Integrating AI into Data-Center Operations

Dr Marco Lorusso, INFN-CNAF

Context

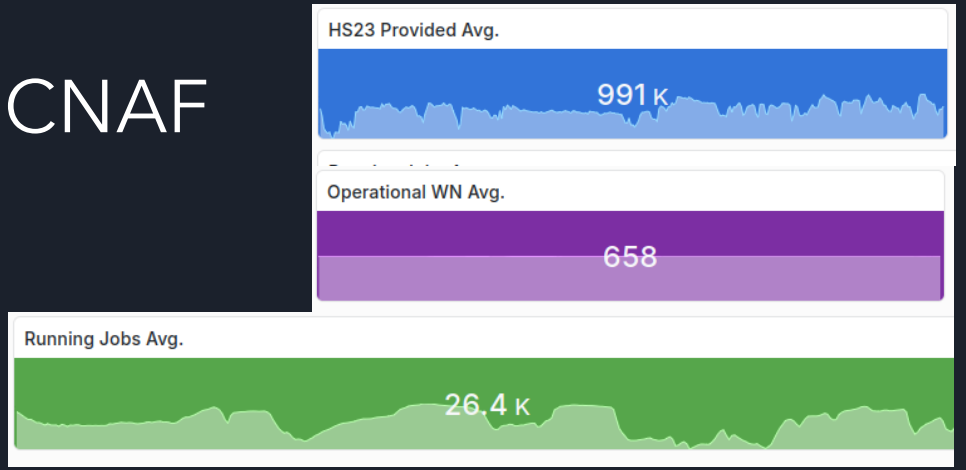
- The increasing **complexity** and **scale** of modern data centers generate operational **environments** where the ability to **detect anomalies**, **predict failures**, and **improve resource** usage is becoming **critically important**.
- Recent advances in **machine learning** and **artificial intelligence** offer **powerful techniques** for extracting actionable insights from **heterogeneous monitoring data**
- In this contribution, we explore an **AI-driven approac** that can be applied to a specific **large-scale computing** facility hosted at **INFN-CNAF** in Bologna.

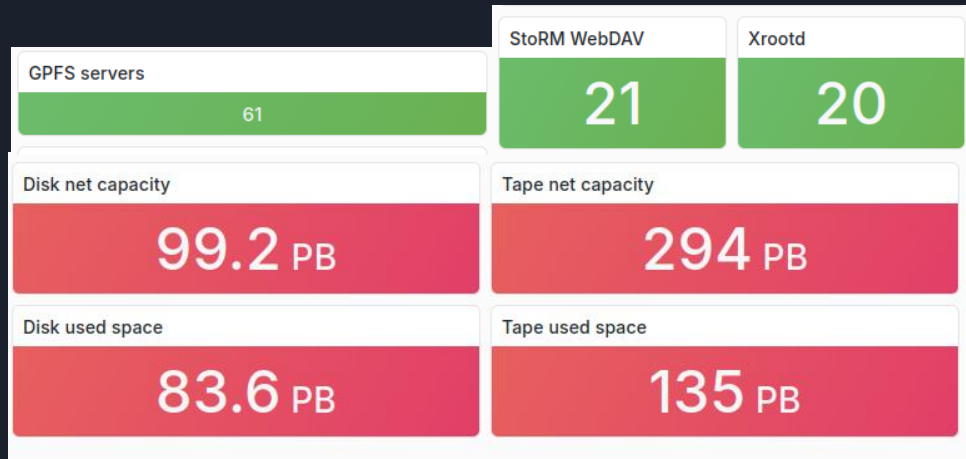
Data Complexity

- **Multiple Data sources**
  - Logs, metrics, monitoring data, and system outputs
- **Heterogeneous formats**
  - Structured (JSON), semi-structured (logs), and time-series metrics
- **Large data volumes**
  - High and continuously growing data amounts
- **Processing Complexity**

