Radar-Based Target Classification for UAV Detection

Real Doppler RAD-DAR Dataset: CNN+BiLSTM vs CNN+xLSTM vs CNN+Transformer

Saadet Büşra ÇAM Bilge Şentürk

EE443 Neural Networks Fall 2025-2026 – Final Report
Bilkent University

Abstract

This report addresses multi-class radar target classification for UAV-related scenarios using the Real Doppler RAD-DAR dataset. Radar signatures are represented as 2D time–frequency maps stored as CSV matrices and categorized into three classes: Cars, Drones, and People. A unified preprocessing pipeline is applied, and three model families are compared: CNN+BiLSTM, CNN+xLSTM (two variants), and CNN+TransformerEncoder. Performance is evaluated using accuracy, macro-F1, per-class precision/recall/F1, and confusion matrices. Across balanced subsets, all models achieve strong performance; the primary challenge remains Cars–Drones confusion, while People is consistently well separated.

1 Introduction

Radar sensing is widely used for target detection and classification because it can capture motion-related signatures even under poor visibility. In UAV detection scenarios, separating drones from non-drone targets such as cars and people is important for reliable monitoring. In this project, we classify radar time–frequency representations into three classes: Cars, Drones, and People. We compare three deep learning approaches:

- CNN+BiLSTM as the recurrent baseline,
- CNN+xLSTM as a stabilized recurrent alternative (two variants),
- CNN+TransformerEncoder as a non-recurrent attention-based approach.

The reason radar is valuable for UAV detection is that it encodes *motion* via Doppler shifts. For a moving target with radial velocity v_r, the Doppler frequency shift is proportional to the velocity:

$$f_D = \frac{2v_r}{\lambda}, \tag{1}$$

where λ is the radar wavelength. For complex targets, not only the bulk motion but also micro-motions (e.g., rotor blades, walking limbs) produce *micro-Doppler* components. These micro-Doppler patterns become visible in time–frequency representations and are highly informative for discrimination between drones, cars, and people. Recent radar ML literature emphasizes that micro-Doppler based classification is effective for UAV-related tasks and benefits from deep learning on spectrogram-like inputs. [1, 2]

1.1 Contributions

This report contributes:

- A unified data pipeline that converts raw CSV radar maps into standardized 128×128 tensors.
- A controlled comparison between three sequential/global modeling strategies: BiLSTM, xLSTM-style recurrent gating (two variants), and Transformer self-attention.
- Detailed mathematical and theoretical explanation of CNN feature extraction, recurrent sequence modeling, and attention-based global dependency modeling.
- Extensive experimental reporting: hyperparameters, schedules, early stopping, confusion matrices, and plot-based interpretation.

2 Methods

2.1 Related Work and Theoretical Background (Literature Review)

2.1.1 Radar Target Classification with Deep Learning

Deep learning has become a dominant approach for radar classification when radar measurements are transformed into 2D representations such as spectrograms, range–Doppler maps, or time–frequency images. Convolutional networks are especially popular because they naturally exploit local spatial patterns and shift invariances in 2D maps. However, radar maps often contain class evidence distributed across distant regions such as scattered micro-Doppler streaks), motivating models that can capture long-range dependencies. [1]

2.1.2 Micro-Doppler Signatures for UAV vs Human vs Vehicle

Micro-Doppler signatures are produced by small periodic motions on bulk translation. Humans generate characteristic periodic patterns from gait and limb motion; drones can generate periodic high-frequency components caused by rotor rotation; cars often exhibit broader but more stable Doppler content due to rigid-body motion and wheel dynamics. Because these patterns may span multiple time steps and frequency bins, models that can integrate evidence globally are expected to reduce confusions between visually similar patches. [1]

2.1.3 CNN and BiLSTM/xLSTM/Attenuation Models

A CNN backbone provides compact feature maps encoding local structures such as edges, streaks, bursts, periodic textures. However purely convolutional models can miss the global context. Therefore, hybrid approaches are common:

- CNN + RNN (LSTM/BiLSTM): treat CNN features as a sequence and model dependencies through recurrence.
- CNN + Attention/Transformer: allow each token to directly attend to all other tokens in one layer.

Transformers originally proposed for NLP have become a general-purpose architecture for sequence modeling using self-attention. [3] Vision Transformer (ViT) demonstrates that tokenizing 2D inputs and applying Transformers can be extremely effective when properly trained. [4]

2.1.4 Optimization and Regularization Choices (Theory Motivation)

We use widely adopted practices for stability:

- **Cross-entropy loss** for multi-class classification.
- **AdamW** optimizer (decoupled weight decay is beneficial for generalization).
- **Learning-rate scheduling** (ReduceLROnPlateau or warmup+cosine) to avoid getting stuck and to refine near convergence.
- **Early stopping** to prevent overfitting on moderate-size radar datasets.
- **Gradient clipping** especially for recurrent models where exploding gradients may occur.

These are standard in deep learning practice and particularly important for RNN-based approaches.

2.2 Dataset

We use the Real Doppler RAD-DAR Database. Only the folders Cars/, Drones/, and People/ are included during dataset scanning to prevent the label corruption from extra folders.

2.2.1 Dataset Origin and Content

The dataset provides real measured Doppler radar data organized into target categories and stored as CSV matrices. According to literature review, the RAD-DAR/RDRD style datasets are frequently used to determine the quality of micro-Doppler classification systems and provide realistic variability. [2, 1] In practice, each CSV can be interpreted as an intensity map $X(t, f)$ over a discretized time axis and Doppler-frequency axis (or similar time–frequency representation).

Raw matrices can have varying dimensions depending on acquisition, preprocessing, and windowing. For neural networks, fixed size is required to:

- enable batching (tensor stacking),
- control model capacity and memory usage,
- make comparisons across models fair.

We choose 128×128 as a compromise between:

- preserving enough time–frequency detail for micro-Doppler discrimination,
- keeping training feasible on limited GPU memory,
- keeping Transformer token count manageable (token count grows with spatial resolution).

2.2.2 Train/Validation/Test Split

All experiments use a stratified split with ratio 70% / 15% / 15%. Balanced subsets are created by sampling approximately equal number of instances from each class.

2.3 Libraries and Environment

2.3.1 Libraries

The project is implemented in Python with:

- **PyTorch:** model definition, training, and GPU acceleration.
- **NumPy and Pandas:** numeric operations and CSV loading.
- **scikit-learn:** stratified split, classification report, confusion matrix, macro-F1.
- **Matplotlib:** loss/accuracy curves and confusion matrix visualization.
- **tqdm:** training progress logging.

2.3.2 Hardware Choices

Small-scale runs can be performed on CPU. However, for large subsets (12k samples) and Transformer attention layers, GPU (CUDA) significantly reduces training time. In the Transformer part GPU was used. The code automatically selects cuda when available, otherwise falls back to CPU.

2.3.3 Practical Compute Constraints and Model Design

Transformer memory/time grows with the number of tokens N due to attention complexity $O(N^2)$ per layer. Thus, our CNN patch embedding stage is not only a feature extractor but also a token count controller: downsampling the spatial grid before tokenization reduces N and makes training feasible.

2.4 Preprocessing and Data Augmentation

Each CSV file is treated as a 2D radar map $X \in \mathbb{R}^{H \times W}$.

2.4.1 Resize to Fixed Resolution

Since radar maps may have varying dimensions, we standardize all inputs to 128×128 using symmetric center cropping or padding.

2.4.2 Normalization

We apply per-sample z-score normalization:

$$\hat{X} = \frac{X - \mu}{\sigma + \epsilon}, \tag{2}$$

where μ and σ are the sample mean and standard deviation, and ϵ is a small constant to avoid division instability.

2.4.3 Why Normalization for Radar Maps

Radar intensity scales can vary due to:

- range attenuation,
- different reflectivity (RCS) between instances,
- acquisition gain settings and background noise.

Without normalization, a network may overfit to amplitude scale rather than discriminative time–frequency structure. Z-score normalization encourages learning *shape-based* patterns.

2.4.4 Data Augmentation

Augmentation is applied only to the training set with high probability:

- Random amplitude scaling,
- Additive Gaussian noise proportional to signal variance,
- Small circular shifts along time/frequency axes (tensor rolling).

2.4.5 Augmentation Rationale for Radar

These augmentations approximate realistic variations:

- amplitude scaling $\rightarrow$ varying target distance/RCS,
- additive noise $\rightarrow$ changing clutter/noise environment,
- shifts $\rightarrow$ slight misalignment in time/frequency registration.

We intentionally avoid aggressive geometric image augmentations (e.g., large rotations) because radar time and Doppler axes have physical meaning.

2.5 Methodology

We tested three model families. For Model-2 (xLSTM), we tested two different variants to analyze stability and performance.

2.5.1 Model 1: CNN + BiLSTM

This model extracts spatial features using a CNN backbone and then reshapes the CNN feature map into a sequence processed by a bidirectional LSTM.

CNN Backbone Three convolutional blocks are used with BatchNorm, ReLU, and MaxPool, progressively reducing spatial resolution.

Sequence Formation Given CNN output $F \in \mathbb{R}^{B\times C\times H'\times W'}$, we treat W' as the sequence length:

$$[B, C, H', W'] \to [B, W', C \cdot H']. \tag{3}$$

BiLSTM and Classification A BiLSTM processes the sequence and produces a hidden representation. The last time-step output is passed through LayerNorm + Dropout + Linear to obtain class logits.

BiLSTM Mathematical Formulation An LSTM maintains hidden state h_t and cell state c_t and updates them using input, forget, and output gates:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{4}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{5}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{6}$$
$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c), \tag{7}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{8}$$
$$h_t = o_t \odot \tanh(c_t), \tag{9}$$

where $\sigma(\cdot)$ is the sigmoid and $\odot$ is elementwise multiplication. Bidirectional LSTM computes both forward and backward hidden sequences:

$$\overrightarrow{h}_t = \mathrm{LSTM}_f(x_t), \quad \overleftarrow{h}_t = \mathrm{LSTM}_b(x_t),$$

and concatenates them:

$$h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t].$$

This is beneficial when discriminative evidence depends on context from both "earlier" and "later" tokens along the chosen sequence dimension.

