine whether those patterns remain useful on new signals. In this experiment, a reference waveform helps refine the spike time after the network classifies a window. Further checks exclude candidates, and nearby duplicate detections are removed. The reported performance therefore describes the network together with these later detection steps.

Published EEG studies give specific reasons to investigate CNN design. Lawhern et al.'s EEGNet used a compact network across four brain–computer interface tasks and performed well with limited training data (7). Schirrmeister et al. compared shallow and deeper networks for EEG decoding and examined which signal features they learned (8). These studies all encourage testing network sizes and learned features to optimize performance. However, they do not determine the optimal architecture for the synthetic spikes we used. It’s important to note that the fixed study settings, including the 100-sample input and the 128-unit fully connected layer, are not backed by widespread empirical support; specific functions and limitations are further explored below.

The primary research question is how STEO-derived, BOTM-derived, MTM-derived, and CNN pipelines compare when given identical synthetic single-channel signals at -3 and 0 dB. Performance is assessed through precision, recall, F1 score, and false detections on event-free signals. The dataset contains 230 parent scenes, each represented at two noise levels, giving 460 arrays across training, validation, calibration, and held-out partitions. The 50 held-out parents provide 100 test arrays, including event-free controls.

A separate residual-CNN extension examines whether further development can reduce false detections while retaining high recall. This extension is reported separately from the four-method comparison. Together, the two analyses examine both the performance of the initial pipelines and a possible direction for improving learned detection before testing on real EEG.

METHODOLOGY

Data Generation

The primary experiment compared four complete detection methods on synthetic, single-channel signals. Each signal contained 1,500 samples recorded at 1 kHz, equivalent to 1.5 seconds. Each inserted spike occupied 100 samples, and amplitudes were expressed in arbitrary units. The generator added these spikes to background variation, measurement noise, and two kinds of artifacts. Spike times were random, but the placement rule kept their 100-sample waveforms from overlapping and limited each positive scene to one through six events. This separated the problem of detecting a spike in contamination from the additional problem of separating overlapping spikes. The generated events were mathematical test shapes; they were not validated models of scalp EEG or extracellular action potentials.

Each basic spike shape combined a downward Gaussian pulse with a smaller upward wave that followed it. Gaussian curves provided a simple way to vary the spike's width and asymmetry. Within the

100-sample waveform, sample positions ran from 0 to 99. The downward pulse was centered at round(99p), where p set its position. Its left Gaussian width was d/5.5 samples; its right width was r times that value. Thus, d controlled sharpness and r controlled how differently the two sides rose and fell. The divisor 5.5 was part of the waveform construction; d was not a measured clinical spike duration.

The upward wave was centered h samples later. Its Gaussian width was w/2.355 samples, using the usual conversion from full width at half maximum to Gaussian standard deviation. Table 1 lists these settings and the amplitude of the upward wave. After combining the two waves, a 100-point Hann taper brought the ends smoothly toward zero. A straight line between the endpoint values was then subtracted to remove any remaining slope across the baseline. Finally, the waveform was scaled so its lowest point, or trough, equaled −1. These steps kept the inserted shapes from introducing abrupt jumps at their boundaries.

Table 1. Settings for the basic synthetic spike shapes. Within each group, the three shapes were selected with equal probability. At 1 kHz, one sample equals 1 ms. The symbols d, p, r, h, and w are defined above; the after-wave amplitude is relative to the downward pulse before final scaling.

Group	Shape	d (ms)	p	r	After-wave amplitude	h (ms)	w (ms)
Training / validation	1	28	0.42	1.30	0.10	27	14
Training / validation	2	42	0.45	1.55	0.18	31	17
Training / validation	3	58	0.48	1.15	0.13	24	20
Calibration	1	32	0.40	1.45	0.12	29	15
Calibration	2	47	0.46	1.25	0.20	34	18
Calibration	3	62	0.50	1.65	0.09	22	21
Test	1	35	0.43	1.20	0.15	30	16
Test	2	50	0.47	1.50	0.11	26	19
Test	3	65	0.44	1.35	0.21	33	22

Individual spikes were varied so that a method could not succeed simply by recognizing exact copies of the basic shapes. The timing of the left and right sides was independently stretched or compressed by factors from 0.85 to 1.15. In 20% of events, a small Gaussian bump was added near the trough: its

amplitude was 0.04–0.12, its width was 2–7 samples, and its center was an integer position from six samples before to eleven after the original trough. In 35% of events, the amplitude was gradually tilted across the waveform by multiplying it by a line from 1 − q to 1 + q, or vice versa, with q between 0.04 and 0.12.

Amplitude also varied smoothly within every spike. Seven evenly spaced random values between −1 and 1 were joined by straight lines. This curve was adjusted to have a mean of zero and a root-mean-square amplitude (RMS) of 1, then multiplied by a random strength between 0.04 and 0.12 and added to 1. The resulting amplitude multiplier could not fall below 0.25. Each waveform was shifted by an integer number of samples from −2 to +2, with exposed positions filled with zeros. A second Hann taper and baseline correction were applied before restoring the trough to −1 and multiplying the whole spike by an amplitude from 0.85 to 1.15. All stated continuous ranges were sampled uniformly. In essence, these changes produced related but nonidentical spikes, with variations in timing, shape, and size.

Spike placement began at sample 50 plus a random waiting time drawn from an exponential distribution with a mean of 500 samples, rounded to the nearest sample. Each later placement added 100 samples, plus another rounded exponential wait with a mean of 400 samples. The fixed 100-sample interval prevented overlap, while the random wait avoided evenly spaced events. Placement stopped before sample 1450 or after six events. If this process produced no event in a scene designated as positive, one was added at a uniformly selected valid location. This preserved the planned distinction between spike-containing and spike-free scenes. Scoring used the actual lowest point of each inserted waveform. Because the shape can vary, this time may differ from its nominal center.

Background variation and measurement noise began as two independent sequences of Gaussian random values. To give them more low-frequency power, each sequence was transformed into frequency components, whose amplitudes were multiplied by f^(−α/2). Here, f is frequency, and α was drawn uniformly from 1 to 2, independently for each sequence. After transforming back, each sequence had its mean removed and was scaled to have an RMS of 1. RMS is the square root of the mean-square power, so it describes the overall amplitude of a changing signal.

The background was then made to grow and shrink gradually over time. Between 3 and 8 evenly spaced control points, with values from 0.5 to 2.0, were joined by straight lines to form an amplitude multiplier. After applying it, the background was centered and scaled to RMS 1 again. Background and measurement noise were multiplied by 0.50 and 0.35, respectively, before adding artifacts. These gains set their relative contributions; the later SNR calculation scaled the combined contamination to the required power. The gains and variation ranges define this test scenario and were not fitted to clinical recordings.

A spike-free array cannot have an event-based SNR because its event power is zero. To set its contamination level, the generator used the mean energy of the three basic spike shapes, multiplied by three and divided by 1,500. The factor three came from the nominal rate of two events per second over 1.5 seconds, with the midpoint event amplitude equal to one. This provided a fixed reference level for the negative controls; it was not the measured event count or power of those controls.

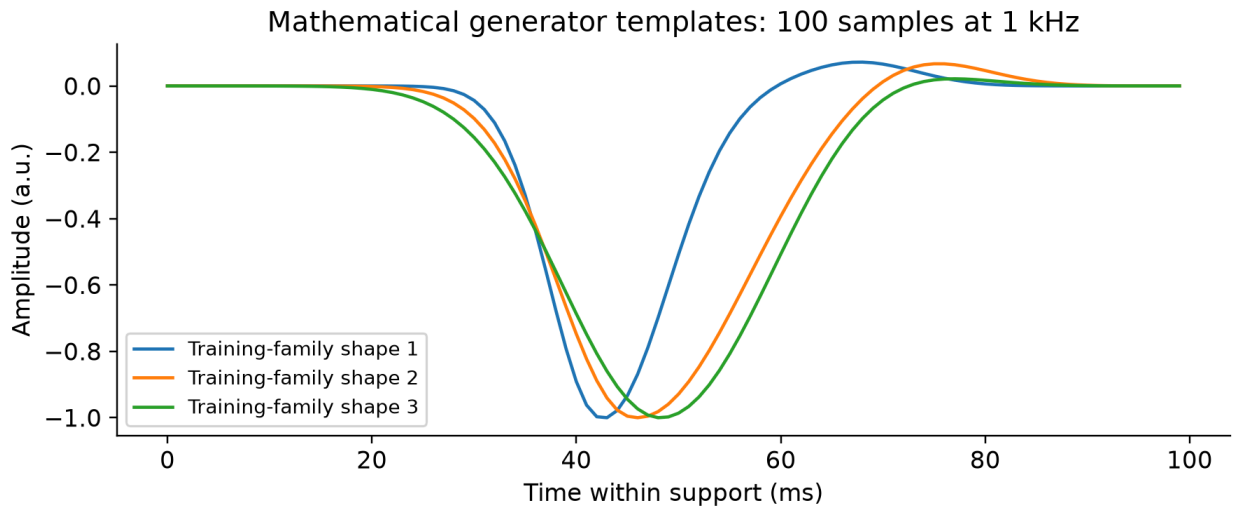


Figure 1. Clean mathematical spike shapes used to generate training signals, with the negative trough normalized to −1. Each waveform spans 100 samples (100 ms at 1 kHz). These illustrations are not clinical recordings or the noisy, training-estimated detector templates.

Every primary scene contained one or two blink-like pulses and one or two burst artifacts. This ensured that both the spike-containing and spike-free tests included interference that could create false detections. Artifact positions were selected uniformly wherever the full artifact fitted, and artifacts were allowed to overlap spikes. The blink-like pulse used a smooth Hann shape, lasted 140–240 samples, and had an amplitude of 1.00–2.50 before the final SNR scaling. A burst lasted 60–140 samples and had an amplitude of 0.60–1.50. It was made from Gaussian noise restricted to 20–100 Hz, tapered at both ends, and scaled so its largest absolute value equaled one before applying that amplitude. Counts and lengths were sampled uniformly over their integer ranges; amplitudes were sampled uniformly over their continuous ranges. These choices created repeatable interference tests, without claiming to reproduce the frequency or severity of artifacts in patients.