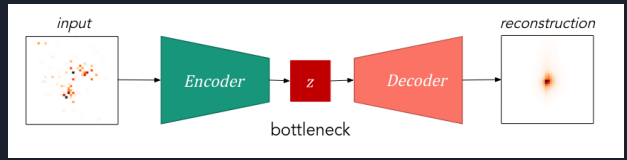
Data Complexity @ CNAF

- **Large-Scale Infrastructure @ CNAF**
  - CERN Tier-1 data center with large **computing** and **storage** capacity
- **High Data Volume**
  - Petabyte-scale storage and thousands of running jobs
  - Metrics, logs, and monitoring data from hosts and services
- **Analytical Challenge**
  - Large and diverse datasets require novel approaches for analysis

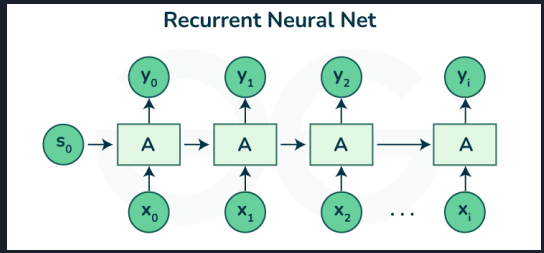




Different ML Approaches for AD

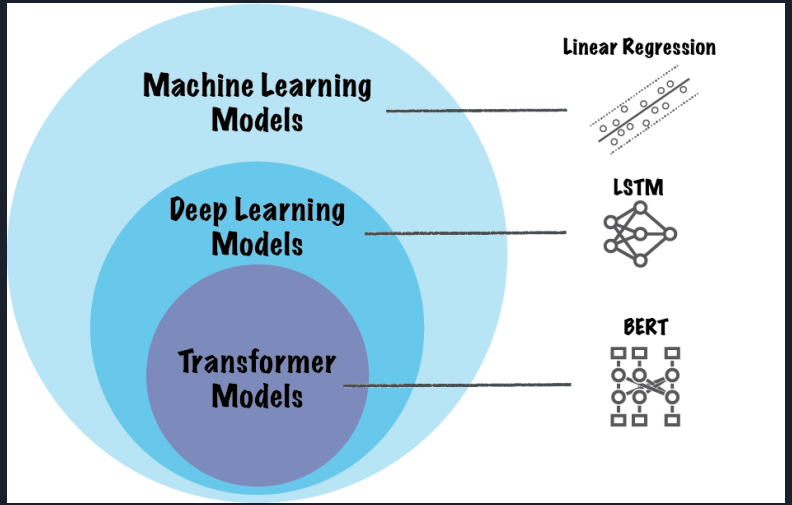


- **Autoencoder**: trained to **reconstruct** their **inputs:**
  - The **encoder compress** the input into a smaller latent space.
  - The **decoder reconstructs** from the compressed representation.
- **Recurrent Neural Network**: designed for processing sequential data
  - the **output** of a neuron at **one time step** is **fed back** as input at the **next time step**
- Transformer: based on the **multi-head attention mechanism:**
  - designed to model relationships **between elements in a set or sequence.**



Transformer

- Traditional ML models often impose **fixed structural assumptions**, such as **locality** or **sequential processing**;
- Transformers instead rely on **attention mechanisms** to learn **relationships directly from the data**;
    - hosts in a distributed system do not have a meaningful spatial ordering;
- Self-attention allows each element in an input representation to **interact with all other elements**;
    - The model learns which elements of the inputs are **relevant to each other**, enabling **flexible modeling** of complex interaction patterns.



Transformer Architecture

A typical Transformer layer consists of two main components:

1. **Self-attention:**
   - exchanges information between elements

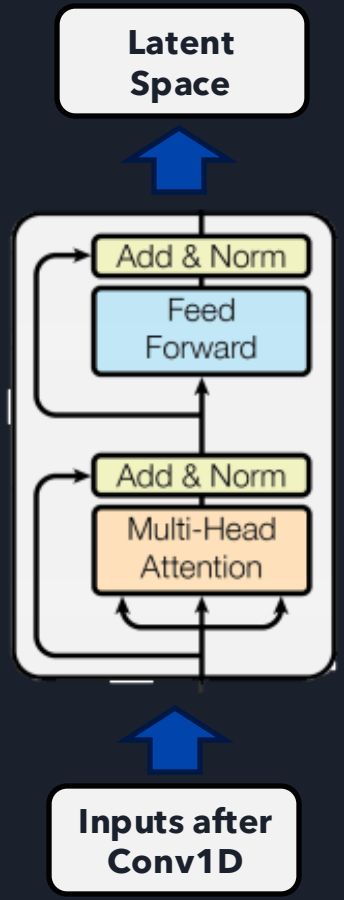
```
attn = tf.keras.layers.MultiHeadAttention(num_heads,key_dim= lat_dim //
                    num_heads, name = "multiheadattention")(e,e)

ee = tf.keras.layers.Add(name = "attn_resid")([e,attn])
ee = tf.keras.layers.LayerNormalization(name = "attn_ln")(ee)
```

2. **Feed-forward transformation:**
   - independently refines each element's representation

```
ffn = tf.keras.layers.Dense(lat_dim * ff_trans_dens, activation = //
                            "relu", name = "ffn1")(ee)
ffn = tf.keras.layers.Dense(lat_dim, activation = "relu", name = //
                            "ffn1")(ffn)

eee = tf.keras.layers.Add(name = "ffn_resid")([ee,ffn])
eee = tf.keras.layers.LayerNormalization(name = "fnn_ln")(eee)
```



Data Overview

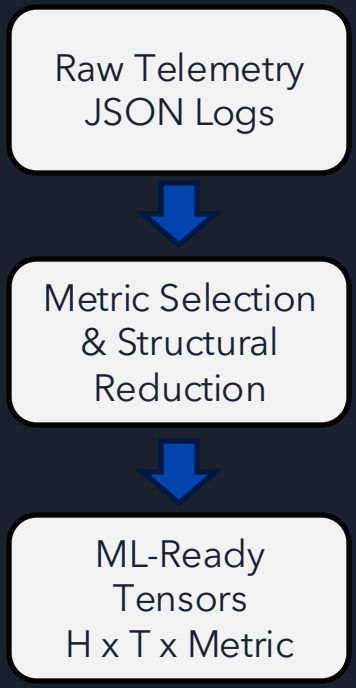
Operational telemetry collected from services managed by the **CNAF Farming Group**, which operates the computing clusters supporting distributed scientific workloads and large-scale data processing.

The data is a **multivariate time series** across **multiple hosts**, which include:

- CPU usage and load
- memory utilization and disk I/O statistics
- process-level resource metrics

Data Preparation and Processing

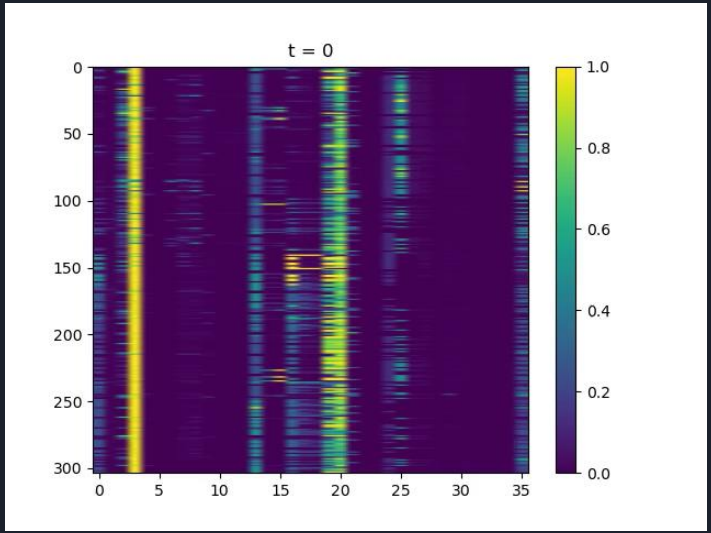
- Starting from multi-gigabyte JSON logs with redundant structure
- For Anomaly Detection data must be processed to:
  - Isolate informative system-level metric
  - Remove noisy and low-signal process-level measurements
  - Remove missing values and inconsistent reporting intervals
- The resulting Dataset contains:
  - **4086** (check number) **time** snapshots for **304** compute **nodes** with **36** system-level **metrics** each = a timeseries with ~11k features each step. (~44M input values)