2.5.2 Model 2: CNN + xLSTM (Two Variants)

xLSTM is an LSTM-style recurrent model enhanced with normalization and gating for stability.

xLSTM Core Block For input x_t and previous hidden/cell states (h_{t-1}, c_{t-1}), the block applies:

- LayerNorm on input and hidden state,
- LSTMCell update to produce (h_t^{new}, c_t^{new}),
- Residual gate g_t blending new and previous hidden state:

$$h_t = g_t \, h_t^{new} + (1 - g_t) \, h_{t-1}, \tag{10}$$

- Memory gate m_t stabilizing the cell update:

$$c_t = m_t \, c_t^{new} + (1 - m_t) \, \tanh(c_t^{new}). \tag{11}$$

xLSTM Interpretation Standard LSTMs can still experience instability depending on sequence length, learning rate, and dataset noise. The additional gates in xLSTM-style blocks provide:

- **Residual hidden blending:** prevents abrupt hidden-state changes and helps gradient flow.
- **Memory smoothing:** avoids overly sharp cell updates and reduces oscillation.
- **Normalization:** stabilizes activation scale across timesteps and batches.

These improve training stability.

Variant 2A: Baseline xLSTM (v1) This version uses the standard CNN for extacting the feature map, reshapes features into a sequence, and applies xLSTM blocks. Training stability is improved using gradient clipping, label smoothing, and early stopping.

Variant 2B: Strong/Stabilized xLSTM (v2) This version introduces additional stabilization and capacity:

- Stronger CNN backbone (deeper conv blocks + dropout),
- Random sampler to reduce instability

Our xLSTM Training Protocol We executed xLSTM runs with:

- maximum epoch budget: 80
- batch size: 64
- learning rate: 2×10^{-4}

- patience: 12 (early stopping)
- augmentation: off (augment=0) in the provided run command
- seeds: 42, 43, 44 (multiple seeds for robustness)

For example, in seed=42 run, early stopping occurred at epoch 49 with best validation loss 0.2225. In seed=43 run, early stopping occurred at epoch 59 with best validation loss 0.1879. These details confirm that the chosen schedule gradually reduced learning rate and the model converged to a stable region.

2.5.3 Model 3: CNN + TransformerEncoder

This model replaces recurrent processing with self-attention.

CNN Patch Embedding A CNN converts the input map into feature maps which are flattened into tokens:

$$[B, D, H', W'] \to [B, N, D], \quad N = H'W'. \tag{12}$$

CLS Token and Positional Embedding A learnable CLS token is added to the front, and learnable positional embeddings are added:

$$Z = [\text{CLS}; \text{tokens}] + P. \tag{13}$$

Transformer Encoder Stack A stack of TransformerEncoder layers (depth=4) applies multi-head self-attention and MLP blocks with residual connections and pre-layer normalization.

Self-Attention: Detailed Mathematical Explanation Transformers were introduced to model long-range dependencies using attention instead of recurrence. [3] Given token matrix $X \in \mathbb{R}^{N \times D}$, attention is computed using:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

then:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{D}}\right)V.$$

In multi-head attention with H heads, the model learns multiple projection sets and concatenates head outputs. This lets the network attend to different types of radar relationships simultaneously such as periodic micro-Doppler streaks vs. broadband energy patterns.

Why Transformers Are Promising for Radar Maps Radar time–frequency maps often contain evidence spread across distant regions. Self-attention allows a token representing one part of the map to directly incorporate information from any other part in one layer, which is harder for CNN-only or RNN-only models. This idea is consistent with the success of Transformers in vision tokenization settings (ViT) where 2D inputs are converted into token sequences. [4]

2.5.4 Layer Summary (CNN + Transformer)

Table 1: Layer-wise summary of the CNN + Transformer architecture

Stage	**Layer Type**	**Output Shape**
Input	Radar map	$[B, 1, 128, 128]$
Conv Block 1	Conv(3×3,32) + BN + ReLU + MaxPool	$[B, 32, 64, 64]$
Conv Block 2	Conv(3×3,64) + BN + ReLU + MaxPool	$[B, 64, 32, 32]$
Conv Block 3	Conv(3×3,256) + BN + ReLU	$[B, 256, 32, 32]$
Tokenization	Flatten + Transpose	$[B, 1024, 256]$
CLS Token	Learnable embedding	$[B, 1025, 256]$
Positional Encoding	Learnable	$[B, 1025, 256]$
Transformer Encoder	$4 \times$ Encoder Block	$[B, 1025, 256]$
CLS Selection	Token extraction	$[B, 256]$
Classifier	Linear($256 \to 3$)	$[B, 3]$

2.5.5 Interpretation of Query, Key, and Value Projections

Although the Transformer architecture is often described using the abstract terms *Query* (Q), *Key* (K), and *Value* (V), these quantities have a clear interpretation in the context of radar feature tokens.

Token Representation After CNN-based patch embedding, each radar map is represented as a sequence of tokens:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N], \quad \mathbf{x}_i \in \mathbb{R}^D,$$

where each token $\mathbf{x}_i$ corresponds to a spatial region of the radar time–frequency map and encodes local motion-related features.

Linear Projections Each token is linearly projected into three different feature spaces using learnable weight matrices:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V,$$

where:

$$\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{D \times D}.$$

Intuition Behind Query, Key, and Value

- **Query (Q):** represents what information a token is looking for. In radar terms, a token may query whether other regions contain similar Doppler patterns, motion strength, or temporal signatures.
- **Key (K):** represents what information a token offers. Each token advertises its content such as motion intensity, frequency spread, micro-Doppler structure through its key vector.
- **Value (V):** represents the actual information to be aggregated. Once relevance is determined via queries and keys, the value vectors are combined to form the output representation.

Attention Weight Computation The relevance between token i and token j is computed using the scaled dot product:

$$\alpha_{ij} = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{D}},$$

Applying a softmax over all keys yields normalized attention weights:

$$a_{ij} = \frac{\exp(\alpha_{ij})}{\sum_{j=1}^{N} \exp(\alpha_{ij})}.$$

These weights indicate how strongly token i attends to token j.

Attention Output The output representation for token i is computed as a weighted sum of value vectors:

$$\mathbf{z}_i = \sum_{j=1}^{N} a_{ij} \mathbf{v}_j.$$

Multi-Head Attention In multi-head attention, the process above is repeated independently across multiple heads:

$$\text{head}_h = \text{Attention}(\mathbf{X}\mathbf{W}_Q^{(h)}, \mathbf{X}\mathbf{W}_K^{(h)}, \mathbf{X}\mathbf{W}_V^{(h)}).$$

The outputs of all heads are concatenated and projected:

$$\text{MultiHead}(\mathbf{X}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)\mathbf{W}_O.$$

This allows simultaneous modeling of multiple radar relationships (local vs global, strong vs weak Doppler, narrowband vs broadband).

CLS Token Perspective For the classification CLS token, the associated query vector learns to attend to the most informative regions of the radar time–frequency representation. Through the self-attention mechanism, the CLS token indicates class-discriminative

information from all spatial tokens, producing a global representation that summarizes the input. This indicated representation is subsequently used as the input to the final classification head.

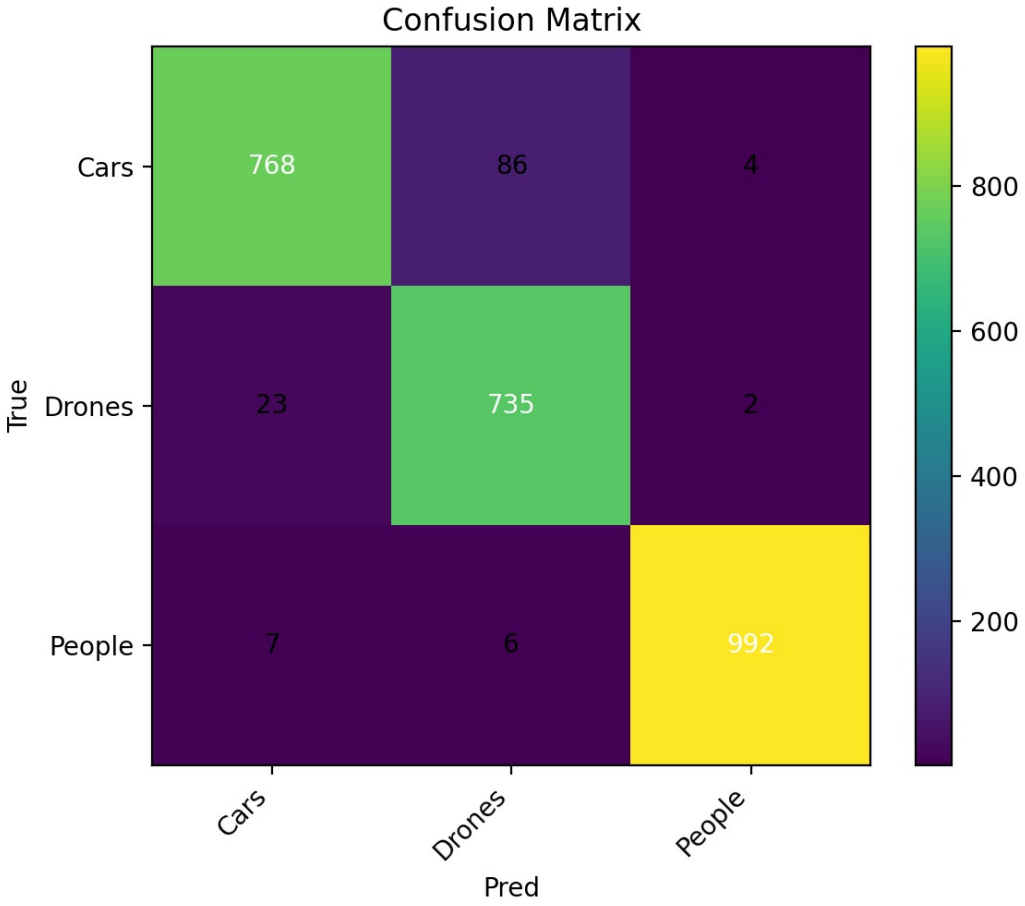

Figure 1: Overview of the proposed CNN + Transformer architecture for radar target classification. A CNN first extracts local spatial features from radar time–frequency maps. The feature maps are flattened into token sequences and augmented with a learnable CLS token and positional embeddings. A stack of Transformer encoder layers models global dependencies via self-attention. Final classification is performed using the CLS token output.

