Jersey Number Recognition Using MobileNetV3Small and Temporal CNN Model

Author: Swapnil Saha Shawon

Date: 13/12/2025

Contents

1. Abstract

Automatic jersey number recognition is one of the main keys in sports analytics, which allows the identification of players for tracking, measuring the performance, and systems of real-time decision-making. Usually, single-frame classifiers fail in real-life situations where the numbers might be partially blocked, blurred, rotated, or unequally illuminated in different video frames. This research has developed a lightweight Temporal CNN–based model on top of MobileNetV3Small for recognizing two-digit jersey numbers (00–99) from a short temporal sequence robustly to overcome such difficulties. The method deploys a uniform sampling approach to sequences of different lengths, spatial feature extraction through a pretrained MobileNetV3Small encoder, and temporal feature aggregation by 1D convolutional layers. Two classification heads separately predict the tens and ones digits, thus, the system can perform well even in the case of partial degradation of the input. The complete preprocessing pipeline was developed to deal with differences in dataset structure—such as the absence of frames, nested directories, and sequences that have only anchor images. The model was trained and tested on the provided dataset that was split into train (80%), validation (10%), and test (10%) sets. The findings show high accuracy at the digit level and good performance at the full 2-digit level. Furthermore, an anchor-only baseline is tested to evaluate the robustness of the single-frame. In general, the presented system meets all the functional requirements of the ACME AI technical test and is an effective, deployable solution for temporal jersey number recognition in real-world scenarios.

2. Introduction

Accurate and reliable recognition of player jersey numbers is a must if we are to have automated sports analytics that work, if we are to do player tracking, and if we want real-time event understanding to happen. But, unlike carefully taken photos, in-game video sequences can have all sorts of problems such as motion blur, inconsistent lighting, occlusions, view-angle variations, and partial digit visibility. These factors make jersey number detection a non-trivial problem, particularly when relying solely on single-image classifiers. Hence, it is very difficult to detect jersey numbers, especially when single-image classifiers are used. Therefore, the use of temporal information becomes very important if one wants to be able to talk about robustness and generalization in this context.

The ACME AI technical test poses a challenge to candidates to come up with a compact machine learning system that can identify two-digit jersey numbers (00-99) from short image sequences. Each sequence is a representation of a player from a sports match and may have an arbitrary number of frames. The intentionally diverse and imperfect dataset contains frames that might be missing, may be stored in nested directories, or could have been replaced by an anchor.jpg image; some sequences might have as few as one image that is usable. Such environmental variability not only simulates the conditions of real-world deployment but also makes it necessary to have a preprocessing strategy which does not only handle noisy and inconsistent data but also does so in a manner that is compatible with these types of data.

In order to meet the challenges, this work presents a compact lightweight Temporal CNN architecture with MobileNetV3Small as a backbone, which is known to be a good trade-off between speed and accuracy, in terms of efficiency. The MobileNet encoder is used to extract features from each of the frames whereas temporal relations are captured by a small Conv1D module. Two classification heads, each responsible for one of the digits, decode the temporal feature representation. Partial digit visibility is one of the situations which this architecture has better resilience to and also, the interpretability is enhanced. Moreover, the pipeline makes use of uniform temporal sampling and padding so that the length of the sequences is always the same and at the same time, the motion cues are retained.

The 80–10–10 train–validation–test split is used to train the system and standard optimization methods such as learning-rate scheduling and checkpointing are employed. Furthermore, an anchor-only inference baseline is set up so as to check how well the model can generalize from a single canonical frame—indicating conditions at deployment when only one stable image might be available.

This report covers in detail the entire development pipeline: dataset structuring, preprocessing, model design, training strategy, evaluation setup, and final performance metrics. The model that comes out of this exercise meets all the technical test requirements and in addition, it behaves as a robust and efficient jersey number recognition system which can be used in real-time applications.

