BC-JNR+: Bicubic Interpolation-Jersey Number Recognition Without Human Pose Estimation

Ghaith Chrit
Computer Science, CMPS, I. K. Barber
Faculty of Science
UBC Okanagan
Kelowna, Canada
ghaith.chrit@ubc.ca

Tanmaya Karmarkar
Computer Science, CMPS, I. K. Barber
Faculty of Science
UBC Okanagan
Kelowna, Canada
tanmayak@student.ubc.ca

Hamza Muhammad Anwar
Computer Science, CMPS, I. K. Barber
Faculty of Science
UBC Okanagan
Kelowna, Canada
hanwar02@student.ubc.ca

Karel Joshua Harjono
Computer Science, CMPS, I. K. Barber
Faculty of Science
UBC Okanagan
Kelowna, Canada
harjono@student.ubc.ca

Amey Karmarkar
Computer Science, CMPS, I. K. Barber
Faculty of Science
UBC Okanagan
Kelowna, Canada
akarmark@student.ubc.ca

***Abstract*—The SoccerNet dataset has been used for several challenges over the past few years, one of which was the jersey number recognition challenge. In this task, participants developed deep learning models to accurately detect and recognize soccer players' jersey numbers from tracklets extracted from real match footage. One team's pipeline for the challenge included a legibility classifier, pose detector, scene text recognition module, and prediction aggregation, achieving a test accuracy of 86.7% and a challenge accuracy of 79.31%. Our work takes this pipeline as a baseline and improves on it by removing the pose detector, which was a performance bottleneck, and adding data augmentation before a new super-resolution module. These changes led to a higher challenge accuracy of 81.70%. In addition to improved accuracy, our pipeline is significantly more optimized, running 1.69 times faster than the original.**

Keywords—SoccerNet, Jersey Number Recognition, Scene Text Recognition, Super Resolution, Human Pose Estimation, Deep Learning, Optimization, Data Augmentation, bicubic interpolation

I. Introduction

SoccerNet [1] is a large-scale dataset that contains images, videos, and labels for soccer video research. Since its release 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.

II. Literature Review

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]. Meanwhile, 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, VitPose++ [5] addresses heterogeneity in the body keypoints while also increasing the throughput and accuracy of the model compared to VitPose.

Once the jersey numbers are spatially enhanced and preprocessed, scene text recognition (STR) techniques are applied to extract the actual numerical information from the image. These techniques are a critical part of the pipeline as they are responsible for interpreting raw pixel data and converting it into meaningful alphanumeric characters that represent player jersey numbers. One of the widely used approaches for STR involves large Transformer-based models, which have consistently demonstrated high accuracy in recognizing text from complex and cluttered natural scenes. Among these, the PARSeq [6] model has emerged as a strong performer due to its autoregressive decoding mechanism and ability to learn character dependencies in sequence modeling tasks.

However, while PARSeq [6] and similar large models excel in accuracy, they often come with increased computational requirements, making them less suitable for deployment in real-time or resource-constrained environments. For scenarios that demand faster inference and lower latency, more efficient and lightweight STR models are typically preferred. One such model is PP-OCRv4-rec [7], an optimized OCR framework that is designed specifically for practical, real-world deployments. This model achieves a favorable trade-off between accuracy and speed, making it highly suitable for mobile applications, embedded systems, and real-time processing pipelines. Another lightweight alternative is SVTR-TINY [8], a compact, Transformer-based model that significantly reduces computational overhead while still retaining strong recognition performance. It is especially attractive for use cases where power consumption, memory footprint, or processing time is limited, such as on lightweight devices or live broadcast overlays. In contrast to these models, CLIP4STR [9] transforms CLIP [10] 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 [11], it is reasonable to assume that a similar result would be found in OCR and STR tasks. Additionally, [12] showed that SR techniques can enhance OCR pipelines by recovering fine-grained text details and mitigating artifacts caused by poor image quality. Text Gestalt [12] 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 [13], [14], DRCT [15] 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 [16] 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 augmentations, [16] was able to improve the performance of baseline STR models without changing the models' architecture. Complementing these approaches, [17] 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 [18], where the winning team heavily relied on data augmentation to improve model robustness and achieve state-of-the-art performance.

III. System design

Our jersey number recognition system is designed as a modular pipeline composed of several sequential stages: frame extraction, image enhancement via super-resolution, scene text recognition (STR), and post-processing. Each stage is tailored to address a specific challenge posed by the SoccerNet dataset, which contains low-resolution tracklets of players in motion. As mentioned previously, the dataset is particularly difficult to process due to the poor quality of many frames, heavy motion blur, limited visibility of the jersey number, and lack of consistent spatial cues. To handle these challenges effectively, our goal was to build a pipeline that could accurately and reliably extract jersey numbers from these video tracklets without relying on human pose estimation, which is computationally expensive and prone to failure under occlusion or unusual poses.

A. *STR Selection*

A major component in our system, and arguably the most critical to the overall performance, is the scene text recognition module. STR is responsible for interpreting the visual content of the frame and extracting a sequence of digits corresponding to the player's jersey number. Our initial implementation was built using PARSeq, a Transformer-based STR model that was also used in the baseline model of the SoccerNet Jersey Number Recognition Challenge. This gave us a strong starting point, as PARSeq had already demonstrated strong performance on the task. PARSeq uses an autoregressive decoding mechanism to predict characters one at a time, leveraging context from previous character predictions to improve accuracy. Its design makes it particularly effective in scenarios where text is partially occluded, warped, or low contrast, conditions that are frequently encountered in soccer footage. Given these strengths and its proven track record on the dataset, we adopted PARSeq as our baseline model and integrated it into our STR stage early on.