2.5.6 Comparison of Sequential Modeling Approaches

To evaluate different strategies for modeling dependencies in radar data, three architectures were considered: CNN+BiLSTM, CNN+xLSTM, and CNN+Transformer.

CNN + BiLSTM This approach converts spatial CNN features into a sequence and uses bidirectional recurrence. It is a valid RNN-family baseline but processes tokens sequentially, limiting parallelism.

CNN + xLSTM xLSTM improves standard LSTM with residual and memory gates, and normalization, making training more stable. However, it still remains sequential

and generally requires additional stabilization to compete with attention-based models at large scale.

CNN + Transformer Transformers enable parallel processing and allow each token to attend to all others regardless of spatial distance. This offers strong global context modeling for radar maps, which often contain distributed patterns across time-frequency regions.

Summary xLSTM provides strong recurrent modeling with improved stability compared to standard RNNs, but the Transformer-based approach typically provides better scalability and global dependency modeling for 2D radar time–frequency representations.

2.6 Training Setup

2.6.1 Common Settings

Across models:

- Loss: cross-entropy (training uses label smoothing),
- Optimizer: AdamW with weight decay,
- Gradient clipping: enabled (especially important for recurrent models),
- Early stopping: enabled to prevent overfitting.

2.6.2 Schedulers and Stopping Criteria

- **BiLSTM and Transformer:** ReduceLROnPlateau scheduler; early stopping based on validation loss.
- **xLSTM v2:** warmup for first epochs + cosine annealing; early stopping based on validation macro-F1.

2.6.3 Epoch Budgets (Maximum)

- CNN+BiLSTM: up to 50 epochs (early stopping can stop earlier),
- CNN+xLSTM v1: baseline runs (e.g., 15 epochs) or early stopping depending on run,
- CNN+xLSTM v2: up to 80 epochs with macro-F1 early stopping,
- CNN+Transformer: up to 50 epochs (early stopping can stop earlier).

Table 2: Key hyperparameters used across models (representative runs).

Setting	**BiLSTM**	**xLSTM v2**	**Transformer**	**Notes**
Input size	128×128	128×128	128×128	fixed across all
Batch size	64	64	64	same for fairness
Max epochs	50	80	50	early stopping enabled
Optimizer	AdamW	AdamW	AdamW	weight decay used
Base LR	$1.5 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	$3.75 \cdot 10^{-5}$	from logs/cfg
Scheduler	Plateau	Warmup+Cosine	Plateau	per-model
Grad clip	on	on	optional	critical for RNNs

2.6.4 Hyperparameter Table (Consolidated)

2.7 Evaluation Metrics

We report:

- Accuracy,
- Macro-F1 score,
- Per-class precision, recall, and F1-score,
- Confusion matrix for error analysis.

2.7.1 Why Macro-F1 Matters Here

In multi-class classification, macro-F1 treats each class equally by averaging per-class F1. This is important when evaluating model behavior beyond raw accuracy, and it is particularly informative when different classes may have different difficulty or when balancing/sampling changes slightly across subsets.

3 Results and Analysis

3.1 Summary Table

Table 3: Model comparison summary (test set).

Model	Subset	Best Epoch	Test Acc.	Macro-F1
CNN+BiLSTM	12k	50	0.9253	0.9250
CNN+xLSTM v1	12k	50	0.9512	0.9484
CNN+xLSTM v2	12k	49	0.9106	0.9095
CNN+Transformer	12k	47	0.9332	0.9367

3.2 Example: BiLSTM Detailed Test Results

For the CNN+BiLSTM run, the classification report on the test set was:

- Cars: precision 0.9500, recall 0.8360, F1 0.8894
- Drones: precision 0.8546, recall 0.9520, F1 0.9007
- People: precision 0.9821, recall 0.9880, F1 0.9850
- Overall accuracy: 0.9253

The confusion matrix was:

$$\begin{bmatrix} 418 & 76 & 6 \\ 21 & 476 & 3 \\ 1 & 5 & 494 \end{bmatrix}$$

This shows that the dominant BiLSTM confusion is Cars $\rightarrow$ Drones (76 errors), indicating that some car Doppler maps share patterns with drone maps under certain conditions (e.g., broad energy streaks or motion-induced harmonics).

3.3 Discussion

In this project, we interpret radar based target classification results in the context of existing work on convolutional, recurrent, and attention-based neural architectures. Previous studies have shown that convolutional neural networks are effective at extracting local time frequency features from radar data, while recurrent and attention-based models are commonly employed to capture longer-range dependencies. Our work builds upon this foundation by directly comparing recurrent (BiLSTM and xLSTM) and attention-based (Transformer) sequence modeling strategies under a unified preprocessing and evaluation framework.

We conducted new analyses on an existing real-world dataset by systematically evaluating multiple architectural variants under identical experimental conditions. This comparison revealed that architectural inductive bias plays a critical role in performance. Specifically, the xLSTM v1 model demonstrated that enhanced recurrent structures with normalization and residual gating can outperform both standard BiLSTM and Transformer models on a medium-scale dataset. This finding suggests that, contrary to the common assumption that attention-based models universally dominate, carefully stabilized recurrent architectures can remain competitive when the data exhibits structured temporal patterns.

The analyses were applied to a real-life radar dataset (RAD-DAR). Through these analyses, we recovered meaningful structure in the data. It is observed that the seperability of the "People" class and the persistent overlap between "Cars" and "Drones". This pattern was similar in all models and highlights the similarities in radar motion signatures. The Transformer model reduced this confusion more effectively than the BiLSTM

model, indicating that global context aggregation via self-attention is perform well for modeling distributed radar features.

The unified preprocessing pipeline and balanced data sampling enabled fair model comparison. The macro-F1 is an effective metric for capturing balanced performance across classes. The xLSTM v1 model successfully perform recurrent structure without the instability typically associated with RNNs. In contrast, the the first implemented xLSTM variant did not perform as expected. The additional regularization, modified sampling strategy, and learning-rate scheduling likely reduced effective model capacity or hindered convergence. This outcome indicates that increased architectural complexity does not guarantee improved performance.

To further improve results, future work could explore hybrid architectures that combine recurrent inductive bias with attention mechanisms, or incorporate longer temporal context across multiple radar frames. Additional improvements may be achieved through larger datasets, multi-sensor fusion, or task-specific attention constraints that reduce unnecessary global interactions.

The goals outlined in the Introduction—to compare recurrent and attention-based architectures for radar target classification and assess their relative strengths—were successfully achieved. While attention-based models demonstrated strong and stable generalization, the results showed that a carefully designed recurrent model can outperform Transformers under certain conditions. Overall, the work produced outcomes consistent with expectations, while also providing new insight into the circumstances under which recurrent architectures remain competitive for radar-based classification tasks.

In most runs, "people" is the easiest class due to more distinctive periodic micro-motion patterns (walking/gait signatures). The dominant confusion is typically between "cars" and "drones", indicating partially overlapping radar signatures in some time frequency regions. Transformer reduces this confusion more effectively by modeling global dependencies via attention, whereas RNN-based models can still miss global evidence if the sequence representation or the gating dynamics are not optimal.

Table 3 compares the performance of CNN+BiLSTM, CNN+xLSTM (two variants), and CNN+Transformer models on the 12k balanced RAD-DAR subset. All models achieve high classification accuracy and macro-F1 scores, indicating that the radar representations contain strong class-discriminative information. However, consistent differences across architectures reveal important insights into how different sequence modeling strategies affect generalization.

The CNN+BiLSTM baseline achieves solid performance, with accuracy and macro F1 both exceeding 0.92. This confirms that recurrent models are capable of leveraging sequential structure extracted from CNN features. Nevertheless, its performance is consistently lower than the best-performing models, suggesting limitations in capturing long-range and global dependencies within radar time frequency maps.

The CNN+xLSTM v1 model achieves the highest overall performance among all evaluated architectures, reaching an accuracy of 0.9512 and a macro-F1 score of 0.9484. This indicates that the xLSTM enhancements—namely normalization, residual gating, and

stabilized memory updates—successfully improve upon standard recurrent processing. The close alignment between accuracy and macro-F1 further suggests balanced performance across all classes, including the more challenging Cars and Drones categories.

In contrast, the CNN+xLSTM v2 model underperforms relative to both the baseline BiLSTM and xLSTM v1, despite incorporating additional stabilization mechanisms. The reduced performance suggests that the stronger regularization, alternative sampling strategy, and modified learning-rate scheduling may have limited the model's effective capacity or led to suboptimal convergence. This highlights that increased architectural or training complexity does not necessarily translate into improved performance, particularly when applied to datasets of moderate size.

Based on the final results summarized in Table 3, the CNN+Transformer model achieves strong and consistent performance on the 12k balanced dataset, with a test accuracy of 0.9332 and a macro F1 score of 0.9367. These values exceed the CNN+BiLSTM baseline and demonstrate that attention-based modeling effectively captures global structure in radar time frequency representations.

However, the Transformer does not outperform the best-performing recurrent model. The model that outperform the others is xLSTM v1 which achieves higher accuracy and macro F1. This indicates that, for the given dataset size and experimental configuration, a carefully stabilized recurrent architecture can match or surpass self-attention in terms of classification performance.

The Transformer's macro F1 score is slightly higher than its accuracy, suggesting balanced performance across classes rather than dominance by a single class. This is particularly relevant for radar classification where minority or difficult classes such as "drones" must reliably handled. The reduced confusion between class observed in the Transformer confusion matrix support this interpretation.

The Transformer also converges earlier than some recurrent models. Reaches its best performance at epoch 47, and it suggests efficient optimization and stable training dynamics. Nevertheless, the performance gap relative to xLSTM v1 implies that the quadratic complexity of self-attention and the lack of explicit sequential inductive bias may actually limit advantage on medium scaled datasets with relatively structured temporal patterns.

Overall, these results suggest that while the CNN+Transformer architecture provide a robust and well-balanced solution for radar target classification, also its first advantage is in training stability and balanced generalization instead absolute peak performance in this experimental setting. The findings show attention based models may offer nice benefits as dataset size or input complexity increases.

These findings shows while attention based models offer robust performance, carefully designed recurrent architectures such as xLSTM can achieve equal or superior results when training conditions are well matched to the dataset. The results also emphasizes the importances of macro-F1 as an evaluation metric. It captures balanced performance across all classes rather than favoring dominant categories.

