Jersey Number Recognition Project Proposal

Course: COSC 419B/519B
Instructor: Dr. Mohamed Shehata
Team Number: 3
Team Members: Ghaith Chrit: 52255320, Hamza Muhammad Anwar: 75686469, Karel Harjono: 93446631, Amey Karmarkar: 57577843, Tanmaya Karmarkar: 66188533

---

1. Introduction

SoccerNet [1] is a large-scale dataset that contains images, videos, and labels for soccer video research. Since being released in June 2018, the database has evolved to include various tasks including camera calibration, action spotting, and player re-identification and tracking. In 2021, the first two SoccerNet challenges were proposed at the ActivityNet workshop of the Conference on Computer Vision and Pattern Recognition (CVPR). The challenges were Action Spotting, which consists of finding all the actions occurring in the videos, and Replay Grounding, which consists of retrieving the timestamp of the action shown in a given replay shot within the whole game.

These challenges were repeated in 2022 and 2023, along with some new ones. Notably, the Player Jersey Number Recognition Challenge [2]. This challenge aimed to identify a player's jersey number within the given tracklets. The task is difficult since the thumbnails are of poor quality (low resolution, excessive motion blur), and only a very small portion of the tracklet may display the jersey numbers. The SoccerNet Jersey Number dataset is composed of 2853 player tracklets taken from the SoccerNet tracking videos. The challenge set is composed of 1211 separate tracklets with hidden annotations. This task was not subsequently repeated in the 2024 and 2025 editions of the SoccerNet challenges.

Because it aids in the development of systems that can precisely identify and recognize players in real-time during a soccer match, this endeavor may prove beneficial for consumer applications. Fan-focused interactive experiences, like player statistics, player monitoring, and in-game analysis, can be made with this data. Furthermore, this technology can be applied to broadcast production to improve the viewing experience and add player labels. The SoccerNet Jersey Number challenge helps create more entertaining and educational consumer applications for soccer fans by promoting player recognition research.

2. Literature Review & Replicated Results

2.1 Related Work

Jersey number recognition in sports videos presents several challenges, including, but not limited to, motion blur, occlusions, and low resolutions. Addressing these challenges requires advances in multiple fields that include humane pose estimation, scene text recognition (STR),

and super-resolution (SR). A thorough review of existing methodologies is essential for identifying effective solutions, evaluating their strengths and limitations, and guiding further improvements in recognition performance.

Human pose estimation plays a crucial role in jersey number recognition by localizing regions of interest (the back of the shirt) [3]. While the baseline model [3], uses one of the most common approaches, ViTPose [4], a vision transformer-based model that achieves high accuracy in pose estimation tasks. However, transformer-based models can be computationally demanding and may struggle in the presence of occlusion. In an attempt to address some of the limitations of VitPose, Pose as Compositional Token (PCT) [5] was proposed. Instead of relying on heatmaps, it represents the human pose as a set of compositional tokens. This approach may improve robustness in challenging conditions, such as partial occlusions and rapid movements, making it a promising alternative for sports video applications. Additionally, VitPose++ [6] improves on VitPose, by addressing heterogeneity in the body keypoints while also increasing the throughput and accuracy of the model compared to VitPose.

Once jersey numbers are localized, STR techniques are applied to extract the numerical information. Large Transformer-based models such as ParSEQ [7], have demonstrated strong performance in recognizing text from natural scenes. However, when implementing such systems in real-time applications, a more lightweight model is often preferred. PP-OCER-v4-rec [8] is an optimized OCR system designed for efficiency in real-world applications, making it suitable for real-time processing. Additionally, SVTR-TINY [9] provides a lightweight transformer-based approach that strikes a balance between performance and computational cost, making it an attractive choice for deployment in resource-constrained environments. CLIP4STR [10] transforms CLIP [11] into a scene text reader by utilizing dual encoder-decoder branches: a visual branch for initial text prediction and a cross-modal branch for refining predictions by aligning visual features with textual semantics, enabling state-of-the-art (SOTA) performance in STR.

Low-resolution text remains a significant challenge in Optical character recognition (OCR) pipelines, as poor image quality can lead to misclassification and segmentation errors. Extrapolating from research in other computer vision areas that showed that higher pixel density in critical regions enhances recognition accuracy [12], it is reasonable to assume that a similar result would be found in OCR and STR tasks. Additionally, [13] showed that SR techniques can enhance OCR pipelines by recovering fine-grained text details and mitigating artifacts caused by poor image quality. Text Gestalt [13] uses the local stroke structure to capture the stroke-aware prior from a Transformer-based recognizer to guide the super-resolution process, ensuring that critical text features are preserved and enhanced. To tackle the information bottleneck caused by abrupt feature map intensity degradation in deep SwinIR-based super-resolution networks [14], [15], DRCT [16] introduces dense-residual connections within the Swin Transformer to stabilize the spatial information flow. Thus, enabling more robust recovery of fine-grained text details while maintaining computational efficiency. Notably, DRCT achieves SOTA performance across multiple benchmark datasets, demonstrating its superiority in enhancing super-resolution quality and preserving critical spatial details.

Research in data augmentation has highlighted its critical role in enhancing the generalization capabilities of STR models. A recent study [17] systematically categorizes data augmentation techniques into eight distinct types: warp, geometry, noise, blur, weather, camera, pattern, and process-based transformations. These strategies enable STR models to better handle

real-world variations, such as changes in lighting, perspective distortions, and diverse environmental conditions. Following these data augmentation, [17] was able to improve the performance of baseline STR models without changing the models' architecture. Complementing these approaches, [18] introduced a self-compositional data augmentation method for scene text detection, which generates new training samples by applying instance-level variations such as translation, scaling, rotation, and curving to text regions within an image. The effectiveness of data augmentation has been underscored in practical applications, including the SoccerNet 2023 Challenges [19], where the winning team heavily relied on data augmentation to improve model robustness and achieve state-of-the-art performance.