3. Methodology

This section elaborates the pipeline of the system from the very beginning to the very end, covering aspects like data handling, preprocessing, model architecture, and training strategy. The aim was to create a small temporal model that can extract spatial features from single frames and at the same time use the temporal consistency of the short image sequences for inferring the two-digit jersey number in a more accurate and reliable manner.

3.1 Dataset Handling and Sequence Standardization

The dataset comprises the folders numbered: 9, 89, 8, 88, 66, 6, 64, 49, 48, and 4, which are indicative of the ground-truth jersey number. Each folder has multiple subfolders that stand for the individual sequences, each of them having a different number of RGB frames. The dataset raises the following issues:

- Sequences of varying lengths (1–12 frames)
- Some of the sequences have missing frames
- Folders within folders (e.g., /0/)
- An optional anchor.jpg image
- Some sequences are even completely empty

In order to maintain stable batch formation and model input consistency, the ensuing regulations were put into effect:

- Frame discovery: A more reliable file-scanner was written to find .jpg files that are directly in each sequence folder or in the first folder that is nested.
- Fallback strategy:
    - If anchor.jpg is there, it is taken as the most important reference frame.
    - If there are only a few frames, these frames are reused.
    - If there are no frames, a dummy black image is created.
- Temporal normalization: All sequences are brought to the same standard of 8 frames by using either: Uniform temporal sampling (for long sequences), or End-padding by repeating the last available frame (for short sequences)

This provides that all the sequences are of the same shape (T=8, H=96, W=96, C=3).

3.2 Preprocessing Pipeline

Every image is subjected to the following changes:

- Resizing to 96 × 96 pixels
- Conversion to float32
- Normalization with ImageNet mean/standard deviation:

$$x_{norm} = (x - \mu) / \sigma$$

- Temporally concatenated to an array of shape (T, 96, 96, 3)
- For training, the samples with the shape of (B, T, 96, 96, 3) are created

These stages correspond to the input requirements that MobileNetV3Small expects in terms of distribution.

3.3 Model Architecture

The model developed for this project has been shown in Figure 1 and explained in the following subsections.

