scientific reports

Check for updates

OPEN

Multi stage generative upscaler recovers low resolution football broadcast images through diffusion models with ControlNet conditioning and LoRA fine tuning

Luca Martini[1]✉, Daniele Zolezzi[2], Saverio Iacono[2], Luca Oneto[2] & Gianni Viardo Vercelli[2]

The reconstruction of low-resolution football broadcast images presents a significant challenge in sports broadcasting, where detailed visuals are essential for analysis and audience engagement. This study introduces a multi-stage generative upscaling framework leveraging Diffusion Models to enhance degraded images, transforming inputs as small as 64×64 pixels into high-fidelity 1024×1024 outputs. By integrating an image-to-image pipeline, ControlNet conditioning, and LoRA fine-tuning, our approach surpasses traditional upscaling methods in restoring intricate textures and domain-specific elements such as player details and jersey logos. The custom LoRA is trained on a custom football dataset, ensuring adaptability to sports broadcast needs. Experimental results demonstrate substantial improvements over conventional models, with ControlNet refining fine details and LoRA enhancing task-specific elements. These findings highlight the potential of diffusion-based image reconstruction in sports media, paving the way for future applications in automated video enhancement and real-time sports analytics.

Keywords Computer Vision, Neural Networks, Image Reconstruction

Generative Artificial Intelligence (genAI) represents a groundbreaking approach to creativity and automation, empowering machines to produce novel and highly realistic data, including images, text, and music. Among the diverse generative models, Diffusion Models have emerged as a powerful technique for high-quality image synthesis. Rooted in the principles of probabilistic modeling, Diffusion Models iteratively refine noise into detailed and coherent representations, achieving remarkable performance in domains like image generation, image inpainting and style transfer.

Diffusion Models have gained traction due to their versatility and robustness, allowing them to excel in challenging tasks where conventional generative approaches, such as Generative Adversarial Networks (GANs), often struggle. These models leverage a forward-backward diffusion process, where images are progressively noised during the forward phase and restored to their original form during the reverse phase. This framework not only provides a stable training environment but also generates results with exceptional fidelity and diversity.

In this study, we focus on leveraging Diffusion Models to reconstruct and upscale low-resolution images, transforming inputs as small as 64×64 pixels into high-resolution outputs of 1024×1024 pixels or greater. This innovative approach goes beyond merely augmenting image resolution; it also reconstructs critical details in areas of the image that are heavily blurred, noisy, or otherwise degraded. The process enables the restoration of fine-grained textures and intricate visual elements, which are often missing or indistinct in the original low-resolution images.

The work presented in this study has been conducted in collaboration with a broadcast company specializing in sports events, particularly football, including major tournaments such as the UEFA Champions League. A significant challenge in this domain arises from low-resolution images generated as cutouts from live camera broadcasts. These cutouts are often used in post-match analysis, where detailed visual content is essential for

[1]Department of Languages and Modern Culture, University of Genova, Genova 16124, Italy. [2]Department of Computer Science and Technology, Bioengineering, Robotics, and Systems Engineering, University of Genova, Genova 16145, Italy. ✉email: luca.martini@edu.unige.it

reviewing critical moments, analyzing player movements, and creating engaging commentary. However, the quality of these images is frequently compromised due to cropping, compression, and real-time processing constraints.

Our approach addresses this challenge by reconstructing and enhancing these low-resolution cutouts, restoring general image structure, specific details and improving overall clarity. This enables broadcasters to deliver high-quality visuals that meet the exacting standards required for detailed analysis and audience satisfaction, while also preserving the authenticity and integrity of the original footage.

To achieve this reconstruction, we leverage the Diffusers library[1] and Flux.1-Dev model[2], employing a systematic multi-stage process to maximize reconstruction accuracy and detail recovery. In the first stage, an image-to-image pipeline is used to reconstruct broader and general details of the low-resolution inputs, providing a foundational enhancement. Following this, we employ a pre-trained ControlNet model[3] to augment finer details and improve the fidelity of the reconstructed images. Finally, to further specialize the reconstruction capabilities of ControlNet in the context of football broadcasts, we train a LoRA (Low-Rank Adaptation)[4] model on a handcrafted football dataset. This targeted fine-tuning enables ControlNet model to better capture and reconstruct domain-specific elements, such as player details, jerseys, and field textures, ensuring the enhanced images meet the high-quality standards demanded in sports broadcasting workflows.

The remaining of the paper is oragnized as folows. Section Related Works, the study reviews prior advancements in image reconstruction, including GAN-based techniques like SRGAN and ESRGAN, and contrasts them with the emerging superiority of Diffusion Models. Section Methods details the reconstruction framework which counts three phases: (i) a first Reconstruction, where input images are standardized and initial upscaling is performed using an image-to-image pipeline, (ii) a second Reconstruction introduces ControlNet to refine finer details like player numbers and jersey textures, and (iii) a Fine-Tuning stage where a custom LoRA model trained on a football-specific dataset to further enhance domain-specific elements. Section Results and Discussion evaluates the effectiveness of the proposed approach, demonstrating improved clarity and structural fidelity, particularly with the integration of LoRA. Finally section Conclusions discusses future improvements, including dataset expansion, Graphical User Interface development and additional fine-tuning strategies for further refinement.

Related works

The field of image reconstruction and enhancement has seen significant advancements in recent years, driven by the development of powerful generative models. Among these, Diffusion Models have emerged as a leading technique, offering superior performance in generating high-quality images from noisy or degraded inputs. Building on the foundational work of "Denoising Diffusion Probabilistic Models" (DDPM)[5], Diffusion Models have demonstrated remarkable versatility across different domain such as inpainting and image synthesis. Their probabilistic framework, which iteratively refines data through a noise-to-data reversal process, has set a new benchmark for generative modeling.

Among the most notable advancements in image reconstruction and enhancement, Generative Adversarial Networks (GANs) have revolutionized the field with their ability to generate perceptually convincing images. The introduction of the Super-Resolution Generative Adversarial Network (SRGAN)[6] marked a turning point in single-image super-resolution tasks. SRGAN employed a novel perceptual loss function that combined content loss, based on high-level feature maps from a pre-trained VGG (Visual Geometry Group) network[7], and adversarial loss to enhance texture details and generate images with realistic characteristics. This architecture demonstrated a significant improvement over traditional methods by focusing not only on pixel-wise accuracy but also on generating visually appealing results, even when the input images were highly degraded.

Building on SRGAN, Enhanced Super-Resolution GAN (ESRGAN) introduced by[8] further advanced the state of the art by addressing some of SRGAN's inherent shortcomings. ESRGAN replaced the vanilla residual blocks in SRGAN with Residual-in-Residual Dense Blocks (RRDB), which improved gradient flow and enabled deeper network architectures. Furthermore, ESRGAN incorporated a relativistic adversarial loss, which improved the discriminator's ability to distinguish between real and fake samples, leading to more stable training dynamics. These innovations resulted in sharper textures, better restoration of fine details, and a noticeable improvement in the perceptual quality of super-resolved images.

Despite the significant improvements made by SRGAN and ESRGAN, their performance often faltered when applied to real-world scenarios involving diverse degradations not seen during training. To address this gap, Real-ESRGAN[9] was introduced, extending the capabilities of GAN-based approaches to handle real-world blind super-resolution tasks. Real-ESRGAN enhanced ESRGAN's architecture by introducing a pure synthetic data generation pipeline for training, which simulated a wide variety of real-world degradations, such as noise, blur, compression artifacts, and downsampling effects. This approach significantly increased the robustness of the model to unseen degradations. Real-ESRGAN also incorporated a second-order degradation model to better approximate real-world data characteristics, along with adaptive weight strategies to balance training between perceptual quality and pixel fidelity. These modifications allowed Real-ESRGAN to produce high-quality outputs that maintained both visual realism and structural accuracy, even under challenging conditions. By emphasizing the diversity and realism of its training data, Real-ESRGAN outperformed its predecessors in tasks where input images suffered from significant and unpredictable degradations.

Despite the improvements brought by ESRGAN and Real-ESRGAN, the need for a more robust approach capable of handling extreme upscaling and severe image degradations prompted the exploration of alternative frameworks. For our scope of work, these limitations of GAN-based approaches prompted the exploration of Diffusion Models, which offer stable training dynamics and the ability to generate high-fidelity details across extreme upscaling tasks. Diffusion Models' iterative denoising process is particularly well-suited for reconstructing not only resolution but also missing or blurred elements, providing a more robust solution for

reconstructing low-resolution, degraded camera cutouts used in post-match analyses of sports events. Notably, the use of Diffusion Models extends beyond generative tasks to support practical applications in image restoration and enhancement. A recent study[10] highlighted the use of Diffusion Models in addressing extreme challenges in image reconstruction by leveraging their iterative refinement mechanism. This method integrates degradation-specific conditioning to address noise, blur, and downscaling artifacts effectively. The study underscores the diffusion framework's adaptability in learning fine-grained details, surpassing traditional and GAN-based approaches in reconstructing visually coherent high-resolution outputs.