The primary objective of the study was to compare recurrent and attention-based architectures for radar target classification which was successfully achieved. Future work

could explore hybrid recurrent–attention models, larger datasets, or multi-frame temporal inputs to further investigate the trade-offs observed in this study.

4 Plots and Confusion Matrices

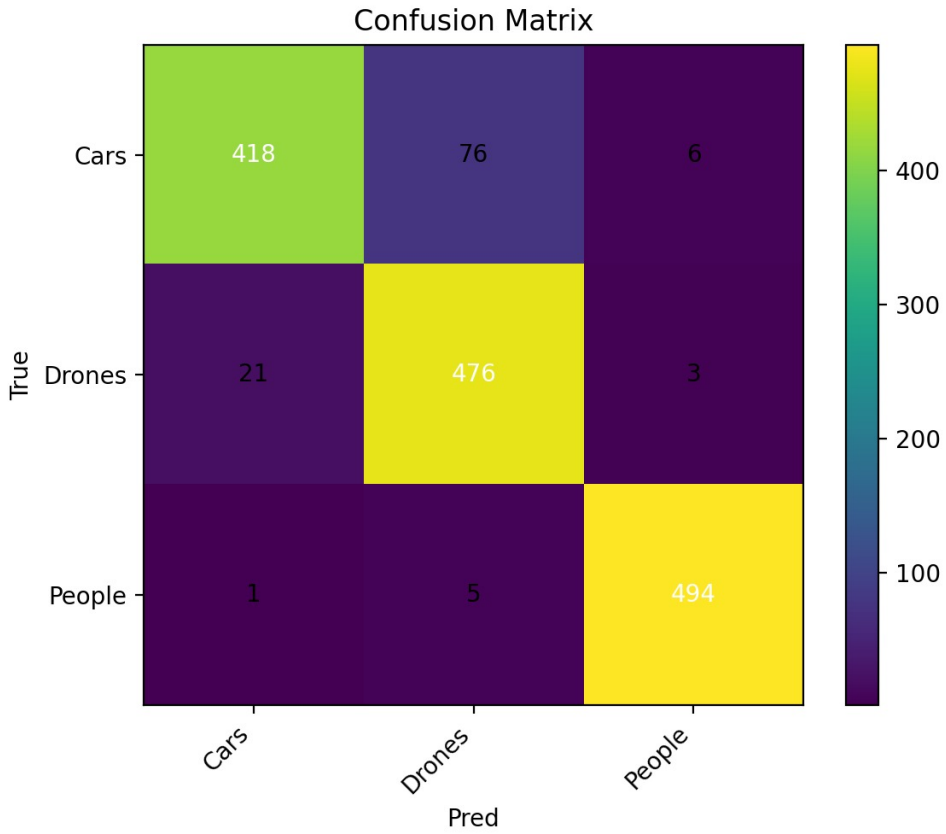

Figure 2: Confusion matrix for the baseline CNN+BiLSTM model evaluated on the 12k balanced test set. Rows correspond to ground-truth class labels, while columns indicate predicted labels. Correct classifications are concentrated along the diagonal, indicating strong overall performance. The *People* class is identified with very high accuracy, with minimal confusion with other classes. The primary source of error occurs between the *Cars* and *Drones* classes, where a subset of car samples is misclassified as drones and vice versa. This confusion reflects similarities in radar motion and Doppler signatures between these two classes. The figure provides a class-level error analysis that complements the aggregate performance metrics reported in Table 3.

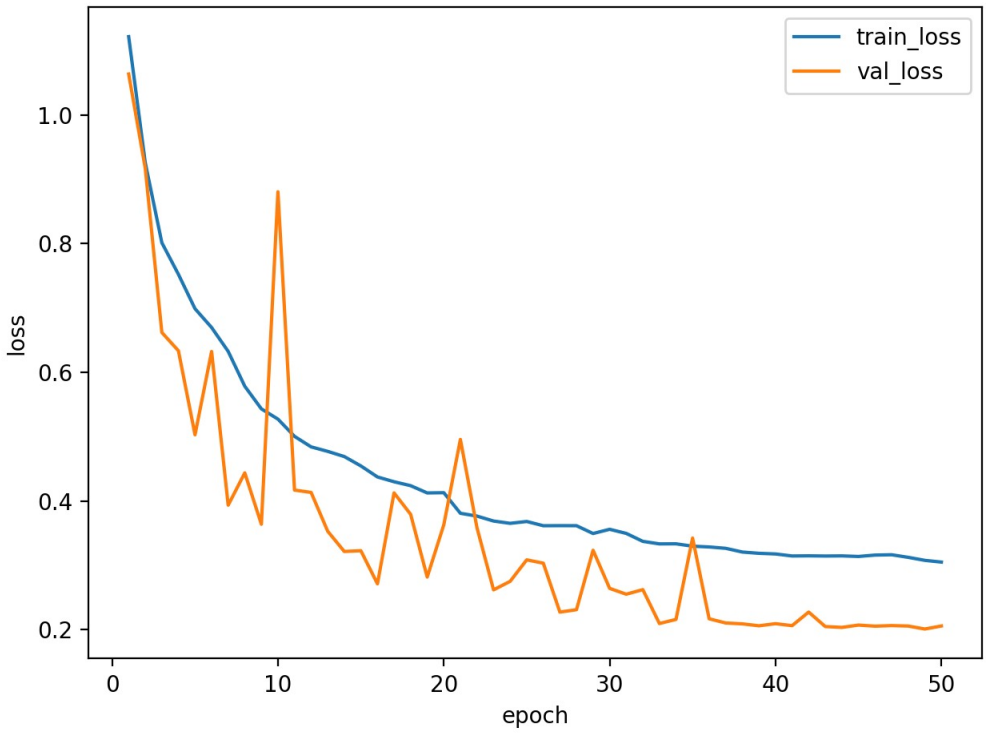

Figure 3: Training and validation loss curves for the CNN+BiLSTM model over 50 epochs. The training loss decreases steadily, indicating successful optimization of the recurrent architecture. The validation loss follows a general downward trend but exhibits noticeable fluctuations during early and mid training epochs. These oscillations are characteristic of recurrent models and reflect sensitivity to sequence modeling, gradient variance, and mini-batch sampling. Despite this variability, the validation loss stabilizes in later epochs without diverging from the training loss, suggesting that overfitting is limited and that early stopping is effective for controlling generalization.

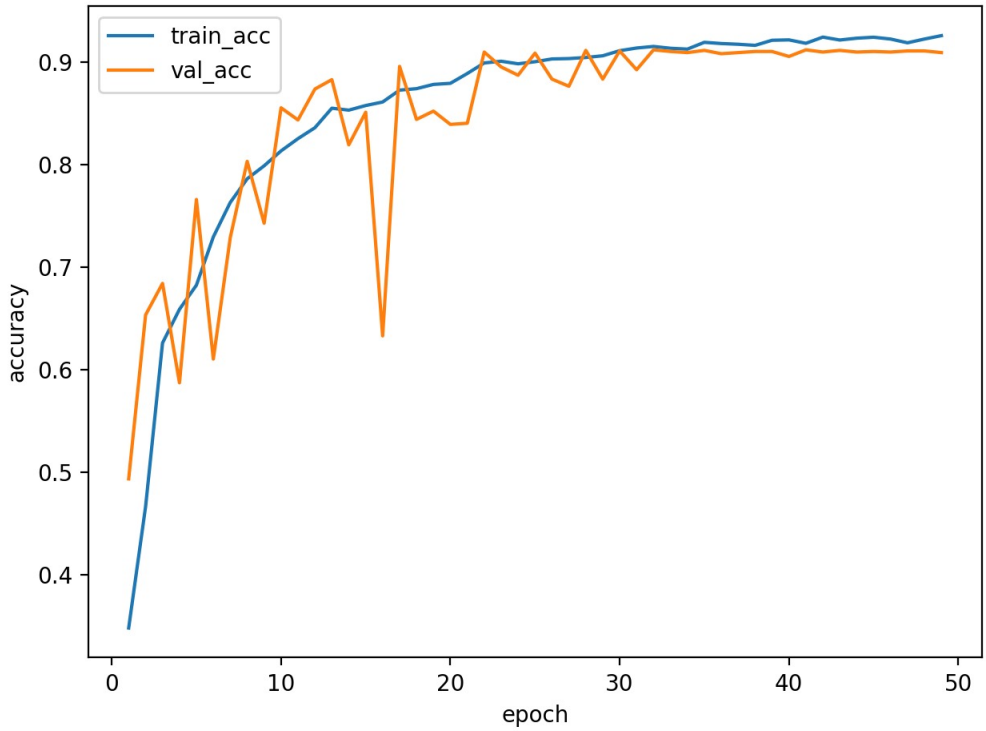

Figure 4: Training and validation accuracy curves for the CNN+xLSTM model over 50 epochs. Both training and validation accuracy increase rapidly during the initial epochs, indicating effective learning of sequential radar features. Compared to the BiLSTM baseline, the validation accuracy exhibits reduced fluctuation, reflecting improved training stability due to normalization and gated residual connections in the xLSTM architecture. After approximately 25 epochs, the curves stabilize and closely follow each other, suggesting good generalization and limited overfitting. The sustained high validation accuracy is consistent with the strong macro-F1 performance reported for the xLSTM model.