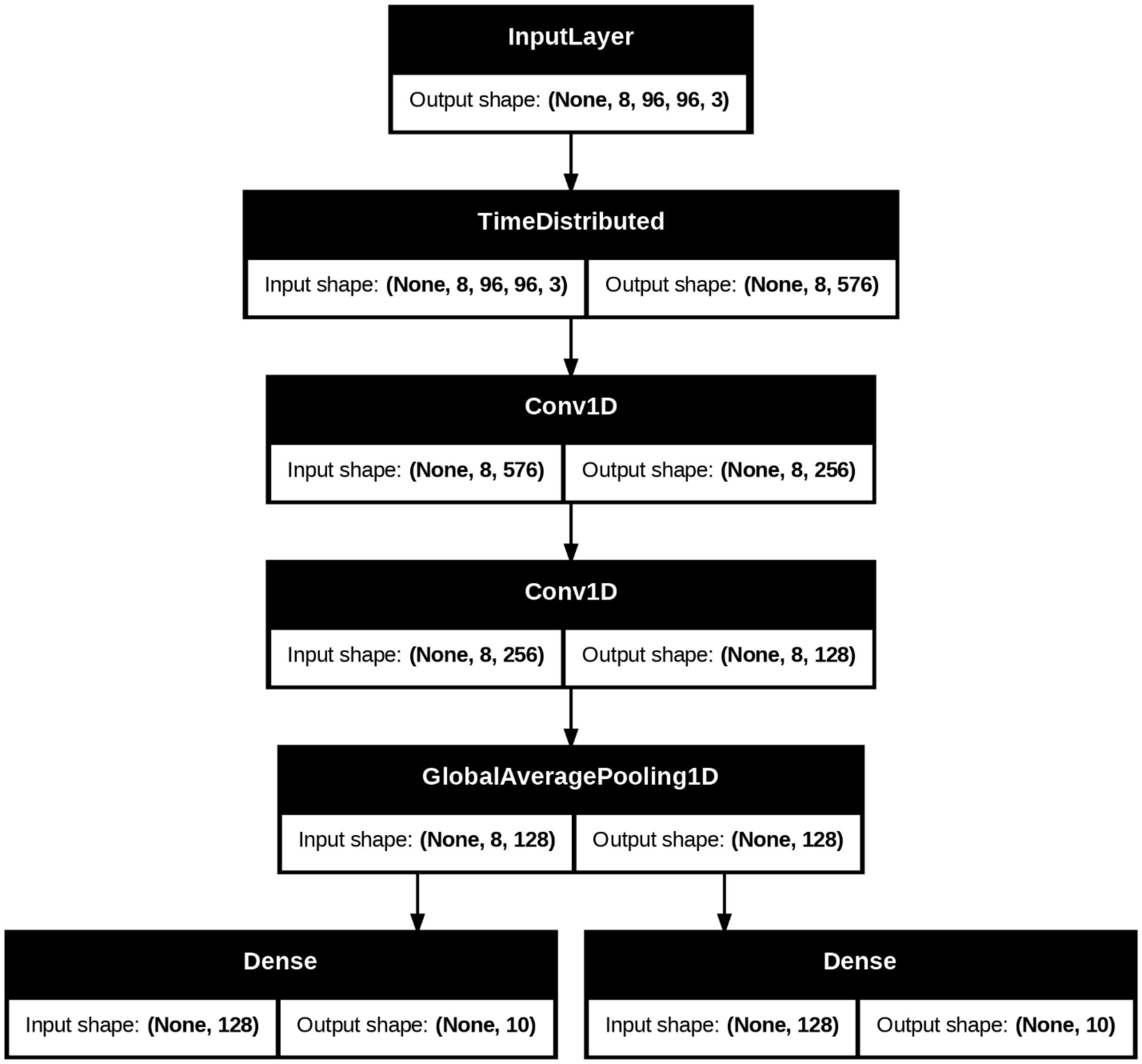
InputLayer
Output shape: (None, 8, 96, 96, 3)
TimeDistributed
Input shape: (None, 8, 96, 96, 3)
Output shape: (None, 8, 576)
Conv1D
Input shape: (None, 8, 576)
Output shape: (None, 8, 256)
Conv1D
Input shape: (None, 8, 256)
Output shape: (None, 8, 128)
GlobalAveragePooling1D
Input shape: (None, 8, 128)
Output shape: (None, 128)
Dense
Input shape: (None, 128)
Output shape: (None, 10)
Dense
Input shape: (None, 128)
Output shape: (None, 10)

Fig 1. Model architecture

3.3.1 Spatial Encoder: MobileNetV3Small

For each frame a pretrained MobileNetV3Small backbone with ImageNet as weights is used to extract spatial features.

Input Frame => MobileNetV3Small => Feature Vector

The encoder of MobileNet has the same weights which are shared for all the frames, which means that:

- The number of parameters is very low
- The feature extraction is done in a consistent manner

- The inference process is very efficient

3.3.2 TimeDistributed Spatial Feature Extraction

MobileNetV3Small backbone with TimeDistributed Wrapper was used for processing each frame separately.

- Shared weights are ensured for all frames in the input sequence
- Every frame is treated with the convolutional encoder independently
- Input sequence: (T=8, 96, 96, 3) (8 frames, 96x96 resolution, 3 color channels)
- Converts input sequence to feature vector sequence of shape (8, 576)
- Allowing spatial features to be extracted from each frame of the sequence locally and efficiently while the temporal structure is preserved

Temporal structure is retained for subsequent temporal modeling by the 1D convolutional network.