Advancements in the capabilities of Diffusion Models have been significantly enhanced by frameworks such as ControlNet, which refine the generative process through the integration of conditional control. The groundbreaking work, *Adding Conditional Control to Text-to-Image Diffusion Models*[11] propose a novel method to incorporate external conditioning signals, such as edge maps, depth information, or pose guidance, into the generation process. This innovation not only improves the flexibility of Diffusion Models but does so without compromising the pre-trained generative capacity of the underlying model. ControlNet introduces a dedicated control branch tailored to process supplementary conditioning inputs while maintaining the original weights of the pre-trained diffusion model. By freezing the core model parameters and focusing training solely on the control branch, this architecture achieves an optimal trade-off between model fine-tuning and stability. This design ensures that the diffusion model preserves its ability to generate high-quality images while enabling precise control over specific aspects of image generation. In comparison to conventional approaches, ControlNet circumvents the need for extensive parameter adjustments, which are often necessary in GAN-based or fully fine-tuned Diffusion Models. Its lightweight and modular framework makes it particularly effective for iterative enhancements in specialized applications, such as real-time sports analysis or domain-specific image restoration.

Building on the advancements introduced by ControlNet, which enable precise conditional control in Diffusion Models, the need for efficient adaptation of such models to diverse domains becomes increasingly evident. While ControlNet provides a framework for integrating external conditioning data, the process of fine-tuning Diffusion Models for specific applications still poses significant computational challenges. This highlights the need for approaches that enable lightweight, task-specific modifications without retraining the entire model, a challenge effectively addressed by techniques like Low-Rank Adaptation (LoRA)[4]. Originally introduced as a technique for efficiently fine-tuning Large Language Models, LoRA has found remarkable relevance in the domain of Diffusion Models[12]. LoRA addresses one of the core challenges in training and adapting large models: the computational cost and memory requirements of updating a vast number of parameters. By introducing a low-rank decomposition framework, LoRA enables the adjustment of model weights in a lightweight and scalable manner, making it an attractive alternative for adapting Diffusion Models to specific tasks or datasets. Furthermore, LoRA's approach of isolating task-specific modifications ensures that the underlying generative capabilities of the diffusion model remain intact, reducing the risk of catastrophic forgetting. This makes LoRA particularly suited for scenarios where Diffusion Models need to handle diverse tasks or datasets with minimal retraining. Recent studies, further highlight the superiority of LoRA over other fine-tuning techniques like Textual Inversion or Dreambooth[13], emphasizing its efficiency and versatility across a wide range of applications. Notably, LoRA has also demonstrated superior computational efficiency, often enabling fine-tuning to be performed even on consumer-grade hardware, making it accessible for a wider range of applications and researchers[13].

Recent developments in diffusion model architecture have marked a shift from the traditional reliance on U-Net backbones to Transformer-based designs called Diffusion Transformers (DiT)[14]. While U-Nets have been the backbone of choice due to their convolutional efficiency and hierarchical feature extraction, they often struggle with capturing long-range dependencies and handling intricate textures. Transformer-based architectures address these limitations through self-attention mechanisms, which enable the modeling of global dependencies and maintain contextual coherence throughout the generation process. These characteristics make Transformers particularly suitable for complex tasks requiring the reconstruction of fine details or the synthesis of high-resolution outputs.

Simultaneously, the Rectified Flow[15] approach has emerged as a significant innovation, enhancing the efficiency and stability of Diffusion Models. Rectified Flow optimizes the noise-to-data reversal process by aligning the transport paths more closely with the shortest trajectories between data distributions. This alignment reduces gradient instability, which is a common issue in traditional models, and ensures smoother training dynamics. Furthermore, Rectified Flow adapts dynamically to the complexity of input data, enabling robust handling of diverse degradations, such as noise, blur, and compression artifacts. These properties allow Rectified Flow to generate high-quality images with fewer inference steps, making it computationally efficient while maintaining output fidelity.

Recent work has further improved diffusion-based super-resolution by combining diffusion denoising with adversarial training and multi-scale feature learning. "Single-Image Super-Resolution with Denoising Diffusion GANs (SRDDGAN)"[16] introduces a conditional GAN discriminator into the diffusion super-resolution pipeline, which permits substantially larger denoising step sizes and yields an $\approx 11\times$ faster inference time compared to conventional diffusion-based SR methods. Remarkably, this acceleration comes without sacrificing quality: SRDDGAN matches or exceeds the standard pixel-level fidelity (e.g., PSNR) and perceptual performance of slower, multi-step diffusion models by leveraging the GAN's adversarial feedback to preserve fine details even under aggressive sampling schedules.

Similarly, "Multi-Scale Adversarial Diffusion Network"[17] employs a progressive, multi-resolution diffusion generator alongside a time-dependent discriminator and a dedicated high-frequency loss. By learning features at multiple scales before reconstructing fine textures, MSADN surpasses previous diffusion SR methods in both fidelity and runtime.

While diffusion models have set a new quality bar for image restoration, their high sampling cost and memory footprint limit deployment. SinSR[18] tackles the first bottleneck by distilling a multi-step teacher into a single-step student using a consistency-preserving loss that concentrates fine-grained gradients normally spread across dozens of sampling steps. Complementary to runtime reduction, BI-DiffSR[19] binarises both weights and activations of a ≈ 4.6 M-parameter UNet via channel-shuffle fusion and timestep-aware modules, shrinking the model to roughly 4.6 M bits while retaining about 98 % of its full-precision PSNR on Urban100. These results confirm that diffusion models can be aggressively compressed, resembling our own LoRA strategy, which greatly reduces trainable parameters without additional quantization and enables fine-tuning on consumer GPUs.

For video sequences, applying diffusion on each frame independently often causes flicker. Upscale-A-Video[20] addresses this by sampling in a shared latent space that is recurrently propagated across neighbouring frames, integrating 3-D convolution and temporal-attention layers inside the U-Net and VAE decoder, and adding a flow-guided latent-propagation module for long-range stability. An optional clip-level text prompt further lets users impose a consistent tone or style on the whole sequence.

In conclusion, for the scope of our work, we leverage State-Of-The-Art (SOTA) advancements in Diffusion Models to tackle the challenges of image reconstruction and enhancement. Our approach incorporates the latest models featuring DiT and Rectified Flow architectures to ensure efficiency and stability in the generative process. Additionally, we utilize ControlNet for precise conditional control and LoRA for lightweight and efficient fine-tuning, enabling task-specific adaptations with minimal computational overhead. This integration of cutting-edge techniques positions our methodology to address diverse image degradation scenarios effectively, setting a robust foundation for high-fidelity reconstruction tasks.

Methods

This study focuses on developing a robust framework to enhance the resolution of low-quality images and reconstruct severely degraded regions, aiming to produce visually coherent and high-fidelity outputs.

The images targeted for upscaling are low-resolution snapshots originating from video production systems commonly used in broadcasting football events. These systems are designed to capture and process live video frames, segmenting them into individual images for transmission to operators. The transmitted images are typically used for tasks such as post-match analysis and reporting. Due to the interlaced nature of 1080p broadcasting, the segmented images are often significantly reduced in size. The snapshots start at an average resolution of around 100×100 pixels, as shown in Fig. 1, presenting a considerable challenge in terms of restoring both clarity and missing details effectively.

Python 3.11 was selected as the programming environment, integrating PyTorch 2.5.1 and CUDA 12.4 to leverage GPU-accelerated computations. The Diffusers library[1] was employed to construct and manage diffusion pipelines, offering fine-grained control and debugging capabilities superior to graphical interfaces like ComfyUI[21]. This modularity facilitated the customization of diffusion processes to meet the specific requirements of restoring and reconstructing broadcast frames.

The workflow was executed using an Nvidia RTX 6000 Ada GPU, which provided the necessary VRAM capacity to handle the computational demands of both training and inference. To monitor hardware performance

Fig. 1. An example of raw image frames segmented from a 1080p interlaced broadcast. These snapshots, manually cut by technical operators during the match, exhibit varying resolutions, often starting as low as 60×60 pixels.

and optimize resource utilization, the nvidia-ml-py3[22] library was employed, enabling real-time tracking of GPU metrics such as memory usage, temperature, and power consumption.

The complete pipeline is presented in Fig. 2, with each stage of the process discussed in its respective section.

First stage of the reconstruction

Before the actual reconstruction, the first phase focuses on standardizing the input images to ensure consistent dimensions and aspect ratios, addressing the variability inherent in the images extracted from live sports broadcasts. This variability, if left unmanaged, can lead to inconsistencies when using models optimized for specific input dimensions. Our ultimate objective was to reconstruct these images to a resolution of 1024×1024, not merely upscaling the images but also restoring critical details such as textures, logos and intricate visual elements that were lost or degraded in the original low-resolution inputs.

To achieve this, a preprocessing step was implemented to resize all input images to a square aspect ratio. The choice of a square format was deliberate, as many Diffusion Models are optimized for square inputs. Non-square images could introduce unintended variations, potentially confounding the outcomes when switching between models. Standardizing the aspect ratio ensures that results remain consistent across different experiments and model configurations.

The resolution for this preprocessing stage was set at 256×256 pixels, balancing computational efficiency and the retention of visual details. The resizing was performed using the Lanczos resampling algorithm[23], a high-quality upscaling method renowned for its ability to preserve edge details while minimizing aliasing. This approach ensured that the resized images retained as much fidelity as possible from their original forms, providing a solid foundation for subsequent stages of image reconstruction and enhancement.

