BETA-AMR: BATCH-CONSISTENT EFFICIENT TEMPORAL MODELING FOR AUTOMATIC MODULATION RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Automatic modulation recognition (AMR) plays a vital role in wireless communications by identifying signal modulation types from raw IQ samples under strict efficiency constraints. Existing methods focus almost exclusively on deployment metrics like parameter size and inference latency, while accuracy may change substantially when the training batch size changes. To study this trade-off, we propose BETA-AMR, a batch-consistent, highly efficient temporal architecture designed for parallel training and efficient single-sample deployment. Rather than relying on sequential state propagation or heavy multi-head attention, BETA-AMR introduces a fully parallel, dilated temporal-convolution backbone that scales the receptive field without sequential $O(T)$ dependencies. Group Normalization avoids batch-dependent statistics in the feedforward path, while experiments separately measure sensitivity to training batch size. Furthermore, a lightweight, non-pairwise temporal aggregation layer adaptively isolates high-information burst regions with minimal computational overhead compared to $O(T^2C)$ self-attention. Under a unified protocol, BETA-AMR achieves the highest accuracy among compared methods under the batch-2048 protocol across three length-128 benchmarks with an ultra-low batch-1 latency of 1.04 ms. When scaling training batch size from 128 to 2048, its accuracy drops by only 0.75 ± 0.14 points on RML2016.10A and 0.80 ± 0.35 points on RML22.01A across multiple seeds. On the longer RML2018 benchmark, expanding its receptive field boosts accuracy by 1.7–1.8 points, bridging the gap to a similarly sized recurrent hybrid to under 0.1 points. Together these results motivate reporting training-batch sensitivity and execution speed alongside accuracy and deployment latency.

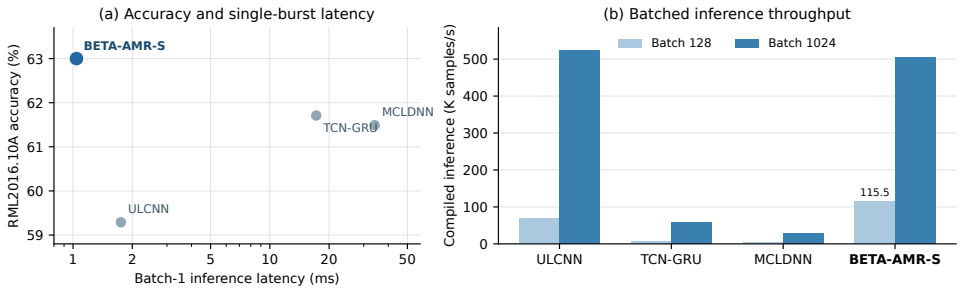

Figure 1: Measured BETA-AMR operating point. Accuracy is from RML2016.10A models trained at batch 2048; latency and throughput are standardized compiled BF16 inference on one RTX 4090. The method path is shown in Figure 2.

1 INTRODUCTION

Automatic modulation recognition (AMR) plays a pivotal role in modern wireless communication systems by identifying the modulation type of received radio signals without prior transmitter parameters. It serves as a foundational building block for cognitive radio, spectrum monitoring, and non-cooperative signal analysis. Since the inception of end-to-end AMR benchmarks (O'Shea & West, 2016; O'Shea et al., 2016), the community has developed diverse deep architectures, ranging

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

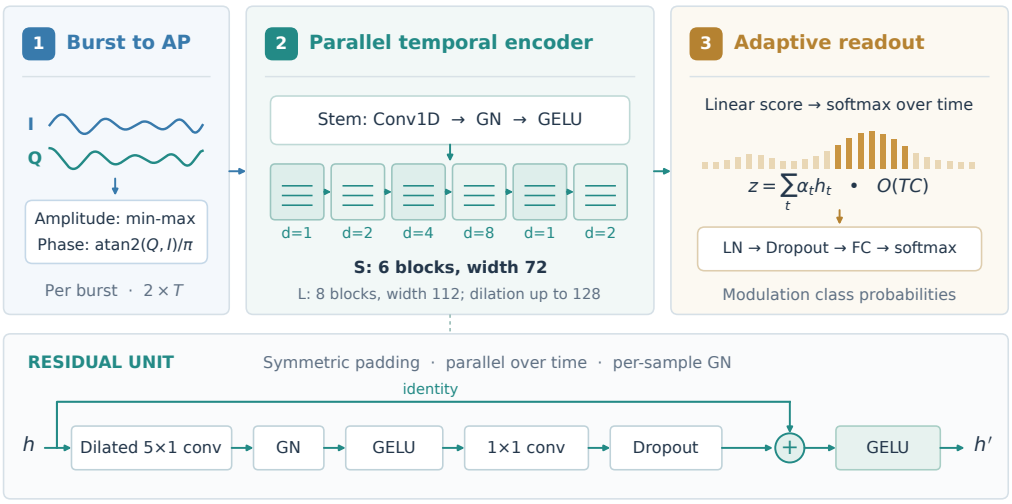

Figure 2: BETA-AMR architecture (schematic). An I/Q burst is transformed to per-burst normalized amplitude and wrapped phase divided by π, encoded by a convolutional stem and dilated residual blocks, then read out by learned temporal weights and a classifier. The inset shows the block operations in code order; the compact variant uses six blocks, while the larger variant extends width and receptive field.

from residual CNNs and recurrent hybrids to Transformer variants and mixture-of-experts models (He et al., 2016; Wang et al., 2021; Qu et al., 2025; Wang et al., 2026).