2.2 Replication of State-of-the-Art Results

2.2.1. Evaluating Off-The-Shelf Models

To establish a baseline performance for Optical Character Recognition (OCR) models, we evaluated two pretrained recognition models, PP-OCR-v4-rec [6] and SVTR-TINY [7], on the 824 cropped images from the result of the pose detector module. Both models were tested using the PP-OCR-v3-det detection model using PaddleOCR running on a CUDA-enabled machine with NVIDIA GeForce RTX 4070 Laptop GPU.

2.2.1.1 PP-OCR V4

Model	Detection Model	Accuracy	Inference Time (sec)
PP-OCR-v4-rec	PP-OCR-v3-det	75.485%	1074.2
SVTR-TINY	PP-OCR-v3-det	57.767%	880.4

Table 1: Comparison of different OCR models based on accuracy and inference time

The models were trained on a large-scale dataset consisting of real-world and synthetic images across different OCR tasks [8]. The dataset includes:

- 97K images for text detection from real-world scenes and synthetic data.
- 600K images for direction classification, including reversed text and vertical fonts.
- 17.9M images for text recognition, with synthetic augmentations to improve robustness.
- Multilingual Support: Additional datasets cover French, Korean, Japanese, German, and alphanumeric symbols.
- Public Dataset Sources: LSVT [20], RCTW-17 [21], MTWI 2018 [22], CASIA-10K [23], SROIE [24], MLT 2019 [25], BDI [26], MSRATD500 [27] and CCPD 2019 [28].

Table 1 shows that SVTR-TINY [9] is the faster model while PP-OCR-v4-rec achieves significantly higher accuracy (75.49% vs. 57.77%) with only a marginal increase in inference time. Although PP-OCR-v4-rec requires slightly more processing time, the trade-off is minimal compared to the substantial gain in accuracy. This suggests that PaddleOCR's PP-OCR-v4-rec is a more effective model for real-world text recognition, particularly in scenarios where accuracy is prioritized over speed. These results indicate that PP-OCR-v4-rec may be better suited for inclusion in, or even as a replacement for, SVTR-TINY in the ensemble method used in the Scene Text Recognition module of the baseline framework for jersey number recognition [3].

2.2.2 Evaluating the Jersey Number Recognition Pipeline

2.2.2.1 Performance Analysis

In order to replicate [1], we first successfully reproduced the results of the original paper, achieving a test accuracy of 86.7%, which is roughly consistent with respect to the reported accuracy of 87.45%. This confirms the correctness of the provided implementation and establishes a reliable baseline for further experimentation. To performance profile the pipeline, we utilized the following resources: an AMD EPYC 7742 64-Core Processor with 256 threads, 64 cores per socket, and an NVIDIA A100-SXM4-80GB GPU. These computational resources enabled us to conduct detailed analyses of runtime and resource utilization, identifying bottlenecks such as sequential processing in the legibility classifier and inefficiencies in the pose estimation module. These insights guided our subsequent optimizations and modifications to improve both speed and efficiency.

2.2.2.2 Code

In order to get a performance profile, we used the nvtx library and added nvtx markers around the routines that were being called. We went through the pipeline and put appropriate markers to give us a breakup of the pipeline. The modified code is pushed to our git repository. We then ran the entire pipeline using nsys to get a report that can be viewed in Nsight systems to analyze and get the required profile. The generated Nsight reports are a link to our repository and are available in Section 8.2 and 8.1, respectively.

2.2.2.3 Performance Analysis: Testing Scenario

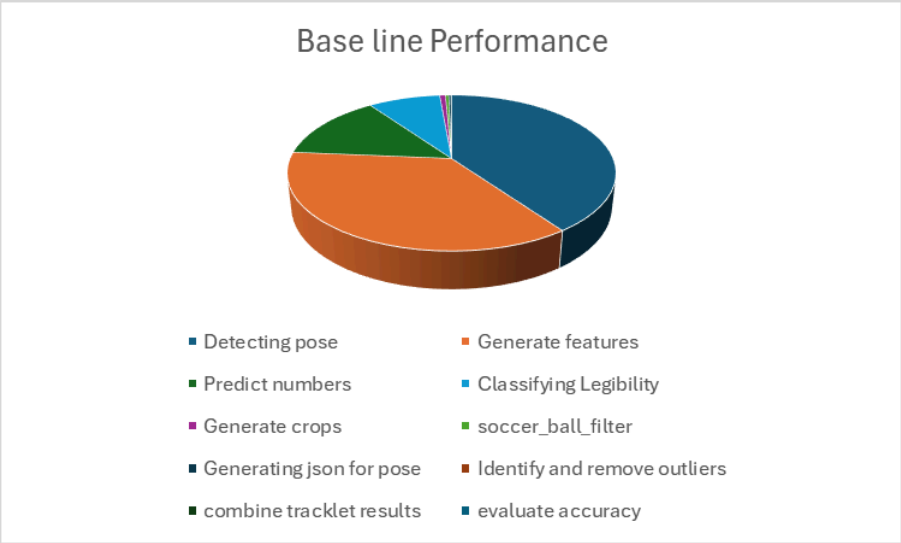

Figure 1: Pie Chart showing the baseline performance of the individual testing components.

We ran the entire pipeline as given in [3] in testing mode with 11 examples to obtain the profile that is given in Figure 1 and Table 2. Less number of examples are used as the purpose here is to get information on the computational performance of the code and not the accuracy results of the model. The performance analysis could be used to better understand what bottlenecks the general framework suffers from, which could narrow down our focus on what to improve in it.

As can be seen from Figure 1, the greatest amount of resources were used by the Pose Detector (ViTPose [2]). After that, generating features also used a lot of computational power, almost equivalent to the pose detector's amount. The jersey number predictions and legibility classifier used almost the same amount of resources, and after these minimal amounts were used for the rest of the computations such as generating crops and removing outliers. Table 2 shows how much time each component took for our testing scenario on 11 examples.