Initially, to achieve the reconstruction, we attempted to use the Real-ESRGAN algorithm, leveraging the Upscayl application[24] for implementation. Upscayl is an open-source, user-friendly tool designed for upscaling and enhancing image resolution using cutting-edge algorithms like Real-ESRGAN. It provides a streamlined interface for leveraging advanced AI models without requiring in-depth technical expertise, making it a popular choice for quick image enhancement tasks. However, as depicted in Fig. 3, the results were visually extremely poor, exhibiting significant noise and distortions. These issues often resulted in the loss of crucial image details. This outcome is likely due to the significant degradation present in the input images, which exceeded Real-

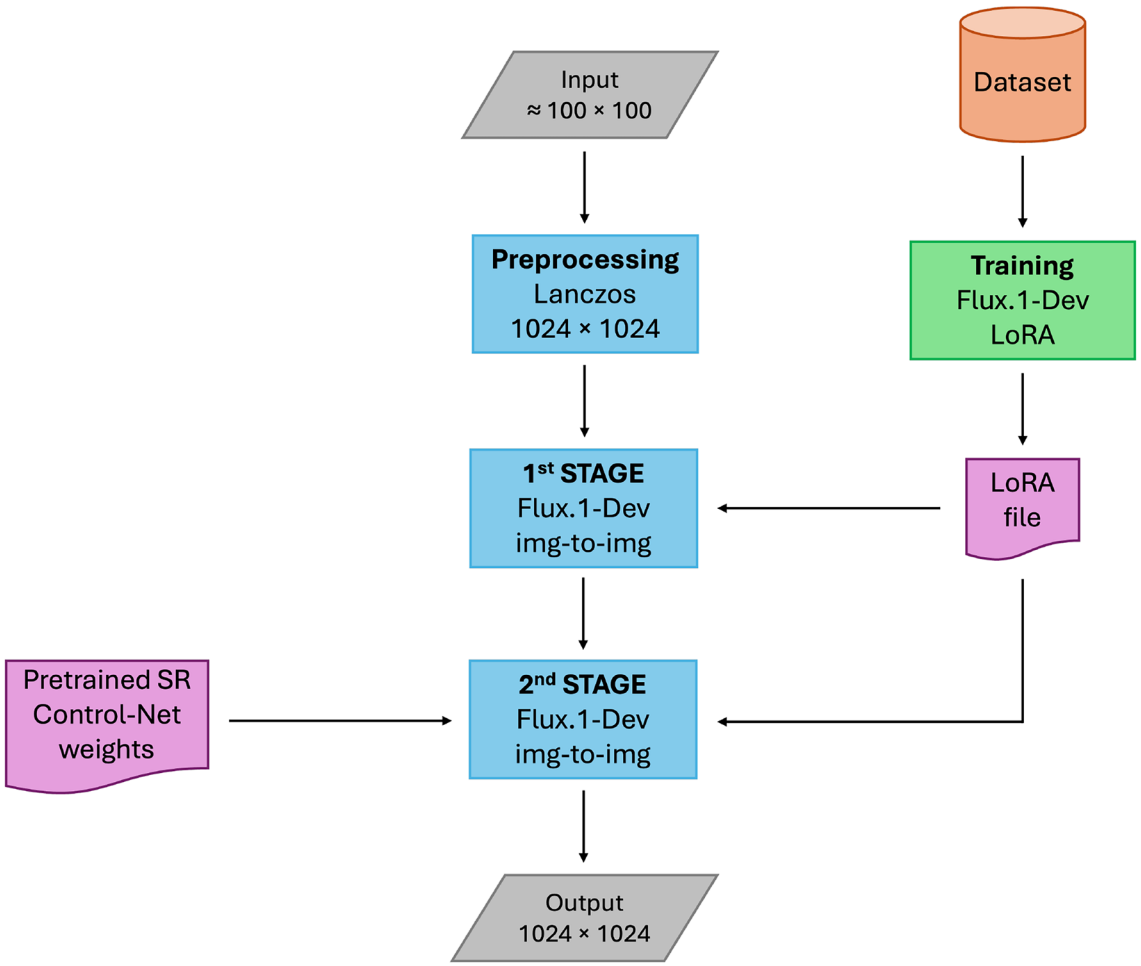

Fig. 2. Complete pipeline flowchart.

Fig. 3. Comparison of image enhancement techniques applied to raw broadcast frames. From left to right, each column represents: the raw image frames resized to 256×256 using Lanczos resampling, the result of applying Real-ESRGAN for 1024×1024 upscaling, the result from the latent x4 upscaler (Stable Diffusion x4 Upscaler), and finally, the Lanczos resampling upscaled to 1024×1024.

```
from PIL import Image
from diffusers import AutoPipelineForImage2Image

pipeline = AutoPipelineForImage2Image.from_pretrained(
                "stabilityai/stable-diffusion-3.5-medium",
                ).to("cuda")

seed = torch.random.seed()
generator = torch.Generator().manual_seed(seed)
prompt = ("A dog")
image = pipeline(
    prompt=prompt,
    image=Image.open("img.jpg"),
    guidance_scale=3.5,
    strength=0.6,
    height=1024,
    width=1024,
    num_inference_steps=80,
    generator=generator
).images[0]

image.save("dog.jpg")
```

Code 1. Example of image-to-image generation pipeline with diffusers library.

ESRGAN's ability to reconstruct satisfactory details. The severe blurring, noise, and compression artifacts in our dataset highlighted the limitations of Real-ESRGAN for this specific application. This prompted us to explore alternative approaches to achieve the desired quality and resolution.

After encountering the limitations of Real-ESRGAN, we next tried using a latent upscaler to enhance the resolution of the degraded images. Specifically, we utilized the Stable Diffusion x4 Upscaler model available in the Diffusers library and developed by StabilityAI[25]. This model is a specialized variant of Stable Diffusion, designed to upscale images by a factor of four while leveraging latent diffusion techniques. Unlike traditional pixel-space upscaling methods, this model operates in the latent space, capturing high-level semantic details before decoding them into high-resolution outputs. The latent diffusion process allows for the generation of more coherent and visually appealing results, especially in scenarios with moderate degradation.

While the Stable Diffusion x4 Upscaler yielded marginally fewer hallucinated details and exhibited less texture loss than Real-ESRGAN, its outputs remained largely unusable for our analysis (Fig. 3). In contrast, simple Lanczos interpolation preserves the original frame with far greater fidelity, achieving the lowest mean absolute error in Table 1. It is important to emphasise that this experiment was conducted purely as a preprocessing step to evaluate whether modern learned upscalers could outperform classical interpolation, not to perform full reconstruction at this stage. Given the minimal quality improvements alongside the substantial computational overhead of both SDx4[25] and Real-ESRGAN, these methods proved unjustified for our workflow. We therefore adopt Lanczos to resample frames to 1024×1204, maximally preserving the input signal before proceeding with our multi-stage pipeline.

After evaluating the limitations of the previously tested methods, we decided to adopt an image-to-image pipeline approach, which is widely recognized in the Diffusers library. This approach leverages the inherent strengths of Diffusion Models in iterative refinement, allowing us to transform low-resolution and degraded images into high-resolution outputs while reconstructing fine details. Since we aimed to evaluate the latest SOTA models, we upscaled the images to 1024×1024 resolution using Lanczos resampling: most of Diffusion Models are optimized for this resolution since are trained on images of this size.

Following the preprocessing step with Lanczos resampling, we began experimenting with various Diffusion Models using the Diffusers library[1] to develop an effective image-to-image reconstruction pipeline. This method leverages the power of Diffusion Models to enhance input images by iteratively refining noise while maintaining the structural coherence of the original image. Unlike generative processes that synthesize images from scratch, the image-to-image pipeline conditions the denoising process on the input image, ensuring that the enhanced output aligns with the original context and layout.

Technically, the pipeline works by first adding a controlled amount of noise to the input image. This noising step, parameterized by a `strength` argument (Code 1), determines the extent of noise introduced into the input image and, as a result, the degree of deviation allowed during the enhancement process. It's crucial to remember that if we don't set the `seed` at the beginning, the noise introduced in each generation will be slightly different because it's sampled randomly from a Gaussian distribution; in other words, `strength` determines only the total amount of noise added to the image, while the sampled noise distribution is set by the `seed` parameter.

`Strength` is one of the most critical parameters in the pipeline, as it directly impacts the generated image[26]. Specifically:

- A higher `strength` value gives the model greater "creativity", allowing it to produce an output that is increasingly different from the initial image. At a `strength` value of 1.0, the initial image is effectively ignored and the generation is only based on the textual prompt.
- A lower `strength` value means the generated image closely resembles the original input, prioritizing fidelity over transformation.

The `strength` and `num_inference_steps` (Code 1) parameters are closely related, as `strength` determines how much noise is applied to the input image, and `num_inference_steps` the number of denoising steps performed on the initial image. This is important because if we use a higher value of `strength`, we introduce more noise, necessitating more inference steps to effectively denoise the image. The Diffusers library[1], for example, with `num_inference_steps` set to 50 and `strength` set to 0.8, adds 40 noise steps (50×0.8) to the input image, followed by 40 steps of denoising to produce the output. This relationship underscores the need to carefully balance these parameters for optimal results: during the denoising phase, the model iteratively removes noise at each step, using the conditioning input (the original image) and textual prompt as guidance. This step-by-step refinement not only restores missing details but also enhances the image with sharper textures and greater visual intricacy.

In our experiments, we observed that Flux.1-Dev, since it employs Rectified Flow, which we remind the reader is a bridge between full ODE and SDE samplers, benefits from a relatively low number of inference steps, ranging from 35 to 50. Increasing the number of steps beyond this range did not yield any substantial

Image ID	Real-ESRGAN	SDx4	Lanczos
attack_higher	5.6	3.6	0.6
attack_lower	4.7	3.8	0.5
mbappé	3.7	4.6	0.5
double	4.1	5.2	0.4
dzeko	3.2	3.1	0.4
double2	6.3	3.6	1.0
young	4.0	3.1	0.4
goalkeeper2	5.0	3.9	0.6
goalkeeper	5.2	3.0	0.5
MAE	**4.6**	**3.8**	**0.6**

Table 1. Distance metric (MAE) between original images and 1024×1024 resampled outputs.

improvement; instead, it frequently led to the generation of more hallucinations. Therefore, for our pipeline we consistently limited the total denoising steps to at most 50.