Despite these advances, efficient AMR research is often summarized through parameter count and batch-1 latency. Batched inference throughput and the accuracy retained when the training batch size changes are also operationally relevant. Recurrent hybrids can capture temporal structure but impose sequential state updates; parallel convolutions offer a shorter critical path. The present work studies that accuracy–efficiency trade-off under a shared AMR protocol. Our retention metric compares *separately trained* models at different training batch sizes; single-sample inference latency is measured independently.

To study this accuracy–efficiency trade-off, we propose BETA-AMR, a parallel temporal architecture evaluated for accuracy, throughput, and single-sample latency. Architecturally, BETA-AMR refactors temporal feature extraction via three core design mechanisms: First, it replaces autoregressive recurrence with stacked 1D dilated convolutions with symmetric padding, expanding the receptive field while keeping time positions parallel within each layer. Second, it enforces channel-group spatial-temporal normalization rather than mini-batch statistics, decoupling feedforward representations from batch size and removing the dependency on batch-wise statistics during single-sample deployment. Third, a lightweight temporal mechanism adaptively weights informative sequence segments to capture burst signatures in $O(TC)$ linear complexity, avoiding quadratic $O(T^2C)$ attention overheads.

Our main contributions are summarized as follows:

- We challenge the standard evaluation paradigm of efficient AMR by examining the practical training and deployment trade-off. We report training-batch retention and compiled inference throughput alongside parameter size and single-sample latency.
- We propose BETA-AMR, a compact framework that eliminates both $O(T)$ sequential recurrent bottlenecks and $O(T^2C)$ dense token interactions. GroupNorm makes its per-sample forward path independent of other inference samples; training-batch sensitivity remains an empirical question.
- Across four public benchmarks, the compact model leads the compared methods on the three short-burst datasets at training batch 2048; the scaled model approaches the recurrent comparator on the longer benchmark. Batch-1 inference latency is 1.04 ms. Component and multi-seed controls characterize the observed accuracy and training-batch behavior.

2 Related Work

Recurrent, hybrid temporal modeling. Because modulation signatures are intrinsically encoded in time-varying amplitude, phase, and instantaneous frequency shifts, sequence modeling is foundational to modern AMR. Pioneering hybrids such as MCLDNN combine convolutional front-ends with LSTM layers (Wang et al., 2021), TLDNN integrates Transformer-LSTM blocks with segment substitution (Qu et al., 2025), and TCN-GRU couples temporal convolutions with GRUs (Wang et al., 2024). Despite their strong representation power, autoregressive state propagation in these recurrent hybrids imposes an inescapable $O(T)$ sequential dependency, creating a severe bottleneck during large-batch GPU training. BETA-AMR fundamentally departs from this paradigm: rather than adding recurrent units, it enforces a fully parallel temporal path with exponential receptive field expansion, preserving maximum training parallelization while capturing rich sequence dynamics.

Convolutional and attention-based alternatives. Pure convolutional sequence models such as TCN and ModernTCN already establish parallel temporal modeling and broad receptive fields (Bai et al., 2018; Luo & Wang, 2024). Our contribution is the evaluated AMR configuration and its joint accuracy–efficiency characterization, not a new convolution principle.

Attention-based architectures and scope differentiation. To bypass recurrent bottlenecks, recent works have explored Transformers, attention mechanisms, and mixture-of-experts designs for AMR (Vaswani et al., 2017; Gao et al., 2023; Wang et al., 2026). Frameworks like MCformer tailor complex-valued attention to radio bursts (Hamidi-Rad & Jain, 2021), while other hybrids incorporate graph structures, multi-domain features, or expert routing (Cai et al., 2022; Wang et al., 2023; Ruikar et al., 2024; Hou et al., 2023; Xing et al., 2024; Di et al., 2024). However, heavy all-to-all token mixing incurs quadratic $O(T^2C)$ memory complexity, making it ill-suited for short, noisy IQ signals under strict latency constraints. Under matched protocols, empirical controls show that attention-heavy backbones do not consistently outperform simpler temporal designs. BETA-AMR translates these insights into core design principles: expanding the receptive field without recurrences, decoupling normalization from mini-batch statistics, and employing lightweight non-pairwise feature aggregation for efficient burst classification.

3 Method

Notations and problem formulation

Notations. Let $\mathbf{x} \in \mathbb{R}^{2\times T}$ represent a received complex-valued radio burst consisting of in-phase (I) and quadrature (Q) components across T discrete time steps, i.e., $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]^\top$. Each signal burst is associated with a ground-truth modulation category $y \in \mathcal{Y} = \{1, 2, \ldots, C\}$, where C denotes the total number of candidate modulation schemes. The received signal is contaminated by channel noise under a specific signal-to-noise ratio (SNR) level $s \in \mathcal{S}$. We define a labeled AMR dataset as $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}, s^{(i)})\}_{i=1}^{N}$, where N is the total number of signal samples.