Although PARSeq showed strong results, we wanted to explore whether more recent off-the-shelf OCR models could outperform it in terms of recognition accuracy or processing efficiency. As the STR field continues to evolve rapidly, we identified two lightweight candidates for evaluation: PP-OCRv4-rec and SVTR-TINY. PP-OCRv4-rec is designed for practical, real-time applications, with a pipeline that integrates text detection and recognition using highly optimized components. SVTR-TINY, on the other hand, is a compact version of a Transformer-based model that emphasizes fast inference and low resource consumption, making it suitable for deployment on edge devices or in low-latency settings. Both models have been shown to perform well on standard OCR benchmarks, and we hypothesized that they might provide improvements over PARSeq, especially when deployed in high-performance computing environments.

To compare these models fairly, we substituted each into our pipeline in place of PARSeq and conducted a series of controlled experiments. Each model was evaluated on the same set of test tracklets, using identical preprocessing steps, augmentation strategies, and output filtering methods. Our aim was to determine whether either model could generalize better to the noisy, low-quality conditions found in SoccerNet videos and whether any speed advantages were worth potential accuracy trade-offs. The results were clear: neither *PP-OCRv4-rec* nor *SVTR-TINY* outperformed *PARSeq*, even in their default, pre-trained configurations. As shown in Table I, PARSeq delivered higher accuracy than both alternatives across multiple subsets of the validation set, particularly in frames with motion blur, partial occlusion, and low contrast. The newer models also exhibited lower confidence scores and a higher rate of digit misclassification.

TABLE I. COMPARISON OF DIFFERENT OCR MODELS BASED ON ACCURACY AND INFERENCE TIME

Model	Accuracy	Inference Time (sec)
PP-OCR-v4-rec	75.49%	1074.2
SVTR-TINY	57.77%	880.4
PARSeq	85.4%	N/A

Based on these findings, we made the decision not to proceed further with fine-tuning PP-OCRv4-rec or SVTR-TINY. Instead, we concentrated our efforts on fine-tuning PARSeq, given that it already outperformed the other models in its out-of-the-box configuration. We hypothesized that with domain-specific augmentation, simulating various environmental effects like zoom blur, motion blur, and lighting distortion, PARSeq could achieve even greater accuracy. This strategy proved successful, as described in the Results section, where we show that fine-tuning PARSeq using augmented SoccerNet samples led to a significant improvement in challenge performance. By focusing on a proven architecture and adapting it to the specific challenges of our dataset, we were able to build an STR module that is both robust and reliable for jersey number recognition.

B. Data Augmentation

During evaluation, we observed that the baseline PARSeq model, though effective on clean images, struggled to generalize to real-world soccer footage where jersey numbers are often affected by motion blur, compression, poor lighting, and environmental artifacts. To address this, we developed a comprehensive data augmentation pipeline to improve the model's robustness and recognition accuracy under such conditions following the STRAug [16] implementation.

We first focused on simulating blur, which is common in fast-paced sports footage. Gaussian blur and defocus blur were used to replicate soft focus and lens misalignment, while motion blur and zoom blur simulated player or camera movement. Glass blur was added to emulate distortion from dirty or fogged lenses. These augmentations aimed to challenge the model's ability to recognize digits in degraded visual conditions.

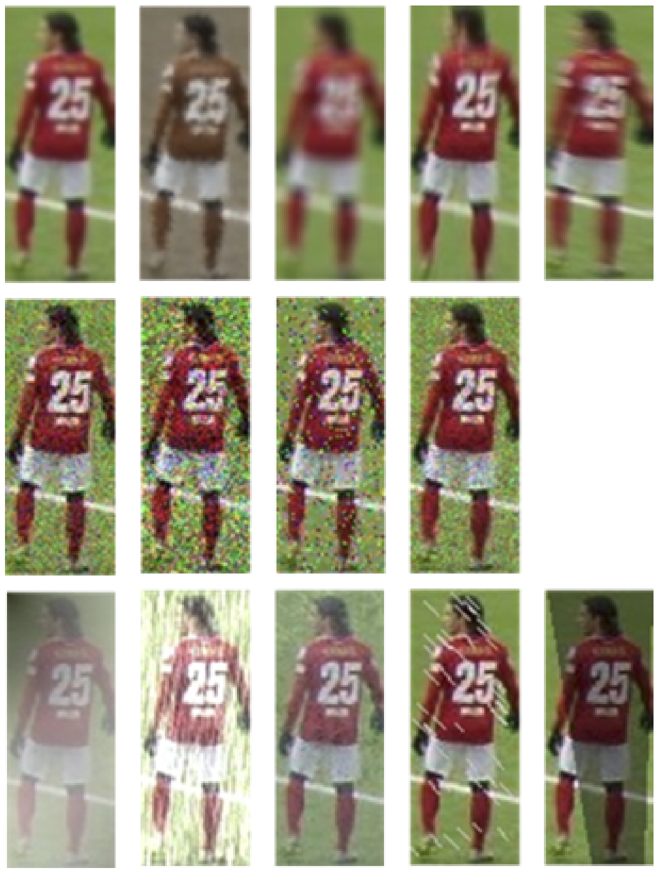


Fig. 1. Different STR-specific data augmentation techniques. The top row is the different blur augmentations, followed by noise, and finally by different weather conditions.

We introduced several types of noise. Gaussian noise and shot noise mimicked sensor-level imperfections, particularly under low-light conditions. Impulse noise, or salt-and-pepper noise, was used to simulate digital corruption, while speckle noise introduced multiplicative texture noise. These noise types helped the model learn to distinguish jersey numbers from visual artifacts that could resemble character strokes. Moreover, to simulate external environmental conditions, we applied weather-based effects. Fog reduces contrast and visibility, snow and frost add bright or semi-transparent overlays, and rain introduces vertical streaks and blur. Shadows were simulated by darkening image regions to reflect uneven field lighting or player occlusion. An example of these data augmentations is shown in Figure 1.

These augmentations were applied randomly during fine-tuning to expose the model to a broad range of visual challenges. Each training batch included both clean and augmented samples, helping the model generalize beyond its original training distribution. As a result, the fine-tuned PARSeq model demonstrated improved accuracy on difficult frames.