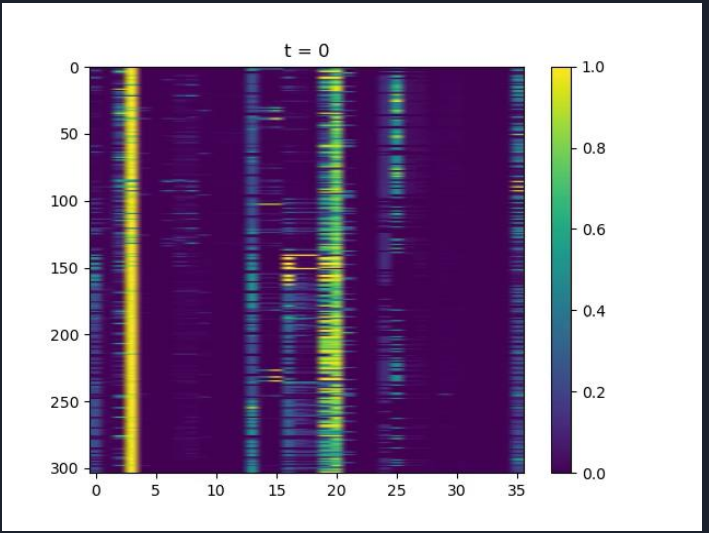
Signal injection

- Data coming from nodes in production
  - Real anomalies are rare
- To evaluate the detection method, anomalies are **synthetically injected** into validation data by simulating realistic events affecting random hosts and metrics:
  - CPU usage spikes
  - Sustained workload shifts
  - Disk I/O bursts
  - Memory leaks
  - Node outages
  - Abnormal process behaviour