Problem statement. In standard deep learning paradigms for AMR, the objective is to learn a parameterized neural network classifier $f(\cdot;\boldsymbol{\theta}) : \mathbb{R}^{2\times T} \to \mathbb{R}^C$ with parameters $\boldsymbol{\theta}$. The network maps an input burst $\mathbf{x}$ to a predicted class probability distribution $\hat{\mathbf{y}} = \mathrm{softmax}(f(\mathbf{x};\boldsymbol{\theta}))$. Model parameters are optimized via cross-entropy loss over a batch $\mathcal{B}$ of size $B = |\mathcal{B}|$:

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{\theta};\mathcal{B}) = -\frac{1}{|\mathcal{B}|}\sum_{(\mathbf{x},y)\in\mathcal{B}}\sum_{c=1}^{C}\mathbf{1}(y=c)\log\hat{y}_c, \tag{1}$$

where $\mathbf{1}(\cdot)$ is the indicator function, and $\hat{y}_c$ denotes the predicted probability for the c-th class. The model parameters are updated iteratively via gradient descent under batch size B:

$$\boldsymbol{\theta}_{\tau+1} = \boldsymbol{\theta}_\tau - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{CE}}(\boldsymbol{\theta}_\tau;\mathcal{B}_B^{(\tau)}), \tag{2}$$

where η is the learning rate, τ represents the training iteration index, and $\mathcal{B}_B^{(\tau)}$ denotes the mini-batch sampled under batch size configuration B.

Evaluation metrics and batch consistency. Conventional AMR protocols evaluate performance using the average classification accuracy across all tested SNR levels $\mathcal{S}$:

$$\mathrm{Acc}_{\mathrm{avg}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{\sum_{(\mathbf{x},y,s) \in \mathcal{D}_s} \mathbf{1}(\arg\max_c f_c(\mathbf{x};\boldsymbol{\theta}) = y)}{|\mathcal{D}_s|}, \tag{3}$$

where $\mathcal{D}_s \subset \mathcal{D}$ represents the subset of bursts under SNR level s. To quantify sensitivity to training batch size, we introduce the *cross-batch retention rate*. Let $\boldsymbol{\theta}_B$ denote the optimal parameters converged under training batch size B. We formally define the accuracy retention ratio when scaling the training batch size from a standard small batch B_s to a throughput-oriented large batch B_l ($B_l > B_s$):

$$\mathrm{Retain}_{B_l \leftarrow B_s} = 100 \times \frac{\mathrm{Acc}_{\mathrm{avg}}(\boldsymbol{\theta}_{B_l})}{\mathrm{Acc}_{\mathrm{avg}}(\boldsymbol{\theta}_{B_s})}\,(\%). \tag{4}$$

In our experiments, we set $B_s = 128$ and $B_l = 2048$. High retention ($\mathrm{Retain}_{B_l \leftarrow B_s} \approx 100\%$) means that the separately trained large-batch model approaches the small-batch model's accuracy. Batch-1 inference latency is measured independently.

Table 1: Dataset summary for the controlled experiments.

Dataset	Samples	Classes	Length	SNR range
RML2016.10A	220K	11	128	-20 to 18
RML2016.10B	1.2M	10	128	-20 to 18
RML22.01A	462K	11	128	-20 to 20
RML2018.01A	>2M	24	1024	-20 to 30

FULLY PARALLEL TEMPORAL CONVOLUTIONAL ENCODER

To avoid the sequential critical path of recurrent models while avoiding the quadratic $O(T^2C)$ memory explosion of dense self-attention, BETA-AMR employs a fully parallel temporal convolutional backbone with dilated receptive field expansion.

Input transformation and feature stem. Given an input burst $\mathbf{x} \in \mathbb{R}^{2\times T}$ containing raw in-phase (I) and quadrature (Q) samples, we first transform it into an Amplitude-phase (AP) representation $\mathbf{x}_{\mathrm{AP}} \in \mathbb{R}^{2\times T}$:

$$a_t = \sqrt{\mathbf{x}(1,t)^2 + \mathbf{x}(2,t)^2 + 10^{-12}},$$
$$\mathbf{x}_{\mathrm{AP}}^{(1,t)} = \frac{a_t - \min_\tau a_\tau}{\max(\max_\tau a_\tau - \min_\tau a_\tau, 10^{-12})}, \quad \mathbf{x}_{\mathrm{AP}}^{(2,t)} = \frac{\mathrm{atan2}(\mathbf{x}(2,t), \mathbf{x}(1,t))}{\pi}.$$

where amplitude is min–max normalized within each burst and the wrapped phase is scaled to $(-1, 1]$. The AP transformed signal is then projected into a high-dimensional feature space via a 1D convolutional stem, GroupNorm, and GELU to yield initial temporal features $\mathbf{H}^{(0)} \in \mathbb{R}^{C\times T}$:

$$\mathbf{H}^{(0)} = \mathrm{GELU}\big(\mathrm{GN}(\mathrm{Conv1D}_{\mathrm{stem}}(\mathbf{x}_{\mathrm{AP}}))\big), \tag{5}$$

where C denotes the channel width.

Dilated temporal residual blocks. The core encoder consists of a stack of L dilated residual temporal blocks with symmetric padding. For a block at depth $l \in \{1, \ldots, L\}$ with odd kernel size K and dilation rate d_l, the dilated 1D convolution at time step t is computed as:

$$\tilde{\mathbf{h}}_t^{(l)} = \sum_{k=0}^{K-1} \mathbf{W}_k^{(l)} \mathbf{h}_{t+d_l(k-\lfloor K/2 \rfloor)}^{(l-1)}, \tag{6}$$

where $\mathbf{W}_k^{(l)} \in \mathbb{R}^{C\times C}$ represents the learnable filter weight for k-th tap. The residual temporal transformation $\phi(\cdot)$ inside the l-th block is formulated as:

$$\mathbf{H}^{(l)} = \mathrm{GELU}\Big(\mathbf{H}^{(l-1)} + \mathrm{Dropout}\Big(\mathcal{W}_{\mathrm{proj}}^{(l)}\big(\mathrm{GELU}(\mathrm{GN}(\tilde{\mathbf{H}}^{(l)}))\big)\Big)\Big),$$

Table 2: Performance evaluation across four benchmark datasets under unified experimental protocols. Avg., Low, High, and Max denote accuracy percentages.