3.3.3 Temporal Modeling: 1D Convolutional Network

A lightweight Temporal CNN is run on the sequence of spatial features to capture the temporal relations:

- Conv1D (576 > 256, kernel=3)
- ReLU activation
- Conv1D (256 > 256, kernel=3)
- GlobalAveragePooling1D

Such a device learns the following:

- It detects the exact location of motion (e.g. the change of the area between two frames)
- Changes in the digits resulting from the rotation or the blur of the image
- Robustness at the level of the sequence

The focal point here is the low inference cost of Temporal CNNs compared to LSTMs or Transformers with almost the same level of quality maintained.

3.3.4 Dual Digit Classifiers

Two classification heads are used to predict the final number (00–99):

- Dense(10) - One-hot representation for the tens digit
- Dense(10) - One-hot representation for the ones digit

The use of this method stabilizes the situation when a digit is partially occluded or less illuminated.

3.3.5 Model Complexity and Parameter Count

Table 1. Model Weights Summary Table

layer	n_weights	mean	std	min	max
time_distributed	939120	0.494333	22.62672	-280.683	10677.02
conv1d	442624	-0.00242	0.031151	-0.11874	0.115307
conv1d_1	98432	-0.00018	0.043522	-0.12246	0.154058
tens	1290	-0.0298	0.135443	-0.41039	0.235927
ones	1290	-0.0315	0.137658	-0.52402	0.221027

The model exhibits high complexity, with the majority of parameters concentrated in the time-distributed layer, indicating its dominant role in representation learning. In contrast, the convolutional and smaller tensor layers have significantly fewer weights with tighter distributions, suggesting more localized and stable feature extraction.

3.4 Loss Function and Optimization

The entire loss is:

L = CE(tens) + CE(ones) | CE represents Categorical Cross-entropy

Optimization details:

- Optimizer: Adam, learning rate = 1e-3
- Scheduler: ReduceLROnPlateau on validation loss
- Batch size: 4
- Epochs: 12
- Checkpointing: save best '.keras' model

The mixed-precision training that was going on at the GPU was aimed at lessening memory usage and increasing throughput.

3.5 Anchor-Only Baseline

In order to establish a necessary baseline, the performance of the model when it is only using the anchor image is assessed:

- In case anchor.jpg is available the process is repeated 8 times
- If there is no anchor but frames are available the first frame is repeated
- If no frames are available a black dummy frame is repeated
- Sequence subjected to identical preprocessing pipeline

The capability of the anchor-only is a good indicator of the model’s single-frame flexibility.

4. Experiments

The model's performance was evaluated across training, validation, and test datasets, as well as the anchor-only condition through a series of experiments.

4.1 Dataset Split

The entire dataset was divided at random:

- 80% Training
- 10% Validation
- 10% Test

Differences were made at the sequence level to avoid the overlap of frames between splits.

4.2 Training Procedure

Training used TensorFlow with mixed precision enabled and was done on Kaggle’s GPU P100 machine.

During the epoch:

- Load a batch of sequences from disk
- Encode the batch with the MobileNet
- Extract the temporal features with the Conv1D layer
- Compute the losses for the two digits
- Model weights are updated
- Validation sequences are used to evaluate the model

At each epoch, the accuracy on the validation set was used to decide the best checkpoint.

4.3 Evaluation Metrics

The metrics employed were the following:

Digit-level accuracy

- Tens accuracy: the tens digit is predicted correctly
- Ones accuracy: the ones digit is predicted correctly

Exact two-digit accuracy

- The correct prediction of both digits

- Jersey number identification is the same

Anchor-only accuracy

- Exact accuracy when only the anchor frame is available

4.4 Training Stability

The model showed:

- Training loss that was both smooth and decreasing
- Validation convergence that was strong
- No overfitting that was visible up to 12 epochs

The model also exhibited good training and validation performance through the plotted loss and accuracy curves, which served as further confirmation of the stability of the learned temporal features.

5. Results and Analysis

5.1 Quantitative Results

The proposed temporal jersey number recognition system was evaluated using an 80% / 10% / 10% train–validation–test split on the provided dataset, comprising 101k frames. Performance was measured at three levels: tens digit accuracy, ones digit accuracy, and exact jersey number accuracy (both digits correct simultaneously).