Training v Test dataset



Time series **with** anomalies



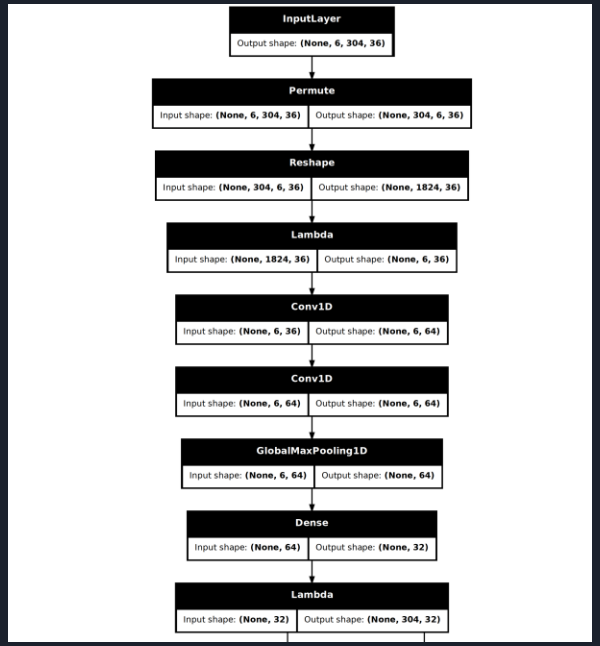
Time series **without** anomalies

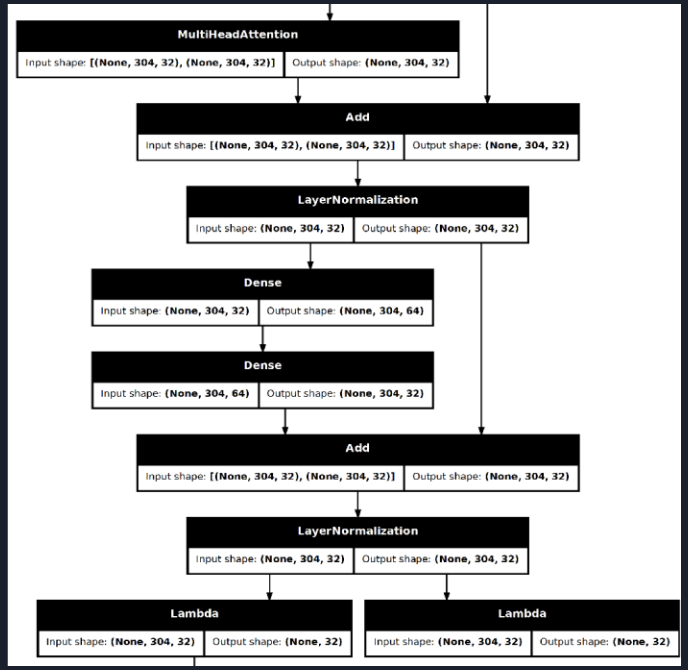
Model Architecture

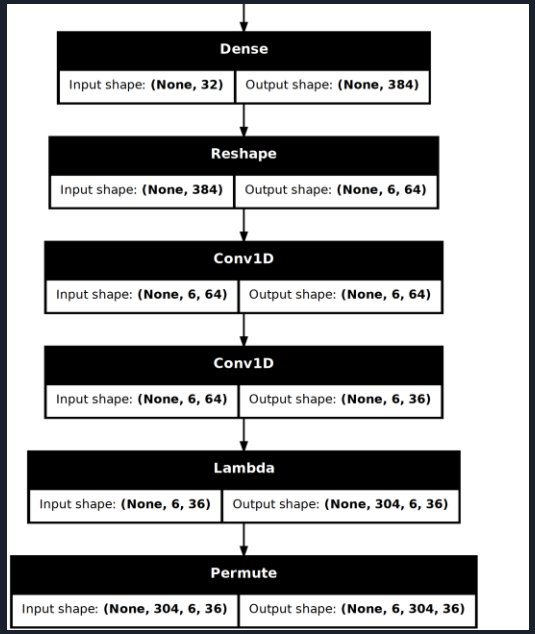
- **Input representation:** telemetry windows arranged as **time steps × hosts × features;**
- **Encoder:**
  - Per-host **temporal Conv1D layers** extract short-term patterns
  - **Global max pooling** produces a compact host representation
- Cross-host modeling:
  - Host embeddings processed with **multi-head self-attention**
  - Residual and feed-forward layers refine the contextual representation
  - A **global latent vector** summarizes the system state
- Decoder:
  - Host embeddings expanded and passed through **temporal Conv1D layers**
  - Reconstructs the original telemetry window