	RML2016.10A					RML2016.10B				
Method	Avg.	Low	High	Max	Params	Avg.	Low	High	Max	Params
ResNet	61.91	32.78	91.03	92.64	157K	64.59	35.91	93.28	93.79	157K
TLDNN	60.56	32.54	88.58	89.77	245K	65.20	37.13	93.27	93.68	245K
AMCNet	60.82	31.67	89.98	91.73	466K	64.36	35.76	92.96	93.43	466K
ULCNN	59.29	31.21	87.37	89.50	9K	61.58	33.73	89.42	90.23	9K
TCN-GRU	61.71	32.59	90.82	91.86	254K	64.51	36.10	92.92	93.28	254K
MCLDNN	61.49	32.77	90.21	91.50	690K	65.01	36.60	93.41	93.88	690K
BETA-AMR-S	**63.00**	**33.49**	**92.51**	**93.59**	190K	**65.30**	**37.06**	**93.54**	**93.98**	190K

	RML22.01A					RML2018.01A				
Method	Avg.	Low	High	Max	Params	Avg.	Low	High	Max	Params
ResNet	68.91	41.11	94.18	98.61	157K	61.37	16.94	89.14	96.42	158K
TLDNN	65.47	38.18	90.29	94.75	245K	61.70	16.41	90.02	96.67	281K
AMCNet	68.05	40.86	92.76	97.32	466K	–	–	–	–	–
ULCNN	63.10	36.78	87.02	90.86	9K	–	–	–	–	–
TCN-GRU	67.57	39.98	92.65	97.46	254K	61.54	15.82	90.12	96.79	255K
MCLDNN	68.95	41.79	93.65	98.05	690K	**63.79**	18.27	92.23	97.46	692K
BETA-AMR-S	**70.00**	**43.12**	**94.44**	**98.46**	190K	61.87	16.17	90.43	96.95	191K
BETA-AMR-L	–	–	–	–	–	63.71	17.94	**92.32**	**97.54**	610K

where $\mathrm{GN}(\cdot)$ denotes group normalization, $\mathrm{GELU}(\cdot)$ is the activation function, and $\mathcal{W}_{\mathrm{proj}}^{(l)}$ is a 1×1 channel projection. The compact model uses $K = 5$ and dilations $(1, 2, 4, 8, 1, 2)$; the larger model uses $(1, 2, 4, 8, 16, 32, 64, 128)$. The block stack has receptive field $R = 1 + (K - 1)\sum_{l=1}^{L} d_l$, before the stem. Time positions are computed in parallel within each layer, leaving an $O(L)$ depth dependency rather than an $O(T)$ recurrent one.

PER-SAMPLE NORMALIZATION AND TRAINING-BATCH SENSITIVITY

GroupNorm computes feature statistics independently for each burst. This is a useful implementation property for single-sample inference, but it does not by itself guarantee accuracy retention when training batch size changes. We report that behavior empirically and compare it with BatchNorm in the component study.

Formally, for a feature tensor $\mathbf{H} \in \mathbb{R}^{B \times C \times T}$ within a mini-batch of size B, traditional $\mathtt{BN}$ computes mean and variance statistics across the mini-batch dimension:

$$\mu_{\mathtt{BN}}(c) = \frac{1}{B \cdot T} \sum_{i=1}^{B} \sum_{t=1}^{T} \mathbf{H}_{i,c,t}, \tag{7}$$

$$\sigma_{\mathtt{BN}}^2(c) = \frac{1}{B \cdot T} \sum_{i=1}^{B} \sum_{t=1}^{T} (\mathbf{H}_{i,c,t} - \mu_{\mathtt{BN}}(c))^2. \tag{8}$$

BatchNorm uses batch statistics during training and stored running statistics at evaluation. Changing training batch size can therefore affect optimization and running estimates, although these effects are not separated in our experiments.

To resolve this, BETA-AMR enforces channel-group spatial-temporal normalization, dividing the C channels into G independent groups. For the i-th burst sample, the statistics are calculated strictly within its own sub-channel representation space across sequence length T:

$$\mu_{\mathtt{GN}}(i, g) = \frac{1}{(C/G) \cdot T} \sum_{c \in \mathcal{C}_g} \sum_{t=1}^{T} \mathbf{H}_{i,c,t}, \tag{9}$$

$$\sigma_{\mathtt{GN}}^2(i, g) = \frac{1}{(C/G) \cdot T} \sum_{c \in \mathcal{C}_g} \sum_{t=1}^{T} (\mathbf{H}_{i,c,t} - \mu_{\mathtt{GN}}(i, g))^2, \tag{10}$$

where $\mathcal{C}_g$ denotes the subset of channels in the g-th group. The normalized and affine-transformed output feature is obtained via:

$$\widehat{\mathbf{H}}_{i,c,t} = \frac{\mathbf{H}_{i,c,t} - \mu_{\mathrm{GN}}(i,g)}{\sqrt{\sigma^2_{\mathrm{GN}}(i,g) + \epsilon}} \cdot \gamma_c + \beta_c, \tag{11}$$

where $\gamma_c, \beta_c \in \mathbb{R}$ are learnable channel-wise scale and shift parameters, and ϵ is a numerical stability constant. At evaluation, GroupNorm does not aggregate over other samples in the batch. This structural property differs from the empirical retention metric, which compares independent optimization runs with training batches of 128 and 2048.