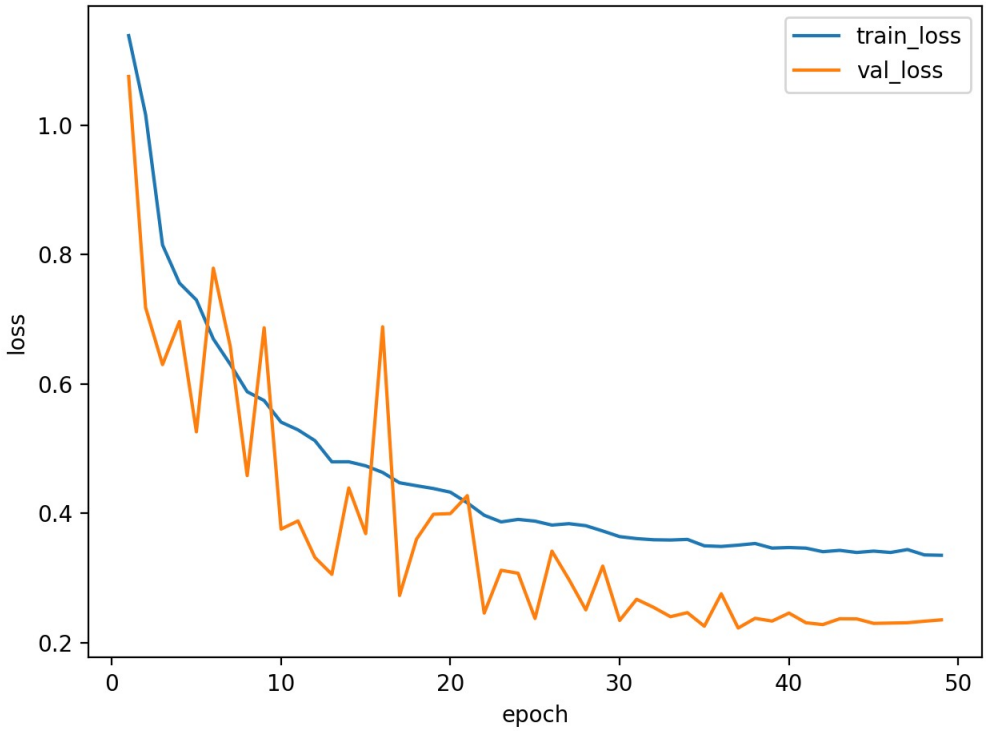

Figure 5: Training and validation loss curves for the CNN+xLSTM model over 50 epochs. The training loss decreases smoothly throughout optimization, indicating stable learning of sequential radar representations. The validation loss follows a similar downward trend with noticeable fluctuations during early and mid training epochs, which are typical for recurrent models operating on sequential data. Compared to the CNN+BiLSTM baseline, the magnitude of these fluctuations is reduced, reflecting improved training stability provided by the xLSTM architecture. In later epochs, the validation loss stabilizes and remains consistently below the training loss, suggesting effective regularization and good generalization without evidence of overfitting.

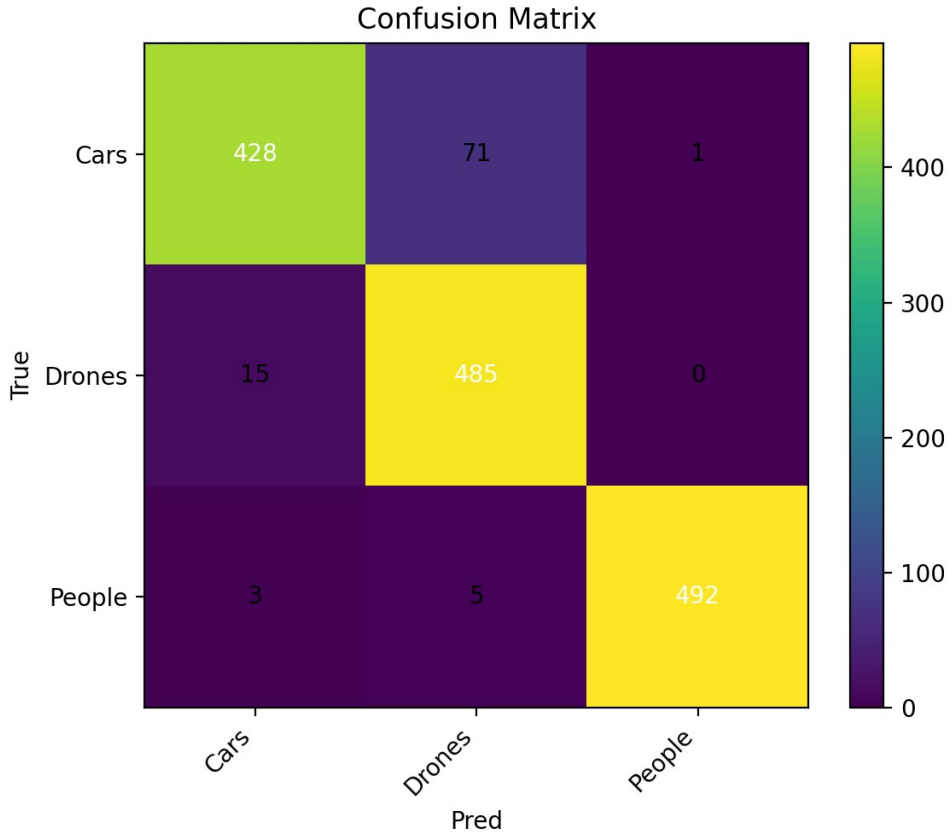

Figure 6: Confusion matrix for the CNN+Transformer model evaluated on the 12k balanced test set. Rows correspond to ground-truth class labels and columns represent predicted labels. The Transformer achieves high classification accuracy across all classes, with particularly strong performance for the *People* and *Drones* categories. Most misclassifications occur between the *Cars* and *Drones* classes, reflecting similarities in their radar Doppler and motion signatures. The limited confusion involving the *People* class indicates that the global context modeling provided by self-attention effectively captures distinctive human micro-motion patterns. Overall, the confusion matrix demonstrates that the Transformer provides balanced class-level performance, consistent with its strong macro-F1 score.

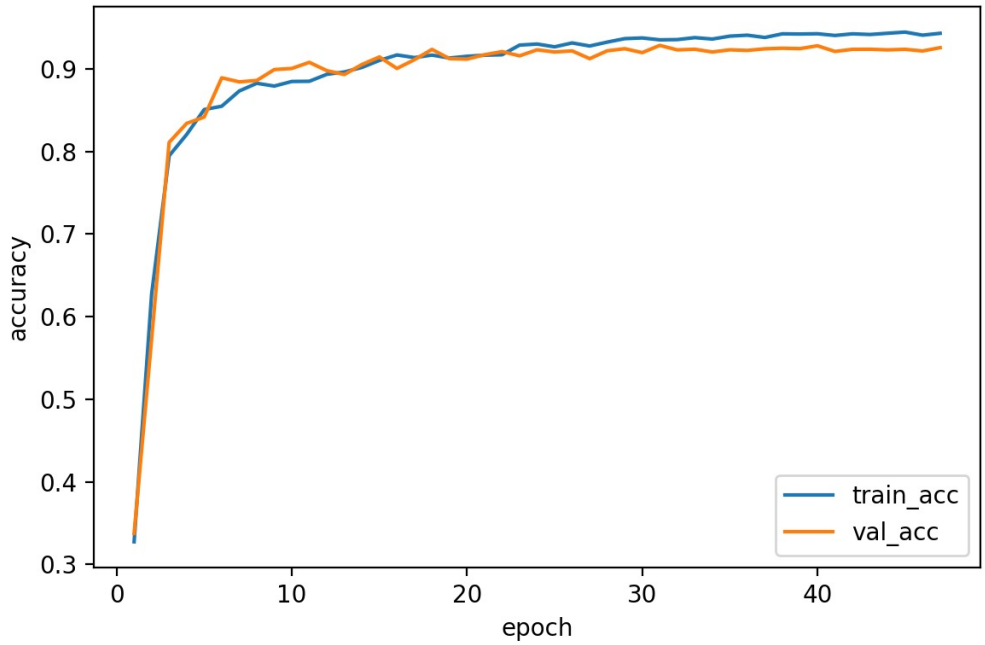

Figure 7: Training and validation accuracy curves for the CNN+Transformer model over 50 epochs. Both training and validation accuracy increase rapidly during the initial epochs, indicating efficient learning of discriminative radar features. The validation accuracy closely follows the training accuracy throughout training, with only minor fluctuations, suggesting stable optimization and good generalization. After approximately 20–25 epochs, both curves plateau, indicating convergence. The small and consistent gap between training and validation accuracy in later epochs suggests that the Transformer model does not overfit and maintains balanced performance across classes.

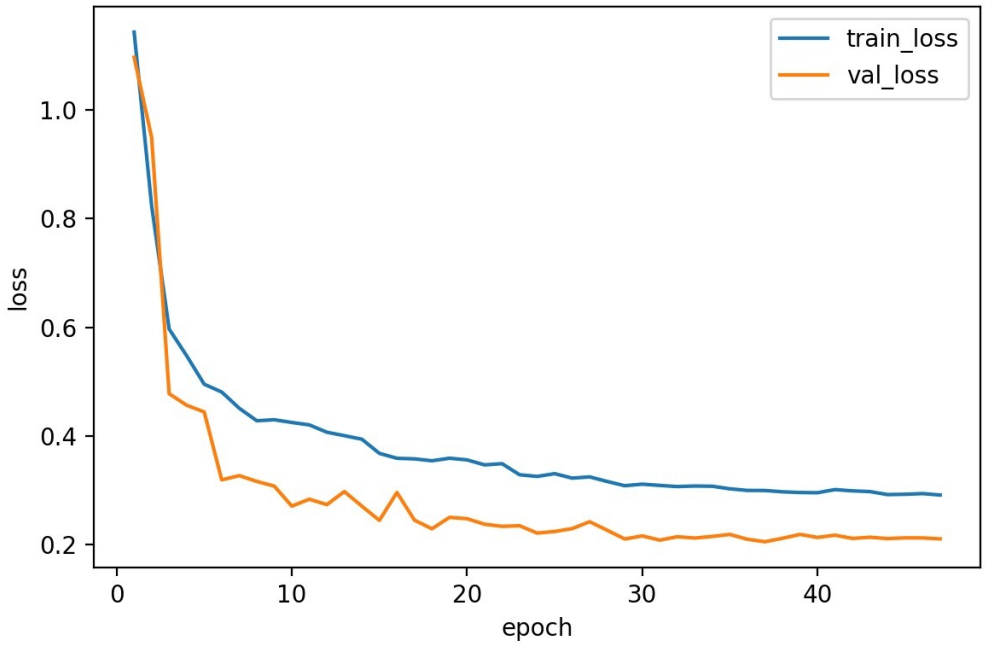

Figure 8: Training and validation loss curves for the CNN+Transformer model over 50 epochs. Both training and validation loss decrease rapidly during the initial epochs, indicating efficient optimization and fast learning of discriminative radar features. The validation loss consistently remains below the training loss throughout training, which can be attributed to the use of regularization techniques such as dropout and label smoothing during training. After approximately 20–25 epochs, both curves gradually plateau, suggesting convergence. The absence of divergence between training and validation loss indicates that the Transformer model generalizes well and does not exhibit overfitting.