Model Architecture

 # parameters: 87,524

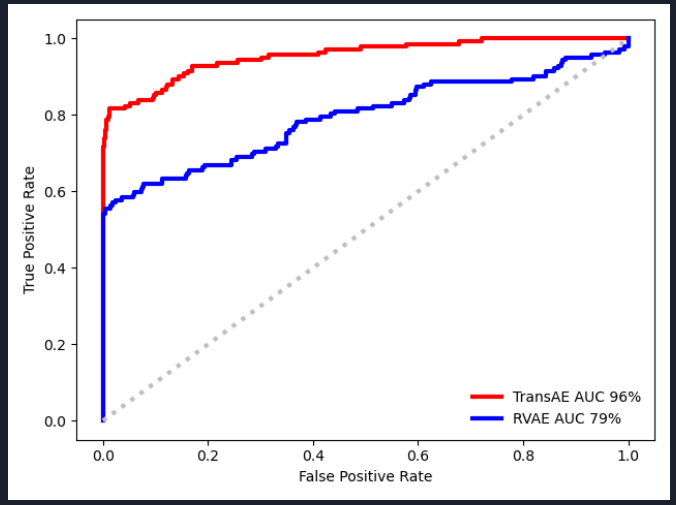


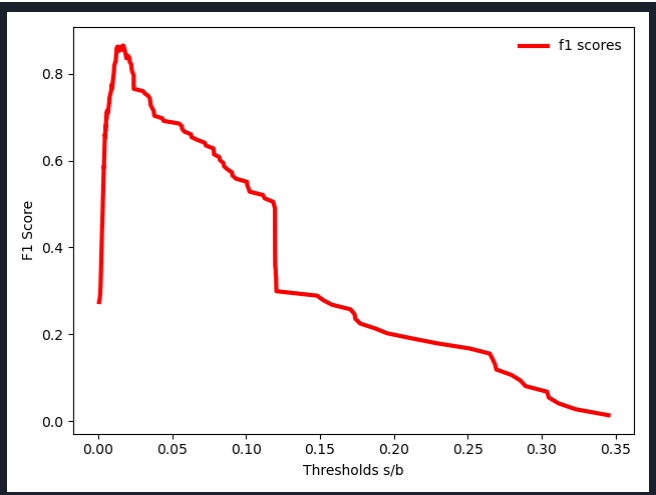




Results

- Injected anomalies provide controlled ground truth labels.
- Anomaly score:
  - Compute reconstruction error **per host**
  - Final score = **average error of the top-k most anomalous hosts**
- This allows quantitative evaluation of the anomaly detection model, e.g. using a ROC curve, its AUC and F1-score:
  - $F1 = \frac{2 \times \mathrm{TP}}{2 \times \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$