Lightweight non-pairwise temporal aggregation

After feature extraction through the encoder, the sequence of frame-level temporal representations $\mathbf{H}^{(L)} = [\mathbf{h}_1^{(L)}, \mathbf{h}_2^{(L)}, \ldots, \mathbf{h}_T^{(L)}] \in \mathbb{R}^{C \times T}$ must be aggregated into a global burst vector $\mathbf{z} \in \mathbb{R}^C$. Instead of constructing an expensive $O(T^2C)$ dense token interaction matrix as in self-attention, we employ a lightweight non-pairwise temporal aggregation mechanism.

Specifically, a learned linear scoring function $g : \mathbb{R}^C \to \mathbb{R}^1$ evaluates the temporal importance of each frame:

$$e_t = g(\mathbf{h}_t^{(L)}) = \mathbf{w}_a^\top \mathbf{h}_t^{(L)} + b_a, \tag{12}$$

where $\mathbf{w}_a \in \mathbb{R}^C$ and $b_a \in \mathbb{R}$ are trainable. The scalar scores e_t are normalized across time via Softmax to yield dynamic temporal alignment weights α_t:

$$\alpha_t = \frac{\exp(e_t)}{\sum_{\tau=1}^{T} \exp(e_\tau)}, \quad \text{s.t.} \sum_{t=1}^{T} \alpha_t = 1. \tag{13}$$

The global burst-level embedding $\mathbf{z} \in \mathbb{R}^C$ is computed as the dynamically weighted sum of temporal features $\mathbf{z} = \sum_{t=1}^{T} \alpha_t \mathbf{h}_t^{(L)}$. Finally, a linear head predicts the modulation class probability distribution $\widehat{\mathbf{y}} \in \mathbb{R}^{|\mathcal{Y}|}$:

$$\widehat{\mathbf{y}} = \mathrm{softmax}(\mathbf{W}_{\mathrm{cls}} \, \mathrm{Dropout}(\mathrm{LN}(\mathbf{z})) + \mathbf{b}_{\mathrm{cls}}), \tag{14}$$

where LN is LayerNorm, $\mathbf{W}_{\mathrm{cls}} \in \mathbb{R}^{|\mathcal{Y}| \times C}$ and $\mathbf{b}_{\mathrm{cls}} \in \mathbb{R}^{|\mathcal{Y}|}$. This pooling mechanism adaptively weights informative burst regions with a computational complexity of only $O(TC)$, avoiding pairwise token mixing.

4 Experiments

Experimental setup and evaluation protocol

Datasets. We evaluate our framework on four benchmark AMR datasets: RML2016.10A, RML2016.10B, RML22.01A (length-128 bursts, 10–11 classes), and the large-scale RML2018.01A (length-1024 bursts, 24 classes) (DeepSig Inc., 2026). To eliminate confounding variables and ensure strict fairness across varying public protocols, all models are evaluated under a unified implementation pipeline. Unless otherwise noted, every baseline uses identical Amplitude-Phase (AP) inputs, Discrete Segment Substitution augmentation (ratio 0.10) (Qu et al., 2025), cross-entropy loss, and AdamW optimization (Loshchilov & Hutter, 2019) with validation-loss scheduling under mixed precision across training batch sizes $B \in \{128, 1024, 2048\}$.

Compared baselines and benchmarks. We benchmark BETA-AMR against six representative baselines across convolutional, recurrent, hybrid, and lightweight families: ResNet (He et al., 2016), TLDNN (Qu et al., 2025), AMCNet (Zhang et al., 2023), MCLDNN (Wang et al., 2021), ULCNN (Guo et al., 2022), and TCN-GRU (Wang et al., 2024). To eliminate system overhead, standardized efficiency metrics are measured on an isolated NVIDIA RTX 4090 GPU using BF16 execution via `torch.compile` and CUDA event synchronization. Batch-1 streaming latency (including preprocessing) is averaged over 1,000 iterations after 200 warmups. Compiled inference throughput ($B \in \{128, 1024, 2048\}$) is evaluated over 200 passes following 50 warmups. Observed training wall-clock times are reported separately in the supplement.

Table 3: Standardized compiled BF16 inference on an RTX 4090. Latency is batch-1 milliseconds; throughput is in K samples/s.

Model	Lat.	Tput@128	Tput@1024	Tput@2048	Params
ULCNN	1.751	67.9	**523.9**	**1086.4**	9K
TCN-GRU	17.205	6.6	57.7	114.5	254K
MCLDNN	34.097	3.6	29.4	56.5	690K
BETA-AMR-S	**1.041**	**115.5**	504.5	465.3	190K

Main recognition performance

Table 2 evaluates the standardized recognition performance across all four datasets trained with batch size 2048.