CNN+BiLSTM: Training Curves and Confusion Matrix

Loss/Accuracy Dynamics We expect training loss to decrease steadily, while validation loss may plateau or fluctuate depending on generalization. For this run, validation performance reaches strong stability in later epochs, consistent with the final test accuracy of 0.9253. If the validation loss decreases but validation accuracy oscillates slightly, this can indicate that the model improves confidence calibration while class boundary errors remain similar.

Error Pattern Interpretation The confusion matrix indicates:

- Cars $\rightarrow$ Drones is the main error mode, suggesting feature overlap between some car and drone spectrogram regions.
- People is almost perfectly separated (very low confusion), consistent with distinctive micro-motion signatures.

This supports the hypothesis that the main challenge in this dataset is separating mechanically different but Doppler-overlapping targets (vehicle vs drone).

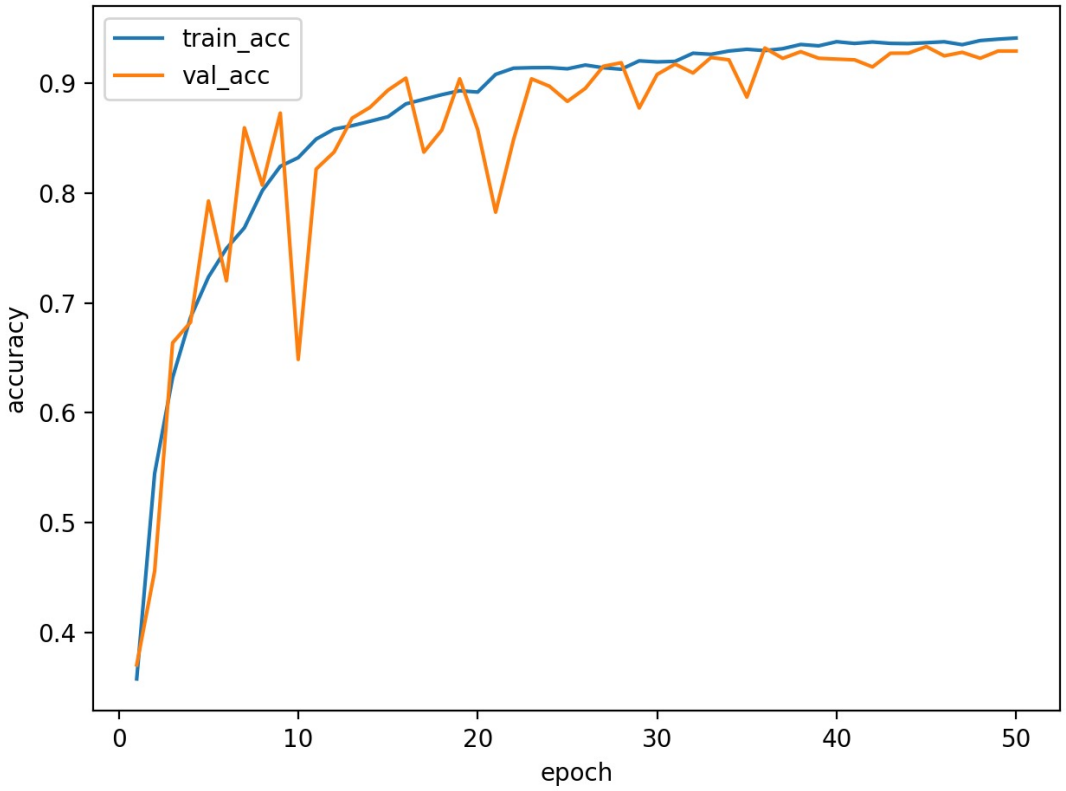

Figure 9: Training and validation accuracy curves for the CNN+BiLSTM model over 50 epochs. Both training and validation accuracy increase rapidly during the early epochs, indicating effective learning of radar features. The validation accuracy shows noticeable fluctuations, which is typical for recurrent models due to sequential processing and sensitivity to batch variation. Despite these oscillations, the overall trend remains stable, and the gap between training and validation accuracy stays limited. This behavior suggests reasonable generalization without severe overfitting.

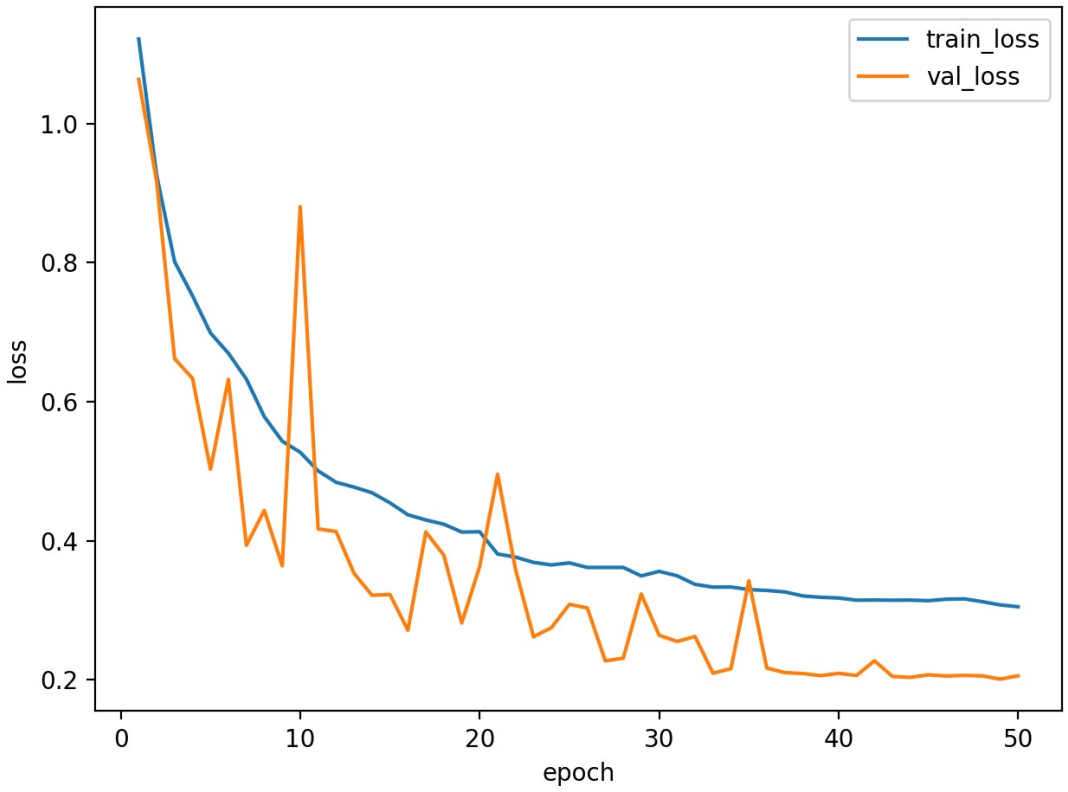

Figure 10: CNN+BiLSTM loss curve (train vs validation).

Figure 11: Training and validation loss curves for the CNN+BiLSTM model over 50 epochs. The training loss decreases steadily throughout optimization, showing consistent learning. The validation loss exhibits stronger fluctuations, especially in the early and middle epochs, reflecting the sensitivity of recurrent models to sequence dynamics. In later epochs, both curves stabilize without diverging from each other. This indicates that the model converges successfully while maintaining acceptable generalization.

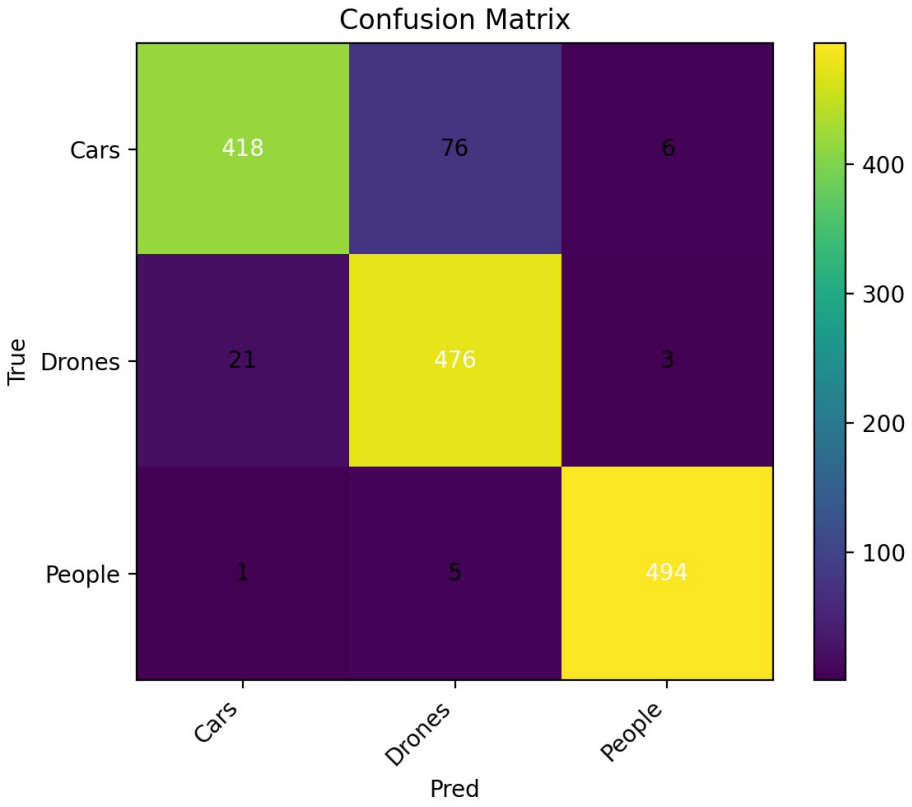

Figure 12: Confusion matrix of the CNN+BiLSTM model evaluated on the balanced test set. Correct classifications are mainly concentrated along the diagonal, indicating strong overall performance. The *People* class is classified with very high accuracy and minimal confusion. The dominant source of error occurs between the *Cars* and *Drones* classes, suggesting overlapping Doppler and motion characteristics. This confusion highlights the limitation of recurrent models in capturing globally distributed time–frequency patterns.