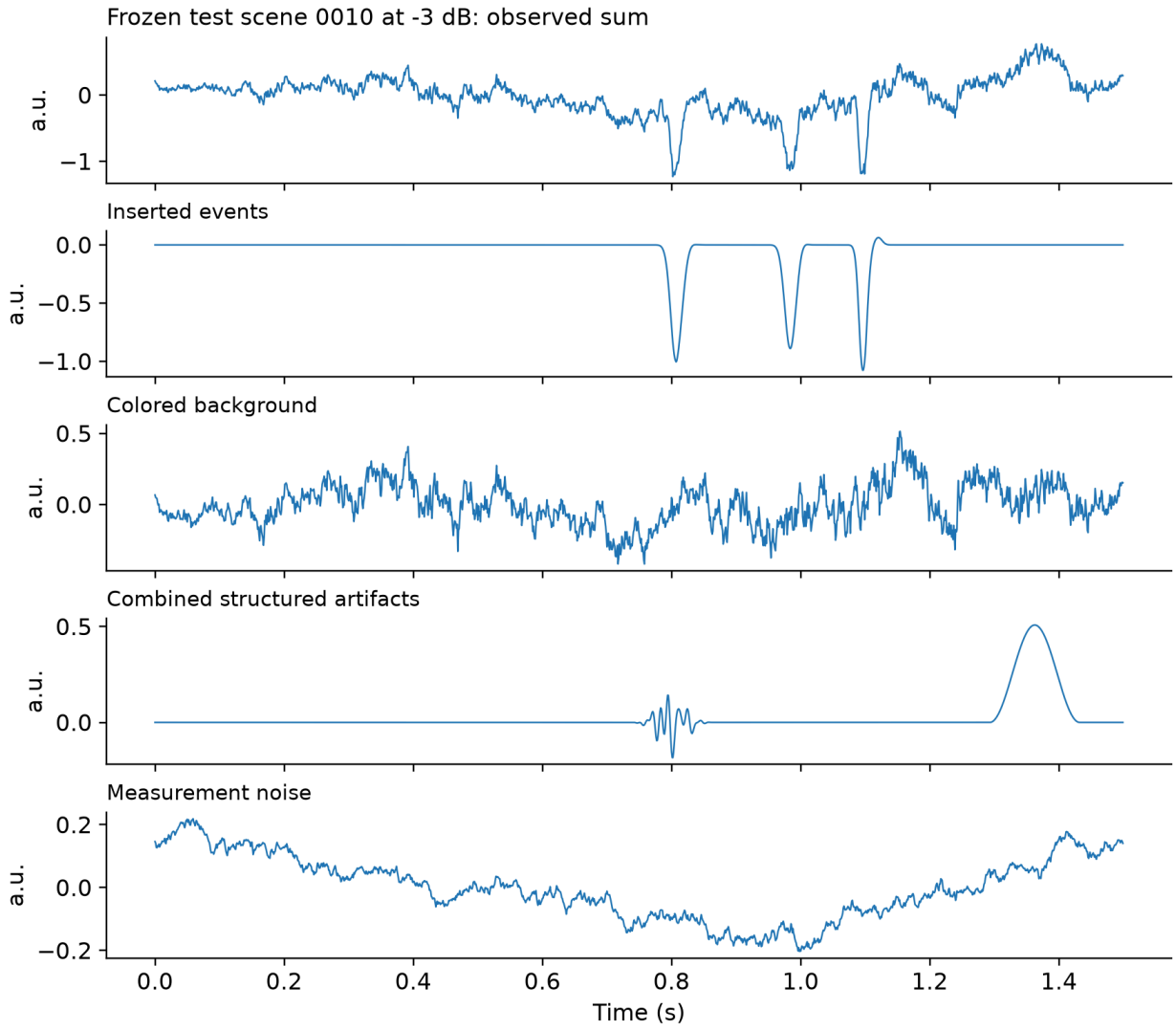


Figure 2. Exact saved components of frozen test scene 0010 at -3 dB: observed signal, inserted event track, colored background, combined blink-like and burst artifacts, and measurement noise. Component panels use their own vertical scales; all amplitudes are in arbitrary units. Their sample-wise sum reconstructs the observation.

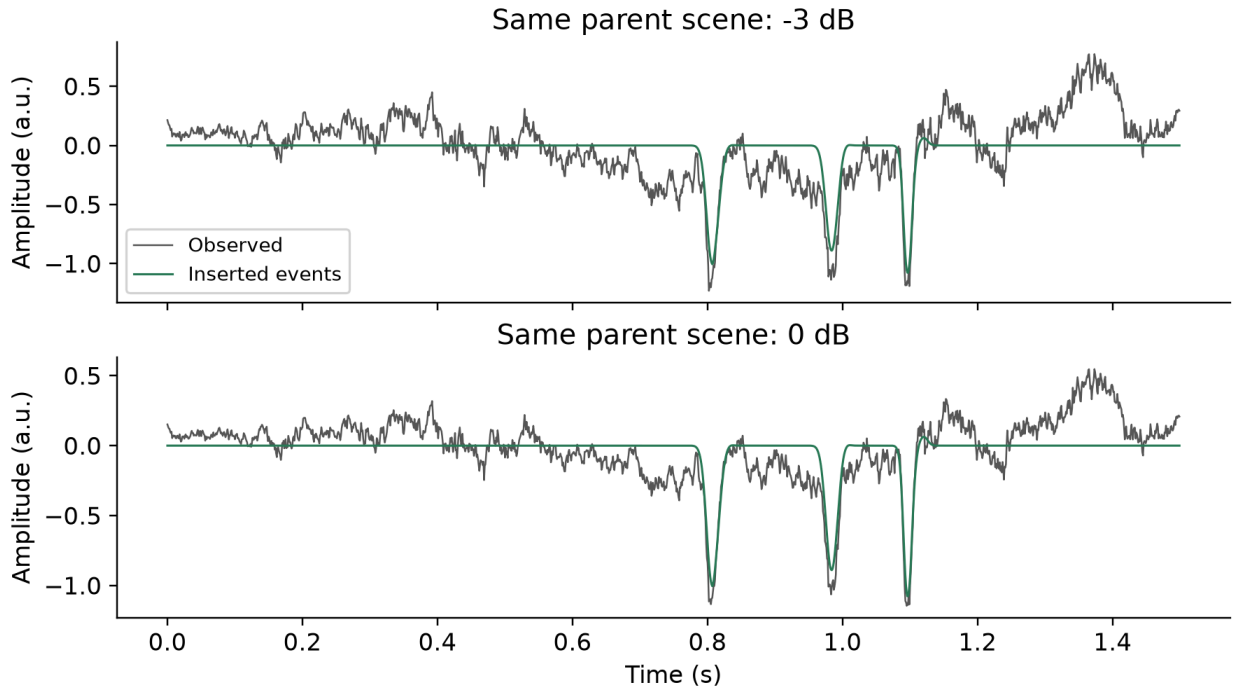


Figure 3. Two observed arrays from frozen test parent scene 0010, at -3 and 0 dB, overlaid with the identical inserted event track. These are paired synthetic conditions, not independent parents. SNR is defined as the full-array event power divided by the combined contamination power.

SNR was calculated from the spike-only signal and the combined background, artifacts, and measurement noise. Power was defined as the average squared amplitude over the full 1,500-sample array, as given in Equation 1. The contamination components were added before calculating their power, so the measurement included how they reinforced or canceled one another. Equations 2 and 3 then set their combined amplitude to produce -3 or 0 dB. This kept the specified power ratio consistent when spike counts or amplitudes changed. It also ensured that artifacts were included in the contamination level rather than added outside the SNR calculation.

$$P(z) = \frac{1}{N} \sum_{i=0}^{N-1} z_i^2$$

Equation 1. Mean-square power. The vector z is any signal component; z_i is its amplitude at sample i; i is the sample index from 0 to N − 1; N is the number of samples (1,500 here); Σ means sum over those samples; and P(z) is the mean squared amplitude, in squared arbitrary amplitude units.

$$D_{\text{achieved}} = 10\log_{10}\left(\frac{P(s)}{P(c)}\right)$$

Equation 2. Achieved SNR. The vector s is the inserted event track, c is the combined scaled contamination, and P denotes the mean-square power from Equation 1. $D_{achieved}$ is the measured SNR in decibels (dB), and $\log_{10}$ is the base-10 logarithm. Event-free scenes have no event-based SNR; their condition labels indicate the reference contamination scale described above.

$$a = \sqrt{\frac{P(s)}{P(c_0)\,10^{D_{\text{target}}/10}}}$$

$$c = a\,c_0, \qquad x = s + c, \qquad c = x - s$$

Equation 3. Setting the contamination amplitude. The scalar a is the amplitude multiplier; c_0 is the unscaled combined contamination; c is the scaled contamination; s is the event track; x is the observed array; and D_{target} is the desired SNR (-3 or 0 dB). P again means full-array mean-square power. The generator multiplies c_0 by a, adds s, then checks Equation 2 using the recovered contamination x − s. The square root converts the required power ratio into an amplitude multiplier.

In short, the generator varied spike shape, event placement, and contamination while retaining the component signals and true event times. This made it possible to compare all four methods on exactly the same inputs and check their errors against known events.

The dataset contained 230 parent scenes and 460 signal arrays. A parent scene was one underlying arrangement of spikes, background, noise, and artifacts. Each parent produced two arrays by changing the contamination amplitude to the -3 dB and 0 dB settings. Pairing the arrays this way lets us compare noise conditions without also changing the underlying events. The two arrays were therefore related observations.

Each parent was assigned to training, validation, calibration, or testing before windows were extracted. Both noise versions and every window from that parent stayed in the same group. Table 2 gives the allocation. Training fitted the models and templates; validation selected the CNN training epoch; calibration selected detection thresholds; and the test set measured final performance. All four methods received the same saved test arrays.

The random seeds were 2026091401 for dataset generation, 2026091402 for model fitting, and 2026091403 for calibration. A seed fixes the sequence of random draws, allowing it to be reproduced.

The partitions used separate random streams and were checked for parent reuse; no parent was shared across partitions.

Table 2. Allocation of parent scenes and their paired signal arrays.

Partition	Parent scenes	Parents with spikes	Parents without spikes	Signal arrays
Training	120	96	24	240
Validation	30	24	6	60
Calibration	30	24	6	60
Held-out test	50	40	10	100
Total	230	184	46	460

Training and validation used separate parents generated from the same three basic spike shapes. Calibration and testing used the different shape settings listed in Table 1. This tested detection on related shapes that were not exact copies of the training shapes. All remained within the same simplified waveform model, so this was a limited test of shape variation. For spike-free scenes, -3 dB and 0 dB identify the reference contamination settings described above.

Compared with the original 20-signal evaluation, the revised design added separate groups for learning, setting selection, and testing, along with spike-free controls. The 230 parents were the total across these roles; only 50 parents were used for the final primary test. The exact allocation was a practical study budget rather than a calculated guarantee of statistical power.

Design rationale and scope

Synthetic signals allow us to check whether each method identifies the same known events under the same level of contamination. Keeping the parents separate also prevented a training window from having a closely related version in the test set. This addresses the risk of obtaining overly favorable results when model selection and final evaluation use the same data (9). Most parents were allocated to training because this stage fitted the CNN and reference templates. Validation and calibration had separate roles, and 50 parents were reserved for testing. These counts made the stages distinct, but the study did not test whether larger training sets would improve performance.

The two SNR conditions gave a clear contrast in contamination strength. At 0 dB, event power and contamination power were equal. At -3 dB, contamination power was about 1.995 times event power, or nearly twice as large. Its RMS amplitude was about 1.413 times that of the event signal. Because the same parent was used at both levels, changes in performance could be examined without changing the spike

arrangement. These ratios describe the full 1.5-second array; an individual spike could still be easier or harder to see depending on the local contamination.

A detection was counted as correctly timed if it fell within 15 ms of the known event. This allowed modest differences between a detector's reported time and the actual waveform minimum. The exact tolerance was a convention of the study. The same predictions were also scored at ±5, ±10, and ±20 ms to show how much the findings depended on it.

Detections within 25 samples of a stronger detection were removed to limit repeated reports of the same spike. This gap was shorter than the generator's 100-sample spacing between spike placements. It was fixed across methods rather than selected separately to improve a particular method's test score.

At 1 kHz, a 100-sample window covers 100 ms, matching the length of the generated spike shapes and reference templates. It therefore gives the CNN a view at the same scale as the event it is being asked to detect. A window near an event's edge can still contain only part of the spike, and a longer context may be needed for a real EEG.

After the two convolution and pooling stages, the network produces 1,600 values describing learned patterns at different positions. The fully connected layer combines those values into 128 outputs before the final four-class decision. It contains $(1{,}600 + 1) \times 128 = 204{,}928$ of the model's 216,100 trainable parameters, or about 95%. Thus, its main role is to combine information across the window, and it accounts for most of the model's size. The study held this layer at 128 units; it did not compare other widths.

During training, dropout randomly set 30% of this layer's outputs to zero on each update (10). This discouraged the network from depending on the same small set of units every time. Training also stopped when performance on the separate validation data stopped improving. These measures reduce reliance on particular training patterns but do not rule out overfitting.

Batch normalization adjusts the scale of intermediate-layer outputs during training (11). Kernel lengths, dropout of 0.30, learning rate of 0.001, label smoothing of 0.05, and the four-epoch stopping rule were held fixed. Their roles are described with the architecture and training settings below. The study evaluated this configuration without claiming that each value was the best possible choice. Importantly, validation selected the training epoch, calibration selected the detection thresholds, and the held-out test measured performance only after those choices were fixed (9).

BOTM and the CNN's timing step used three detector templates estimated only from labeled, noisy training signals. These were built from 544 short sections of the noisy training signals: 166 examples of shape 1, 212 of shape 2, and 166 of shape 3. Each section contained 100 samples, with the known spike minimum placed at sample 50. Its average amplitude was subtracted from every sample, so its baseline was centered around zero. The sections for each shape were then averaged, sample by sample, to create a single template. This gave the methods examples learned from noisy training signals instead of the clean shapes used to generate the test spikes. Then, the CNN used its selected template to refine the estimated spike time after classifying a window, with these templates separate from the filters learned inside the

network. MTM also began with training-derived reference shapes and fitted its own template groups, as described below. No detector received the clean test-generation shapes or test labels.

BOTM also estimated noise covariance from 9,670 training windows that contained no part of an inserted spike. Covariance describes how contamination at different sample positions varies together; averaging it equally with a scaled identity matrix made its inverse more stable. Equation 6 gives the calculation. The methods shared the same underlying training scenes, although they used those scenes in different fitting procedures.