Length-128 benchmarks. Across all three short-burst datasets (RML2016.10A, RML2016.10B, RML22.01A), BETA-AMR-S (190K parameters) achieves top average accuracy ($\mathrm{Acc}_{\mathrm{avg}}$), reaching 63.00% on 16A (outperforming ResNet's 61.91% and TCN-GRU's 61.71%) and 70.00% on RML22 (98.46% peak). The ultra-lightweight ULCNN (9K) illustrates the limits of extreme compression, trailing BETA-AMR-S by 3.71, 3.72, and 6.90 percentage points across the three benchmarks, respectively.

Long-sequence scaling on RML2018.01A. On the length-1024 benchmark, scaling capacity from BETA-AMR-S (191K) to BETA-AMR-L (610K) boosts average accuracy from 61.87% to 63.71%. BETA-AMR-L establishes top high-SNR (92.32%) and peak (97.54%) rates, narrowing the gap against the heavy recurrent hybrid (MCLDNN, 63.79%) to a negligible 0.08 percentage points without introducing $O(T)$ sequential bottlenecks.

Inference efficiency and latency

Figure 1 and Table 3 relate recognition accuracy to standardized compiled inference on a dedicated RTX 4090 GPU. These are operator-level measurements; the supplementary long-burst comparison separately reports observed training wall-clock times.

Streaming batch-1 latency. BETA-AMR-S measures 1.04 ms per burst, including AP preprocessing. Table 3 gives $1.68\times$, $16.5\times$, and $32.7\times$ lower latency than ULCNN, TCN-GRU, and MCLDNN, respectively, under this compiled benchmark.

Batched inference throughput. At batch 128, BETA-AMR-S processes 115.5K samples/s, the highest value in Table 3. At batch 1024, it processes 504.5K samples/s, versus 57.7K and 29.4K for TCN-GRU and MCLDNN; ULCNN reaches 523.9K. At batch 2048, ULCNN has the highest throughput. Thus the measured advantage is the combination of short-burst accuracy, low single-sample latency, and competitive batched inference, while training-time comparisons require the separately reported wall-clock data.

Cross-batch stability and consistency

Figures 3(a) and (b) evaluate accuracy preservation across training batch sizes ($B \in \{128, 1024, 2048\}$). Across short-burst datasets, BETA-AMR-S exhibits high resilience, dropping only 0.53 percentage points from $B = 128$ to 2048 (99.20% retention), whereas TCN-GRU and TLDNN lose about 1.3 and 2.06 points, respectively. Multi-seed controls confirm this stability, restricting degradation to 0.75 ± 0.14 points on 16A and 0.80 ± 0.35 points on RML22 with zero instances of large-batch collapse.

Normalization transfer and architectural synergy

Normalization-transfer experiments in Supplementary Table S8 show that replacing BN with GN affects backbones differently. The component ablation in Table 5 is especially informative: the BN

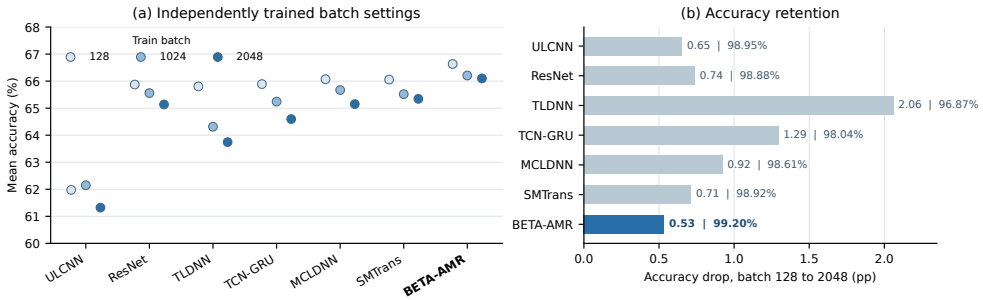

Figure 3: Short-burst accuracy across independently trained batch settings. Each point in (a) averages RML2016.10A, RML2016.10B, and RML22.01A; (b) shows the batch-128 to batch-2048 drop and retention computed from those means. BETA-AMR has a 0.53-point drop and 99.20% retention in this sweep.

Table 4: Architecture probes (under the same internal probe setting, distinct from Table 3) evaluating recognition accuracy, drop (Δ), retention rate (Ret.), and batch-1 inference latency. BatchF., IQF-lite, and SMT-lite denote batch-focused, IQ-feature-only, and spatio-temporal mixing variants, respectively.

Model	Acc@128	Acc@2048	Δ	Ret.	Lat.
BatchF.	49.07	55.25	-6.18	–	0.98
IQF-lite	63.97	63.73	0.24	**99.63**	3.46
SMT-lite	65.03	64.54	0.49	99.25	2.00
Ours	**65.57**	**65.04**	0.53	99.20	**1.16**

variant drops only 0.03 points between batches 128 and 2048, whereas the full GN model drops 0.60 points. GN therefore cannot be credited as the cause of the smaller drop under this metric. The full model has higher absolute accuracy at both batch sizes, which is the supported result.

Multi-seed quantification. The repeated runs over multiple random seeds further rule out random seed variance as the cause of performance retention. On RML2016.10A, BETA-AMR-S achieves $63.66 \pm 0.06\%$ average accuracy at batch 128 and $62.91 \pm 0.09\%$ at batch 2048. On RML22.01A, the corresponding accuracies are $70.56 \pm 0.09\%$ and $69.75 \pm 0.29\%$. The small standard deviations ($\leq 0.29\%$) support reproducibility of these within-model measurements, although baseline uncertainty and optimization-budget effects remain separate questions.