Stage Name	Time Taken (sec)
Detecting pose	68.750
Generate features	62.104
Scene Text Recognition	23.672
Classifying Legibility	14.478
Generate Crops	1.158
Soccer Ball Filter	0.653446
Generating Json For Pose	0.476208
Identify and Remove Outliers	0.072719
Combine Tracklet Results	0.035379
Evaluate Accuracy	0.001451
Entire pipeline	171.403

Table 2: The breakdown of runtime for each routine (Testing)

The time taken for each routine reflects the amount of computational resources used in Figure 1. Pose detection and feature generation take the most amount of time. Followed by the jersey number prediction and legibility classification take similar amounts of time. The rest of the routines take negligible amounts of time to complete.

2.2.2.4 Analysis of GPU utilization

Stage Name	Metric	Metric Value
Classifying Legibility	GPU utilization	18.90%
	Time Taken in Kernels	79 %
	Time Taken for Memory Transfers	21 %
Generating Features	GPU utilization	28.20%
	Time Taken in Kernels	98 %
	Time Taken for Memory Transfers	2 %
Detecting Pose	GPU utilization	89.00%
	Time Taken in Kernels	99 %
	Time Taken for Memory Transfers	1 %
Predicting Numbers	GPU utilization	17.00%
	Time Taken in Kernels	98 %
	Time Taken for Memory Transfers	2 %

Table 3: GPU Utilization of each program (Testing)

Table 3 shows how much of the GPU each program used to perform computations during the time the pipeline was run. We can see that legibility classification, feature generation, and jersey number prediction do not use much of the GPU capacity. This provides us an avenue of

improvement by modifying the code to allow these components to use more of the GPU power to increase pipeline performance and decrease time taken. Based on the profile, we analyze that:

1. Top Four Time-Consuming Routines: These routines represent the primary computational bottlenecks in the pipeline and will be the focus of our optimization efforts.
2. GPU Utilization: Current GPU usage is suboptimal, indicating potential for significant performance gains through better resource management.
3. Legibility Classifier: By streaming and overlapping the processing of different examples, we can achieve more efficient GPU utilization and reduce idle time, leading to improved throughput and overall performance.

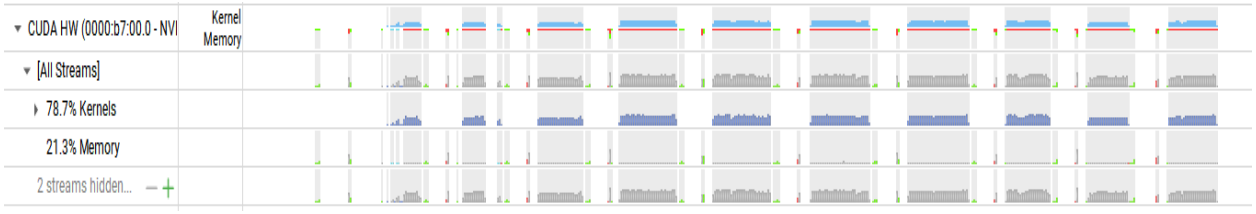

Figure 2: Visualization of memory usage of the legibility classifier

As can be seen in Figure 2, the legibility classifier is working on each test sample in serial. The pipeline's performance can be improved by parallelizing this process, therefore reducing computation time.

2.2.2.5 Performance Analysis: Training Scenario

We ran the entire pipeline as given in [3] in training mode with 8 examples and 1 epoch to get the profile given below. A smaller number of examples and only 1 epoch are used as the purpose here is to get information on the computational performance of the code and not to get a trained model.

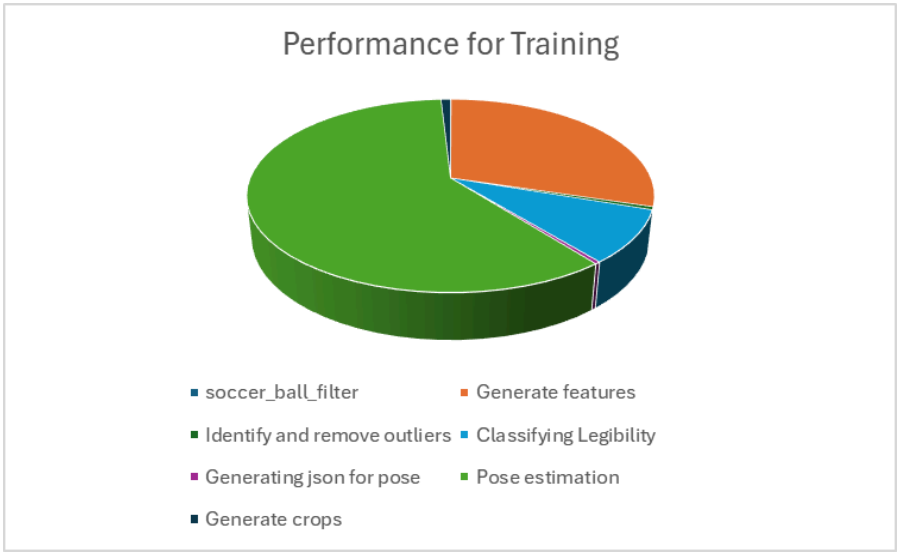

Figure 3: Pie Chart showing the baseline performance of the individual training components.

Figure 3 shows that similar to the testing scenario, the most computational resources were used during pose estimation. The next most resource-consuming component was the feature generation. After that was legibility classification. The rest of the components were not resource-heavy.