C. Super Resolution

We noticed that the input-output of the pose detection (and cropping) step and the input to the STR model had very different image sizes (35×44 compared to 128×32). Thus, we attempted to apply super-resolution (SR) models to bridge this gap. We chose DRCT [15] due to its effective use of residual connections, which help preserve critical information during the upsampling process, and TextGestalt [12], which is specifically designed for text-focused tasks and aims to enhance textual details in low-resolution images.

TABLE II. COMPARISON OF VARIOUS TECHNIQUES TO IMPROVE THE RESOLUTION OF THE INPUT IMAGE TO THE STR MODEL

SR Model	Test Accuracy	Change
Baseline (No SR)	86.7%	NA
Text Gestalt [13]	83.1%	-3.6%
DRCT [16]	87.1%	+0.4%
Bicubic Interpolation	87.2%	+0.5%

As shown in Table II, applying DRCT increased the final model accuracy by 0.4%, while TextGestalt led to a significant reduction in accuracy by 3.6%. To understand these results, one can refer to Figure 2, taken from [15], which illustrates how the information flow, represented by feature map intensities, diminishes sharply at the end of multiple SR networks, indicating potential information loss—a phenomenon that likely extends to TextGestalt. Although time constraints prevented us from running the same experiment on TextGestalt, it is evident that DRCT, thanks to its residual connections, avoids such issues.

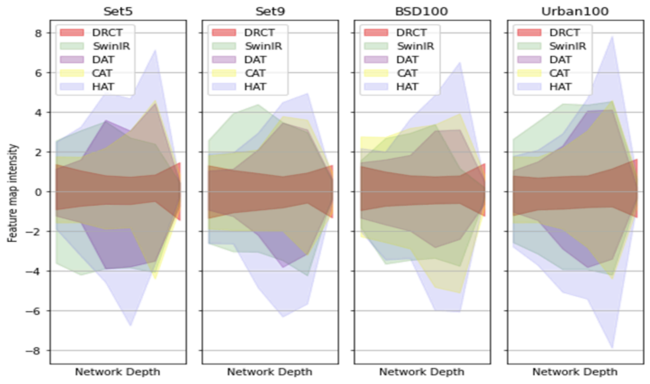


Fig. 2. Information flow as the network depth increases in various SR models. Figure from [15].

Additionally, this difference in performance is further reflected in the outputs of these SR models, as seen in Figure 3. Here, DRCT's output exhibits fewer artifacts and sometimes even functions as a deblurring mechanism, enhancing image clarity and quality. Since we wanted to mainly focus on increasing the efficiency of the pipeline, it seemed counterintuitive to add a new compute-intensive model into the workflow. Taking inspiration from the concept of supersampling in computer graphics, where higher-resolution rendering is used to reduce aliasing and artifacts before downsampling, we decided to upscale the input image to an intermediate size and then downscale it to the required STR input size. This approach was adopted to minimize artifacts while maintaining computational efficiency. As can be seen in Table II, this method improved performance even more than DRCT. Although the STR model is a black-box model, we can gain insight into this improvement by examining an instance where DRCT misclassified the input compared to bicubic interpolation.

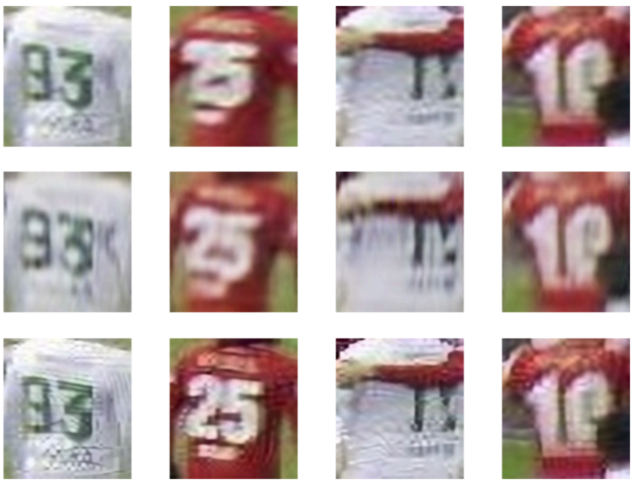


Fig. 3. Samples of the images inputted to the STR model without SR (Top), after Text Gestalt (Middle), and DRCT (Bottom).

As shown in Figure 4, DRCT, due to the warping of the shirt, altered the digit "1" to resemble a "7," causing the classification to change from "11" to "17." This issue did not occur with bicubic interpolation when applied in the technique mentioned earlier, which preserved the integrity of the original input. While these results are impressive on their own, it is important to note that applying these SR models without fine-tuning represents a domain generalization challenge, which is one reason for their relatively lower performance.

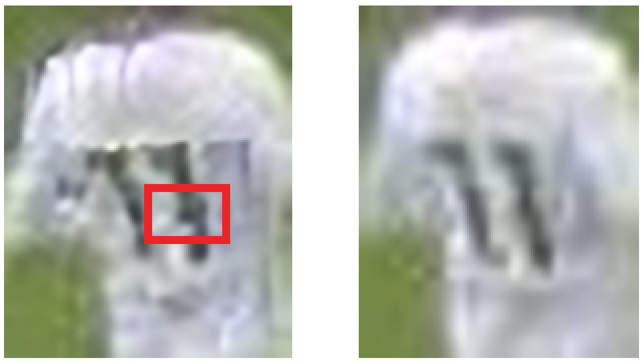


Fig. 4. A sample image where the DRCT (left) introduced an artifact that caused the jersey number to look like 17, while the interpolation technique (right) did not introduce the artifact.