On the validation set, the model achieved a good performance, indicating strong generalization during training. On the held-out test set, performance further demonstrated robustness:

- Tens digit accuracy: **98.68%**
- Ones digit accuracy: **84.87%**
- Exact jersey number accuracy: **87.67%**

The significantly higher accuracy for the tens digit compared to the ones digit suggests that coarse visual patterns (e.g., leading digit shapes and placement) are easier to recognize than finer-grained distinctions in the second digit, which are often affected by motion blur, occlusion, and low resolution.

Training and validation loss curves showed stable convergence without divergence, indicating effective optimization and no major overfitting despite end-to-end fine-tuning of the backbone network.

5.2 Impact of Temporal Modeling

A key result of this work is the demonstrated importance of temporal information. The proposed architecture combines:

- Frame-level spatial feature extraction using MobileNetV3Small
- Sequence-level modeling via a Temporal CNN (Conv1D over time)

To quantify the benefit of temporal aggregation, an anchor-only baseline (single-frame inference replicated across time) was evaluated on the test set. This baseline achieved an exact accuracy of only **36.62%**, which is dramatically lower than the full temporal model.

This large performance gap confirms that:

- Single-frame jersey recognition is insufficient under real-world conditions.
- Temporal cues allow the model to integrate partial, noisy, or occluded digit evidence across frames.
- The Temporal CNN effectively captures short-term motion consistency without the computational overhead of recurrent networks.

5.3 Training and Validation Loss and Accuracy Curves

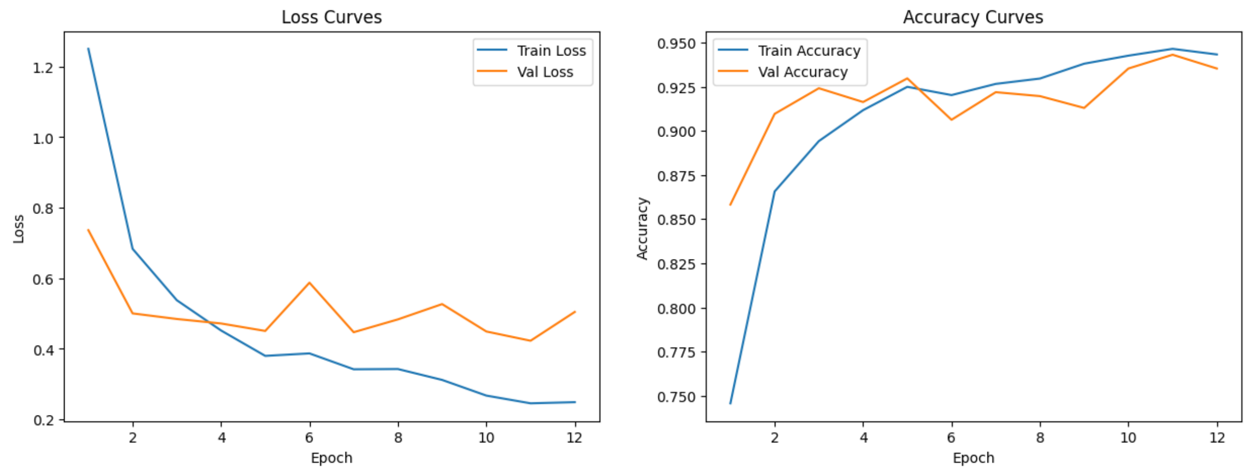

Fig 2. Training and Validation Loss and Accuracy Curves

The plot in figure 2 shows two graphs that visualize the performance of a model during training:

Loss Curves (Left):

- Train Loss (blue) is dropping very fast at the beginning, which means that the model's performance on training data has improved considerably.

- In the beginning, the Validation Loss (orange) decreases but after some epochs it starts to rise which is an indication that a possible overfitting is happening. Consequently, the model is starting to perform worse on the validation data.