2.2.2.6 GPU Utilization Analysis

Stage Name	Duration (sec)
Pose Estimation	75.867
Generating Features	37.713
Classifying Legibility	11.187
Generating Crops	1.152
Identifying and Removing Outliers	0.628537
Generating json for pose	0.464575
Soccer Ball Filter	0.055517
Entire Pipeline	127.069

Table 4: The breakdown of runtime for each program/routine (training)

Table 5 shows how much of the GPU each program used to perform computations during the time the pipeline was run during the training scenario. Similar to the testing scenario, we can see that legibility classification, feature generation, and jersey number prediction do not use much of the GPU capacity. Similar improvements can be made to the training scenario as suggested for the testing scenario to increase pipeline performance

Stage Name	Metric	Metric Value
Classifying Legibility	GPU utilization	19.20%
	Time Taken in Kernels	80 %
	Time Taken for Memory Transfers	20 %
Generating Features	GPU utilization	17.70%
	Time Taken in Kernels	99.1 %
	Time Taken for Memory Transfers	0.1 %
Detecting Pose	GPU utilization	88.20%
	Time Taken in Kernels	99.4 %
	Time Taken for Memory Transfers	0.6 %

Table 5: GPU Utilization of each program (Training)

2.2.2.7 Challenges

Installations

We found that the code given by [3] was outdated, which required us to downgrade many of our dependencies and libraries such as Cuda. Otherwise, we had to modify the given code for it to be compatible with the newer versions of the libraries. The pose detector (VitPose) took the most time and effort to modify. Also, each part of the pipeline had to be run in separate environments for all of them to work.

System resources

As the general framework of [3] consists of many parts, running them required great amounts of computational power that not every team member had access to. This slowed down our progress more as only some members could replicate the results of the general pipeline, or test

out other alternative methods to the individual components of the pipeline.

Time taken
As mentioned previously, replicating the original framework took quite some time, which prevented us from moving forward and analyzing which parts of the pipeline could be improved to enhance the performance of the general framework.

3. Proposed Pipeline

3.1 Overview of Proposed Approach

3.1.1 Super Resolution Model

According to [13], super-resolution (SR) techniques have been successfully applied to OCR pipelines to recover fine-grained text details in noisy or low-resolution images. [13] have demonstrated that resolution enhancement can mitigate artifacts caused by poor image quality, improving character segmentation and recognition accuracy. The decision to integrate Text Gestalt [13], and DRCT [16] was based on the need to address OCR sensitivity to low-resolution text, as many models struggle with small or blurred text. SR techniques have been shown to improve readability by restoring lost details [13]. Additionally, Text Gestalt and DRCT are designed to correct slight misalignments in text, potentially providing cleaner inputs for recognition models. Given the success of SR in other computer vision applications [12], we hypothesized that improved text quality would yield higher OCR accuracy in the context of jersey number recognition.

3.1.2 Scene Text Recognition Model

The Scene Text Recognition (STR) module is designed to recognize text within cropped images generated by the pose estimation module. This is typically achieved by fine-tuning pretrained recognition models using the training set of the jersey dataset. The current state-of-the-art framework utilizes PARSeq, which has demonstrated strong out-of-the-box performance. However, ZZPM reported better results using an ensemble-based approach, where multiple STR models were combined via a voting mechanism for final recognition. Inspired by this, we propose expanding the STR module to include a diverse set of models, incorporating PARSeq alongside PPOCRV4 [29]—a more recent model that was unavailable at the time of previous reports. This ensemble strategy aims to enhance robustness and improve jersey number recognition accuracy by leveraging the strengths of multiple architectures.

Using the data augmentation techniques discussed in Section 3.3, we plan to train PARSeq and PPOCRV4 STR models on randomly augmented data to improve the robustness of STR model predictions. By introducing a broader range of augmentations, including synthetic distortions, varying lighting conditions, and font transformations, we aim to better simulate real-world scenarios where jersey numbers may appear blurred, partially occluded, or distorted. This augmentation-driven approach, combined with an ensemble learning strategy, is expected to create a more adaptable and accurate STR module, ultimately improving recognition consistency across real-world game environments and enhancing overall test accuracy.

3.1.3 Human Pose Estimation Model

The current state-of-the-art model for the pose detector on the MPII dataset is the Pose as Compositional Tokens (PCT) method that considers joint dependency, introduced in [5]. In this method, the human body is first decomposed into smaller parts called token features by a compositional encoder. Each token feature encodes a sub-structure of the pose, which is then quantized using a shared codebook. This converts them into discrete indices. The encoder, codebook, and decoder are trained together to minimize reconstruction error and ensure accurate pose representation. The encoder, codebook, and decoder are trained together by minimizing reconstruction error, ensuring highly accurate pose representation. For the classification task, the model predicts the category of each token. Finally, the decoder network uses those predictions to reconstruct the full pose. This method is robust against occlusions and works for both 2D and 3D pose estimation. Moreover, it has a faster inference speed than the ViTPose (Huge) [4].

Additionally, a year after the original ViTPose [4] method was released, it was succeeded by ViTPose++ [6]. ViTPose++ builds upon the original ViTPose by introducing knowledge factorization, which involves using task-agnostic and task-specific feed-forward networks within the transformer. This allows it to handle heterogeneous body keypoint categories more effectively. Additionally, ViTPose++ demonstrates improved transferability of knowledge between models, enabling large models to transfer knowledge to smaller ones via a simple knowledge token. This results in state-of-the-art performance across multiple datasets, including MS COCO, AI Challenger, and others, without compromising inference speed.

Given their exceptional performance in human pose estimation, ViTPose++ and the PCT method present promising alternatives to the approach utilized in the pipeline referenced in [3]. Replacing the existing method with ViTPose++ or PCT could enhance accuracy and robustness in pose estimation tasks.

3.2 Model Architecture

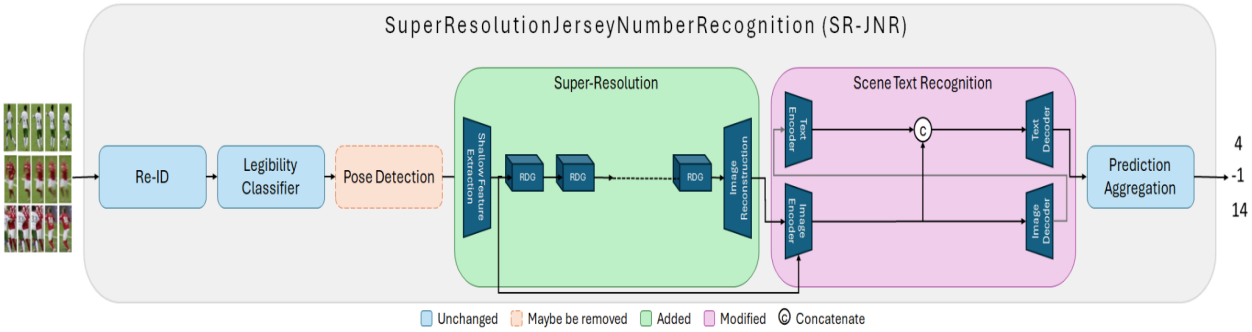

Figure 4: Proposed architecture of the modified Jersey Number Recognition pipeline (SR-JNR). The SR model follows DRCT [14], and the STR model is based on CLIP4STR [8]. However, the final model selection depends on the evaluation results.