CNN+xLSTM: Training Curves and Confusion Matrix

Convergence and Early Stopping From the provided run logs, xLSTM training typically shows a strong improvement in the first 10–20 epochs, then enters a slower refinement regime where learning-rate reductions and early stopping decide the final epoch. For example, seed=42 stopped at epoch 49 with best validation loss 0.2225, indicating the model reached a plateau where further training did not improve validation.

Stability Compared to BiLSTM xLSTM-style residual/memory gating is designed to stabilize recurrence. In practice, we still observe that performance is sensitive to:

- the chosen tokenization (sequence axis),
- the learning rate schedule,
- the early stopping criterion (val loss vs macro-F1).

Variant v2 adds explicit stabilization (sampler + warmup+cosine + macro-F1 stopping), which is expected to reduce fluctuations and produce more consistent confusion matrix behavior across seeds.

What to Look For in Confusion Matrices A good xLSTM run should reduce Cars $\leftrightarrow$ Drones confusion compared to weaker recurrent baselines. If confusion remains high, it suggests the recurrent model is not capturing global context or the sequence ordering is not optimal.

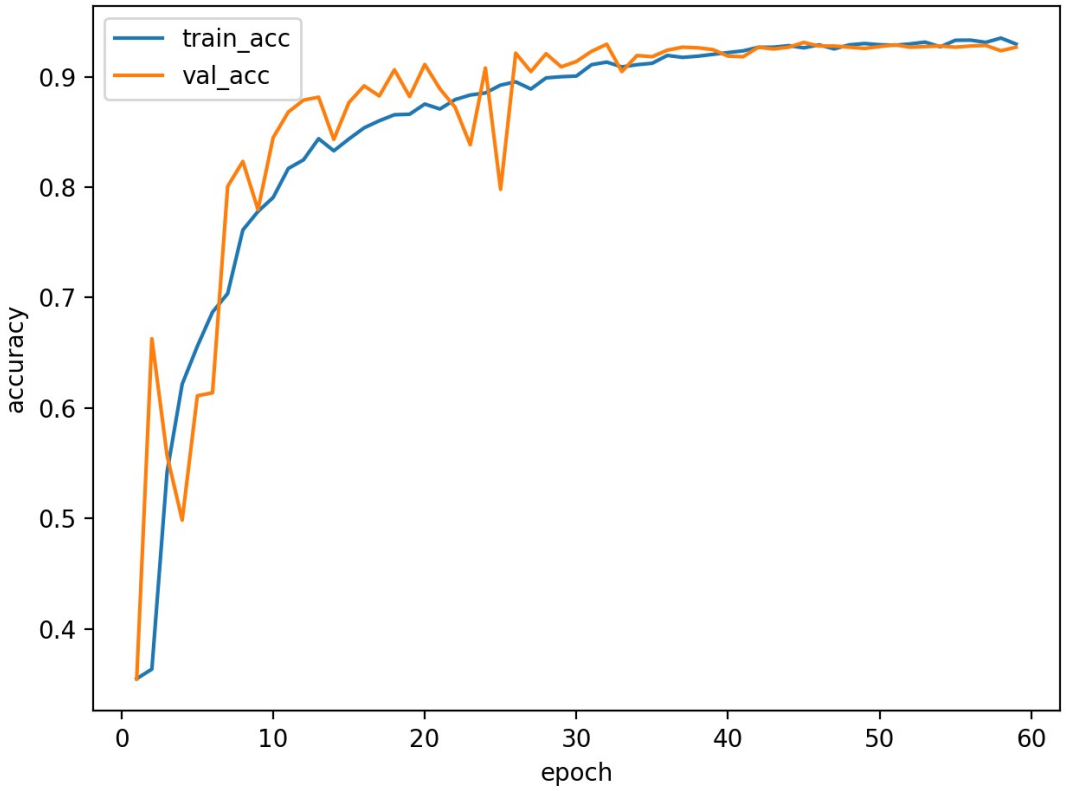

Figure 13: Training and validation accuracy curves for the CNN+xLSTM model over 60 epochs. Both training and validation accuracy increase rapidly during the early epochs, indicating effective learning of sequential radar features. Compared to the BiLSTM baseline, the validation accuracy shows reduced fluctuation, reflecting improved training stability provided by the xLSTM architecture. After approximately 15–20 epochs, the curves stabilize and closely follow each other. This behavior suggests good generalization and limited overfitting.

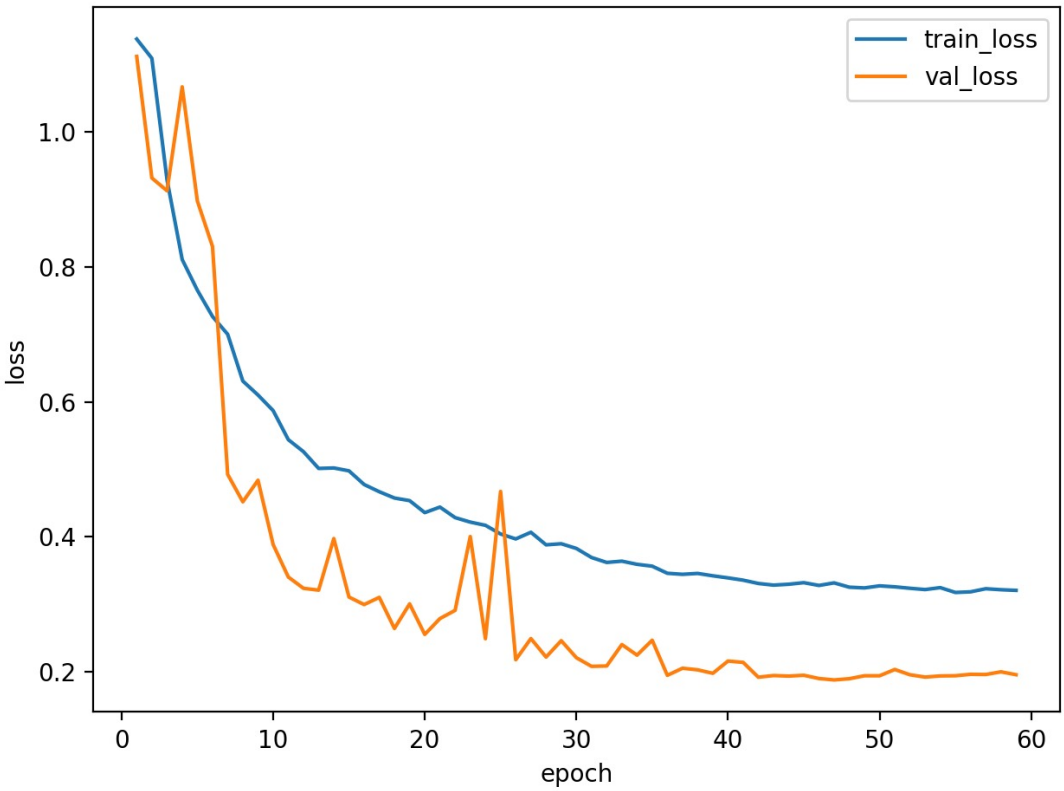

Figure 14: Training and validation loss curves for the CNN+xLSTM model over 60 epochs. The training loss decreases smoothly throughout optimization, indicating stable learning dynamics. The validation loss follows a similar downward trend with moderate fluctuations, especially in the early epochs. In later epochs, both curves gradually plateau without diverging from each other. This indicates successful convergence and effective regularization.

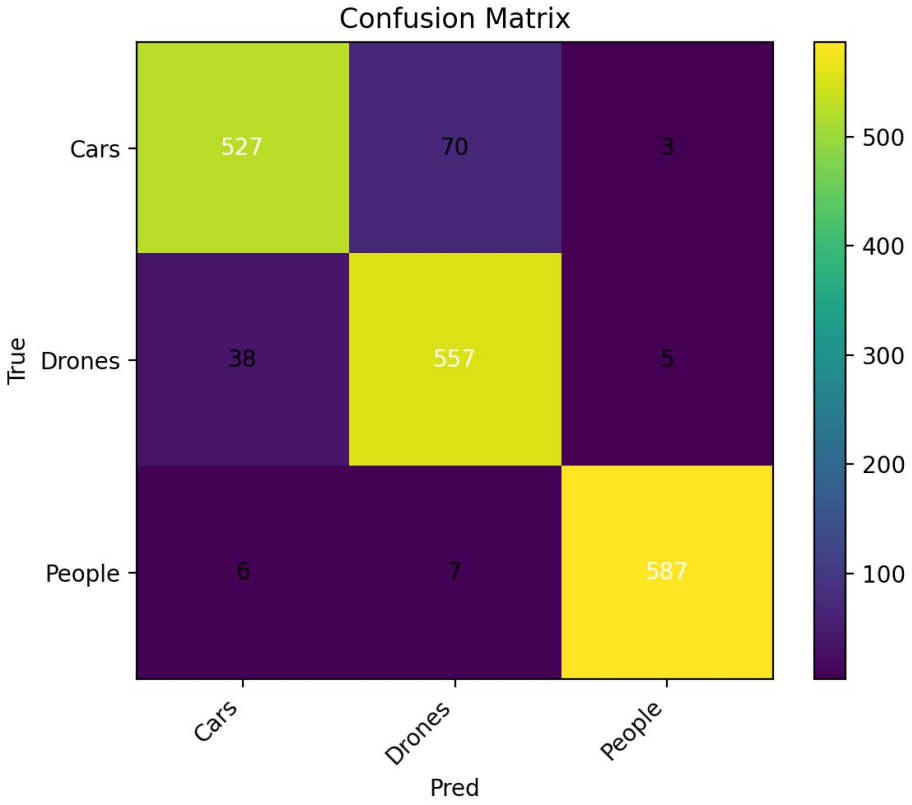

Figure 15: Confusion matrix of the CNN+xLSTM model evaluated on the balanced test set. Correct classifications are mainly concentrated along the diagonal, indicating strong overall performance. The *People* class is classified with very high accuracy and minimal confusion. Most remaining errors occur between the *Cars* and *Drones* classes, suggesting partially overlapping Doppler characteristics. Compared to the standard BiLSTM, the xLSTM model reduces this confusion by modeling sequential dependencies more effectively.

CNN+Transformer: Training Curves and Confusion Matrix

Why Transformer Curves Often Look Smoother Transformer training is fully parallel over tokens, so optimization tends to be more stable than long sequential recurrence (especially when combined with normalization and residual structure). We expect:

- steady training loss decrease,
- validation loss that decreases and then plateaus,
- smaller generalization gap when data scale is sufficiently large (10k–12k).