D. Fine-tuning Hyperparameters

When fine-tuning the PARSeq model, we performed some hyperparameter tuning, with the most prominent change being the shift from the 1cycleLR scheduler [19] to Cosine Annealing [20] with a linear warmup. While this approach exhibited slower improvement initially, it prevented the learning rate from becoming too small too quickly, which would have necessitated training over an excessive number of epochs. This was because the 1cycleLR aggressively reduces the learning rate after the initial phase, often leading to slow convergence in later stages when adjustments are still needed. In contrast, Cosine Annealing with a linear warmup provides a more gradual and controlled decay, allowing the model to sustain meaningful updates throughout training without requiring an impractically large number of iterations.

E. *Performance Improvements*

We ran the entire pipeline on 802 frames in order to generate the performance analyses. We cannot run performance analyses on the entire dataset, as the files generated are too big and cannot be transferred and viewed on a laptop. Ideally, such experiments are run only on one iteration, but as this is a pipeline and as each program is doing something different, we decided to use more frames in order to get a better picture.

In order to get the performance, we used Nvidia Nsight systems along with nvtx markers. You can use nvtx markers by installing the nvtx library. We put nvtx markers for the entire program and for individual components/ programs in the pipeline. Based on the output of the first profile, we could add more markers where we thought we needed more information.

Table III gives us the breakdown of each individual component.

TABLE III. BREAKDOWN OF TIMING FOR INDIVIDUAL COMPONENTS OF THE PIPELINE.

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

We then plotted a pie chart to visualize the results better, which is shown in Figure 5.

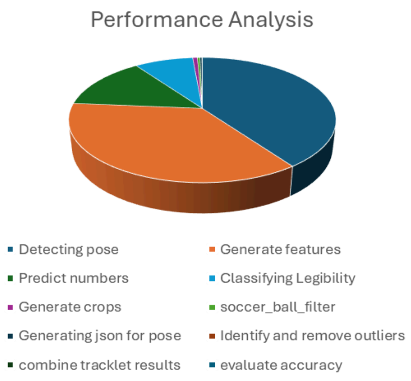


Fig. 5. Timings for 802 frames run in testing mode.

Examining the profiles obtained on the initial pipeline, we observe that four programs take a major chunk of the time. All four programs have been ported onto GPU's but further analyses showed that GPU utilization is low, so we are not effectively using the computational power.

TABLE IV. GPU UTILIZATION OF EACH COMPONENT ALONG WITH COMPUTE / MEMORY BREAKUP

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 %

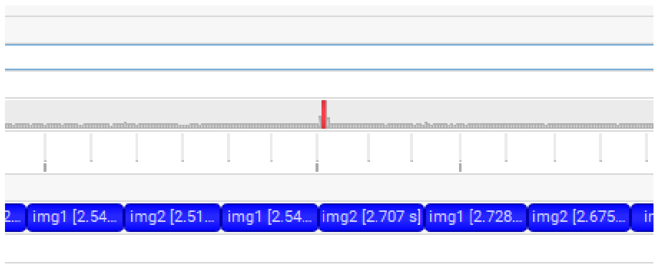


Fig. 6. Profile showing each tracklet is processed in serial.

The GPU utilization obtained showed that the pose estimation model was using resources well and that we would not be able to gain much more improvement by optimizing it. As pose detection is not required in the BC-JNR+ model, we modified the pipeline and obtained good performance.

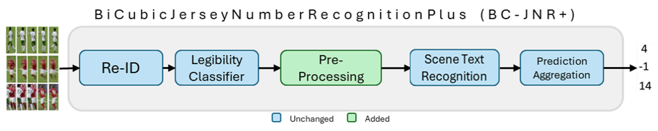


Fig.7. The BC-JNR+ pipeline after removing the pose detector.

The other three programs showed utilization below 30%. Hence, we decided to profile the code to generate further details on the GPU utilization. In this code, we can see that each frame is processed serially, and processing on that frame is done on the GPU.

The profile shows that each tracklet is being processed serially. Hence, we decided to experiment with streams. In order to use streams, we needed to use Numba, and thus, we had access to CUDA streams. By doing so, we could overlap the processing of tracklets.

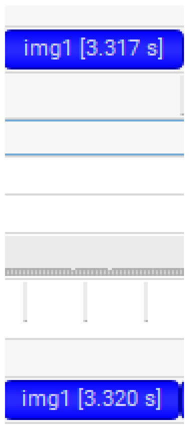


Fig. 8. Running tracklets in parallel.

The reason for using streams was to overlap copy / compute operations in order to get better utilization of GPU resources.

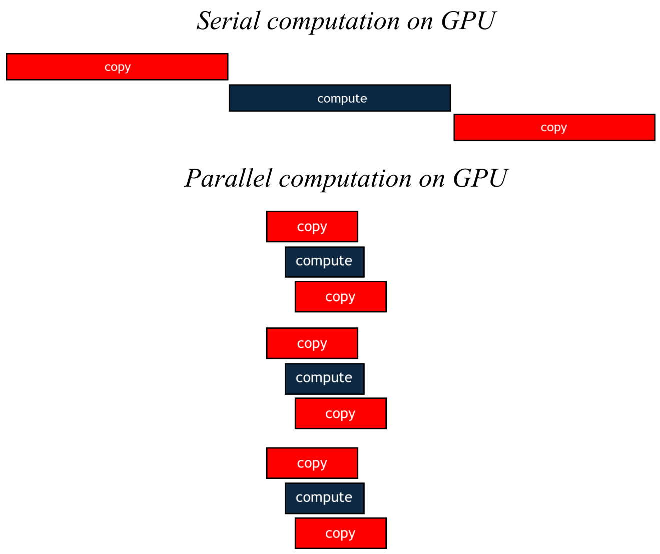


Fig. 9. Serial computation of copy/compute converted to parallel using streams.

Further, using the numba library, we could optimize numpy arrays to work in parallel and also on the GPU. By default numpy works on CPU.

a) NVMath-python

nvmath-python (Beta) [21] is an open-source Python library, providing Python programmers with access to high-performance mathematical operations from NVIDIA CUDA-X math libraries. nvmath-python provides both low-level bindings to the underlying libraries and higher-level Pythonic abstractions.