In this section, we detail the architecture of the deep learning models employed in our pipeline, as presented in Figure 4, highlighting modifications and improvements over the baseline model described in [3]. We retain the feature extraction methodology from [3], leveraging the pre-trained ResNet50 architecture for robust feature representation and reusing the Re-ID module to preserve the ability to distinguish between different text instances effectively. To improve GPU resource utilization, as discussed in Section 2.2.2, we propose to modify the legibility classifier to reduce memory overhead and enhance parallelism during training, after which it will be frozen to ensure stability. The pose estimation component will be updated to incorporate a more efficient model as outlined in Section 3.1.3, though it may be omitted based on an ablation study and following the approach adopted by the winners of [19]. A key enhancement involves integrating a Super-Resolution (SR) model to improve text clarity, selecting the DRCT [16] model due to its superior performance compared to Text Gestalt [13], as demonstrated in Section 4.2. To address the loss of auxiliary information during compression, we plan to introduce a residual connection from the shallow feature detection stage to the visual encoder in the Scene Text Recognition (STR) model, preserving critical spatial and contextual details. Given the central role of the STR model, we will update it to utilize CLIP4STR [10], which has demonstrated SOTA performance in STR tasks through contrastive learning and multi-modal embeddings. Our training strategy involves freezing the legibility classifier and pose estimation model (if retained) while fine-tuning the SR and STR models to adapt them to the dataset. To address data imbalance, as shown in Section 3.3, we plan to replace the Cross-Entropy (CE) loss function used in [3] with a focal loss, which has proven effective in image segmentation tasks with heavy class imbalance [30], offering a more adaptive solution than the current CE-based approach with three hyperparameters. Hyperparameter optimization will focus on the SR and STR models, including learning rate schedules and balancing factors for the focal loss, using empirical evaluation. This revised architecture reflects a strategic blend of established techniques and innovative modifications, aiming to deliver a more robust and scalable solution for the jersey number detection task.

3.3 Data Preprocessing & Augmentation

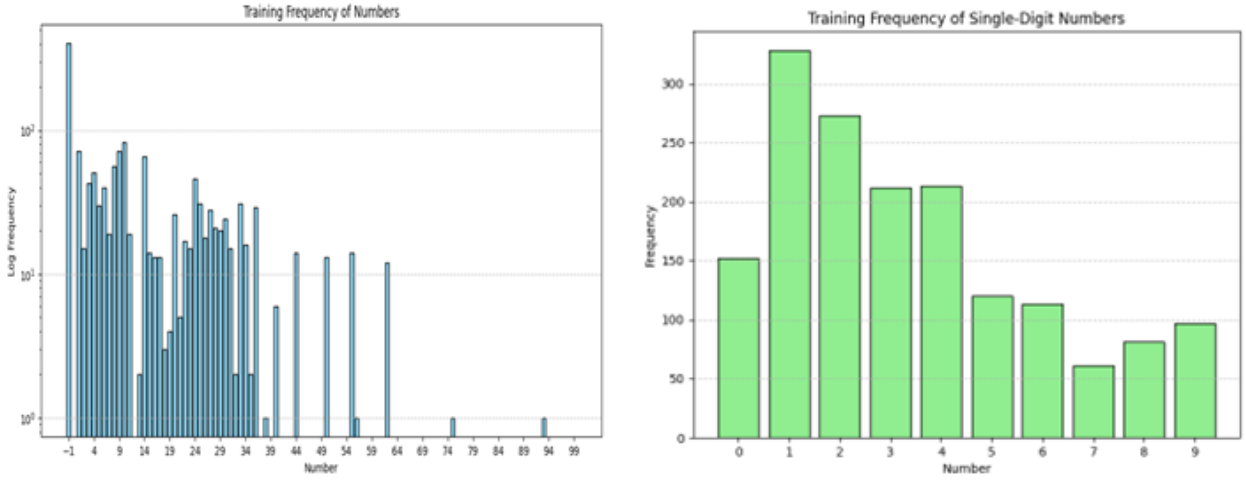

Figure 5: Distribution of the jersey numbers in the training dataset (Left) and distribution of each digit in the training dataset (Right)

The SoccerNet dataset [2], which will be used in this study, will undergo several preprocessing and augmentation steps to enhance model robustness and address challenges such as class imbalance. As shown in Figure 5, the dataset exhibits significant imbalance, not only across different classes but also within individual digits in the training set. While this imbalance could pose a challenge, prior work [31] has demonstrated that synthetic data generation, to counter this imbalance, is unnecessary, and simple data augmentation techniques can effectively improve model performance. To leverage these findings, we adopt a comprehensive data augmentation strategy informed by recent studies. For instance, [17] categorizes augmentation methods into eight groups: Warp (e.g., curved, distorted, and stretched text styles), Geometry (e.g., perspective, shrink, and rotation transformations), Noise, Blur, Weather, Camera effects (e.g., contrast, brightness, compression, and pixelation), Pattern (masking out certain regions), and Process (e.g., posterize, solarize, invert, equalize, auto-contrast, sharpness, and color adjustments). Among these, [17] highlights that models benefit most from Blur and Noise augmentations. Similarly, [31] emphasizes the effectiveness of cropping (Crop), which slightly cuts the top, bottom, left, and right ends of the text to make STR models more robust, and rotation (Rot) for handling rotated, perspective, or curved texts. Building on these insights, we plan to incorporate RandAugment [32], focusing on Noise, Rotation, and Weather-based augmentations to account for diverse environmental conditions, such as varying weather during different matches, to ensure the model generalizes well to real-world scenarios.