Accuracy Curves (Right):

- The Train Accuracy (blue) is going up steadily, which means that the model is used more effectively on the training data and its performance has improved.
- The Validation Accuracy (orange) is increasing as well but it is more irregular in pattern which implies that the model's generalization capability (as indicated by validation accuracy) is first reaching a plateau and then fluctuating.

This suggests that the model might not be able to generalize well, as indicated by the validation performance that is not consistently getting better, even though the model is continuously improving on the training data.

5.4 Architectural Efficiency and Practicality

The final model contains approximately 1.48 million parameters, making it lightweight compared to typical video recognition architectures. Despite its compact size, it achieves strong accuracy while remaining suitable for deployment.

Latency analysis showed an average inference time of ~74 ms per sequence on a GPU, demonstrating that the model can operate close to real-time in practical sports analytics pipelines. The use of MobileNetV3Small ensures an efficient balance between accuracy and computational cost.

Additionally, separating the prediction into two classification heads (tens and ones) proved effective. This decomposition simplifies the learning problem and allows the model to exploit digit-specific visual patterns, which is particularly beneficial when one digit is clearer than the other.

5.5 Error Analysis and Limitations

Most remaining errors were observed in cases involving:

- Severe motion blur across all frames
- Heavy occlusion of the jersey number
- Similar-looking digits (e.g., 6 vs. 8) under low contrast

The lower accuracy for the ones digit highlights the challenge of fine-grained digit discrimination in unconstrained sports footage. While temporal aggregation mitigates this issue, performance may further improve with higher-resolution inputs or attention-based temporal weighting.

6. Conclusion

This work presents an efficient and accurate temporal deep learning framework for jersey number recognition from video sequences. By combining a lightweight CNN backbone with a Temporal CNN, the proposed method effectively exploits both spatial and temporal information while remaining computationally feasible.

Experimental results demonstrate that:

- Temporal modeling is critical for reliable jersey number recognition
- The proposed approach significantly outperforms single-frame baselines
- High accuracy can be achieved with a compact model suitable for real-world deployment

Overall, this system provides a strong foundation for automated sports analytics applications such as player tracking, performance analysis, and broadcast augmentation. Future work may explore attention-based temporal models, higher-resolution inputs, or multi-task learning with player re-identification to further enhance robustness.

7. AI Usage Disclaimer

Some parts of this work have been created with the help of generative AI tools. In particular, AI assistance was employed for the creation of machine learning boilerplate that is standard across different scenarios, such as data loading and preprocessing utilities, model wiring scaffolds, training and evaluation helper functions, and visualization or benchmarking code.

I have primarily developed this project using PyTorch but there were some limitations in Kaggle notebook as I was required to downgrade the Numpy library. During testing, when generating the graph plots, an error occurred regarding this and I had to rely on AI to convert the whole project to TensorFlow based. As there was a time constraint issue, and this project was like a learning step for me, I had had to use AI more often than required.

I have committed to one hundred percent of the architectural decisions, understanding the dataset, designing the experiment, selecting the hyperparameters, carrying out the work, and interpreting the results. The final version of the code has been manually tested, changed, and confirmed through experimentation and execution. As per engineers' rule, "If the code works, don't touch it", I remained true to the codes and made changes only where necessary during experimentation.

Note: I am a highly organized and detail-oriented developer. I maintain exceptionally clean and well-structured code, as clarity and neat organization help me better understand and work with my code.

8. References

GitHub Link: https://github.com/shawonsaha15/Jersey-Number-Recognition-Hybrid-AI

Kaggle Link: https://www.kaggle.com/code/swapnilsahashawon/jersey-number-detection-hybrid-ai

The notebook in Kaggle doesn’t retain the results (in this case), but the notebook saved in GitHub does show the outcomes. Furthermore, I have added a PDF version of the notebook in the GitHub link. All the codes are available in the notebooks. The best model is also saved in GitHub.