ADF & TransApp: A Transformer-Based Framework for Appliance Detection

Using Smart Meter Consumption Series

Adrien Petralia, Philippe Charpentier, Themis Palpanas

September 24, 2025

Overview

Mathematical Setup

Time Series Definition:

- Electrical consumption time series: $X = (x_1, x_2, \ldots, x_T)$
- Each element $x_j \in \mathbb{R}^1_+$ represents consumption at timestamp i_j
- Very low frequency: sampled every 15-60 minutes

Appliance Detection Problem:

Binary Classification Task

Given: Collection of consumption time series $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$ of variable lengths
Goal: Predict presence/absence of appliance a in time series X_i

Challenge: Long series (10k-20k points), variable lengths, low sampling frequency

Dummy Example

Input: Smart meter data from household over 30 days

- $X = (x_1, x_2, \ldots, x_{1440})$ where $T = 1440$ (30 days $\times$ 48 readings/day)
- x_j = electricity consumption in kWh at 30-minute intervals
- Example values: $x_1 = 0.5$, $x_2 = 0.8$, $x_3 = 1.2$, ...

Output: Binary label for appliance presence

- $y = 1$: Dishwasher present in household
- $y = 0$: No dishwasher in household

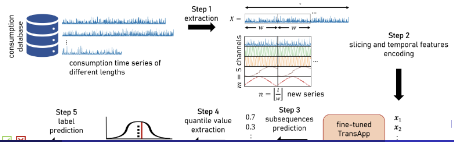
Key Challenge: At low sampling rates, individual appliance signatures are smoothed out and difficult to detect directly.

Framework Overview

Core Idea: Fragment long consumption series into manageable subsequences

Algorithm 1 ADF Framework

Input: Consumption series X of length l
Step 1: Extract series from database
Step 2: Slice into $n = \lfloor l/w \rfloor$ subsequences of length w
Step 3: Add temporal encoding features
Step 4: Apply TransApp classifier to each subsequence
Step 5: Merge predictions using quantile-based aggregation
Output: Final binary prediction



Detailed Mathematical Process

Step 2 - Subsequence Creation:

$$n = \left\lfloor \frac{l}{w} \right\rfloor \tag{1}$$

$$\mathbf{x}_i = X_{(i-1)w+1:(i-1)w+w} \quad \text{for } i = 1, 2, \ldots, n \tag{2}$$

Step 3 - Temporal Encoding:

$$Te_{sin}(i_t) = \sin\left(\frac{2\pi i_t}{p}\right) \tag{3}$$

$$Te_{cos}(i_t) = \cos\left(\frac{2\pi i_t}{p}\right) \tag{4}$$

Where $p = 24$ for hours, $p = 7$ for days
Result: Each subsequence $\mathbf{x}_i$ becomes $\mathbf{x}_{w \times m}$ where m is number of channels (consumption + temporal features)

Prediction Merging Strategy

Step 4 - Individual Predictions:

- TransApp predicts probability $p(\mathbf{x}_i)$ for each subsequence
- Results in probability vector: $P_X = (p(\mathbf{x}_1), p(\mathbf{x}_2), \ldots, p(\mathbf{x}_n))$

Step 5 - Quantile Aggregation:

$$\alpha_a^* = \arg \max_{\alpha \in \{0, 0.5, \ldots, 0.95, 1\}} S(y_{true}, y_\alpha^{pred}) \tag{5}$$

$$\text{Final Prediction} = \text{round}(Q_{P_X}(\alpha_a^*)) \tag{6}$$

Where $Q_{P_X}(\alpha_a^*)$ is the α_a^*-th quantile of P_X

Intuition: Instead of simple majority voting, use quantile that maximizes validation performance

Concrete Example

Input Series: X with length $l = 2048$ points (21 days of 30-min data)
Subsequence Creation: Choose $w = 256$ (2.67 days)

- $n = \lfloor 2048/256 \rfloor = 8$ subsequences
- $\mathbf{x}_1 = (x_1, x_2, \ldots, x_{256})$
- $\mathbf{x}_2 = (x_{257}, x_{258}, \ldots, x_{512})$
- ... and so on