In addition to augmentation, we will explore semi-supervised learning techniques, depending on the time and dataset constraints, to further address this data imbalance. Specifically, following the approach in [31], we plan to consider Pseudo-label Semi-Supervised Learning, which involves three steps: (1) training the model on labeled data, (2) using the trained model to generate pseudo-labels for unlabeled data, and (3) combining labeled and pseudo-labeled data to retrain the model. This method has been shown can improve performance. Depending on time constraints and data availability, we will evaluate the feasibility of incorporating this technique to mitigate the impact of class imbalance. Overall, our preprocessing pipeline combines normalization (reusing the ImageNet dataset mean and standard deviation), targeted

augmentation strategies, and potentially pseudo-labels to ensure the model is both robust and capable of handling the inherent challenges of the dataset.

4. Proposed Enhancements & Performance Improvements

4.1 Technical details

Our proposed enhancements aim to improve accuracy, speed, and resource efficiency by addressing key bottlenecks that were described in earlier section. However, in short, to enhance speed and GPU utilization, we plan to parallelize the legibility classifier, which currently processes data sequentially (Section 3.1). For pose estimation, the large size of the ViTPose [4] model makes it inefficient for real-time applications; thus, we aim to replace it with a smaller model or potentially remove it if experiments confirm its minimal impact on accuracy (as discussed earlier). To address misclassification caused by poor image quality in STR models [13], we will incorporate an SR model and introduce a residual connection from the SR to the STR model, reducing feature loss during downsampling (Section 4.2). Finally, while ParSEQ [7] is accurate, its computational cost motivates us to explore lighter alternatives like SVTR-TINY [9] or PP-OCRv4 [8], resorting to CLIP4STR [10] if necessary since we will aim to improve accuracy before efficiency. These changes optimize performance while maintaining efficiency and scalability.

4.2 Justification

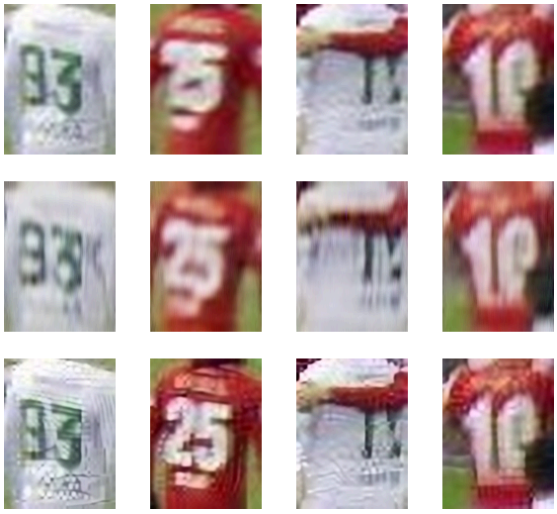

Figure 6: Samples of the input to the STR task without SR (Top), after Text Gestalt (Middle), and DRCT (Bottom)

Our design choices are motivated by the limitations identified in prior work and validated through empirical experiments. It is intuitive to expect that upgrading key components, such as the STR model and human pose estimation module, would enhance overall pipeline performance. However, to substantiate our hypothesis that incorporating a super-resolution (SR) model could improve accuracy, we conducted an experiment comparing the baseline model [3] with two SR

models: Text Gestalt and DRCT. As shown in Table 6, applying DRCT to the input data improved the test accuracy from 86.7% (baseline) to 87.1%, even without additional training. This demonstrates the potential of SR models to enhance STR performance by improving input image quality. Conversely, Text Gestalt resulted in a performance drop to 83.1%, likely due to its inability to handle the specific characteristics of our dataset, as illustrated in Figure 6. These findings confirm the importance of selecting an appropriate SR model, with DRCT proving to be better suited for our use case. Additionally, replacing the Cross-Entropy (CE) loss with a focal loss, as discussed earlier, helps to address the challenge of class imbalance in the dataset. This choice is supported by [30], which demonstrated that the focal loss helps models focus on low-frequency samples during training, improving performance in imbalanced scenarios. Similarly, our decision to adopt targeted data augmentation strategies is grounded in evidence from prior studies. For instance, [31] showed that augmentations like cropping and rotation significantly improve robustness, while [17] highlighted the benefits of noise and blur augmentations. By exposing the model to diverse variations in text appearance and environmental conditions, these techniques should enhance generalization, particularly in real-world settings where factors like weather and camera effects can degrade image quality.

Thus, our modifications—ranging from integrating DRCT for super-resolution to adopting a focal loss and advanced data augmentation—are directly informed by empirical evidence, either performed by us or other published work and address critical limitations in the baseline pipeline. These choices collectively aim to improve accuracy, robustness, and efficiency while mitigating challenges such as class imbalance and poor input quality.

SR Model	Test Accuracy	Change
Baseline (No SR)	86.7%	NA
Text Gestalt [13]	83.1%	-3.6%
DRCT [16]	87.1%	+0.4%

Table 6: Comparison of accuracy of the vanilla model and with Text Gestalt and DRCT models applied to the input data as shown in Figure 4

5. Experimental Plan & Evaluation Metrics

5.1 Experiment Design

We plan to conduct a series of experiments to evaluate the impact of our proposed modifications, beginning with an analysis of pose estimation by running the vanilla model [3] with its original pose estimation module, a variant using PCT [5], and a configuration without pose estimation to measure its effect on performance, accompanied by profiling to assess computational efficiency. To address inefficiencies in GPU usage, we will parallelize the legibility classifier and compare its performance against the vanilla model through profiling similar to Section 2.2.2. Additionally, we will validate the necessity of an SR model by testing a bicubic interpolation as a naive baseline, demonstrating that simply adding pixels does not improve text recognition, with results compared similarly to Table 6. We will also evaluate different STR models (i.e., SVTR-TINY, PP-OCRv4, CLIP4STR), noting that initial findings suggest these models require fine-tuning for fair comparisons since the ParSEQ using in [3] was finetuned. Finally, we will train and test the complete modified pipeline shown in Figure 4 to assess the cumulative impact of our enhancements. The dataset, pre-split as described in [2], will be used

for training, validation, and testing, with final results submitted on the challenge dataset. All experiments will be conducted on the same hardware setup detailed in Section 2.2.2, including an AMD EPYC 7742 64-Core Processor with 256 threads and an NVIDIA A100-SXM4-80GB GPU, ensuring consistency and reproducibility in terms of profiling.