In addition to `strength` and `num_inference_steps`, other parameters (Code 1) play a crucial role in controlling the behavior and output of the pipeline:

- The `seed` parameter is used to initialize the random number generator, ensuring reproducibility of results. By setting a specific seed value (e.g., `seed = 5`), the same noise pattern is generated for each run. This consistency is particularly useful for comparing different configurations or testing enhancements. The `generator` is a PyTorch object initialized with the manual seed, providing fine-grained control over the random sampling process.
- The `prompt` parameter defines the textual description used to guide the diffusion process.
- The `guidance_scale` controls the influence of the prompt on the generated image. Higher values prioritize adherence to the prompt at the cost of potentially reduced coherence with the original image, while lower values allow the model more freedom to balance the prompt and the input image.

These parameters, when tuned carefully alongside `strength` and `num_inference_steps`, enable precise control over the pipeline, ensuring high-quality reconstructions that align with both the input image and the descriptive prompt.

Initially, we explored a range of open-source models to tackle the challenges of reconstructing fine-grained details in highly degraded images. This included Stable Diffusion XL[27], the most recent Stable Diffusion 3.5 in both *medium*[28] and *large*[29] configurations, as well as several fine-tuned models designed to produce realistic outputs. Notable among these were models available on platforms such as CivitAI[30], including RealVis_V4.0[31]. Although these models exhibited significant strengths in general applications, they fell short in meeting the specific requirements of extreme upscaling and intricate detail recovery within the football domain, as shown in Fig. 4.

After evaluating various open-source models, we selected FLUX.1-dev[32] as our primary model due to its superior performance in our football-specific context. FLUX.1-dev is an open-weight, guidance-distilled model designed for non-commercial applications. It is directly distilled from FLUX.1-pro, achieving similar quality and prompt adherence capabilities while being more efficient than standard models of the same size[2]. The FLUX.1 architecture is based on a hybrid design that combines multimodal and parallel diffusion transformer blocks, scaled to 12 billion parameters. This structure enhances the model's ability to generate high-quality images with detailed textures and complex visual elements.

We observed superior performance with FLUX.1-dev compared to other models, particularly against Stable Diffusion 3.5 which is particularly surprising given that Stable Diffusion 3.5, as one of StabilityAI's latest models, should be expected to compete closely with FLUX.1-dev. Although Stable Diffusion 3.5 utilizes three Text Encoders, two of these, CLIP-L/14[33] and T5XXL[34], are also incorporated in FLUX.1-dev. Both models share a DiT backbone[2835] and leverage the Rectified Flow method to enhance training efficiency and stability in the diffusion process. Despite these architectural similarities, Stable Diffusion 3.5 delivered far inferior results in most of our experiments, failing to achieve the same level of detail recovery and visual coherence. FLUX.1-dev consistently excelled in reconstructing fine details and intricate textures in our specific domain, making it the clear choice for our workflow. Additionally, FLUX.1-dev demonstrated significantly better coherence with the original image, even when used with high `strength` values, avoiding excessive deviation from the input. Moreover, FLUX.1-dev exhibits superior prompt adherence and enhanced reconstruction capabilities for company brands and logos, a feature crucial for applications in a broadcast company.

However, this high-quality output comes at the expense of increased computational demands. Running the full FLUX.1-dev model with both text encoders active requires a minimum of 24 GB of VRAM, highlighting

Fig. 4. Comparison of outputs from image-to-image pipelines using various models. From left to right, each column represents: the original raw frames, the outputs of Stable Diffusion XL, RealVis_v4.0, Stable Diffusion 3.5-Medium, Stable Diffusion 3.5-Large, and FLUX.1-dev.

```
import torch
from diffusers import FluxImg2ImgPipeline
from PIL import Image

pipe = FluxImg2ImgPipeline.from_pretrained(
                            "black-forest-labs/FLUX.1-dev",
                            torch_dtype=torch.bfloat16
                            )
pipe.enable_model_cpu_offload() # entire model is offloaded

seed=5
generator=torch.Generator().manual_seed(seed)
prompt= # prompt changes based on the input image
image=pipe(
    prompt=prompt,
    image=resize_image_square("imgs/99_Donnarumma.jpg", 1024),
    # custom function for Lanczos resampling
    num_images_per_prompt=3
    guidance_scale=3.5,
    strength=0.75,
    height=1024,
    width=1024,
    num_inference_steps=65,
    generator=generator
).images
```

Code 2. Our pipeline used for the first image reconstruction stage.

```
import torch
from diffusers import FluxControlNetModel
from diffusers.pipelines import FluxControlNetPipeline

controlnet = FluxControlNetModel.from_pretrained(
                  "jasperai/Flux.1-dev-Controlnet-Upscaler",
                  torch_dtype=torch.bfloat16)
pipe = FluxControlNetPipeline.from_pretrained(
                  "black-forest-labs/FLUX.1-dev",
                  controlnet=controlnet,
                  torch_dtype=torch.bfloat16)

pipe.enable_model_cpu_offload()

control_image = Image.open("reconstructed1024.jpg")
seed = torch.random.seed()
generator = torch.Generator().manual_seed(seed)
image = pipe(
    prompt= # prompt changes based on the input image,
    control_image=control_image,
    num_images_per_prompt=3
    controlnet_conditioning_scale=0.5,
    num_inference_steps=35,
    guidance_scale=3.5,
    height=1024,
    width=1024,
    generator = generator
).images
```

Code 3. Pipeline used for the second image reconstruction stage.

the model's resource-intensive nature. Additionally, running FLUX.1-dev on consumer-grade hardware, even with 24 GB of VRAM, can be challenging without optimizations such as CPU offloading and memory-efficient computation techniques[36]. These optimizations help manage the substantial memory requirements and computational load, making it feasible to deploy the model in environments with limited resources. To address these challenges, we utilized the following optimization techniques:

- Bfloat16 precision: This format, supported by modern GPUs and CPUs, reduces memory usage by using 16-bit floating-point precision while retaining the dynamic range of 32-bit floats. This allows for faster computation and reduced memory requirements without a significant impact on the quality of the outputs[37].
- Model CPU offloading: This feature offloads specific model components, such as the attention blocks and text encoders, to the CPU during both training and inference. By leveraging the memory bandwidth of the CPU, this approach reduces the GPU memory burden and enables running large models like FLUX.1-dev on hardware with limited VRAM. Additionally, it ensures smooth operation by transferring data between the GPU and CPU dynamically during processing[36].

Our workflow begins with Lanczos resampling (`resize_image_square` function in the Code 2), a precise and efficient method for resizing/upscaling low-resolution inputs to the required 1024×1024 resolution for processing within the Flux.1-dev framework. This resampling stage is critical, as it preserves essential details while mitigating artifacts typically introduced by other interpolation methods, thereby creating a clean and reliable baseline for downstream processing.

In the next stage, a Flux.1-dev image-to-image pipeline was configured with a `strength` parameter set at 0.75 and `num_inference_steps` fixed at 80 (Code 2). These values were meticulously calibrated to strike an optimal balance between maintaining structural integrity and enhancing fine details. The pipeline utilized a carefully crafted prompt, tailored to align with the characteristics of the input images and the desired output quality. Additionally, to ensure consistency, a fixed `seed` was employed throughout the process, enabling systematic exploration of configurations and facilitating the identification of the best settings for the task. However, in some cases, we kept the `seed` random and used the `num_images_per_prompt` parameter to generate multiple images at once, allowing us to evaluate the outputs and select the most suitable one for our needs.

At this developmental stage, prompts were manually crafted, leveraging information provided by the broadcast technician about specific elements within the scene, such as numbers on players' shirts. This knowledge proved invaluable in enhancing the accuracy and coherence of the generated outputs. For instance, an example prompt used was: "A goalkeeper wearing a green jersey and matching shorts with the number 99 is diving low to make a save. The player is focused on controlling the ball, which is on the ground ahead of him. The scene unfolds on a soccer field, with the blurred background featuring the goal area and field markings".

The `guidance_scale` parameter was set to 3.5, as it proved to deliver the best results in terms of prompt adherence, ensuring the outputs remained aligned with the intended characteristics defined by the crafted prompts.

Despite leveraging the Nvidia RTX 6000 Ada GPU with 48 GB of VRAM for inference, `enable_model_cpu_offload` were employed to circumvent memory exhaustion and maintain stability. This optimization ensured sufficient memory availability for subsequent stages of reconstruction. Furthermore, the use of components like the AutoEncoder, two Text Encoders, and DiT in bfloat16 format significantly accelerated the inference process while maintaining computational efficiency and improving latency.

Second stage of the reconstruction

After completing the initial reconstruction with an image-to-image pipeline (Fig. 2), we achieved significantly enhanced 1024×1024 images compared to the degraded originals. This stage successfully restored substantial amounts of detail and structure, elevating the visual quality of the outputs. However, a consistent challenge emerged during these experiments: many reconstructed images exhibited a noticeable degree of blurriness.

The primary cause of this issue was the need to constrain the model's deviation from the original input. To achieve this, the `strength` parameter in the image-to-image pipeline was kept below 0.8. While this precaution was crucial for preserving the overall structure and critical elements of the input images, it also limited the model's ability to introduce finer details or sharpen features effectively. Consequently, the resulting images, though significantly improved, often lacked the crispness and clarity required for high-fidelity applications such as sports broadcast analysis.