We decided to experiment with the NVMath library because it offers high-performance, GPU-accelerated operations that go beyond what NumPy typically provides. NVMath gives us direct, pythonic access to NVIDIA's optimized math libraries, enabling near-native performance for complex operations like advanced matrix multiplications and FFTs (Fast Fourier Transforms). With its ability to fuse kernel operations (such as incorporating bias and scaling into matrix multiplications) into single calls, it minimizes overhead and makes our computational workflows significantly more efficient.

Moreover, NVMath is built to harness the power of GPUs and parallel processing, allowing us to leverage NVIDIA hardware to perform computations concurrently at a massive scale. Its integration with GPU-based libraries like CuPy, PyTorch, and RAPIDS—as well as interoperability with traditional CPU-based tools such as NumPy and SciPy—means we can maintain familiar data structures while accelerating performance. This parallelized approach not only boosts speed for deep learning and data processing tasks but also provides the flexibility to fuse and optimize operations using device callback functions, making it a superior alternative to the regular NumPy library.

b) Fusing Epilog Operations with Matrix Multiplication Using nvmath-python.

The implementation using the RELU_AUX_BIAS epilog is faster than its naive counterpart, providing a significant performance gain, as shown in the following benchmark.

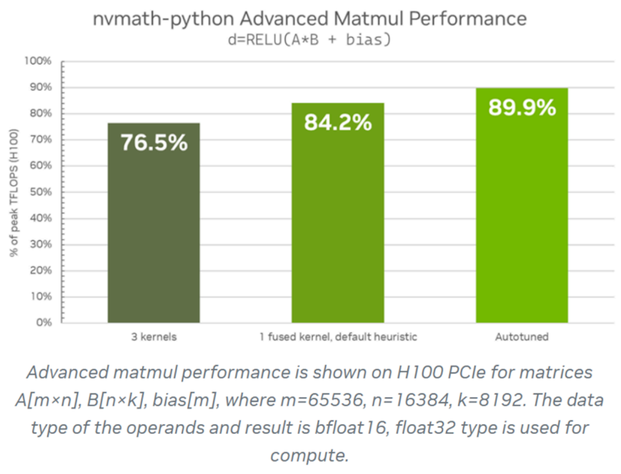
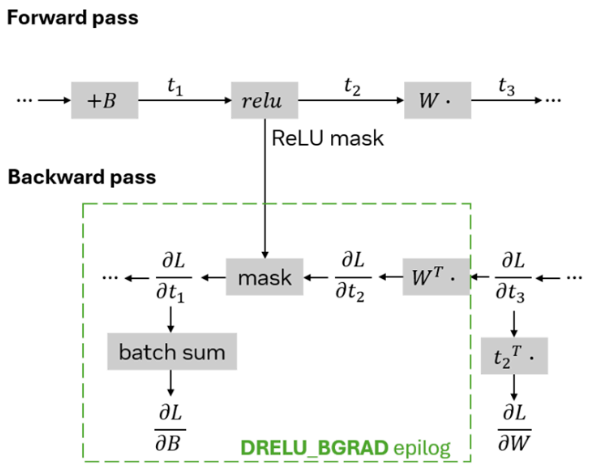
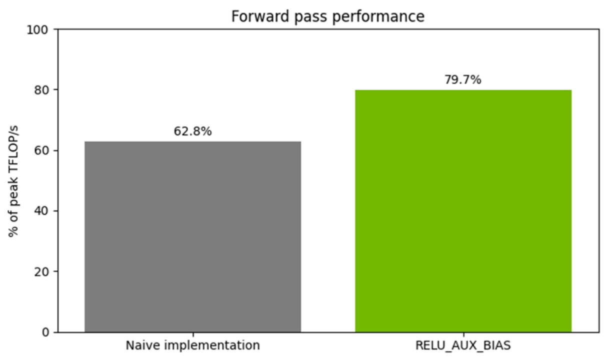


Fig.10. Performance using fused relu-matmul-bias.

As our training modules involve both forward and backwards passes, replacing numpy with nvmath seems to give us a speedup while replicating the same results.


Forward pass
+B
t_1
relu
t_2
$W \cdot$
t_3
ReLU mask
Backward pass
$\frac{\partial L}{\partial t_1}$
mask
$\frac{\partial L}{\partial t_2}$
$W^T \cdot$
$\frac{\partial L}{\partial t_3}$
batch sum
$t_2{}^T \cdot$
$\frac{\partial L}{\partial B}$
DRELU_BGRAD epilog
$\frac{\partial L}{\partial W}$



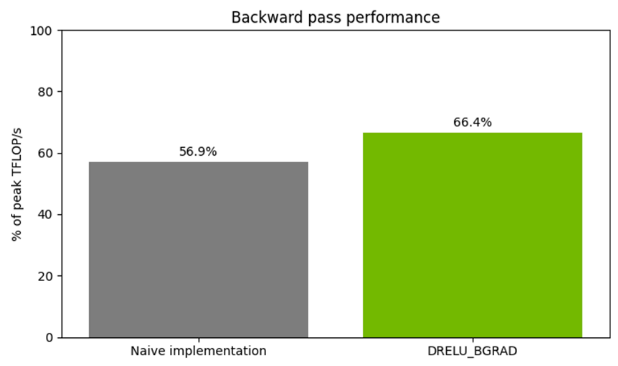


Fig. 11. Comparison of forward and backward pass (nvmath/numpy).

We were able to update the legibility classifier by replacing numpy with nvmath and improve the performance of training.

c) Bottleneck

The main bottleneck in the pipeline is that each individual component is reading the image and doing some processing. Then, the next program does the same. Hence, there are a lot of I/O operations involved. In the testing phase, if we could process the pipeline differently by keeping the images in the GPU memory and doing multiple steps, we could gain substantially better performance. In order to get this performance, we would have to fuse the programs and change the order of computations in order to gain the effects of the resources available..