5.2 Evaluation Metrics

To comprehensively assess the performance of our proposed pipeline, we will utilize a combination of accuracy-based and efficiency-focused metrics. The primary metric for evaluating recognition performance is Top-1 Accuracy, as it aligns with the standard reported in [4] and provides a clear measure of the model's ability to predict the correct output. Additionally, to view the impact of data imbalance highlighted earlier, we will compute Precision, Recall, and F1-Score. These metrics will help quantify the impact of data augmentation strategies and the focal loss in mitigating bias toward majority classes while improving performance on underrepresented samples. Finally, we will measure Inference Time through performance profiling to evaluate the computational efficiency of the pipeline, ensuring that our modifications strike a balance between accuracy and speed. Together, these metrics provide a holistic view of the model's effectiveness, robustness, and practicality in real-world applications.

6. References

[1] S. Giancola, M. Amine, T. Dghaily, and B. Ghanem, “SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2018, pp. 1792–179210. doi: 10.1109/CVPRW.2018.00223.

[2] A. Cioppa, A. Deliège, S. Giancola, B. Ghanem, and M. Van Droogenbroeck, “Scaling up SoccerNet with multi-view spatial localization and re-identification,” *Sci. Data*, vol. 9, no. 1, p. 355, Jun. 2022, doi: 10.1038/s41597-022-01469-1.

[3] M. Koshkina and J. H. Elder, “A General Framework for Jersey Number Recognition in Sports Video,” 2024, *arXiv*. doi: 10.48550/ARXIV.2405.13896.

[4] Y. Xu, J. Zhang, Q. Zhang, and D. Tao, “ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation,” Oct. 13, 2022, *arXiv*: arXiv:2204.12484. doi: 10.48550/arXiv.2204.12484.

[5] Z. Geng, C. Wang, Y. Wei, Z. Liu, H. Li, and H. Hu, “Human Pose as Compositional Tokens,” Mar. 21, 2023, *arXiv*: arXiv:2303.11638. doi: 10.48550/arXiv.2303.11638.

[6] Y. Xu, J. Zhang, Q. Zhang, and D. Tao, “ViTPose++: Vision Transformer for Generic Body Pose Estimation,” Dec. 14, 2023, *arXiv*: arXiv:2212.04246. doi: 10.48550/arXiv.2212.04246.

[7] D. Bautista and R. Atienza, “Scene Text Recognition with Permuted Autoregressive Sequence Models,” Jul. 14, 2022, *arXiv*: arXiv:2207.06966. doi: 10.48550/arXiv.2207.06966.

[8] Y. Du *et al.*, “PP-OCR: A Practical Ultra Lightweight OCR System,” Oct. 15, 2020, *arXiv*: arXiv:2009.09941. doi: 10.48550/arXiv.2009.09941.

[9] Y. Du *et al.*, “SVTR: Scene Text Recognition with a Single Visual Model,” arXiv, 2022. doi: 10.48550/ARXIV.2205.00159.

[10] “CLIP4STR: A Simple Baseline for Scene Text Recognition With Pre-Trained Vision-Language Model | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Feb. 26, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10816351

[11] A. Radford *et al.*, “Learning Transferable Visual Models From Natural Language Supervision,” Feb. 26, 2021, *arXiv*: arXiv:2103.00020. doi: 10.48550/arXiv.2103.00020.

[12] A. Ali and Y.-G. Kim, “Deep Fusion for 3D Gaze Estimation From Natural Face Images Using Multi-Stream CNNs,” *IEEE Access*, vol. 8, pp. 69212–69221, 2020, doi: 10.1109/ACCESS.2020.2986815.

[13] J. Chen, H. Yu, J. Ma, B. Li, and X. Xue, “Text Gestalt: Stroke-Aware Scene Text Image Super-resolution,” *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 1, Art. no. 1, Jun. 2022, doi: 10.1609/aaai.v36i1.19904.

[14] X. Chen, X. Wang, J. Zhou, Y. Qiao, and C. Dong, “Activating More Pixels in Image Super-Resolution Transformer,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 22367–22377. doi: 10.1109/CVPR52729.2023.02142.

[15] Z. Chen, Y. Zhang, J. Gu, L. Kong, X. Yang, and F. Yu, “Dual Aggregation Transformer for Image Super-Resolution,” Aug. 11, 2023, *arXiv*: arXiv:2308.03364. doi: 10.48550/arXiv.2308.03364.