To address the limitations of the first stage, the second stage of the reconstruction process leverages ControlNet, a conditional refinement network designed to provide precise control over the generative process. By introducing auxiliary conditioning inputs, ControlNet enhances the model's capacity to refine specific aspects of the image while maintaining adherence to its original context.

In this stage, we employed a pre-trained ControlNet model, Flux.1-dev-ControlNet-Upscaler[3]. The inputs to this model were the reconstructed 1024×1024 outputs from the first stage. This ControlNet model was trained using a synthetic complex data degradation scheme that involved artificially degrading real-life images by combining several degradation techniques, including Gaussian and Poisson noise, image blurring, and JPEG compression, in a manner similar to Real-ESRGAN[9]. The `controlnet_conditioning_scale` parameter (Code 3) in ControlNet pipelines determines the influence of ControlNet's output on the base U-Net or DiT model during image generation. Specifically, the outputs from ControlNet are multiplied by this scale factor before being added to the residual connections in the U-Net or DiT. Adjusting this parameter allows control over how strongly the conditioning input affects the final generated image[38].

The incorporation of ControlNet enabled us to apply fine-grained modifications, targeting areas where detail restoration was most needed. For example, ControlNet effectively sharpened blurred player numbers, refined jersey textures, and restored field markings while preserving the broader composition of the image, as shown in Fig. 5. The pipeline was optimized by setting the `controlnet_conditioning_scale` parameter between 0.5 and 0.65, ensuring a balanced influence of the conditioning input relative to the textual prompt. The textual prompts used in this stage were identical to those used in the first stage.

Fig. 5. Comparison of Control-Net pipeline outputs. From left to right, each column represents: the original raw frames, the output images without the first stage of the reconstruction, the output images with the first stage of the reconstruction.

However, this enhancement comes at the cost of increased computational requirements. Integrating ControlNet with the Flux.1-Dev pipeline incurs an additional VRAM overhead of approximately 4 GB, bringing the total VRAM usage to 30 GB for generating a single image. This high memory demand necessitates the use of hardware with sufficient VRAM capacity for stable and efficient execution.

Furthermore, it is crucial to point out that the first stage of reconstruction was not only instrumental in improving the degraded images but also fundamental for the success of the subsequent ControlNet pipeline. When the raw image frames were fed directly into the ControlNet pipeline, the results were extremely poor (Fig. 5), as the severe degradation in the original frames overwhelmed the model's capacity for conditional refinement. The initial reconstruction provided a critical baseline by reducing noise, enhancing structure, and restoring key elements, thereby creating a reliable foundation for ControlNet to build upon.

By leveraging ControlNet in this second stage, we successfully overcame the limitations of the initial reconstruction pipeline. This multi-stage framework, which combines the strengths of image-to-image pipelines

```
{
    "/workspace/imgs/imago1050756553.jpg": {
        "caption": "A player from A.C. Milan, in a red and black striped jersey with white
             shorts, marked with the number 11, is attempting to score as he approaches
            the goal. He is surrounded by two players from Inter Milan, in black and blue
            striped jerseys, one with the number 28 and another with the number 20, trying
             to block him. The goalkeeper from Inter Milan, wearing a black kit, is
            preparing to make a save. The background shows the net, advertising boards,
            and a partially filled stadium with spectators."
    },
    "/workspace/imgs/imago1050756556.jpg": {
        "caption": "A player from Inter Milan, with short hair and white skin tone, in a
            black and blue striped jersey with black shorts, is in possession of the ball
            while being challenged by a player from A.C. Milan, with curly hair and white
            skin tone, in a red and black striped jersey with white shorts. Another A.C.
            Milan player, with short hair and dark skin tone, also in a red and black
            striped jersey, is positioned nearby, observing the play. The action is taking
             place on a grass field, with advertising boards and out-of-focus spectators
            in the background."
    },
    # ...
}
```

Code 4. Example of the structured JSON file with captions describing the actions, appearance, and context of players and the stadium in a soccer image.

```
[[datasets]]
resolution = [1024, 1024]
enable_bucket = true
max_bucket_reso = 1024

  [[datasets.subsets]]
  image_dir = '/workspace/imgs'
  metadata_file = '/workspace/captions_kohya.json'
  flip_aug = true
  num_repeats = 1
  shuffle_caption = false
```

Code 5. Toml file for dataset configuration.

with conditional refinement, provides a robust solution for reconstructing and enhancing degraded images in sports broadcasting contexts.

Fine-tuning

The results achieved so far using the multistage framework, which combines an image-to-image pipeline and ControlNet, have been satisfactory in most cases. However, we believed there was room for further improvement in the final outputs. To address this, we decided to train a LoRA for the FLUX.1-Dev model, aiming to enhance its reconstruction capabilities specifically for football actions such as interceptions, headers, or sliding tackles, as well as details like shirt logos and player skin tones.

A key aspect of this work is the custom dataset we developed for training. Since we could not find a proper football dataset online with high-quality images paired with captions, we created one ourselves. The dataset comprises high-resolution images paired with descriptive captions, and all the images were sourced internally from the broadcast company we collaborate with, ensuring both quality and specificity to the task.

For training, we utilized two RTX 6000 Ada GPUs to accelerate the process and relied on the KohyaSS repository[39]. This repository offers highly customizable Python scripts tailored for fine-tuning various Diffusion Models, enabling precise adjustments to achieve optimal results.

Dataset

The broadcast company provided us with 460 high-quality football images, which will undergo various augmentation techniques to achieve a satisfactory number before training. To accelerate the dataset creation process, we decided to use ChatGPT Plus[40], specifically creating a custom GPT, tailored to generate detailed captions for the included images. This custom GPT was designed to provide rich descriptions of key elements commonly observed in football-related imagery.

The primary goal of the GPT was to generate precise captions capable of explaining the depicted game elements, actions, and the players involved. The sole input required from the user during the image upload

```
accelerate launch flux_train_network.py
    --pretrained_model_name_or_path= #path to FLUX.1-dev
    --clip_l= #path to Text Encoder 1, CLIP_L
    --t5xxl= #path to Text Encoder 2, T5XXL
    --ae= #path to Autoencoder
    --dataset_config= #path to dataset .toml file
    --output_dir= #path to the output directory
    --output_name= #output name
    --save_model_as safetensors
    --sdpa
    --highvram
    --gradient_checkpointing
    --persistent_data_loader_workers
    --max_data_loader_n_workers 2
    --mixed_precision bf16
    --save_precision bf16
    --learning_rate 1e-4
    --max_train_epochs 10
    --train_batch_size 4
    --gradient_accumulation_steps 1
    --network_train_unet_only
    --network_dim 8
    --network_alpha 8
    --network_module networks.lora_flux
    --cache_latents_to_disk
    --cache_text_encoder_outputs_to_disk
    --optimizer_type adamw8bit
    --timestep_sampling shift
    --discrete_flow_shift 3.1582
    --save_every_n_epochs 2
    --model_prediction_type raw
    --seed 42
    --guidance_scale 1.0
    --log_with tensorboard
    --logging_dir "/workspace/"
```

Code 6. Script and parameters used for LoRA training (multi-line is only for better readability).

process was identifying the teams present in the image, which was necessary when describing them for the first time. The GPT was designed with a specific set of instructions to ensure the generation of detailed, accurate, and professional captions for soccer-related images. Its primary objective was to produce captions that adhered to strict guidelines, guaranteeing consistency and clarity across all outputs. The rules emphasized the need for descriptive and neutral language, restricting the captions to observable elements within the image to avoid subjective interpretations or assumptions.

Captions were required to begin with a precise identification of all visible individuals within the image. The GPT was instructed to provide an accurate count of these individuals while clearly distinguishing their roles, such as players, referees, coaches, or spectators, whenever possible. Specific attention was given to players' uniforms, ensuring detailed descriptions of their appearance. When players wore their team's first jersey (home kit), the team name was explicitly mentioned in the caption. For second or third kits, team names were deliberately omitted to avoid ambiguity.

Descriptions of uniforms included a detailed breakdown of jersey and shorts colors, allowing for visual clarity and consistency. For instance, captions highlighted combinations such as "red jersey with white shorts" or "blue jersey with yellow accents". Uniform-specific details were dynamically associated with the respective team within the GPT chat when the user provided this information during the image upload process. Allowing the user to input the relevant context at the time of upload reduced the significant time burden of predefining all potential jersey-and-shorts combinations for every team. The process also accommodated the variability in secondary kits, which often change from season to season. Efficiency and adaptability were thereby ensured while maintaining the accuracy and relevance of captions.

Jersey numbers were included in the caption only when fully visible in the image. The GPT was instructed to link jersey numbers to specific actions performed by players, such as "Player number 9 is kicking the ball". Names of players were added only if clearly visible, ensuring captions relied exclusively on observable and unambiguous information. This approach allowed the GPT to learn the association between jersey numbers and specific player actions effectively.

Players' appearances were meticulously described, with hairstyles categorized into predefined groups such as bald, short, or long hair to ensure consistency across captions. Unique features, including braids, ponytails, or other distinctive arrangements, were incorporated when clearly visible, enhancing both the richness and accuracy of the descriptions. The GPT was explicitly instructed to provide descriptions that avoided any culturally biased or subjective terminology, focusing solely on observable features. Skin tones were described using neutral and

professional language, emphasizing an observational and objective approach. The categorization avoided overly simplistic labels and focused on respectful descriptors that accounted for the diversity of individuals depicted in the images. Such guidelines were implemented to ensure that the captions remained precise, inclusive, and free from any unintended biases, aligning with the overall objective of creating a comprehensive and fair representation of the individuals in the dataset.