IV. Results

Table V: Final performance comparison on the challenge dataset

	Accuracy	Performance/speed up	Inference Time (803 Frames)
Baseline	79.31%	–	187.36 *sec*
BC-JNR+ (ours)	81.70%	1.69X	133.15 *sec*

After fine-tuning the PARSeq model as part of our proposed pipeline, which we refer to as BC-JNR+, and as mentioned in earlier sections, we tested its performance on the challenging split of the SoccerNet Jersey Number Recognition Task. We compared our results to the original model by Koshkina and Elder [3]. As can be seen in Table V, the accuracy improved by 2.39%, while the analytical evaluation demonstrated that our pipeline was 1.69 times faster. Furthermore, experimental testing on 803 frames revealed an even more pronounced speedup, with our method being 28.93% faster in practice. These findings underscore the effectiveness of BC-JNR+ in achieving both higher accuracy and greater computational efficiency.

V. Discussions

The results presented in Section IV confirm our assumption that the baseline model struggled to generalize effectively on unseen data, as evidenced by the significant drop in performance across splits: a validation accuracy of 95~99%, a test accuracy of 86.7%, and a challenge accuracy of 79.31%. By incorporating diverse data augmentation techniques and training on the entire dataset with an 80/20 train/validation split, we observed notable improvements. This was particularly beneficial since STR-specific data augmentation techniques were applied in approximately 90% of cases, reducing overfitting to the training distribution. Our findings align with the results reported in the STRAug paper [16], which demonstrated that blurring and noise augmentations generally enhance the performance of STR models. Additionally, our assumption that players may compete in varying weather conditions motivated the use of weather-based data augmentation. However, this hypothesis requires further validation through an ablation study to determine the individual contributions of each augmentation technique.

Our results also indicate that pose detection is unnecessary and that removing it improves both performance and accuracy. Nevertheless, a comprehensive ablation study is needed to fully assess the impact of this removal. Furthermore, fine-tuning the model became essential after eliminating ViTPose, as the model had previously learned to rely on the jersey number being consistently centered in the frame. Without fine-tuning, the accuracy on the test split dropped drastically to approximately 44%, underscoring the importance of adapting the model to the new input distribution. These insights highlight the need for systematic experimentation to optimize the pipeline further and ensure robust generalization across diverse scenarios.

Additionally, while the streaming component of our pipeline contributed to a slight increase in performance, its benefits were offset by the substantial number of I/O operations in the pipeline. These operations significantly impacted the overall performance, overshadowing the gains achieved through optimizations like nv-math. This trade-off between intermediate saving of the results and computational overhead highlights the challenges of balancing efficiency with the need for transparency in monitoring model behavior and identifying potential bugs in real-time applications. Further exploration is needed to minimize I/O bottlenecks while preserving the accuracy improvements brought by streaming, potentially through more efficient data handling or parallel processing strategies.

VI. Future Work

Due to time and resource constraints, we were unable to test out other modifications to try to make further performance and accuracy improvements. However, we have many ideas that we believe will better our pipeline and excel its performance to new heights. These ideas will be listed and elaborated on below.

A. End-to-end fine-tuning

One idea that can be implemented is to try finetuning the Super-Resolution (bicubic interpolation) and Scene Text Recognition modules end-to-end. Because we obtained an increased accuracy by fine-tuning the PARSeq STR module, there is a high possibility that doing the same with the SR module will give better results as well. This approach will allow the SR module to learn task-specific features that are directly beneficial to the STR module, leading to more accurate and efficient jersey number recognition [22]. End-to-end training holds several advantages that will be detailed below:

1) Task-Specific Feature Learning: Traditional SR methods focus on general image quality enhancement, which may not align with the specific needs of text recognition. By training the SR and STR modules together, the SR component can learn to emphasize features that are most

relevant for text recognition, such as character edges and stroke patterns. This targeted enhancement facilitates better recognition outcomes. [23]

2) Error Propagation Mitigation: In a sequential pipeline where SR and STR are trained separately, errors from the SR stage can propagate and amplify in the STR stage. End-to-end training allows the system to adjust both modules simultaneously, reducing the compounding of errors and improving overall robustness.

3) Optimized Resource Utilization: Joint training can lead to more efficient models by eliminating redundant computations and focusing resources on features that contribute directly to the end goal of accurate text recognition. This can result in faster inference times and lower computational costs.

By adopting an end-to-end training strategy for the SR and STR modules, the system can achieve a more cohesive and effective enhancement of low-resolution text images, leading to superior recognition performance.

B. Replace PARSeq with CLIP4STR

Another modification that can be made to increase our bicubic jersey number recognition pipeline's accuracy is to replace the current STR module (PARSeq) with an STR technique that achieved SOTA status on many datasets, CLIP4STR.

Integrating CLIP4STR into our jersey number recognition pipeline could significantly enhance accuracy. CLIP4STR leverages pre-trained vision-language models to improve scene text recognition (STR). Unlike traditional STR methods that rely solely on visual features, CLIP4STR incorporates both visual and linguistic information, resulting in more robust text recognition [9].

CLIP4STR has demonstrated state-of-the-art performance across multiple STR benchmarks. For instance, models like the CLIP4STR-B and CLIP4STR-L achieve remarkable accuracies on the COCO-Text, CUTE80, HOST, and Street View Text (SVT) datasets. CLIP4STR-L (DataComp-1B) achieved #1 Global Rank on the CUTE80 dataset. This indicates its effectiveness in recognizing text in natural scenes, which is directly relevant to our task of jersey number recognition [24].

By replacing the current PARSeq STR module with CLIP4STR, we can capitalize on its advanced architecture that combines visual and linguistic cues. This integration is expected to improve the model's ability to accurately recognize jersey numbers, even in challenging conditions such as varying lighting, angles, or partial occlusions. Implementing CLIP4STR could thus be a strategic enhancement to our recognition pipeline, potentially leading to more reliable and accurate results.