Evaluation Metrics

A reported detection is counted as correct when it could be paired with a known event within ±15 samples, including detections exactly 15 samples away. At 1 kHz, this is ±15 ms. Events and detections were considered in chronological order, and the matching rule found the maximum number of valid pairs. Each event and each detection could belong to only one pair. Paired detections were true positives (TP); unpaired detections, including duplicate reports of one event, were false positives (FP); and events without a detection were false negatives (FN). The same saved predictions were also scored at ±5, ±10, and ±20 ms to assess the effect of timing tolerance without changing any detector parameters.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$F_1 = \frac{2\,\text{Precision}\,\text{Recall}}{\text{Precision} + \text{Recall}}$$

Equation 4. Detection metrics. TP is the number of matched predictions (true positives); FP is the number of unmatched predictions, including duplicates (false positives); FN is the number of unmatched true events (false negatives). Precision is the fraction of reported detections that are correct; recall is the fraction of true events found; F1 is their harmonic mean. All are dimensionless proportions, reported as percentages where indicated. The scoring utility returns zero when a metric's denominator is zero.

- Recall describes sensitivity to known inserted events, including events obscured by noise or artifacts.
- F1 combines precision and recall into a single score, giving equal weight to both measures.

Scoring asked whether a method found an event at the correct time. A correctly timed detection counted as a TP even if the method assigned the wrong shape label. The CNN produced four probabilities: one for background and one for each of the three synthetic spike shapes used in training. These labels helped it choose a reference template, but they were not clinical diagnoses or verified biological spike types. During testing, every detector received only the combined noisy signal. Known event times, clean spike

shapes, separate contamination components, artifact locations, and SNR labels were withheld from the detectors and used only to check their results.

For the main comparison, precision, recall, and F1 were calculated for each spike-containing test array and then averaged separately across the 40 arrays at each SNR. This gave each scene equal weight, rather than giving more weight to scenes with more spikes. Mean F1 was the average of the individual F1 scores; it was not recalculated from mean precision and mean recall.

As a second summary, TP, FP, and FN were added across all 50 arrays per condition before calculating the pooled metrics. This included false detections in spike-free scenes. The ten spike-free arrays provided 15 seconds of spike-free test signals per condition. Their false-detection counts, counts scaled to one minute, and fraction with any detection were also reported separately. They were excluded from the primary F1 average because recall has no true events to measure in a spike-free array. Whenever a metric's denominator was zero, the scoring program returned zero; separate reporting avoided treating that convention as a meaningful negative-scene F1 score.

Software

The primary benchmark used Python 3.11.9, TensorFlow 2.21.0, and Keras 3.13.2. The separate residual-CNN extension used PyTorch 2.6.0+cu124 with CUDA 12.4 and cuDNN 9.1 on an NVIDIA GeForce RTX 4060 Ti. It did not replace the TensorFlow model in the primary comparison.

- NumPy 2.3.5 and SciPy 1.17.1 supported array operations, signal generation, and signal processing.
- scikit-learn 1.8.0 supplied K-means clustering and ROC calculations for fitting the MTM-derived training-only slope thresholds.
- Matplotlib 3.11.0 was used for figures.
- Run-records preserve the dataset configuration, random seeds, trained weights, fitted templates, selected thresholds, per-array detections, and software versions. File hashes identify the frozen inputs and outputs. Reproducing reported scores requires those same saved artifacts and the stated matching and averaging rules; rerunning training can introduce numerical differences.

Algorithm Implementations

Smoothed Teager Energy Operator

The STEO-derived detector calculated the Teager energy for each sample and its immediate neighbors, then smoothed the result using a five-sample moving average. At the signal boundaries, it did not treat the last sample as a neighbor of the first. This avoided an artificial connection between the two ends. Equation 5 defines the energy measure and threshold.

$$\psi[n] = x[n]^2 - x[n-1]x[n+1]$$

$$T = \text{median}(E) + 1.4826\, k\, \text{median}(|E - \text{median}(E)|)$$

Equation 5. STEO-derived energy and threshold. The observed amplitude at sample n is x[n], and ψ[n] is its Teager energy. E is the energy sequence after five-sample smoothing, T is the detection threshold, and k is the multiplier selected by calibration. Median means the middle value; | | means absolute value. The median absolute deviation measures spread: it is the median of the absolute distances from the median value. Multiplying this distance by 1.4826 puts it on the same scale as standard deviation for a normal distribution. This explains the constant; it does not assume that the actual energy values follow that distribution.

One threshold was calculated for each 1.5-second array, excluding the first and last three samples from both threshold estimation and detection because they were affected by the boundary calculation. Calibration selected k = 9 from {2, 4, 6.5, 9, 12}. Within each continuous region above the threshold, the detector reported the time of the largest energy value. The threshold stayed fixed within that array.

This implementation differs from Semmaoui et al.'s published method. Their Equations 28–30 use the Q_n scale estimator to estimate noise autocorrelation, which enters their calculation of the STEO output statistics and threshold (2). Here, the threshold was calculated directly from the median and median absolute deviation of the smoothed energy. The comparison therefore evaluates this STEO-derived implementation, rather than a complete reproduction of their threshold method.

All four methods removed nearby duplicate candidates by considering the strongest score first. Once a candidate was kept, other candidates within 25 samples on either side were discarded, including those exactly 25 samples away. If scores were tied, the earlier candidate was kept. This procedure is commonly called nonmaximum suppression.

However, the methods can identify the same spike but report slightly different times. STEO marks the peak of its calculated energy measure, which emphasizes rapid changes in the signal. The template-based methods instead estimate the spike's lowest point. Thus, scores changes when detections had to fall within 5, 10, 15, or 20 milliseconds of the known spike time were checked.

Bayes Optimal Template Matching

The BOTM-derived implementation used three 100-sample, mean-subtracted templates estimated from labeled noisy training windows. Equation 6 defines the covariance matrix used for scoring.

$$C = 0.5\,C_e + [0.5\,v + \max(10^{-6}v, 10^{-10})]I$$

Equation 6. Stabilized noise covariance. C_e is the sample covariance matrix of event-free training windows; v is the mean of its diagonal entries; I is the 100-by-100 identity matrix; and C is the resulting scoring covariance matrix. The max operation selects the larger of its two arguments. This mixture retains some measured sample-to-sample dependence while making matrix inversion more stable; the small positive term prevents a zero-diagonal adjustment.

$$g(w, t) = w^T C^{-1} t - \tfrac{1}{2} t^T C^{-1} t + \ln\left(\tfrac{0.1}{3}\right)$$

$$g(w, t) > \ln(0.9) - \delta$$

Equation 7. BOTM-derived score and acceptance rule. The vectors w and t contain the candidate window and one fitted template, each with its average amplitude subtracted. C is the covariance matrix from Equation 6, C^{-1} is its inverse, and superscript T means transpose. The function g(w,t) returns the match score, and ln is the natural logarithm. Before considering the match, the calculation assigns a total event weight of 0.1, divided equally among the three shapes, and a background weight of 0.9. These prior weights favor the background unless the waveform provides sufficient evidence for an event; they are fixed assumptions in this implementation, not measured event frequencies. The calibrated offset δ adjusts the acceptance cutoff.

BOTM compared all eligible positions and templates and accepted the match with the highest score. It recorded the event time, subtracted 80% of the fitted template at that position, and searched the remaining signal again. The subtraction reduced the contribution of an already accepted event to later searches. The remaining signal is the residual; this term is separate from the residual connections used in the CNN extension. Candidates within 25 samples of an accepted event were excluded.

The factor 0.8 left part of the fitted waveform in place rather than assuming a perfect template match. Its value was fixed, and the study did not test whether it was optimal. The iteration limit was the signal length divided by 26, rounded up, reflecting the minimum spacing between accepted detections. An additional eligible event at that limit triggered an error instead of silent truncation. Calibration selected $\delta = -4$ from $\{-4, -2, 0, 2, 4\}$.

This single-channel adaptation uses the template and covariance-weighted scoring principle of Franke et al. (3) but does not reproduce their full multichannel spike-sorting system. The 0.5 covariance shrinkage weight, event/background priors, threshold grid, subtraction factor, and suppression gap are the

implementation settings, not values prescribed by that paper. Therefore, the noise model approximates structured artifacts, so Bayesian optimality is not fully expressed in this study. The generator also prevents overlap between the 100-sample spike waveforms, so the subtraction step does not recover overlapping spikes.

Multi-Template Matching

MTM first averaged the three training-derived templates to form one general reference shape, called the universal template. It scanned the training signals for regions with cosine similarity to this reference of at least 0.20. Cosine similarity compares the waveform shapes after removing the mean and scaling by the lengths of the two vectors; a value near 1 indicates a close match. The best match in each passing region proposed an event time, around which a 100-sample window was extracted. Using 100 samples kept the window length consistent with the fitted templates.

After the slope checks described below, the passing training windows were grouped into three clusters by K-means. This algorithm groups similar waveforms and uses the average of each group as a template. Three groups kept the number of templates consistent with the three training shapes, although the clusters were learned rather than assigned those labels. K-means used seed 2026091402 and ten initializations. If fewer than three windows passed, the code instead used the original three training-average templates. In the reported run, 644 windows passed, so this fallback was not needed. Each resulting template was shifted so its lowest point was at sample 50, filling exposed positions with zeros.

The slope checks measured how a candidate waveform changed before and after its midpoint. Each window was centered and divided by its standard deviation, and a straight line was fitted separately to its first 50 and last 50 samples. The two signed slopes retained whether each half rose or fell. Training candidates were labeled positive if their proposed time fell within 15 samples of a known event; otherwise, they were labeled negative.

For each slope, the acceptance direction was determined by whether positive candidates had a higher or lower median value than negative candidates. The cutoff was then chosen to maximize sensitivity minus the false-positive rate, known as Youden's J. This used training labels to select a cutoff that separated the two groups. If only one label was present, both directions remained +1, and both thresholds remained −1,000,000, making the slope checks effectively nonrestrictive. This run included both labels.

At detection, a candidate had to match the universal template, match at least one fitted template, and pass both slope checks. Calibration selected a universal-template similarity of 0.50 and a fitted-template similarity of 0.65 from {0.20, 0.35, 0.50} and {0.35, 0.50, 0.65}, respectively.

Compared with Jiang et al. (4), this MTM-derived version uses a training-mean universal template rather than their triangular proposal template, 100-sample (another way of saying 100-ms) rather than 150-ms signal windows, and a fixed three-cluster model. It uses two fitted half-window slope checks and does not implement their small-cluster pruning or final combination of Teager energy, duration, and amplitude

checks. No test-scene clustering or test-label threshold fitting was allowed. The reported result applies to these disclosed changes, not to the full published MTM method.

Convolutional Neural Network

Architecture Overview

The primary one-dimensional CNN processes 100-sample windows and returns four class probabilities: background and three training-defined spike shapes. It has two convolutional layers, a 128-unit fully connected layer, and a four-unit output layer. The fitted model has 216,100 trainable parameters.