[16] C.-C. Hsu, C.-M. Lee, and Y.-S. Chou, “DRCT: Saving Image Super-Resolution away from Information Bottleneck,” in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2024, pp. 6133–6142. doi:

10.1109/CVPRW63382.2024.00618.
[17] R. Atienza, “Data Augmentation for Scene Text Recognition,” Aug. 16, 2021, *arXiv*: arXiv:2108.06949. doi: 10.48550/arXiv.2108.06949.
[18] D. Zhu, L. Wang, and D. Tao, “Self-compositional data augmentation for scene text detection,” in *International Conference on Cloud Computing, Performance Computing, and Deep Learning (CCPCDL 2023)*, K. Subramaniam and S. Saxena, Eds., Huzhou, China: SPIE, May 2023, p. 67. doi: 10.1117/12.2679227.
[19] A. Cioppa *et al.*, “SoccerNet 2023 challenges results.” arXiv, 2023. doi: 10.48550/arXiv.2309.06006.
[20] Y. Sun, J. Liu, W. Liu, J. Han, E. Ding, and J. Liu, “Chinese Street View Text: Large-Scale Chinese Text Reading With Partially Supervised Learning,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019, pp. 9085–9094. doi: 10.1109/ICCV.2019.00918.
[21] B. Shi *et al.*, “ICDAR2017 Competition on Reading Chinese Text in the Wild (RCTW-17),” Sep. 26, 2018, *arXiv*: arXiv:1708.09585. doi: 10.48550/arXiv.1708.09585.
[22] M. He *et al.*, “ICPR2018 Contest on Robust Reading for Multi-Type Web Images,” *2018 24th Int. Conf. Pattern Recognit. ICPR*, pp. 7–12, Aug. 2018, doi: 10.1109/ICPR.2018.8546143.
[23] W. He, X.-Y. Zhang, F. Yin, and C.-L. Liu, “Multi-Oriented and Multi-Lingual Scene Text Detection with Direct Regression,” *IEEE Trans. Image Process. Publ. IEEE Signal Process. Soc.*, Jul. 2018, doi: 10.1109/TIP.2018.2855399.
[24] Z. Huang *et al.*, “ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction,” in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, Sep. 2019, pp. 1516–1520. doi: 10.1109/ICDAR.2019.00244.
[25] N. Nayef *et al.*, “ICDAR2019 Robust Reading Challenge on Multi-lingual Scene Text Detection and Recognition -- RRC-MLT-2019,” Jul. 01, 2019, *arXiv*: arXiv:1907.00945. doi: 10.48550/arXiv.1907.00945.
[26] D. Karatzas, S. R. Mestre, J. Mas, F. Nourbakhsh, and P. P. Roy, “ICDAR 2011 Robust Reading Competition - Challenge 1: Reading Text in Born-Digital Images (Web and Email),” in *2011 International Conference on Document Analysis and Recognition*, Sep. 2011, pp. 1485–1490. doi: 10.1109/ICDAR.2011.295.
[27] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, “Real-time Scene Text Detection with Differentiable Binarization,” Dec. 03, 2019, *arXiv*: arXiv:1911.08947. doi: 10.48550/arXiv.1911.08947.
[28] Z. Xu *et al.*, “Towards End-to-End License Plate Detection and Recognition: A Large Dataset and Baseline,” in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Cham: Springer International Publishing, 2018, pp. 261–277. doi: 10.1007/978-3-030-01261-8_16.
[29] “PP-OCRv4技术报告 - PaddleOCR Documentation.” Accessed: Feb. 27, 2025. [Online]. Available: https://paddlepaddle.github.io/PaddleOCR/main/en/ppocr/blog/PP-OCRv4_introduction.html
[30] B. H. Tran, T. Le-Cong, H. M. Nguyen, D. A. Le, T. H. Nguyen, and P. L. Nguyen, “SAFL: A Self-Attention Scene Text Recognizer with Focal Loss,” in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2020, pp. 1440–1445. doi: 10.1109/ICMLA51294.2020.00223.
[31] J. Baek, Y. Matsui, and K. Aizawa, “What If We Only Use Real Datasets for Scene Text Recognition? Toward Scene Text Recognition With Fewer Labels,” Jun. 05, 2021, *arXiv*: arXiv:2103.04400. doi: 10.48550/arXiv.2103.04400.

[32] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “RandAugment: Practical automated data augmentation with a reduced search space,” Nov. 14, 2019, *arXiv*: arXiv:1909.13719. doi: 10.48550/arXiv.1909.13719.

7. Timeline & Milestones

Our estimated timeline is summarized in Table 7, which provides a clear breakdown of tasks and deadlines.

Milestone	Description	Estimated Completion Date
Search for (Unlabeled) Data	Search for unlabeled datasets for pseudo-label semi-supervised learning	March 10
Experiment with Pose Estimation	Test the aforementioned Human Pose Estimation to focus on the final model pipeline architecture	March 10
Experiment with STR	Test the aforementioned STR to focus on the final model to use in the pipeline	March 10
Experiment with Bicubic Interpolation	Test the bicubic interpolation to provide concrete results for the need for SR models	March 10
Implement the different data augmentation methods	Implement the different data augmentation methods aforementioned with [15] code	March 10
Implement Proposed Pipeline	Implement enhancements: parallelization, pose estimation analysis, and focal loss	March 17
Train and Test the Implemented Pipeline	Train and test the implemented model from the previous task	March 24

Table 7: Summary of the different milestones and predicted timelines for project completion.

8. Team Contributions

This project results from a collaborative effort, with each team member contributing to specific aspects of the work. Below is a detailed breakdown of each member's roles and responsibilities:

- Ghaith: Focused on pipeline evaluation, including testing and integrating STR and SR models into the workflow. Improved the base model's performance through experimentation and optimization. Contributed to the report writing and formatting and conducted a portion of the literature review to support the theoretical foundation of the project.
- Tanmaya: Concentrated on pipeline evaluation and performance profiling, ensuring that computational efficiency and resource utilization were thoroughly analyzed. Assisted in drafting sections of the report related to experimental results and profiling outcomes.
- Karel: Was responsible for testing and experimenting with various STR models, comparing their performance, and fine-tuning them for optimal results. Contributed to documenting these experiments and their implications in the final report.
- Hamza: Led the literature review process, identifying key papers and summarizing relevant findings to guide the project's methodology. Played a significant role in structuring and writing sections of the report to ensure clarity and coherence.
- Amey: Specialized in performance profiling, analyzing the runtime and resource consumption of different pipeline components alongside Tanmaya. Provided actionable insights to improve the computational efficiency of the system.

8.1 GitHub Repository

- Our team's codebase is hosted on GitHub.

8.2 Our results are distributed as follows:

- Performance profiling results can be found here and here.
- Vanilla pipeline evaluation results can be found here.
- Text Gestalt output can be found here.
- DRCT output can be found here.