Temporal Encoding: Each subsequence becomes 256×5 matrix

- Channel 1: Original consumption values
- Channels 2-3: Hour encoding $(\sin, \cos)$
- Channels 4-5: Day encoding $(\sin, \cos)$

Predictions: $P_X = (0.2, 0.8, 0.9, 0.1, 0.7, 0.3, 0.85, 0.6)$ **Final Result:** If optimal $\alpha^* = 0.7$, then $Q_{P_X}(0.7) = 0.8 \rightarrow \text{Label} = 1$

Hybrid CNN-Transformer Design

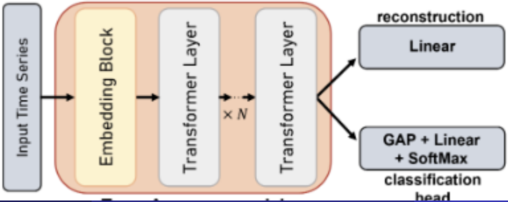
Core Components:

1. **Embedding Block:** Convolutional feature extraction
2. **Transformer Block:** Long-range dependency modeling
3. **Task-specific Heads:** Classification or reconstruction

Input/Output Flow:

$$\mathbf{x}_{w \times m} \xrightarrow{\text{Embedding}} \mathbf{z}_{w \times d_{model}} \xrightarrow{\text{Transformer}} \mathbf{z}_{w \times d_{model}} \xrightarrow{\text{Head}} \text{Output} \tag{7}$$

Where $d_{model} = 96$ is the latent dimension



Convolutional Feature Extraction

Architecture: 4 stacked Residual Units (ResUnits)
Each ResUnit contains:

- 1D Convolutional layer
- GeLU activation function
- BatchNormalization layer
- Residual connection

Dilation Pattern:

$$\text{ResUnit}_i : \quad d = 2^i \quad \text{for } i = 1, 2, 3, 4 \tag{8}$$

Purpose:

- Exponentially increasing receptive fields
- Preserves time dimension (stride $= 1$)
- Provides inductive bias for local patterns

Modified Attention Mechanism

Architecture: N stacked Transformer layers ($N = 3$ or 5)
Each layer contains:

1. Layer Normalization
2. Multi-Head Diagonally Masked Self-Attention (DMSA)
3. Layer Normalization
4. Position-wise Feed-Forward Network (PFFN)
5. Residual connections after DMSA and PFFN

Key Innovation - DMSA:

- Masks diagonal elements of attention matrix
- Attention scores: $A_{ii} = 0$ after softmax
- Emphasizes inter-token relationships
- Reduces overfitting on small datasets

No Positional Encoding: Temporal features already encoded in input

Detailed Mathematical Operations

Input: Subsequence $\mathbf{x}_{w \times m}$ where $w = 256$, $m = 5$
Embedding Block:

$$\mathbf{h}_1 = \text{ResUnit}_1(\mathbf{x}_{256\times 5}) \rightarrow \mathbf{h}_1^{256\times 32} \tag{9}$$
$$\mathbf{h}_2 = \text{ResUnit}_2(\mathbf{h}_1) \rightarrow \mathbf{h}_2^{256\times 64} \tag{10}$$
$$\mathbf{h}_3 = \text{ResUnit}_3(\mathbf{h}_2) \rightarrow \mathbf{h}_3^{256\times 96} \tag{11}$$
$$\mathbf{z} = \text{ResUnit}_4(\mathbf{h}_3) \rightarrow \mathbf{z}^{256\times 96} \tag{12}$$

Transformer Block:

$$\mathbf{z}' = \text{DMSA}(\text{LayerNorm}(\mathbf{z})) + \mathbf{z} \tag{13}$$
$$\mathbf{z}'' = \text{PFFN}(\text{LayerNorm}(\mathbf{z}')) + \mathbf{z}' \tag{14}$$

Output: Final representation $\mathbf{z}_{256\times 96}$

Self-Supervised Pretraining + Supervised Fine-tuning

Motivation:

- Large amounts of unlabeled smart meter data available
- Limited labeled appliance data
- Transformer architectures benefit from pretraining