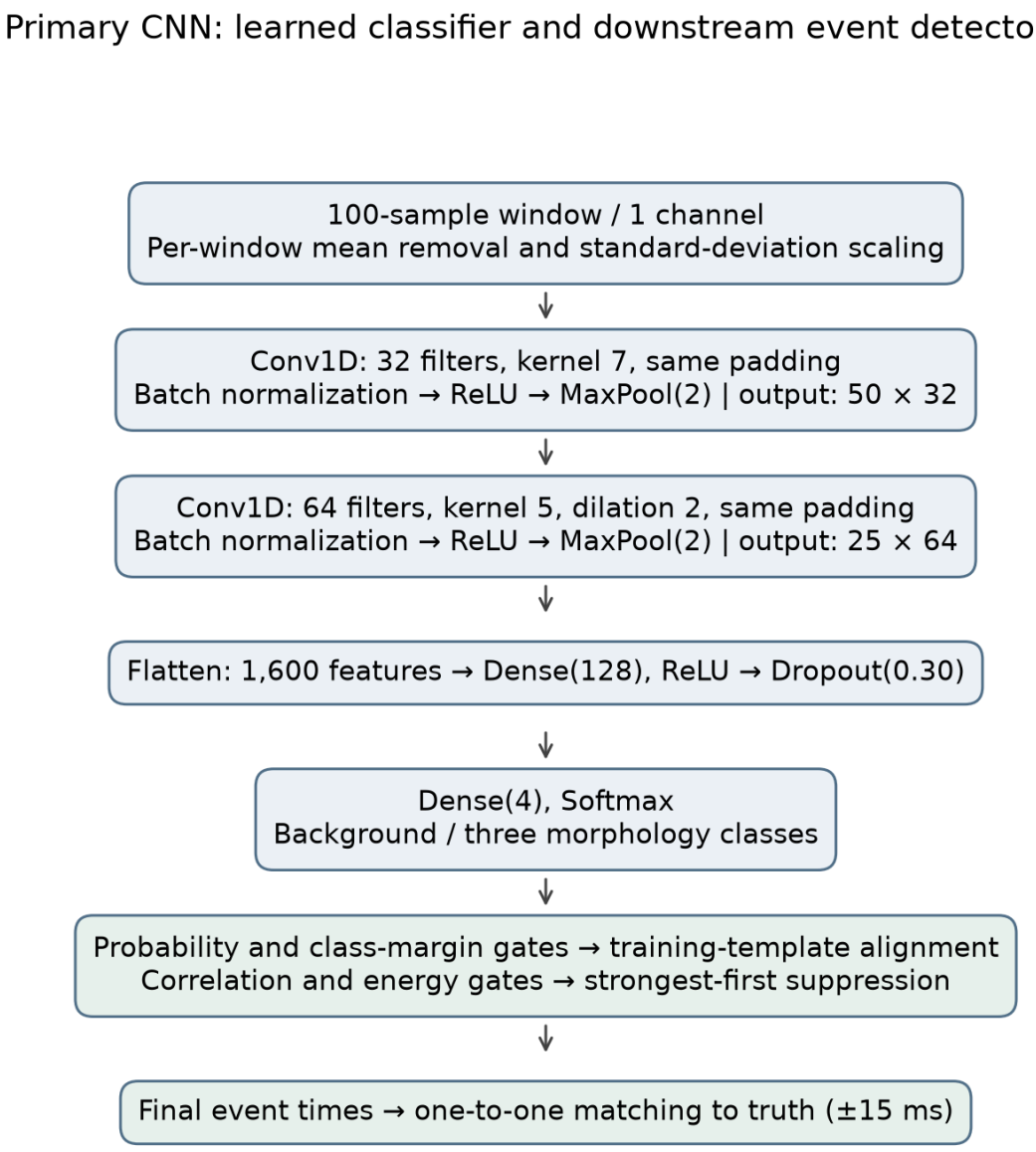


Figure 4. Primary two-convolution CNN and downstream detection pipeline. The learned classifier has two convolutional and two dense layers; template-based timing adjustment, candidate acceptance checks, and removal of nearby duplicate detections are separate processing stages. The residual CNN is described separately in Supplement S1.

Inference and Spike Detection

A sliding window traverses the signal; windows predicted as any spike type become candidates. Candidates were filtered by probability and morphology checks and then consolidated into spike times.

Inference/detection parameters

- Scan: 100-sample windows, stride 1 sample only.
- Candidate rule: the highest-probability class must be one of the three spike shapes, rather than background.
- Gates:
  - Probability gate: winning non-background probability ≥ 0.35 and top-minus-runner-up probability ≥ 0.10.
- Shape checks: correlation with the predicted shape's template must be at least 0.70, and the smaller of the two local energy values must be at least 20% of the larger (Equation 8).
- Post-processing: score-first suppression within an inclusive 25-sample gap; ties favor the earlier event.
- Scoring tolerance: ±15 samples from the known negative minimum of the inserted spike.

Architecture parameters

- Input window: 100 × 1
- Class set: {non-spike, spike-1, spike-2, spike-3}
- Convolution block 1: 32 filters, each covering seven samples. Zero-padding keeps the output length unchanged, and the convolution adds no bias. Batch normalization adjusts output scales; ReLU replaces negative outputs with zero. Max pooling then keeps the larger value in each pair of neighboring positions.
- Convolution block 2: 64 filters, each using five positions spaced two samples apart (dilation 2). This covers a broader portion of the first block's output without adding additional filter weights. Padding, the absence of convolution bias, batch normalization, ReLU, and two-position max pooling follow the same rules as block 1.
- Final layers: the remaining 25 positions across 64 feature channels are arranged into a list of 1,600 values. A fully connected layer combines them into 128 outputs and applies ReLU, followed by dropout of 0.30 during training. A second fully connected layer produces four values; softmax converts them into four class probabilities that sum to one.

Training

Every primary-CNN window had its mean subtracted and was divided by its standard deviation. The standard deviation used all 100 samples, and the divisor could not fall below 10^{-8}. This kept window scales consistent and prevented division by zero, but also removed differences in overall amplitude between windows. The same operation was applied during training and detection.

Positive training windows placed the known spike minimum at sample 40, 45, 50, 55, or 60, counting from zero. These positions covered the range that the later template search could locate. Windows extending beyond an array were excluded. Background windows were drawn at 25-sample steps and could contain no part of an inserted spike. They were randomly reduced to match the positive count, so background examples did not dominate training. This original rule provided clear negative examples but did not teach the CNN how to handle a spike at the edge of a window; that issue motivated the separate sampling experiment in Supplement S1. Validation used the same extraction rules on separate parents. The primary benchmark used the final trained CNN.

Training parameters

Table 3. CNN training and validation windows by label. Background and all three spike labels combined were balanced within each partition. Multiple windows can come from a single parent and are not independent recordings. Every window stayed within its parent's assigned partition.

Window label	Training	Validation
Background	2,722	636
Spike shape 1	834	230
Spike shape 2	1,058	200
Spike shape 3	830	206
Total	5,444	1,272

- Split: parent-level training and validation partitions; no random window-level 80/20 resplit.
- Optimizer: Adam, learning rate 0.001; seeded reshuffling each epoch.
- Loss: categorical cross-entropy, which penalizes assigning too little probability to the labeled class. Label smoothing of 0.05 changed the target from probability 1 for the correct class and 0 for the others to 0.9625 for the correct class and 0.0125 for each other class. This discouraged training toward completely certain outputs.
- Batch size: 32

- Training: at most 20 epochs, where one epoch is one pass through the training windows. Training stopped after four consecutive epochs without a decrease in validation loss, and the weights from the lowest-loss epoch were restored. The primary detector used the weights from epoch 8. This selected the training duration using validation data rather than the final test results.

Data generation and noise levels

The CNN received the same saved 100 held-out arrays as the other three methods: 50 parents, each rendered at -3 dB and 0 dB. Per condition, 40 arrays contained events, and 10 were event-free. These were not newly generated method-specific test sets. Model weights, fitted templates, and calibrated detection settings were frozen before evaluation.

Post-processing gating system

The reported CNN result includes more than the network's four-class prediction. The network proposed a spike shape; the corresponding training template then helped estimate its time. Probability, waveform-correlation, and local-energy checks rejected candidates, and duplicate removal produced the final event list. The following rules describe the complete detector used in the primary comparison.

- P_MIN = 0.35: one of the three spike classes had to have the highest probability among all four classes, and that probability had to be at least 0.35.
- PROB_MARGIN = 0.10: the winning probability minus the second-highest of all four probabilities had to meet this margin.
- Template correlation, RHO = 0.70: the predicted shape's template was shifted from −10 to +10 samples relative to the candidate window. At each shift, Pearson correlation was calculated using only the overlapping samples, with each overlapping portion centered and divided by its vector length. The shift with the highest correlation was kept, and that correlation had to reach 0.70. This compares shapes while allowing a small timing offset.
- The predicted class selected the template used for the timing check. Because the test generator used related but different shapes, these class labels were not treated as exact test-shape identities.
- Scan step, STRIDE = 1: the window moved forward by one sample at a time, so every valid 100-sample window was evaluated.
- SUPPRESSION_GAP = 25 samples: accept the stronger candidate first and suppress nearby candidates, rather than accepting the first threshold crossing.
- MAX_LAG = 10 samples: search lags −10 through +10, then estimate event time from the selected lag and the fitted template's negative minimum.

For the CNN energy check, a candidate had to satisfy Equation 8.

$$\frac{\min(L, R)}{\max(L, R, 10^{-12})} \geq 0.20$$

Equation 8. Local energy balance. L is the sum of squared observed amplitudes in the 20 samples before the candidate time. R is the corresponding sum starting at that time and continuing for 20 samples. Min selects the smaller energy, and max selects the largest of the two energies and 10^{-12}; this small lower bound prevents division by zero. A ratio of at least 0.20 required the smaller side to contain at least one-fifth of the larger side's energy. Candidates failing this check were excluded from the final detection list. This check was intended to filter out windows containing only part of a spike. The cutoff was fixed; its individual effect on correct and false detections was not isolated.

The local energy check used the observed signal before window standardization. A candidate was rejected if either 20-sample interval extended outside the array. This kept the calculation the same for every accepted candidate.

Calibration of gate parameters

All methods used one common selection objective: maximize pooled event F1 across the 60 calibration arrays, including false detections in negative scenes and both SNR conditions. Ties favored fewer false positives, and then the first configuration in the declared grid. Calibration fitted detection settings, not CNN weights, and its scores were development evidence rather than held-out results.

- CNN candidate grid:
  - Winning-class probability: {0.35, 0.50, 0.65}.
  - Probability margin: {0.00, 0.10}.
  - Template correlation: {0.30, 0.50, 0.70}. Together, the three lists above gave 18 combinations of settings.
  - Fixed across candidates: stride 1, lag ±10 samples, energy ratio ≥ 0.20, suppression gap 25 samples.
- Selected CNN configuration:
  - P_MIN = 0.35; PROB_MARGIN = 0.10; RHO = 0.70.
  - Calibration counts: TP = 117, FP = 26, FN = 1.
  - Pooled calibration precision = 81.82%; recall = 99.15%; F1 = 89.66%.
  - The other searches tested five STEO threshold multipliers, five BOTM threshold offsets, and nine MTM similarity-threshold pairs. Each search used that method's adjustable quantities, while all methods shared the same calibration arrays and F1 selection objective. The search sizes were disclosed rather than treated as equal tuning budgets.

One selected configuration per method was locked before held-out evaluation; no test-score-based retuning was used for the primary benchmark.

Evaluation protocol

Evaluation used the shared test arrays, saved CNN weights, training-fitted templates, and detection settings selected on calibration data. The following CNN values are means across the 40 spike-containing arrays at each noise level. They are separate from the calibration scores and from metrics calculated after adding counts across all test arrays:

- -3 dB: mean precision = 94.29%, mean recall = 98.87%, mean F1 = 95.90%.
- 0 dB: mean precision = 94.29%, mean recall = 100.00%, mean F1 = 96.53%.

RESULTS

All four methods were evaluated on the same 100 held-out arrays from 50 parent scenes. Each parent contributed a -3 dB and a 0 dB version. At each noise level, 40 arrays contained 111 known events in total, and 10 arrays contained no events.

Table 4 reports the mean precision, recall, and F1 across the 40 spike-containing scenes at each SNR, using one-to-one matching within ±15 ms. Counts added across scenes and false detections in spike-free arrays are reported separately. These results describe performance within this synthetic test.

Algorithm	SNR (dB)	Achieved SNR (mean ± SD)	Precision (%)	Recall (%)	F1 (%)
STEO	-3	-3.00 ± 0.00	54.81	71.21	57.97
STEO	0	0.00 ± 0.00	66.77	92.04	74.26
BOTM	-3	-3.00 ± 0.00	98.67	100.00	99.22
BOTM	0	0.00 ± 0.00	100.00	100.00	100.00
MTM	-3	-3.00 ± 0.00	75.52	76.54	70.71
MTM	0	0.00 ± 0.00	78.29	85.96	77.30
CNN pipeline	-3	-3.00 ± 0.00	94.29	98.87	95.90

CNN pipeline	0	0.00 ± 0.00	94.29	100.00	96.53

Table 4. Primary held-out mean precision, recall, and F1 (%), each averaged independently over 40 event-containing scenes per SNR. All methods received identical arrays. STEO, BOTM, and MTM denote the derived implementations described in Methods; CNN denotes the primary two-convolution network with its later detection checks. Event-free controls are excluded from these means. Achieved SNR was calculated over the positive scenes; SD rounds to 0.00 dB because contamination was scaled to the target, apart from floating-point error.