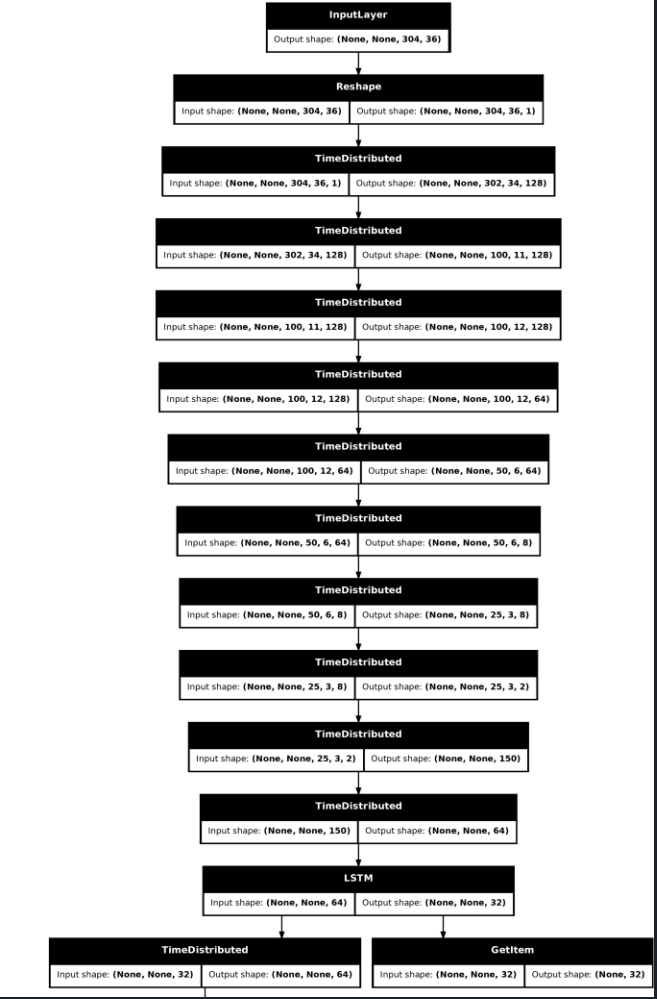


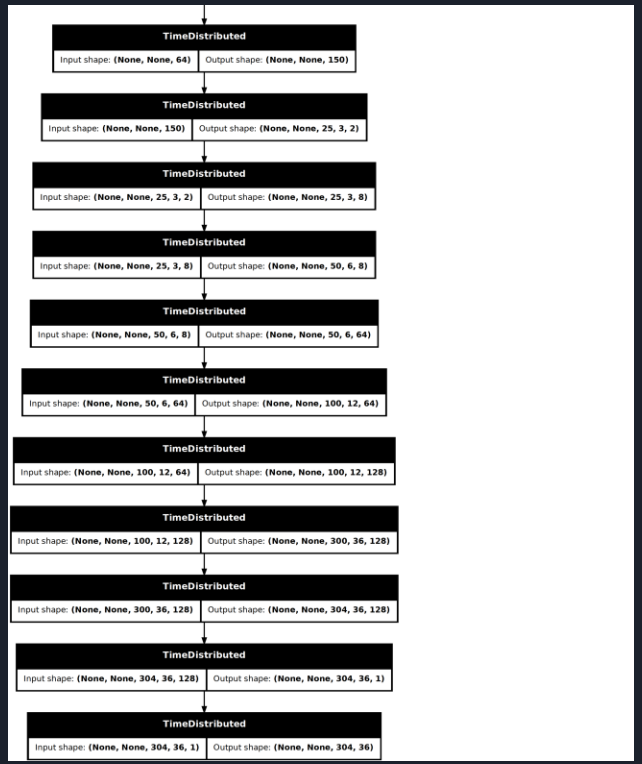
Conclusion

- **Autoencoders** provide a powerful tool for unsupervised anomaly detection in multivariate telemetry;
- **Attention mechanisms** enable modeling dependencies between hosts, allowing to detect cluster-wide behaviour;

Future Directions:

- **Improve model robustness** by incorporating additional feature engineering;
- Create **more complex and generalized architectures** for modeling temporal and cross-host dependencies;
- Develop a **dedicate data pipeline** for ingestion and preprocessing, enabling **systematic experimentation** and facilitating **deployment in the future**.


InputLayer
Output shape: (None, None, 304, 36)
Reshape
Input shape: (None, None, 304, 36)
Output shape: (None, None, 304, 36, 1)
TimeDistributed
Input shape: (None, None, 304, 36, 1)
Output shape: (None, None, 302, 34, 128)
TimeDistributed
Input shape: (None, None, 302, 34, 128)
Output shape: (None, None, 100, 11, 128)
TimeDistributed
Input shape: (None, None, 100, 11, 128)
Output shape: (None, None, 100, 12, 128)
TimeDistributed
Input shape: (None, None, 100, 12, 128)
Output shape: (None, None, 100, 12, 64)
TimeDistributed
Input shape: (None, None, 100, 12, 64)
Output shape: (None, None, 50, 6, 64)
TimeDistributed
Input shape: (None, None, 50, 6, 64)
Output shape: (None, None, 50, 6, 8)
TimeDistributed
Input shape: (None, None, 50, 6, 8)
Output shape: (None, None, 25, 3, 8)
TimeDistributed
Input shape: (None, None, 25, 3, 8)
Output shape: (None, None, 25, 3, 2)
TimeDistributed
Input shape: (None, None, 25, 3, 2)
Output shape: (None, None, 150)
TimeDistributed
Input shape: (None, None, 150)
Output shape: (None, None, 64)
LSTM
Input shape: (None, None, 64)
Output shape: (None, None, 32)
TimeDistributed
Input shape: (None, None, 32)
Output shape: (None, None, 64)
GetItem
Input shape: (None, None, 32)
Output shape: (None, 32)


TimeDistributed
Input shape: (None, None, 64)
Output shape: (None, None, 150)
TimeDistributed
Input shape: (None, None, 150)
Output shape: (None, None, 25, 3, 2)
TimeDistributed
Input shape: (None, None, 25, 3, 2)
Output shape: (None, None, 25, 3, 8)
TimeDistributed
Input shape: (None, None, 25, 3, 8)
Output shape: (None, None, 50, 6, 8)
TimeDistributed
Input shape: (None, None, 50, 6, 8)
Output shape: (None, None, 50, 6, 64)
TimeDistributed
Input shape: (None, None, 50, 6, 64)
Output shape: (None, None, 100, 12, 64)
TimeDistributed
Input shape: (None, None, 100, 12, 64)
Output shape: (None, None, 100, 12, 128)
TimeDistributed
Input shape: (None, None, 100, 12, 128)
Output shape: (None, None, 300, 36, 128)
TimeDistributed
Input shape: (None, None, 300, 36, 128)
Output shape: (None, None, 304, 36, 128)
TimeDistributed
Input shape: (None, None, 304, 36, 128)
Output shape: (None, None, 304, 36, 1)
TimeDistributed
Input shape: (None, None, 304, 36, 1)
Output shape: (None, None, 304, 36)