Capacity scaling on long-sequence bursts

To investigate whether the length-1024 performance gap stems from non-recurrent structural bottlenecks or capacity limits, Supplementary Table S9 and Figure 4 evaluate capacity scaling on RML2018.01A. The compact BETA-AMR-S (191K) achieves 62.07% accuracy at batch 1024, trailing heavy recurrent backbones due to its length-128 calibrated receptive field and width. However, expanding width ($C = 72 \rightarrow 112$) and dilations ($d_L \leq 128$) enables BETA-AMR-L (610K) to gain $+1.71$ points at batch 1024 (63.78%) and $+1.84$ points at batch 2048 (63.71%). Crucially, BETA-AMR-L closes the gap against the heavy recurrent hybrid (MCLDNN, 692K) to a negligible 0.08 points (63.71% vs. 63.79%) while securing top stability with the lowest cross-batch drop (0.07 points). These results confirm two operational regimes: BETA-AMR-S provides optimal latency for short-burst edge deployment, while the architecture scales capacity for long sequences without $O(T)$ recurrent or $O(T^2C)$ attention bottlenecks.

Protocol sensitivity and hyperparameter controls

To test whether model rankings depend on these choices, we conduct sensitivity controls across learning rates and segment substitution (SS). On RML2016.10A ($B = 2048$), BETA-AMR-S demonstrates high learning-rate robustness, varying by only 0.30 percentage points across 5×10^{-4} to 2×10^{-3}. In contrast, TCN-GRU improves from 61.71% to 62.11% when tuned to 2×10^{-3},

Table 5: Component ablation averaged over RML2016.10A and RML22.01A across training batch sizes; the BN variant has a smaller batch-size drop but lower absolute accuracy.

Variant	Acc@2048	Acc@1024	Acc@128
Full	**66.50**	66.60	67.10
No dilation	65.88	65.96	66.21
BN	66.26	66.40	66.29
AvgPool	66.38	**66.81**	**67.16**
Kernel 3	66.29	66.60	66.98
No dropout	65.75	66.15	66.38

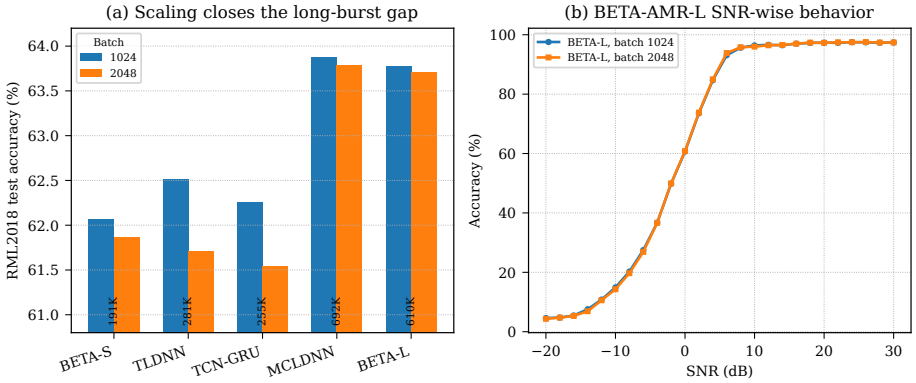

Figure 4: Capacity scaling on RML2018.01A. The larger BETA-AMR model approaches the recurrent comparator at training batches 1024 and 2048; its SNR-wise curves at the two batches nearly overlap. Results and model sizes are detailed in Supplementary Table S9.

yet remains 0.89 points behind default-trained BETA-AMR-S. As a shared augmentation protocol, adding SS improves BETA-AMR-S, TCN-GRU, and ULCNN on 16A by 2.75, 2.81, and 3.18 points, respectively. On RML22.01A, its effect becomes backbone-dependent: BETA-AMR-S gains 3.09 points, TCN-GRU gains 0.48 points, and ULCNN shifts by -0.45 points. Crucially, BETA-AMR-S maintains top rank on 16A without SS and stays competitive on RML22, confirming that AP+SS enhances general representations.

5 Conclusion

BETA-AMR combines a parallel dilated temporal encoder, per-sample normalization, and learned temporal pooling for AMR. Under the shared protocol, the compact model leads the compared methods on three short-burst benchmarks at training batch 2048, while retaining low measured batch-1 GPU latency. Its independently trained large-batch variants show modest accuracy changes across multiple seeds; the scaled model approaches the recurrent comparator on long bursts. The results support reporting accuracy, throughput, training-batch sensitivity, and single-sample latency together. Fixed-compute training and representative deployment hardware remain important tests of the broader systems benefit.

AI Use Statement

Generative AI tools were used to aid and polish writing, retrieve and discover related work, assist with research ideation and execution, and draft sections of this paper. They also assisted with adapting the manuscript to the ICLR 2027 template and checking figure and table placement and PDF layout. The authors take responsibility for the final text, citations, experiments, and claims.

REFERENCES

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. In *ICLR Workshop*, 2018. URL https://arxiv.org/abs/1803.01271. https://arxiv.org/abs/1803.01271.

Jun Cai, Feng Gan, Xiaoyang Cao, and Wei Liu. Signal modulation classification based on the transformer network. *IEEE Transactions on Cognitive Communications and Networking*, 8(3):1348–1357, 2022. doi: 10.1109/TCCN.2022.3163523. URL https://ieeexplore.ieee.org/document/9766054/. https://ieeexplore.ieee.org/document/9766054/.