Smoothed Teager Energy Operator (STEO)

STEO achieved mean precision, recall, and F1 of 54.81%, 71.21%, and 57.97% at -3 dB, and 66.77%, 92.04%, and 74.26% at 0 dB. It generated 21 false detections in the 15 seconds of spike-free test signals at each condition (84/min), with detections in 9 of 10 negative scenes. These short synthetic tests support descriptive comparisons only.

Bayes Optimal Template Matching (BOTM)

BOTM-derived achieved mean precision, recall, and F1 of 98.67%, 100.00%, and 99.22% at -3 dB, and 100.00% for all three measures at 0 dB. Across all 50 scenes per condition, it detected all 111 true events with five false detections at -3 dB and none at 0 dB. Two of the -3-dB false detections occurred in one event-free scene. Perfect observed scores in the 0-dB sample do not establish zero underlying error risk.

Multi-Template Matching (MTM)

MTM-derived achieved mean precision, recall, and F1 of 75.52%, 76.54%, and 70.71% at -3 dB, and 78.29%, 85.96%, and 77.30% at 0 dB. All-scene pooled counts were TP/FP/FN = 77/41/34 and 90/41/21, respectively. Each condition contained seven event-free false detections across seven negative scenes (28/min over 15 seconds).

Convolutional Neural Network (CNN)

The primary CNN pipeline achieved mean precision, recall, and F1 of 94.29%, 98.87%, and 95.90% at -3 dB, and 94.29%, 100.00%, and 96.53% at 0 dB. All-scene pooled counts were TP/FP/FN = 109/10/2 and 111/10/0, respectively. It produced three false detections in two event-free scenes at each condition (12/min over 15 seconds). Its observed recall was high, but its mean F1 remained below BOTM-derived in both conditions.

Comparative summary

BOTM-derived had the highest observed mean F1 in both primary conditions, followed by the CNN, MTM-derived, and STEO. Tightening the timing tolerance to ±5 ms reduced CNN mean F1 to 88.18% at -3 dB and 88.73% at 0 dB; its ±10-, ±15-, and ±20-ms scores were identical. BOTM-derived mean F1 was

97.97% and 100.00% at ±5 ms. This analysis reused the same detections and did not add independent observations. The residual-CNN findings are reported separately in Supplement S1 and do not replace these four-method results.

The following figures show common-input examples (Figures 5–8), primary scene-averaged metrics (Figure 9), event-free controls (Figure 10), and timing-tolerance sensitivity (Figure 11).

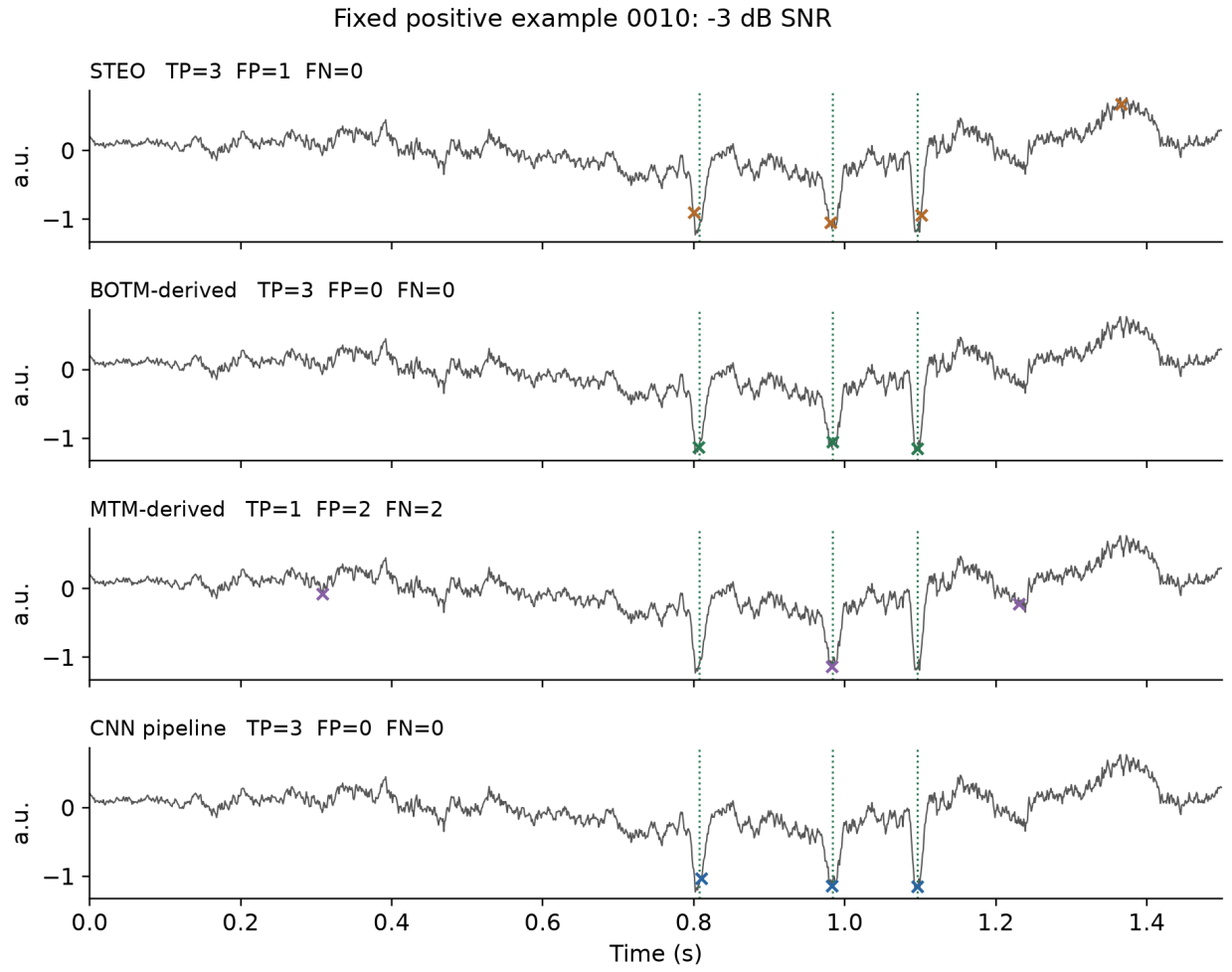


Figure 5. Fixed event-containing test scene 0010 at -3 dB, shown identically for all four pipelines. Dotted green lines indicate true events; crosses indicate detections. Panel TP/FP/FN counts use one-to-one matching within ±15 ms. This example illustrates behavior, not average performance.

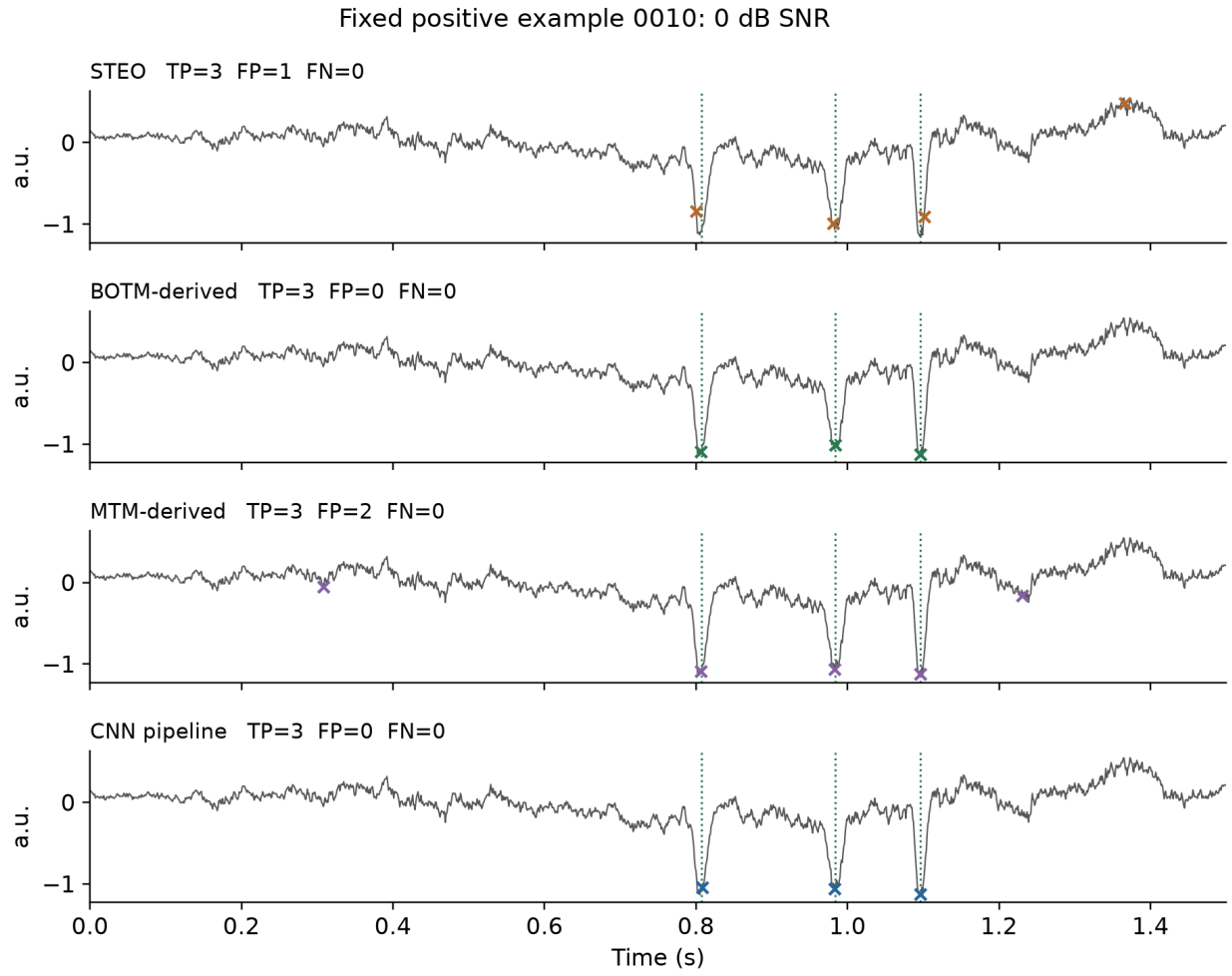


Figure 6. The paired 0-dB version of test parent scene 0010, with the same event track as Figure 5. All four pipelines receive the identical observed array. Dotted lines mark truth and crosses mark detections; counts use ±15-ms matching.