Instructing the GPT to adhere to these principles not only enhanced the quality of the captions but also reinforced the importance of ethical considerations in automated descriptive tasks. Every effort was made to ensure that descriptions contributed to an accurate yet impartial portrayal of players, avoiding any potential misinterpretation or cultural insensitivity.

The background of the image was described with attention to the stadium's occupancy and other visible elements. Captions specified whether the stadium was empty (no spectators), half full (moderate crowd), or at full capacity (sold-out stadium). The inclusion of these details provided additional context regarding the atmosphere and scale of the event. Observations also extended to identifying prominent background elements such as billboards, goalposts, or the corner flag whenever they were visible in the image. Providing these contextual elements contributed to a richer and more complete description of the scene, ensuring that the captions captured not only the primary action but also the environment in which it occurred.

The actions and movements of players were described in detail, focusing on activities such as running, kicking, tackling, celebrating, or interacting with the ball or other players. Captions provided clear and specific descriptions, for example: "Two players are contesting possession of the ball in a sliding tackle". Spatial relationships between players, objects, and field landmarks were explicitly outlined to enhance clarity, such as "Player number 10 is positioned near the penalty box". Consistency in terminology was emphasized, ensuring that similar actions were described using the same phrases across different captions. This standardization was critical for training the GPT to recognize and accurately describe recurring patterns of play.

Uniform descriptions were mandatory for all visible players, detailing the colors of both jerseys and shorts worn by each team. Captions included statements like "The player is wearing a blue jersey with white shorts" or "The red team is in a red jersey and red shorts, while the opposing team is wearing a white jersey with black shorts". Such precision provided consistency and allowed for clear differentiation between teams, contributing to the overall comprehensiveness of the captions.

Descriptions focused exclusively on elements that were clearly visible, ensuring accuracy and reliability. Sentences were crafted to be concise, grammatically correct, and written in clear English, prioritizing clarity and readability. These guidelines supported the creation of high-quality, objective captions that aligned with the project's goals for precision and consistency.

The captions were required to be formatted in a single JSON file by the KohyaSS scripts, where each image file path was directly linked to its respective caption (Code 4). This design ensured seamless integration into downstream applications or analyses while maintaining high-quality descriptive standards. Captions were required to be accurate, objective, and formatted correctly for immediate use. By embedding the image paths as keys in the JSON structure, the need for an image_name field was eliminated, avoiding redundancy. This approach streamlined the process, ensuring that each caption was directly associated with its corresponding image path, thereby enhancing the overall efficiency and usability of the dataset.

Training

For training the LoRA model on our football dataset, we leveraged the KohyaSS[39] training pipeline: this was chosen due to its support for fine-tuning large-scale Diffusion Models and its flexibility in accommodating advanced training strategies.

The Kohya pipeline expects the dataset configuration to be defined in a separate .toml file (Code 5), where a variety of configuration parameters can be set[41]. This structured approach enables precise control over the dataset and its preparation.

The main parameters that are important and peculiar to point out are:

- `enable_bucket`: This parameter activates the bucket-based resolution mechanism, which dynamically groups images into resolution "buckets" according to predefined sizes. This approach ensures efficient GPU memory utilization by resizing images to fit within the allocated buckets, leading to faster and more consistent training. The upper resolution limit for bucket allocation is determined by `max_bucket_reso`, meaning that images exceeding this resolution threshold are resized accordingly to remain within the defined constraints.
- `flip_augmentation`: Enables horizontal flipping as a data augmentation technique. This method effectively doubles the dataset size by generating mirrored versions of images, which is particularly useful in the domain of football imagery. Since football actions exhibit left-to-right symmetry without altering the semantics of the scene, this augmentation enhances generalization without introducing inconsistencies.
- `num_repeats`: Specifies the number of times each image-caption pair is repeated during training. This parameter ensures a balanced representation of underrepresented samples, such as specific football-related actions (e.g., headers, tackles), thereby improving the model's ability to generalize. Given that our dataset comprises only 460 images, we set this value to 5 to increase exposure to each image, thereby facilitating effective learning by the model.

The application of this optimizations and augmentations resulted in a dataset of 2300 images, coming from 460×5 repeats, which are randomly and horizontally flipped.

Since we used two Nvidia RTX 6000 Ada GPUs for training, to fully leverage the hardware capabilities the environment was configured using the Accelerate library[42] in a `multi_gpu` distributed type setup. This

enabled efficient distribution of computational tasks across both GPUs, optimizing memory usage and reducing overall training time.

During the configuration of `Accelerate`, we explored integrating the DeepSpeed library[43] to optimize memory efficiency further and enable advanced features such as ZeRO (Zero Redundancy Optimizer). This technique significantly reduces memory requirements by partitioning optimizer states, gradients, and model parameters across GPUs. Specifically, we experimented with ZeRO Stages 1, 2, and 3 to evaluate their compatibility with our setup. However, several challenges arose: Stages 2 and 3 were found to be incompatible with Kohya scripts, while Stage 1, despite being explicitly supported through the `deepspeed` argument, encountered a matrix multiplication issue within PyTorch, preventing successful execution. These limitations necessitated alternative optimization strategies for memory efficiency and scalability. Furthermore, at this time, fine-tuning scripts for LoRA remain highly experimental, with some implementations not even attempting to incorporate DeepSpeed support.

In addition, we attempted to utilize Torch Dynamo[44] with both `'inductor'` and `'cudagraphs'` as backends in an effort to improve performance. Dynamo is a Python-level Just-In-Time (JIT) compiler that intercepts Python bytecode execution to extract PyTorch operations into an FX Graph, which can then be optimized and executed by custom backends. However, we observed no significant speedup during training, suggesting that these optimizations may not yet be well-suited for KohyaSS scripts or our specific training.

Despite some optimization frameworks like TorchDynamo not providing significant speedups, KohyaSS scripts offer various optimization techniques to accelerate training and optimize VRAM usage. The training script (Code 6) we employed leverages several key optimizations to enhance performance and stability:

- `gradient_checkpointing`: This technique reduces memory consumption by recomputing activation functions during the backward pass rather than storing them in GPU memory. This allows for training larger models without exceeding VRAM limits, albeit at the cost of slightly increased computation time[45].
- `highvram`: This optimizes tensors allocation to fully utilize available GPU memory. This is particularly useful for large-scale models, ensuring that memory fragmentation is minimized and computational resources are used efficiently.
- `sdpa` (Scaled Dot-Product Attention): It improves the efficiency of Attention computation on supported hardware by leveraging optimized Key, Query and Values matrices multiplications in Transformers models[46].
- Latent and Text Encoder Output Caching: By enabling `cache_latents_to_disk` and `cache_text_encoder_outputs_to_disk`, we minimize redundant computations, reducing both training time and memory consumption. This is particularly effective when working with large datasets, as it avoids recomputing intermediate representations for each training iteration[39].
- Selective Network Training: The `network_train_unet_only` argument ensures that only the U-Net/DiT component of the model is updated during training, while keeping other components such as the text encoders frozen. This significantly reduces the number of trainable parameters, leading to faster convergence and lower VRAM requirements while still allowing effective fine-tuning of the diffusion model[39].

When fine-tuning with LoRA, two key parameters define the network structure and influence on the original model:

- Network Rank: Determines the dimensionality of the learned weight updates. A higher rank allows the network to learn more expressive adaptations but requires more VRAM and often more epoch to converge.
- Network Alpha: Acts as a scaling factor for these learned weights. In other words, it adjusts how much the LoRA weights contribute to the overall model adjustments.

It is crucial that Network Alpha does not exceed Network Rank. While it is technically possible to set an Alpha higher than Rank, doing so often leads to unintended LoRA behavior and instability in training. The Alpha value directly influences the effective learning rate. If Alpha is smaller than Rank, the strength of the LoRA weight updates is reduced[45]. For example: if Alpha = 8 and Rank = 16, then the effective weight strength is $8 \div 16 = 0.5$. This means that the effective learning rate is only half of the original Learning Rate setting.

The other training arguments are kept at their default values as specified in the Flux and KohyaSS repositories, particularly for configurations related to discrete flow shift and learning rate[32,39]. These defaults ensure compatibility and stability while leveraging the optimizations provided by both frameworks.

With a batch size of 4, a Network Rank (`network_dim`) of 8 and a training over 10 epochs, the total training time was approximately 13 hours. As shown in Fig. 6, the loss remained relatively stable after an initial sharp decrease, indicating that the model reached a steady state without significant fluctuations. While minor oscillations persisted, they were within an expected range due to batch variability.

It is also important to note that LoRA training often does not exhibit a strong loss convergence compared to full fine-tuning. This is because LoRA modifies only a subset of parameters (low-rank matrices) rather than the full model, meaning the error may plateau at a higher value while still producing effective results. Despite this, the model's performance remains robust as long as the loss stabilizes without excessive divergence.

Results and discussion

The proposed reconstruction framework yielded satisfactory results already in the second stage of the process, which leveraged ControlNet for refining image details. However, further improvements were sought by training a custom LoRA to enhance the reconstruction of specific elements such as shirts, logos, and football action scenes.