Training Pipeline:

1. **Step 1:** Self-supervised pretraining on unlabeled consumption data
2. **Step 2:** Supervised fine-tuning on labeled appliance data

Masked Reconstruction Task

Objective: Learn consumption patterns without appliance labels
Masking Process:

- Randomly mask 50% of consumption channel
- Average mask segment length: $l_m = 24$ time steps (12 hours)
- Keep temporal encoding channels untouched

Architecture: TransApp + Reconstruction Head

$$\mathbf{z}_{w \times d_{model}} \xrightarrow{\text{Linear Layer}} \hat{\mathbf{x}}_{w \times 1} \tag{15}$$

Loss Function: Mean Absolute Error on masked elements

$$\mathcal{L}_{MAE} = \frac{1}{\#M} \sum_{i \in M} |\hat{x}_i - x_i| \tag{16}$$

Where $\#M$ is the number of masked elements

Appliance Classification Task

Objective: Detect specific appliances using learned representations
Architecture: TransApp + Classification Head

$$\mathbf{z}_{w \times d_{model}} \xrightarrow{\text{Global Avg Pool}} \mathbf{z}_{d_{model}} \xrightarrow{\text{Linear}} \text{logits}_2 \quad (17)$$

Training Details:

- Freeze or fine-tune pretrained weights
- Binary classification for each appliance type
- All subsequences inherit label from full series

Loss Function: Cross-entropy loss

$$\mathcal{L}_{CE} = -\sum_{c=1}^{2} y_c \log(\sigma(\text{logits}_c)) \quad (18)$$

Performance Benefits Analysis

Representation Learning Benefits:

1. **Pattern Recognition:** Learns general consumption patterns across households
2. **Temporal Dependencies:** Captures daily/weekly consumption rhythms
3. **Noise Robustness:** Develops robust features through reconstruction
4. **Data Efficiency:** Leverages abundant unlabeled data

Experimental Evidence:

- TransAppPT (pretrained) vs TransApp (no pretraining)
- Average improvement: 1-2 percentage points in Macro F1-score
- Larger improvements with more pretraining data
- TransAppPT-l (pretrained on 200k series) achieves best results

Scaling Effect: Performance increases proportionally with pretraining data size

Concrete Training Scenario

Pretraining Phase:

- Dataset: 200,000 unlabeled consumption series
- Input: Masked subsequences $\mathbf{x}_{256\times5}$
- Target: Reconstruct original consumption values
- Duration: 100 epochs

Fine-tuning Phase:

- Dataset: 3,000 labeled series (dishwasher detection)
- Input: Complete subsequences $\mathbf{x}_{256\times5}$
- Target: Binary labels (dishwasher present/absent)
- Duration: 50 epochs

Result:

- TransApp (no pretraining): 0.564 Macro F1
- TransAppPT (pretrained): 0.594 Macro F1
- **Improvement: +3.0 percentage points**

Performance Summary

Datasets:

- CER: 3,470 Irish households, 9 appliance types
- EDF 1: 4,701 French households, 7 appliance types
- EDF 2: 200,000 unlabeled series for pretraining

Key Findings:

1. ADF improves all baseline classifiers by 2-5 percentage points
2. TransAppPT achieves best overall performance (rank 1-2)
3. Pretraining on large unlabeled data provides significant gains
4. Framework scales efficiently to long time series

Best Results:

- Water Heater detection: 0.855 Macro F1
- Electric Vehicle detection: 0.825 Macro F1
- Average across all appliances: 0.746 Macro F1

Key Contributions and Impact

Technical Contributions:

1. **ADF Framework:** Scalable approach for long, variable-length series
2. **TransApp Architecture:** Hybrid CNN-Transformer with DMSA
3. **Two-step Training:** Effective use of unlabeled data
4. **Temporal Encoding:** Novel approach for time-aware features

Practical Impact:

- Enables real-world appliance detection for energy suppliers
- Supports personalized energy services and recommendations
- Contributes to energy transition goals
- Scalable to millions of smart meters

Future Directions:

- Multi-appliance detection
- Real-time inference
- Cross-domain adaptation