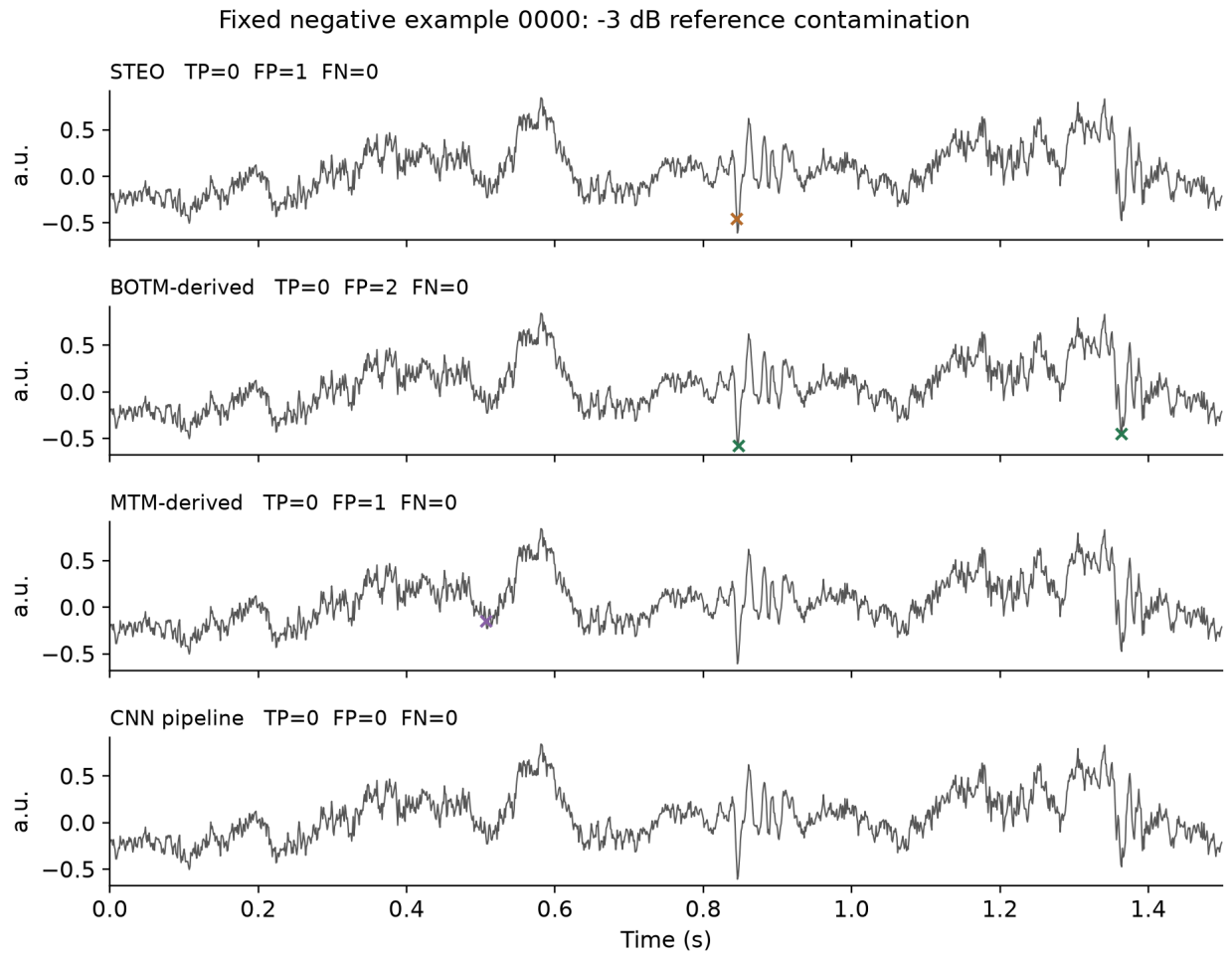


Figure 7. Fixed event-free test example at the -3-dB reference contamination setting, shown identically for all methods. Every detection is a false positive. Actual event-based SNR is undefined because no events are present; the setting uses the generator's positive-scene reference power.

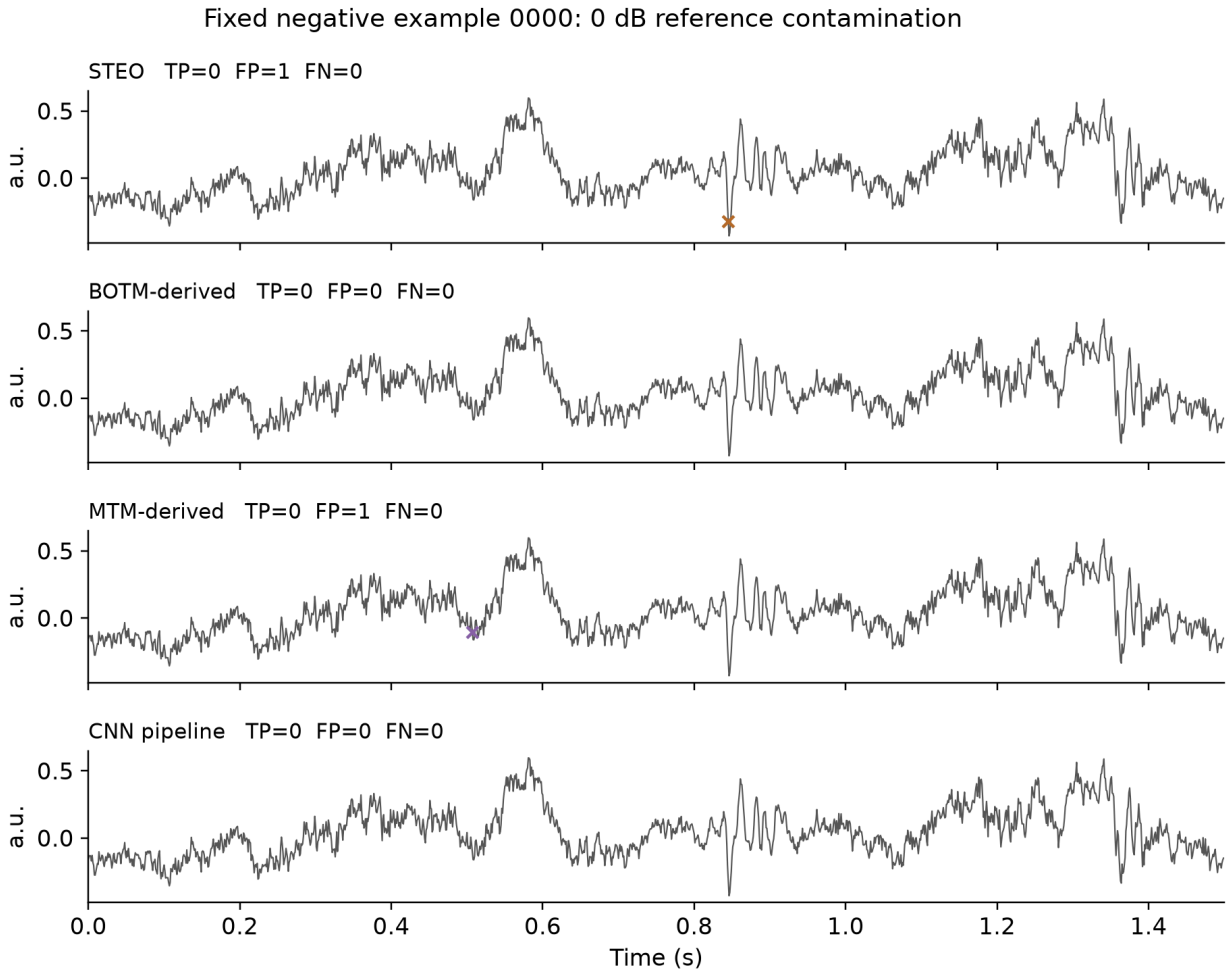


Figure 8. Paired event-free example at the 0-dB reference contamination setting. All methods receive the same observed array and every detection is false. Figures 7–8 are paired conditions, not independent parents.

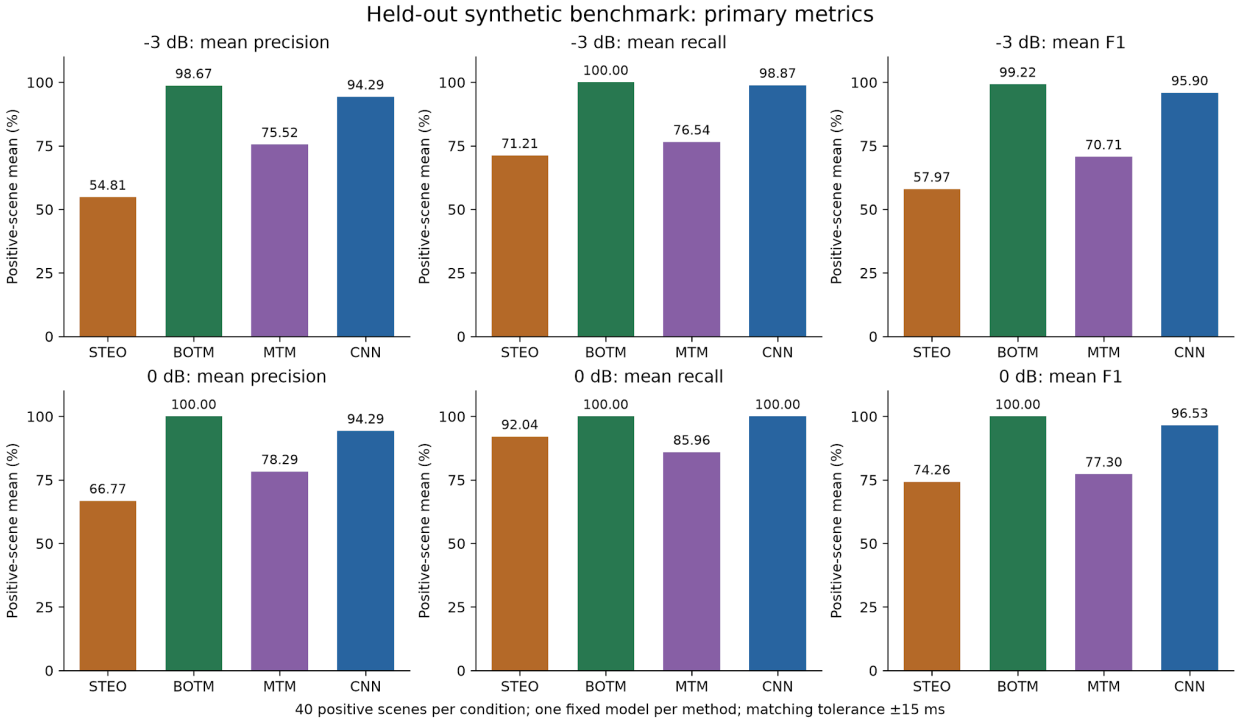


Figure 9. Frozen primary benchmark: precision, recall, and F1 averaged independently across 40 event-containing scenes per condition. These are scene means, not metrics computed from pooled counts. Both noise versions of each parent were retained in the same partition. STEO, BOTM, and MTM denote derived implementations; CNN denotes the primary two-convolution pipeline. Event-free controls are reported separately.

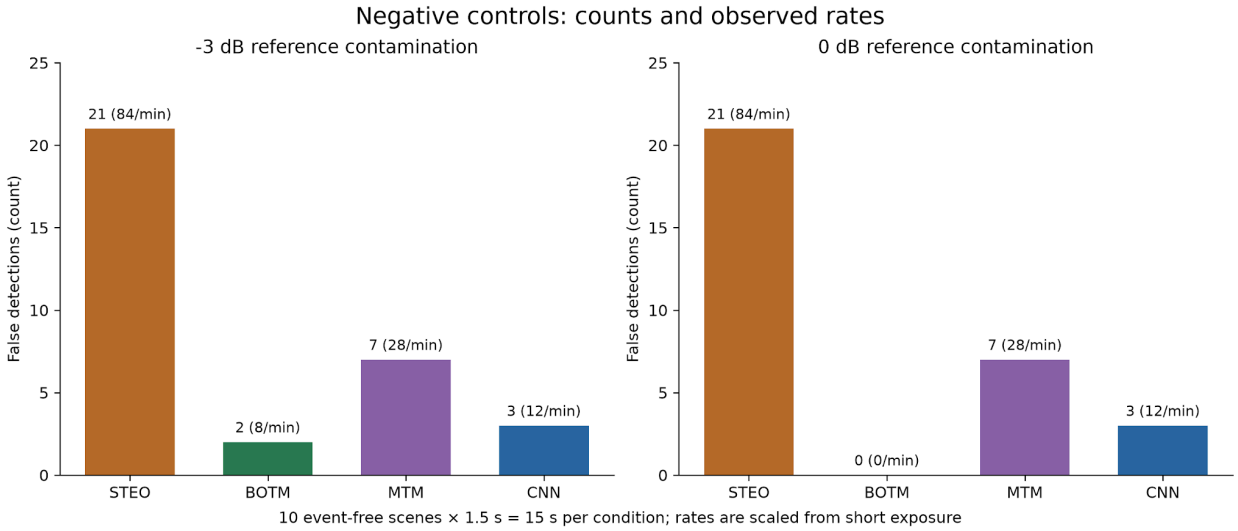


Figure 10. False detections on ten event-free scenes per condition (15 s of spike-free test signals). Labels show counts and rates scaled to one minute; these are not measurements from minute-long clinical recordings. Noise labels indicate reference-contamination settings, not an event-based SNR in the absence of events.