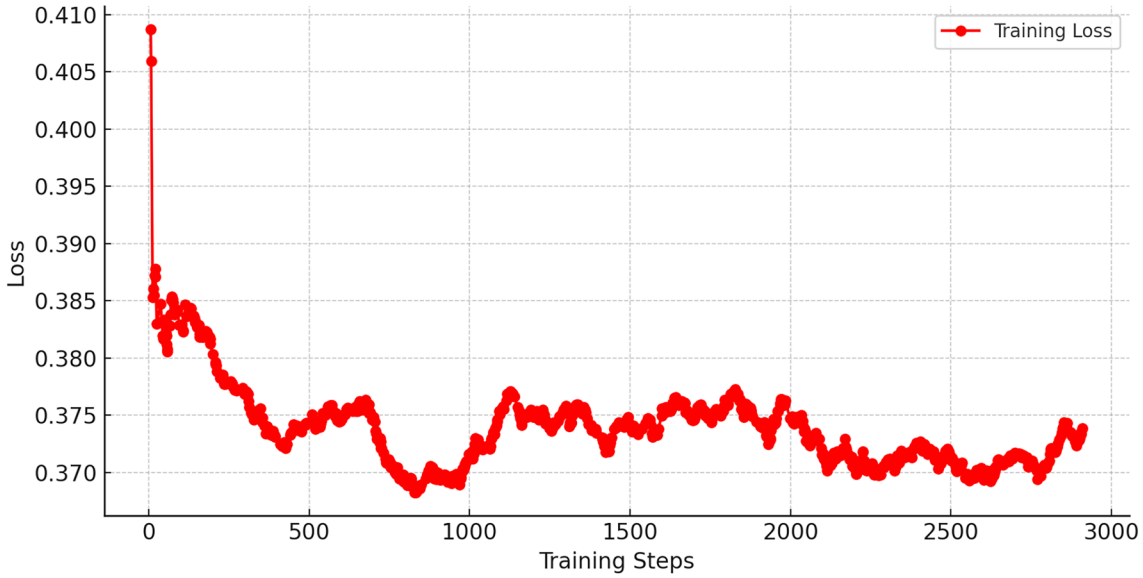

Fig. 6. Training loss over steps.

As illustrated in Fig. 7, the ControlNet pipeline with the LoRA weights attached demonstrated significant improvements in several aspects of the reconstruction process. Notably, the clarity and sharpness of player uniforms were enhanced, resulting in more accurate jersey numbers, colors and less hallucinated writings. Also, overall players physical appearance was improved with less distortions and noise. In particular, finer details, such as hands, shoes, and general players pose, exhibited greater accuracy and definition in the LoRA-enhanced outputs. This improvement was particularly evident in fast-moving actions, where diffusion-based reconstructions traditionally struggle to retain structure due to motion blur. The ability of the LoRA-tuned model to recover these details suggests that the fine-tuning process successfully learned and integrated domain specific visual cues from the football dataset.

Despite these advances, some difficulties remained; in particular, recreating the intricate details of shirt logos and brand lettering proved challenging, even with our trained LoRA. This outcome was anticipated, as our experiment relied on a relatively small dataset and was intended primarily to evaluate LoRA's ability to preserve macro-structural features, such as player pose and silhouette, rather than to reproduce fine-grained textures. Overriding Flux's strong pretrained biases to capture team and logo-specific details will require a substantially larger, domain-specific corpus and more extensive adaptation strategies, which were beyond the scope of this preliminary study. Given the promising results on structural accuracy, we plan to pursue this direction by assembling an expanded dataset of broadcast cut-outs to enable the model to learn team-specific jersey details and further improve reconstruction of logos and text. It is important to note that the code for implementing the LoRA remains substantially the same as Code 3, except for the addition of the line `pipe.load_lora_weights(cross_attention_kwargs="scale": 0.9)`, which enables developers to control the extent to which the LoRA weights are incorporated into the original model (with a value of 1 being equivalent to using the fully fine-tuned LoRA)[47].

A particularly compelling example of the benefits provided by the trained LoRA can be observed in the reconstruction of a goalkeeper image, Fig. 8. As shown, the image generated without using LoRA exhibited noticeable deformities; notably, the two legs of the goalkeeper appeared to fuse into one. In contrast, when employing our trained LoRA, the reconstruction maintained a clear and distinct separation between the two legs. This improvement is especially significant considering the extremely low starting resolution of 64 x 64, which typically poses substantial challenges for preserving fine structural details.

Furthermore, it is worth noting that the LoRAs created by the KohyaSS repository adhere to the Black Forest Lab (BFL) format, which differs from those trained with Diffusers-based repositories[47]. The KohyaSS repository provides a Python script called *convert_flux_lora.py* to convert LoRAs between the BFL format and the Diffusers format. We conducted inference experiments with both formats after converting our LoRA into Diffusers, and observed no notable differences in inference speed or output quality, highlighting that in the latest versions of Diffusers ($> 0.30.3$), any discrepancies between the formats have largely become negligible.

Because raw broadcast cut-outs lack paired high-resolution references, we performed a complementary evaluation by collecting web images of football actions, synthetically degrading them to our LR domain, and running them through the pipeline. As shown in Fig. 9, LoRA fine-tuning markedly reduces both structural and chromatic artifacts: in the first sample, the goalkeeper's and player's feet, which fuse together without LoRA, remain anatomically distinct; in the second, the ball's color and shape closely match the ground truth while no spurious wristband appears; and in the third, LoRA restores the correct number of fingers and reproduces a tattoo that the non-LoRA output omits. Importantly, our LoRA fine-tuning and the associated image-metadata processing did not target specific team logos or uniform details at this stage; rather, we focused on improving

Fig. 7. Comparison of reconstruction quality across two stages with and without LoRA fine-tuning. The first column represents the first stage of reconstruction, with the left image being the output without LoRA and the right image incorporating LoRA. The second column represents the second stage of reconstruction (ControlNet), following the same pattern: left without LoRA and right with LoRA.

overall player anatomy preservation and color fidelity. Given these encouraging results, we will extend our fine-tuning to a much larger, domain-specific dataset that incorporates diverse team logos and uniforms to also enhance the reconstruction of brand logos and jersey details.

To further quantify reconstruction fidelity, we applied the same MAE metric described earlier: we computed the MAE between our manually degraded LR images and their high-resolution ground truth, and between our pipeline's reconstructions and the same ground truth. As shown in Table 2, the average MAE rises modestly from 19.0 (degraded vs GT) to 26.2 (generated vs GT), reflecting that our model often adds plausible high-frequency details, details that a pixel-wise loss nonetheless penalises even when they improve perceptual quality. Interestingly, one image (img 3) actually sees its MAE drop from 16.2 to 15.1 after reconstruction, indicating that the framework can recover lost structure and reduce absolute error in certain cases. The per-image MAE range (15.1–38.0) further underscores that MAE alone does not fully capture the anatomical and chromatic fidelity gains delivered by our multi-stage pipeline. In future work, we will complement MAE with perceptual metrics (e.g. SSIM, LPIPS) to better reflect the true visual improvements achieved.

Despite the strong performance of our two-stage pipeline, which is expressly designed to minimise hallucinations by running the LoRA-fine-tuned Flux model at low strength values (0.55–0.66) so that the original signal dominates Stage 1 before texture refinement in Stage 2, we acknowledge that occasional hallucinations can still arise in the second stage. Nevertheless, for the features that matter most to sport analysts (structural accuracy and colour fidelity) the residual error rate remains acceptable. Crucially, the framework is not yet production-ready: it is currently deployed only for offline post-match analysis, where operators have sufficient time to inspect each frame and discard any image that exhibits implausible content.

In summary, the integration of ControlNet and LoRA within our pipeline proved highly effective for football broadcast image reconstruction. The results indicate that leveraging targeted fine-tuning techniques can significantly enhance structural fidelity, particularly for elements that are crucial for accurate and aesthetically convincing sports imagery. It is important to note, however, that despite these advancements, controlling the generated output remains a nuanced challenge. While the modularity of Diffusion Models enhanced by