C. Pose Detection Module

In our current BC-JNR+ pipeline, we removed the pose detection module, ViTPose. This is because it was one of the modules that took the most time to run and acted as a bottleneck, so removing it optimized our pipeline and significantly increased its performance. However, there is a possibility that reincorporating it or another human pose estimation module into the pipeline can improve the performance after fine-tuning the SR and STR modules end-to-end.

Several new pose detection techniques have been developed since ViTPose was first released, such as its successor, ViTPose++, or others, such as the human pose as compositional tokens (PCT) method. Adding any of these before the SR module may enhance its performance once the player's position in the image is determined.

For example, ViTPose++ builds upon the original ViTPose architecture, employing plain, non-hierarchical vision transformers as backbones to extract features for human pose estimation. This model demonstrates remarkable scalability, ranging from approximately 20 million to 1 billion parameters, balancing throughput and performance effectively. Its flexibility in attention mechanisms, input resolutions, and training strategies makes it a versatile choice for various pose estimation tasks. Integrating ViTPose++ into our pipeline could leverage its enhanced capabilities, potentially improving accuracy without substantially compromising processing speed [5].

The PCT approach offers a structured representation by decomposing a human pose into discrete tokens, each characterizing a substructure with several interdependent joints. This compositional design effectively models joint dependencies, achieving small reconstruction errors efficiently. Notably, PCT has demonstrated resilience in scenarios involving occlusion, maintaining performance where other methods may falter. Incorporating the PCT method into our pipeline could enhance robustness, particularly in challenging conditions where occlusions are prevalent [25].

Reintroducing a pose detection module like ViTPose++ or implementing the PCT method could provide the SR module with more precise information regarding the player's position and posture. This additional context can guide the SR module to focus on relevant image regions, enhancing the quality of the super-resolved output. Subsequently, the STR module benefits from clearer, more detailed inputs, potentially leading to higher recognition accuracy.

D. Lanczos interpolation

The interpolation method we implemented in our BC-JNR+ pipeline for the super resolution module was bicubic interpolation. Bicubic interpolation is an extension of cubic interpolation for interpolating data points on a two-dimensional regular grid. It estimates the intensity of new pixels based on the weighted average of the 16 nearest pixels (a 4×4 grid) surrounding the target pixel. This method results in smoother and more visually appealing images compared to simpler methods like bilinear interpolation, which considers only 4 neighboring pixels (a 2×2 grid) [26], and even the bicubic interpolation.

However, to enhance the accuracy of our pipeline, it is beneficial to explore alternative interpolation methods for super-resolution. Different interpolation techniques can affect the quality of the upscaled images, potentially leading to better recognition performance. One promising alternative is Lanczos interpolation.

Lanczos interpolation employs a sinc function windowed by another sinc function, effectively reducing aliasing artifacts and preserving image details [27]. It uses a convolution kernel to interpolate pixel values, typically considering a window of a predefined size around each pixel [28]. This method is known for producing sharp and high-quality images, making it a popular choice for tasks requiring precise image scaling.

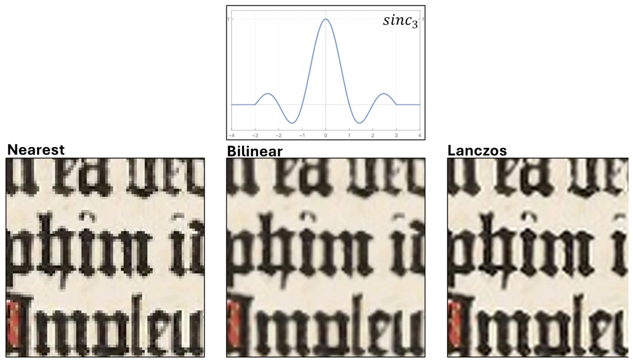


Fig. 12. Comparison between three interpolation techniques on an excerpt from The Gutenberg Bible, including Lanczos interpolation. The sinc function Lanczos interpolation uses is shown above [29].

Implementing Lanczos interpolation in the pipeline instead of the bicubic interpolation could enhance the clarity and detail of upscaled images, potentially leading to improved accuracy.

E. nvmath-python

Integrating NVIDIA's nvmath-python library into our BC-JNR+ pipeline, which currently utilizes NumPy for mathematical computations, could significantly enhance performance and efficiency. nvmath-python offers high-performance, Pythonic access to NVIDIA's CUDA-X Math Libraries, enabling seamless offloading of computations to NVIDIA GPUs.

nvmath-python is designed to work in conjunction with popular Python packages, including GPU-based packages like CuPy, PyTorch, and RAPIDS, as well as CPU-based packages like NumPy, SciPy, and scikit-learn. By leveraging nvmath-python, our pipeline can achieve higher throughput and reduced latency, crucial for real-time jersey number recognition tasks [30].

Furthermore, nvmath-python delivers performance comparable to its underlying C libraries, with minimal overhead. This efficiency ensures that Python applications can achieve near-native speeds, making it suitable for computationally intensive tasks in deep learning and data processing [30].

By transitioning our mathematical computations from NumPy to nvmath-python, we can harness NVIDIA GPUs' computational power more effectively, leading to a more efficient and responsive BC-JNR+ pipeline. This enhancement is particularly valuable in real-time applications where performance is critical [31].

F. Merging processing modules

Optimizing data transfer between CPU and GPU is crucial for enhancing the performance of GPU-accelerated pipelines. In the current pipeline, multiple system calls to individual Python programs result in repeated image reads and processing steps. Each of these operations involves transferring data from CPU memory to GPU memory and back, introducing significant overhead that can degrade performance.

To accomplish this, we use some strategies such as batch processing. Instead of processing images individually, we can aggregate multiple images into batches. This approach reduces the frequency of data transfers between CPU and GPU, thereby minimizing overhead and enhancing throughput.