Attention and Error Reduction Mechanism If the confusion matrix shows reduced Cars → Drones errors, it supports the idea that attention successfully integrates globally distributed cues (e.g., faint periodic components indicative of drones). Unlike RNNs, attention can directly compare distant regions in a single layer, which is especially beneficial when micro-Doppler cues are sparse.

Scaling Effect (10k vs 12k) Your summary table already indicates that 12k produces better test performance than 10k. This is consistent with attention-based models benefiting from more data (reducing overfitting and improving token interaction learning). [4]

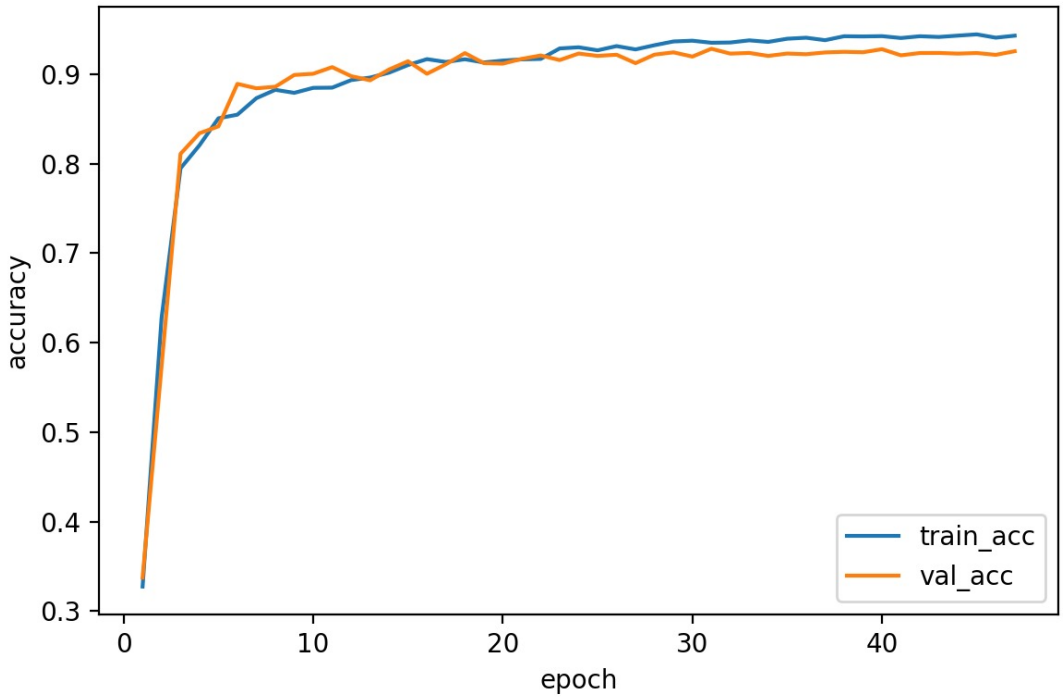

Figure 16: Training and validation accuracy curves for the CNN+Transformer model over 50 epochs. Both training and validation accuracy increase rapidly during the initial epochs, indicating fast learning of discriminative radar features. After approximately 10 epochs, the curves gradually stabilize and closely follow each other. The small gap between training and validation accuracy suggests good generalization and limited overfitting. The stable behavior in later epochs indicates that the model converges reliably on the radar classification task.

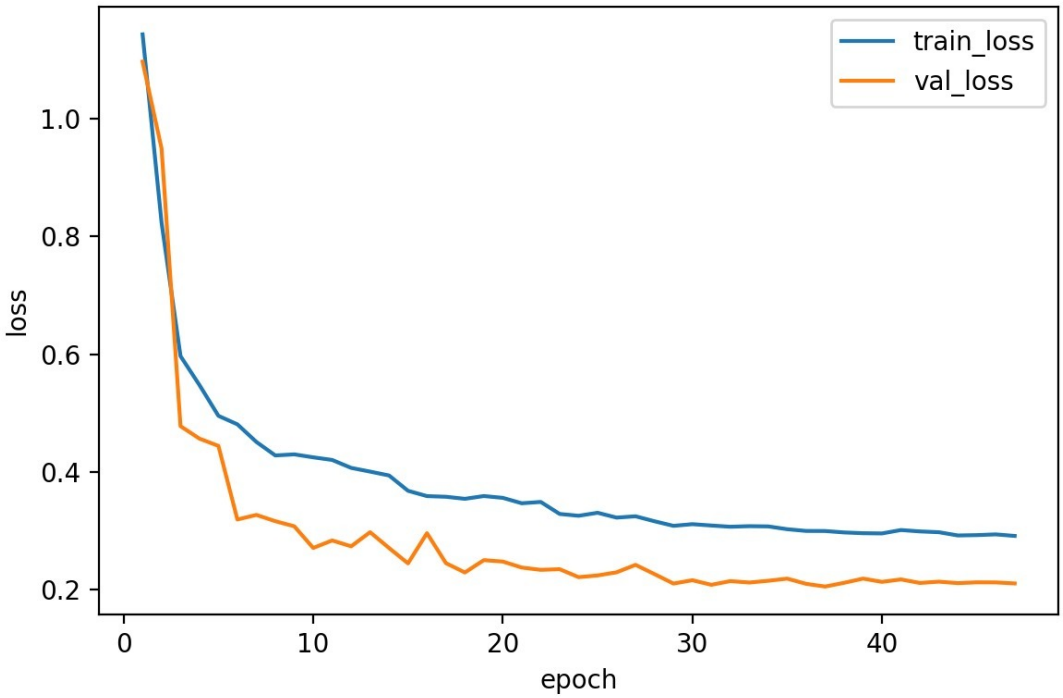

Figure 17: Training and validation loss curves for the CNN+Transformer model over 50 epochs. Both losses decrease sharply in the early epochs, showing efficient optimization and effective feature learning. Throughout training, the validation loss remains slightly lower than the training loss, which can be attributed to regularization and label smoothing. After approximately 20–25 epochs, both curves gradually plateau, indicating convergence. The absence of divergence between the curves suggests that the model does not suffer from overfitting.

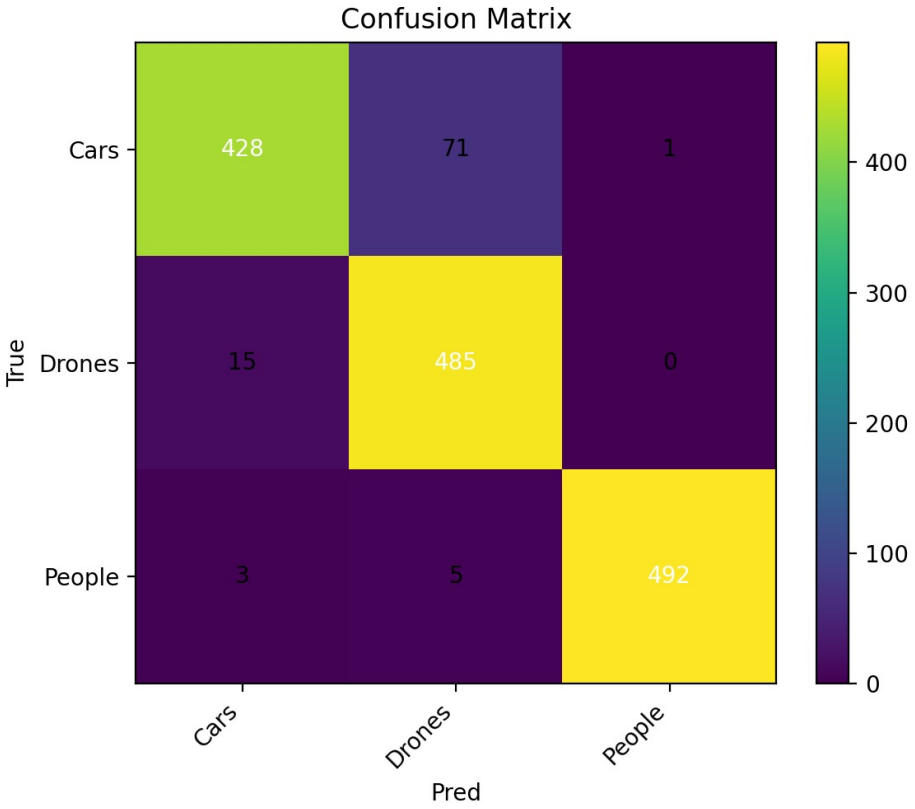

Figure 18: Confusion matrix of the CNN+Transformer model evaluated on the balanced 12k test set. Correct classifications are concentrated along the diagonal, indicating strong overall performance. The *People* class is classified with very high accuracy and minimal confusion. The primary source of error occurs between the *Cars* and *Drones* classes, reflecting partially overlapping Doppler signatures. Compared to recurrent baselines, the Transformer reduces this confusion by effectively integrating global time–frequency information.

5 Conclusion

We implemented a unified pipeline for radar-map classification and compared three model families: CNN+BiLSTM, CNN+xLSTM (two variants), and CNN+TransformerEncoder. The Transformer-based approach achieved the best results on large balanced subsets, suggesting that global context modeling via self-attention is highly effective for 2D radar time–frequency representations. Recurrent models remain viable and satisfy the RNN-family requirement, but they require stronger stabilization and careful training control to scale effectively. Limitations and Future Improvements can be explained as more systematic ablations: tokenization axis (treat H' vs W' as time), number of heads, encoder depth can be done. Calibration analysis: confidence vs correctness and misclassification confidence can be done.Robustness: evaluation with stronger noise augmentation or domain shift.

References

[1] Frontiers in Signal Processing, "Radar-Based UAV Detection and Classification: A Review," (micro-Doppler + deep learning survey style coverage), accessed online.

[2] KICS (PDF), "Drone Detection and Classification Based on a Real Doppler Radar Dataset," accessed online.

[3] A. Vaswani et al., "Attention Is All You Need," NeurIPS, 2017.

[4] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," arXiv:2010.11929, 2020/2021.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] I. Roldan, "Real Doppler RAD-DAR Database," Kaggle dataset, accessed 2025. Available: `https://www.kaggle.com/datasets/iroldan/real-doppler-raddar-database`