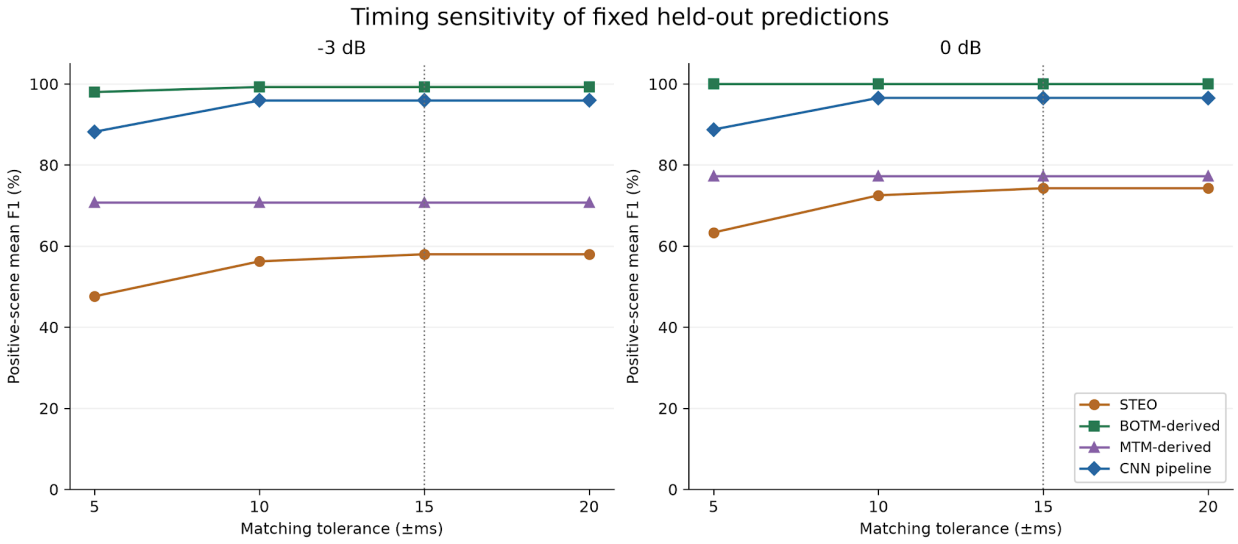


Figure 11. Timing-tolerance sensitivity of the same frozen predictions. Each point is mean scene-level F1 over 40 event-containing scenes per condition after matching within ±5, ±10, ±15, or ±20 ms. The dotted line marks the primary ±15-ms convention. No model was retrained or threshold retuned for this analysis.

DISCUSSION

The two experiments show different strengths of CNN-based detection. In the primary comparison, the compact CNN detected 109 of 111 events at −3 dB and all 111 at 0 dB. BOTM-derived detected all 111 at both noise levels and achieved the highest mean F1. In the extension, three residual CNNs produced fewer false detections overall on new signals but missed more events than BOTM. Their thresholds were chosen using calibration data to retain at least the primary CNN's event-detection count at each noise level while reducing false detections. Separately, the test on new signals required evidence of fewer false detections than BOTM and a 95% confidence interval showing a recall decrease of less than one percentage point. However, this was a study-specific limit, without any clinical standard. The extension addressed a different question and used different threshold-selection rules, so it is reported alongside the four-method comparison rather than replacing it.

STEO

STEO's mean F1 increased from 57.97% at -3 dB to 74.26% at 0 dB. It performed better with weaker contamination, but its false detections on spike-free signals and sensitivity to scoring tolerance show limitations in this implementation. Examining the energy trace around individual errors could help identify where it confused interference with spikes or placed a detection too far from the event minimum. The overall scores alone cannot determine the cause of each error or show whether changing the threshold would correct it.

BOTM

BOTM-derived achieved mean F1 scores of 99.22% and 100.00% at -3 and 0 dB, respectively. This strong performance was obtained using templates and noise covariance estimated from training data, without access to clean test templates. One possible explanation is that the related synthetic waveform families made template matching particularly effective. That explanation remains a hypothesis, however, because the experiment did not independently vary waveform similarity across conditions.

MTM

MTM-derived achieved mean F1 scores of 70.71% and 77.30%, below BOTM-derived and the primary CNN. Its detection process combines template formation, similarity checks, and slope checks, creating several possible points where an event could be missed or a false candidate accepted. Examining these stages separately would help identify which changes might improve the detector. The present scores describe this implementation, rather than the performance of every multi-template matching method.

CNN

The primary CNN detected 109 of 111 events at -3 dB and all 111 at 0 dB. This shows that a compact network, along with template-based timing adjustment and candidate-acceptance checks, could detect the constructed spikes with high sensitivity. The remaining challenge was to reduce false detections without

rejecting true events. This provided a specific reason to investigate the revised training examples and residual network reported in Supplement S1.

The residual-CNN extension provides a more specific result to build on. Across 500 new parent scenes, all three trained residual models produced fewer false detections than the BOTM-derived model. The average reduction was 51.46%, with a recall of 99.486% calculated after averaging the detected and missed event counts across the three trained models. This improvement came with a recall decrease of 0.4656 percentage points, and every residual model missed more events than BOTM. Because recall remains a priority, the next challenge is to retain the reduction in false detections while recovering those missed events. Supplement S1 reports the full counts, intervals, and results at each noise level.

Overall findings

BOTM-derived had the highest mean F1 in the primary comparison. The residual CNNs later reduced the total number of false detections, but each missed more events. Their thresholds had also been selected differently from BOTM's, so the results do not show an advantage at equal recall. They show a setting in which the CNN accepted fewer false events while maintaining high recall. Whether that balance could reduce review time or improve clinical decisions needs testing on real EEG.

Limitations and future work

The main limitation is the use of synthetic data. Known spike times and separate signal components enabled controlled comparisons, but the generated waveforms do not capture the full variety of real EEG. The two noise versions of each parent and the overlapping training windows are related observations, not independent patients or recordings. The primary test included 50 parents and only 15 seconds of event-free data per noise condition. Testing the extension on 500 new parents increased the evaluation sample, but retained the same generator and related waveform families.

Future work should first address the residual CNN's missed events while preserving its lower false-detection count. Changes should be developed using training and validation data, then evaluated on an independent, unseen set of signals. Longer event-free recordings would provide a broader test of false detections. More varied spike shapes, different contamination patterns, and real EEG with verified event annotations would help determine how far the findings extend beyond the present synthetic conditions.

The contribution of the residual connection remains uncertain. He et al. showed how adding a shortcut around a group of layers can make deeper networks easier to train (12). Here, normalization and training examples also changed as the architecture did. In the controlled development comparison, adding the shortcut to the deeper network with revised sampling changed calibration false detections by +3, −4, and −1 across the three seeds, with no missed calibration events in either model. Thus, the shortcut helped in two runs but worsened false detections in one.

Only three residual models were tested on the new confirmation signals. The confidence intervals describe variation among sampled parent scenes for those fixed models. They do not capture the full

variation that could arise from repeatedly training new models. More training seeds and comparisons that change one factor at a time would help separate these effects.

These next steps would test whether the findings extend beyond the waveform shapes and contamination patterns represented by this generator.

CONCLUSION

This study compared three conventional approaches—STEO-derived, BOTM-derived, and MTM-derived—with a compact CNN pipeline on identical held-out synthetic signals. Separate training, validation, and calibration partitions were used before testing. BOTM-derived achieved the highest mean F1 at both -3 and 0 dB, while the CNN detected nearly all inserted events. The primary comparison therefore supports high CNN sensitivity in these conditions, but not overall superiority over the conventional methods.

The separate extension evaluated a CNN with four convolutional layers and a residual shortcut, trained with additional negative window types. Across three trained models, it produced 51.46% fewer false detections on average than BOTM-derived, but recall was 99.486% compared with 99.952%. This was a decrease of approximately 0.466 percentage points. Across both noise conditions combined, the 95% confidence interval for this decrease was 0.225–0.742 percentage points, below the study's chosen limit of one percentage point. Together with the evidence of fewer false detections, this met the two requirements set before testing. The one-percentage-point limit was not established as clinically acceptable, and the requirements were not met separately at both noise levels. Different threshold-selection objectives also prevent interpreting the result as a CNN advantage at equal recall.

This study provides a four-method comparison under controlled conditions and a separate example of CNN development reducing false detections. It also identifies the cost: more missed spikes. The next step is to improve this balance and test it on a wider range of synthetic signals and annotated real EEG data. The results support continued investigation into learned spike detection, while leaving open its clinical usefulness and the independent benefit of residual connections.

ACKNOWLEDGEMENT

I would like to thank Behnoush Rostami for her guidance in developing this research paper.

REFERENCES

1. Sharma R, Meena HK. Emerging trends in EEG signal processing: a systematic review. SN Comput Sci. 2024;5(4):415. doi: 10.1007/s42979-024-02773-w.

2. Semmaoui H, Drolet J, Lakhssassi A, Sawan M. Setting adaptive spike detection threshold for smoothed TEO based on robust statistics theory. IEEE Trans Biomed Eng. 2012;59(2):474-82. doi: 10.1109/TBME.2011.2174992.

3. Franke F, Quian Quiroga R, Hierlemann A, Obermayer K. Bayes optimal template matching for spike sorting – combining Fisher discriminant analysis with optimal filtering. J Comput Neurosci. 2015;38(3):439-59. doi: 10.1007/s10827-015-0547-7.

4. Jiang T, Wu D, Gao F, Cao J, Dai S, Liu J, et al. Improved spike detection algorithm based on multi-template matching and feature extraction. IEEE Trans Circuits Syst II Express Briefs. 2022;69(1):249-53. doi: 10.1109/TCSII.2021.3092141.

5. Rajwal S, Aggarwal S. Convolutional neural network-based EEG signal analysis: a systematic review. Arch Comput Methods Eng. 2023;30(6):3585-615. doi: 10.1007/s11831-023-09920-1.

6. O'Shea K, Nash R. An introduction to convolutional neural networks. arXiv [Preprint]. 2015. arXiv:1511.08458. doi: 10.48550/arXiv.1511.08458.

7. Lawhern VJ, Solon AJ, Waytowich NR, Gordon SM, Hung CP, Lance BJ. EEGNet: a compact convolutional neural network for EEG-based brain-computer interfaces. J Neural Eng. 2018;15(5):056013. doi: 10.1088/1741-2552/aace8c.

8. Schirrmeister RT, Springenberg JT, Fiederer LDJ, Glasstetter M, Eggensperger K, Tangermann M, et al. Deep learning with convolutional neural networks for EEG decoding and visualization. Hum Brain Mapp. 2017;38(11):5391-420. doi: 10.1002/hbm.23730.

9. Cawley GC, Talbot NLC. On over-fitting in model selection and subsequent selection bias in performance evaluation. J Mach Learn Res. 2010;11:2079-107.

10. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15:1929-58.

11. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Bach F, Blei D, editors. Proceedings of the 32nd International Conference on Machine Learning; 2015; Lille, France. PMLR; 2015. p. 448-56. (Proceedings of Machine Learning Research; vol. 37).

12. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. arXiv [Preprint]. 2015. arXiv:1512.03385. doi: 10.48550/arXiv.1512.03385.

SUPPLEMENT S1. RESIDUAL-CNN DEVELOPMENT AND FRESH SYNTHETIC CONFIRMATION

S1.1 Methods and implementation rationale

The extension was designed to examine two possible improvements: changing the training windows and adding depth with or without a residual shortcut. It remained separate from the four-method benchmark. Three architectures were tested: the two-convolution baseline transferred to PyTorch, a deeper CNN, and an equally deep CNN with a shortcut around its added layers. Each architecture was trained using both the original windows and the revised windows described below. Three seeds were used for each

combination—2026091402, 2026092620, and 2026092621—giving 3 architectures × 2 sampling rules × 3 seeds = 18 trained models. The two deeper networks started with identical weights for each paired comparison and had the same number of parameters. This made it possible to compare the shortcut without also changing model size. The original test arrays and earlier confirmation arrays were not used to select these models.

Both deeper CNNs added two convolutional layers after the second pooling stage and before the fully connected layers. Each new layer had 64 input and output channels and used filters covering three neighboring positions. Keeping 64 channels meant the input and output had the same dimensions and could be added directly. The three-position filters added another stage of local shape processing without reducing the number of positions through more pooling. These choices increased depth while leaving the final classifier unchanged.

The added block applied convolution, batch normalization, ReLU, a second convolution, and another batch normalization. The plain network then applied ReLU to the block's output. The residual network first added the block's input back to its output, then applied ReLU. This shortcut follows the idea developed by He et al. (12): the added layers can learn a change in the incoming representation, while the original information has a direct path forward. It is a feed-forward CNN shortcut, not a recurrent loop or memory of earlier windows.

Each deeper model had 240,932 trainable parameters, compared with 216,100 in the baseline. At the deepest convolutional stage, a value could depend on a 42-sample region of the original input instead of 26 samples. The fully connected layer still combined information from across the whole 100-sample window. These were controlled comparisons of specific model sizes; no search established that those sizes were optimal. The residual model reported here contained four convolutional layers and two fully connected layers.