DeepSig Inc. DeepSig datasets for radio deep learning. https://www.deepsig.ai/datasets/, 2026. URL https://www.deepsig.ai/datasets/.

C. Di et al. SOAMC: A semi-supervised open-set recognition approach for automatic modulation classification. *Electronics*, 13(21):4196, 2024. doi: 10.3390/electronics13214196. URL https://www.mdpi.com/2079-9292/13/21/4196. https://www.mdpi.com/2079-9292/13/21/4196.

Jiaxin Gao, Qinglong Cao, and Yuntian Chen. MoE-AMC: Enhancing automatic modulation classification performance using mixture-of-experts, 2023. URL https://arxiv.org/abs/2312.02298. https://arxiv.org/abs/2312.02298.

Lantu Guo, Yu Wang, Yun Lin, Haitao Zhao, and Guan Gui. Ultra lite convolutional neural network for fast automatic modulation classification in low-resource scenarios, 2022. URL https://arxiv.org/abs/2208.04659. https://arxiv.org/abs/2208.04659.

Shahab Hamidi-Rad and Swayambhoo Jain. MCformer: A transformer based deep neural network for automatic modulation classification. In *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2021. doi: 10.1109/GLOBECOM46510.2021.9685815. URL https://ieeexplore.ieee.org/document/9685815/. https://ieeexplore.ieee.org/document/9685815/.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90. URL https://arxiv.org/abs/1512.03385. https://arxiv.org/abs/1512.03385.

Dongbin Hou, Lixin Li, Wensheng Lin, Junli Liang, and Zhu Han. ClST: A convolutional transformer framework for automatic modulation recognition by knowledge distillation. *arXiv preprint arXiv:2312.17446*, 2023. URL https://arxiv.org/abs/2312.17446. https://arxiv.org/abs/2312.17446.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101. https://arxiv.org/abs/1711.05101.

Donghao Luo and Xue Wang. Moderntcn: A modern pure convolution structure for general time series analysis. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/pdf?id=vpJMJerXHU.

Timothy J. O'Shea and Nathan West. Radio machine learning dataset generation with gnu radio. In *Proceedings of the 6th GNU Radio Conference*, 2016. URL https://www.gnuradio.org/grcon/grcon16/presentations/radio_machine_learning_dataset_generation/. https://www.gnuradio.org/grcon/grcon16/presentations/radio_machine_learning_dataset_generation/.

Timothy J. O'Shea, Johnathan Corgan, and T. Charles Clancy. Convolutional radio modulation recognition networks. In *Proceedings of the International Conference on Engineering Applications of Neural Networks*, pp. 213–226, 2016. URL https://arxiv.org/abs/1602.04105. https://arxiv.org/abs/1602.04105.

Yunpeng Qu, Zhilin Lu, Rui Zeng, Jintao Wang, and Jian Wang. Enhancing automatic modulation recognition through robust global feature extraction. *IEEE Transactions on Vehicular Technology*, 74(3):4192–4207, 2025. doi: 10.1109/TVT.2024.3486079. URL https://arxiv.org/abs/2401.01056. https://arxiv.org/abs/2401.01056.

J. D. Ruikar et al. HCTC: Hybrid convolutional transformer classifier for automatic modulation recognition. *Electronics*, 13(19):3969, 2024. doi: 10.3390/electronics13193969. URL https://www.mdpi.com/2079-9292/13/19/3969. https://www.mdpi.com/2079-9292/13/19/3969.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762. https://arxiv.org/abs/1706.03762.

D. Wang et al. Automatic modulation classification based on CNN-transformer graph network. *Sensors*, 23(16):7281, 2023. doi: 10.3390/s23167281. URL https://www.mdpi.com/1424-8220/23/16/7281. https://www.mdpi.com/1424-8220/23/16/7281.

Jiale Wang, Wupeng Xie, Yaxin Mu, Xin Liu, Zhilong Zhao, and Jingwei Zhang. Mixture-of-experts transformer for automatic modulation recognition, 2026. URL https://arxiv.org/abs/2606.09085. https://arxiv.org/abs/2606.09085.

Na Wang, Yunxia Liu, Liang Ma, Yang Yang, and Hongjun Wang. Multidimensional CNN-LSTM network for automatic modulation classification. *Electronics*, 10(14):1649, 2021. doi: 10.3390/electronics10141649. URL https://www.mdpi.com/2079-9292/10/14/1649. https://www.mdpi.com/2079-9292/10/14/1649.

Zhao Wang, Weixiong Zhang, Zhitao Zhao, Ping Tang, and Zheng Zhang. Robust automatic modulation classification via a lightweight temporal hybrid neural network. *Sensors*, 24(24):7908, 2024. doi: 10.3390/s24247908. URL https://www.mdpi.com/1424-8220/24/24/7908. https://www.mdpi.com/1424-8220/24/24/7908.

Huijun Xing, Xuhui Zhang, Shuo Chang, Jinke Ren, Zixun Zhang, Jie Xu, and Shuguang Cui. Joint signal detection and automatic modulation classification via deep learning, 2024. URL https://arxiv.org/abs/2405.00736. https://arxiv.org/abs/2405.00736.

Jiawei Zhang, Tiantian Wang, Zhixi Feng, and Shuyuan Yang. AMC-Net: An effective network for automatic modulation classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023. URL https://arxiv.org/abs/2304.00445. https://arxiv.org/abs/2304.00445.