We can also implement a unified data loading mechanism that reads all necessary images into memory at once. This strategy avoids redundant disk I/O operations and facilitates more efficient data handling [32].

Another idea is to design the pipeline to perform data transfers concurrently with computation. By overlapping these operations, the GPU can process data while simultaneously receiving the next batch, leading to more efficient utilization of resources.

Minimizing dynamic memory allocation and deallocation during runtime can also increase performance, as these operations can introduce latency. Pre-allocating memory buffers and reusing them can lead to more predictable and efficient performance.

G. Unified model

Transitioning from a segmented pipeline to an integrated, end-to-end model can significantly enhance both the explainability and performance of machine learning systems. Segmented pipelines, which involve chaining multiple distinct models or processing steps, often suffer from compounded errors, increased complexity, and reduced transparency. Each component's output serves as the subsequent component's input, making it challenging to trace and interpret the origin of errors or decisions within the system [33].

In contrast, a unified model processes data holistically, providing a singular framework where decisions are made based on the entire input context. This integration facilitates more straightforward interpretation and debugging, as the model's internal representations and decision pathways are consolidated. Furthermore, end-to-end models can leverage shared features and joint optimization, often leading to improved performance and generalization.

Research supports the advantages of integrated models over segmented approaches. For instance, a study on explainable deep learning methods highlighted that incorporating cascaded models to refine outputs can enhance both performance and interpretability [34]. Additionally, discussions in the machine learning community emphasize the importance of transparency in complex pipelines, noting that end-to-end models often provide clearer insights into the decision-making process [35].

Acknowledgment

We would like to express our gratitude to Professor Mohamed Shehata and the teaching assistants for their invaluable guidance throughout the project. Additionally, we would like to thank NVIDIA for giving us access to one A100 GPU for our development. Finally, we sincerely appreciate our colleagues and other teams in the class.

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] 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.

[6] 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.

[7] 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.

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

[9] "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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] R. Atienza, "Data Augmentation for Scene Text Recognition," Aug. 16, 2021, *arXiv*: arXiv:2108.06949. doi: 10.48550/arXiv.2108.06949.

[17] 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.

[18] A. Cioppa *et al.*, "SoccerNet 2023 challenges results." arXiv, 2023. doi: 10.48550/arXiv.2309.06006.

[19] L. N. Smith and N. Topin, "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates," May 17, 2018, *arXiv*: arXiv:1708.07120. doi: 10.48550/arXiv.1708.07120.

[20] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," May 03, 2017, *arXiv*: arXiv:1608.03983. doi: 10.48550/arXiv.1608.03983.

[21] "nvmath-python," NVIDIA Developer. Accessed: Apr. 07, 2025. [Online]. Available: https://developer.nvidia.com/nvmath-python

[22] D. Park and S. P. Ko, "NCAP: Scene Text Image Super-Resolution with Non-CAtegorical Prior," Apr. 01, 2025, *arXiv*: arXiv:2504.00410. doi: 10.48550/arXiv.2504.00410.

[23] W. Wang *et al.*, "TextSR: Content-Aware Text Super-Resolution Guided by Recognition," Oct. 20, 2019, *arXiv*: arXiv:1909.07113. doi: 10.48550/arXiv.1909.07113.

[24] "Papers with Code - CLIP4STR: A Simple Baseline for Scene Text Recognition with Pre-trained Vision-Language Model." Accessed: Apr. 04, 2025. [Online]. Available: https://paperswithcode.com/paper/clip4str-a-simple-baseline-for-scene-text-1

[25] 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.

[26] "(PDF) SUPER RESOLUTION IMAGE RECONSTRUCTION BY USING BICUBIC INTERPOLATION," in *ResearchGate*, doi: 10.13140/RG.2.1.4909.0401.

[27] K. Turkowski, "Filters for common resampling tasks," in *Graphics gems*, USA: Academic Press Professional, Inc., 1990, pp. 147–165.

[28] Amanrao, "Image Upscaling using Bicubic Interpolation," Medium. Accessed: Apr. 04, 2025. [Online]. Available: https://medium.com/@amanrao032/image-upscaling-using-bicubic-interpolation-ddb37295df0

[29] F. Mazzoli <f@mazzo.li>, "Lánczos interpolation explained." Accessed: Apr. 07, 2025. [Online]. Available: https://mazzo.li/posts/lanczos.html

[30] Q. News, "NVIDIA Unveils High-Performance Math Library For Python Applications." Accessed: Apr. 04, 2025. [Online]. Available:

https://quantumzeitgeist.com/nvidia-unveils-high-performance-math-library-for-python-applications/

[31] "Overview — NVIDIA nvmath-python." Accessed: Apr. 04, 2025. [Online]. Available: https://docs.nvidia.com/cuda/nvmath-python/latest/overview.html?utm_source=chatgpt.com

[32] pszilard, "Answer to 'Techniques to Reduce CPU to GPU Data Transfer Latency,'" Stack Overflow. Accessed: Apr. 04, 2025. [Online]. Available: https://stackoverflow.com/a/6514816

[33] A. Andres, A. Martinez-Seras, I. Laña, and J. D. Ser, "On the Black-box Explainability of Object Detection Models for Safe and Trustworthy Industrial Applications," *Results Eng.*, vol. 24, p. 103498, Dec. 2024, doi: 10.1016/j.rineng.2024.103498.

[34] S. Mohagheghi and A. H. Foruzan, "Developing an explainable deep learning boundary correction method by incorporating cascaded x-Dim models to improve segmentation defects in liver CT images," *Comput. Biol. Med.*, vol. 140, p. 105106, Jan. 2022, doi: 10.1016/j.compbiomed.2021.105106.

[35] B. O.-T. Renders Jean-Michel, "Explainability matters in machine learning pipelines," Naver Labs Europe. Accessed: Apr. 04, 2025. [Online]. Available: https://europe.naverlabs.com/blog/explainability-matters-in-machine-learning-pipelines/