The revised training windows addressed a specific mismatch. The CNN could detect part of a spike anywhere within its 100-sample window, but the subsequent template search could report an event only near the center. All three fitted templates had their lowest point at sample 50, and the search shifted them by up to ten samples in either direction. The reportable region was therefore samples 40–60. Labeling a window positive merely because it contained a spike near its edge could encourage a detection that this window could not place correctly.

All 2,722 original positive windows were retained, with spike minima at samples 40, 45, 50, 55, or 60. The 2,722 negatives were replaced with 1,361 windows containing an off-center spike or tail, 680 spike-free windows overlapping an artifact, and 681 ordinary spike-free background windows. Half of the negatives thus targeted partial spikes, while the remaining half covered artifact and background cases. The quotas preserved the total number of examples and the positive–negative balance, so any benefit could not simply be attributed to more training windows.

For an off-center negative example, every spike minimum had to be at sample 30 or earlier, or sample 70 or later. Samples 31–39 and 61–69 were excluded from this labeling boundary to avoid an abrupt switch

from positive to negative across almost identical windows. The ten-sample separation from the positive region matched the scale of the alignment search. Its exact width was a fixed choice, not an established optimum.

Under this rule, class 0 meant that the window contained no spike whose minimum lay in the central region the detector could report. It could still contain part of a spike elsewhere. During detection, the window continued moving one sample at a time, allowing the same event to appear centrally in another window. No events were removed from the scoring labels, and no boundary windows were skipped during detection.

All 18 models used the same revised validation set: 636 positive windows, 318 off-center negatives, 159 artifact negatives, and 159 ordinary negatives, for a total of 1,272. They also shared the same parent partitions, optimizer settings, loss, and calibration arrays. This prevented different validation examples from deciding when each model stopped training.

Each window had its average amplitude subtracted, then was divided by the same fixed value, 0.25602530655904754. This value was the RMS across all samples of the original 5,444 training windows after separately removing each window's mean. It was calculated once in the earlier normalization experiment and reused across all extension models. Unlike dividing by each window's own standard deviation, this preserved amplitude differences between windows. No validation or test data were used to calculate the scale.

Adam used a learning rate of 0.001 and epsilon 10^{-7}, the small constant in its denominator. Batch size was 32, label smoothing was 0.05, and dropout was 0.30. Training ran for at most 20 epochs and stopped after 4 epochs without a decrease in validation loss. Because sampling, normalization, and architecture differed from the primary CNN, directly comparing that model with the residual CNN cannot identify which single change caused the improvement.

Each extension model was tested on the same 18 combinations of probability, probability-margin, and template-correlation cutoffs using the calibration arrays. A setting was eligible only if it detected at least as many true events at each SNR as the primary CNN had detected there, with no more than the primary CNN's 26 false detections in total. Among eligible settings, selection chose the setting with the fewest false detections, then the one with the most true detections, then the first setting in the listed search order. This aimed to reduce false detections without sacrificing the primary CNN's calibration event counts.

All three selected residual models used a winning spike-class probability of at least 0.50 and a template correlation of at least 0.70. Their probability-margin cutoff was zero, meaning the winning class did not need an additional positive separation from the runner-up. The template timing search, local energy check, and duplicate-removal rules were unchanged. Table S1 records the training duration for each residual model. The calibration rules used the existing benchmark as a development reference; they did not guarantee the same recall on new signals.

Table S1. Training duration of the three residual models. The best epoch supplied the saved weights; training could continue for four further epochs while checking for improvement.

Training seed	Best epoch	Stopping epoch	Optimizer updates
2026091402	5	9	1,539
2026092620	8	12	2,052
2026092621	4	8	1,368

Confirmation used 500 new parent scenes from ten generator seeds, 2026092801–2026092810. There were 400 spike-containing and 100 spike-free parents, each created at -3 and 0 dB, giving 1,000 arrays. The signal generator, shape families, and contamination settings stayed unchanged. This larger set increased the number of test scenes; it did not add training data. The size was fixed before evaluation as a practical budget, without a formal calculation of the sample needed to prove superiority.

The main comparison tested all three residual models trained with revised windows against the fixed BOTM-derived detector with $\delta = -4$. Two additional CNN groups provided context. The two-convolution comparison models, labeled “Matched baseline” in Figure S1, were trained in the same 18-model experiment using the original windows. The residual models used the revised windows, so the training examples were not identical between these groups. The “Previous CNN” models in Figure S1 were the three two-convolution models from the earlier experiment that divided every mean-centered window by the same training-derived scale. They were transferred from TensorFlow to PyTorch, retaining their trained weights and detection settings. They were distinct from both the original primary CNN and the newly trained baselines. No weights, thresholds, or generator settings were changed after confirmation began, and no single best training seed was selected.

For the main comparison, TP, FP, and FN were averaged across the three residual models before calculating pooled recall. The models' predictions were not combined into an ensemble. Differences were calculated as residual CNN minus BOTM for false detections per array and pooled recall.

To estimate variation across signals, parents were sampled with replacement 10,000 times using seed 2026092811. Sampling was performed separately within each generator-seed group and within spike-containing and spike-free groups. Whenever a parent was sampled, both the noise versions and every model's results for that parent were kept together. This paired-parent bootstrap preserved the shared inputs and the dependence between the two noise conditions. The 2.5th and 97.5th percentiles of the resulting differences formed the approximate 95% confidence intervals.

Before testing, we set two requirements for the comparison with BOTM using both noise levels together. First, the entire 95% confidence interval for the difference in false detections per array (residual CNN minus BOTM) had to be below zero. Second, the entire 95% confidence interval for the recall difference

had to be above −1 percentage point. In other words, the comparison required evidence of fewer false detections while limiting the supported decrease in recall to less than one percentage point. This limit was a study-specific choice, not an established clinically acceptable loss. It was used to assess the results on new signals, not to select detection thresholds. These intervals apply to the fixed models and this signal generator. With rare errors and only three trained models, they do not describe every possible future training run.

S1.2 Results

All 18 models detected every calibration event and stayed within the limit of 26 false detections used when selecting thresholds. Table S2 shows the mean false detections across three training seeds for each architecture and training-window choice, pooling both noise levels. Because these same calibration data selected the thresholds, the numbers describe development performance rather than an independent test.

Table S2. Mean calibration false detections across three trained models for each architecture and window choice, pooled over both noise levels. Every model detected all calibration events.

Architecture	Original windows	Revised windows
Two-convolution baseline	7.33	7.00
Deeper plain CNN	6.67	6.00
Residual CNN	6.00	5.33

The revised residual condition had the lowest mean false-detection count, motivating its evaluation on new signals. However, neither the revised sampling nor the residual shortcut reduced false detections in every paired training run.

On the new confirmation signals, the residual CNNs averaged 27.67 false detections compared with BOTM-derived's 57, a reduction of 51.46%. Precision and recall calculated from the averaged counts were 98.678% and 99.486%, respectively. BOTM-derived achieved 97.326% precision and 99.952% recall. The residual CNNs therefore made fewer false reports, but also missed more true events. Their high recall should not be described as unchanged recall.

Table S3 gives the event counts for each residual model and the BOTM-derived model. Every residual model produced fewer false detections and more misses than BOTM.

Table S3. Confirmation event counts over all 1,000 arrays. TP means correctly detected events, FP means false detections, and FN means missed events. Noise-paired presentations of a spike are counted separately here; statistical resampling still uses whole parents.

Method / training seed	**TP**	**FP**	**FN**
Residual / 2026091402	2,062	27	14
Residual / 2026092620	2,066	21	10
Residual / 2026092621	2,068	35	8
BOTM-derived	2,075	57	1

The average residual-minus-BOTM difference was −0.02933 false detections per array, with a 95% interval from −0.046 to −0.014. The recall difference was −0.4656 percentage points, with an interval from −0.7415 to −0.2252. Across both noise levels together, the false-detection interval was entirely below zero and the recall-difference interval entirely above −1 percentage point, satisfying both requirements set before testing. Nevertheless, the recall interval remained entirely below zero, supporting a recall loss rather than equivalent sensitivity.

The reduction in false detections was concentrated at -3 dB: the residual models averaged 24, compared with BOTM's 55. At 0 dB, the residual models averaged 3.67 false detections and 2.67 misses, compared with BOTM's two false detections and no misses. At -3 dB, the recall-loss interval reached 1.143 percentage points. Thus, meeting the false-detection and recall-loss requirements with both noise levels combined did not mean meeting them separately at each level.

Against the newly trained two-convolution baseline, mean false detections fell from 34.67 to 27.67 and mean misses from 11.67 to 10.67. In that comparison, the baseline used the original training windows, and the residual model used the revised windows. Both architecture and sampling changed, so the difference cannot be attributed to the shortcut alone.

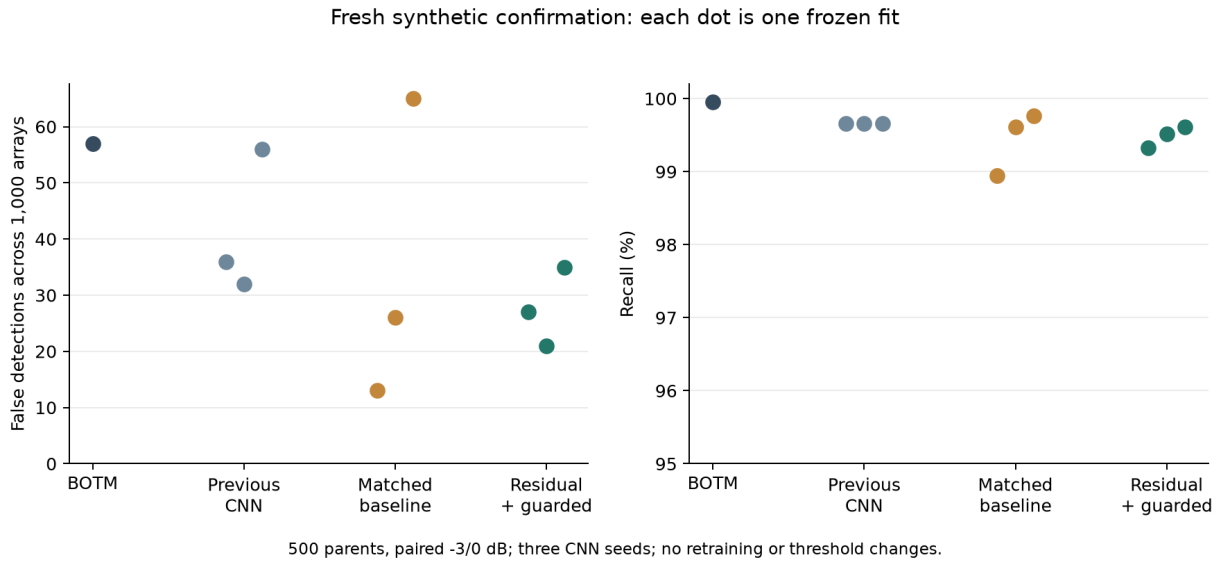


Figure S1. Evaluation on 500 new synthetic parents and 1,000 paired-noise arrays. Each CNN point represents one trained model whose weights and detection settings were fixed before testing; BOTM-derived has one point because only one fitted detector was tested. “Previous CNN” denotes the earlier two-convolution models that divided mean-centered windows by one training-derived scale, not the primary TensorFlow model. “Matched baseline” denotes the two-convolution comparison models trained in the new development experiment using the original windows; the residual models used revised windows. All three residual models produced fewer false detections overall than BOTM-derived, while all had lower recall. The recall axis is truncated to 95–100% to show differences; see the reported TP/FP/FN counts for their absolute scale.

S1.3 Interpretation

All three residual models produced fewer false detections overall than BOTM on the new signals, which makes this a useful direction for further development. However, their thresholds were selected to minimize false detections while detecting at least as many calibration events at each noise level as the primary CNN and producing no more than 26 false detections in total. BOTM retained the threshold chosen to maximize F1 in the primary calibration. Part of the difference could therefore come from how strictly each method accepted candidates.

The extension does not show fewer false detections at the same recall, because every residual model missed more events. Nor does it show that the residual shortcut alone caused the improvement. The next test is whether the missed events can be recovered while retaining the reduction in false detections, followed by an evaluation on real EEG data.