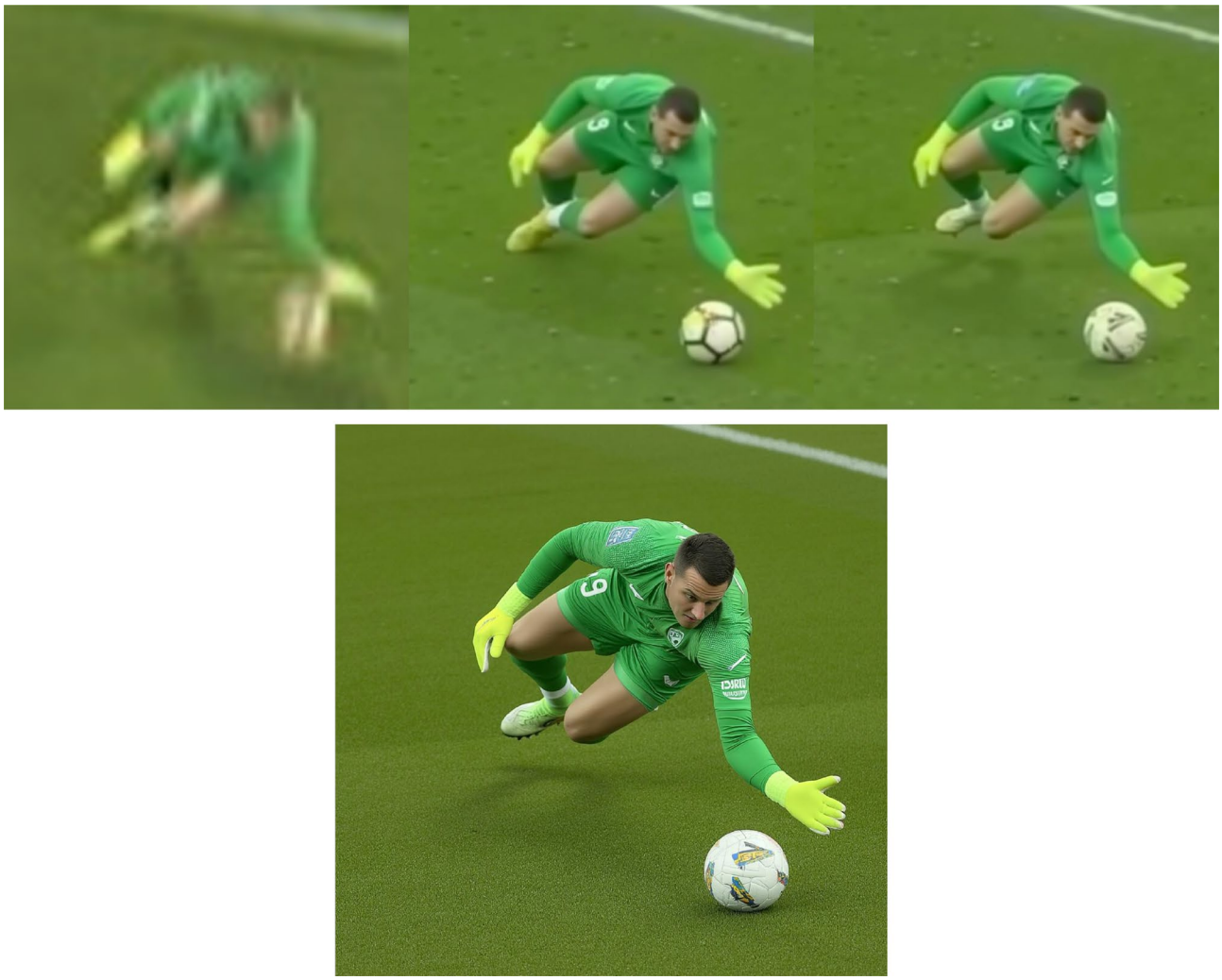

Fig. 8. The first row presents the original raw frame at 64×64 resolution, followed by the reconstructed output from the first stage without LoRA, and then the output incorporating LoRA. The second row highlights the final reconstruction result after the second stage (ControlNet), demonstrating the significant enhancement in detail and realism achieved with LoRA and ControlNet.

ControlNet and LoRA offers a straightforward pathway to inject domain-specific details, this same flexibility can sometimes lead to unpredictable variations, where minor tweaks in input or parameter settings may result in disproportionate changes in the output. This duality underscores the ongoing need for more robust control strategies to ensure consistent, high-quality reconstructions in complex visual scenarios.

Conclusion

This study explored the use of Diffusion Models for reconstructing and enhancing low-resolution football broadcast images. By integrating ControlNet and LoRA fine-tuning, we achieved substantial improvements in image clarity and structural fidelity. Our results show that while ControlNet alone effectively refined general details, the addition of a custom LoRA significantly enhanced the reconstruction of jerseys, logos, and motion elements, ensuring greater precision in uniform textures. These enhancements highlight the effectiveness of domain-specific fine-tuning for specialized image restoration tasks.

Beyond football broadcasts, our findings demonstrate the broader potential of diffusion-based reconstruction techniques in applications requiring high-fidelity image recovery from degraded inputs. In this context, we are considering the development of specialized LoRA modules trained specifically on individual football teams or players. Such targeted fine-tuning is expected to further enhance reconstruction capabilities by capturing team-specific or player-specific visual features, thereby improving the accuracy of restoring details like uniform patterns, emblems, and distinctive motion characteristics.

Moreover, our experiments revealed that in some cases the initial reconstruction stage produced better results without applying LoRA. To address this variability, we are planning to implement a parallel execution strategy that concurrently generates both LoRA-enhanced and non-LoRA outputs. This approach will facilitate

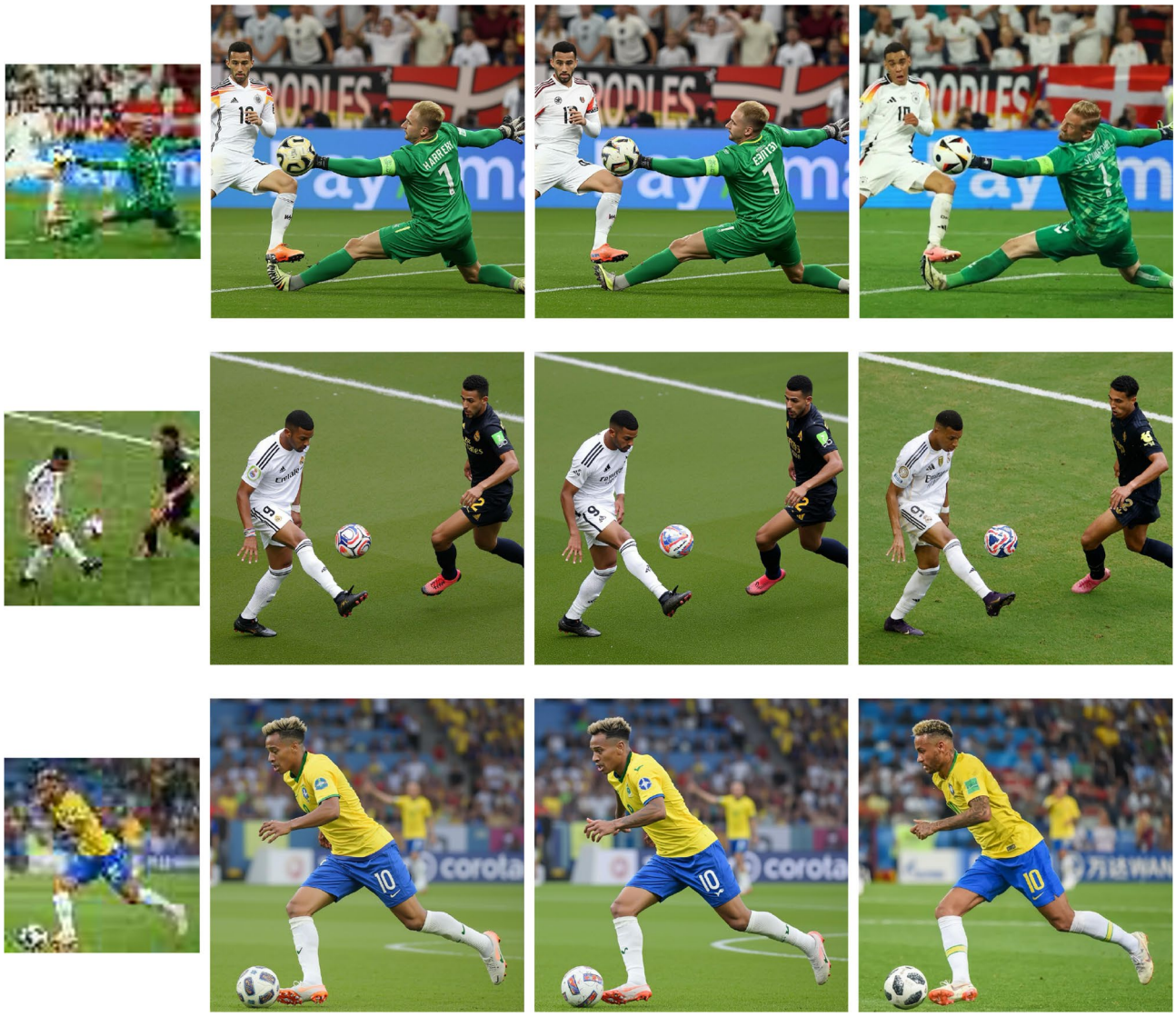

Fig. 9. The first column displays the downsampled images at a resolution of 64×64. The second and third columns present the reconstructed images without and with LoRA, respectively. The fourth column contains the ground-truth images.

rapid comparison and enable the selection of the optimal reconstruction based on the specific characteristics of each input image, enhancing both the efficiency and quality of the process.

In addition, we plan to develop a user-friendly graphical user interface (GUI) to streamline the process and make the pipeline accessible to less technical users. This GUI will simplify the deployment and use of our system, further broadening its practical applicability.

Furthermore, future work will focus on scaling our approach to a larger dataset containing more than 1,500 images, further refining the model's ability to reconstruct football-related details with higher accuracy. We also plan to explore the possibility of fine-tuning a Vision Model for automatic, offline captioning of dataset images, reducing reliance on external AI services for metadata generation. Finally, we intend to explore the potential for converting these images into three-dimensional model representations using future Img-to-3D models. By leveraging generative AI with targeted fine-tuning and adaptive reconstruction strategies, this research sets a foundation for next-generation image enhancement techniques, effectively bridging the gap between traditional upscaling methods and AI-driven super-resolution.

Image ID	Degraded vs GT	Generated vs GT
img1	21.2	38.0
img2	19.5	27.1
img3	16.2	15.1
img4	25.3	31.1
img5	18.1	25.2
img6	19.8	24.4
img7	21.6	25.8
img8	19.3	27.0
img9	17.9	21.6
img10	18.6	27.2
img11	15.9	24.2
img12	21.6	35.5
img13	20.5	25.9
img14	16.7	17.8
img15	17.8	27.2
img16	19.6	21.9
img17	19.0	23.0
img18	20.9	24.4
img19	19.6	28.6
img20	18.0	32.3
img21	20.2	25.5
img22	22.6	34.2
img23	21.7	22.3
img24	17.6	28.3
img25	16.2	31.5
img26	17.8	26.8
img27	19.4	21.9
img28	13.6	20.8
img29	18.8	23.5
img30	16.2	27.5
MAE	**19.0**	**26.2**

Table 2. Mean absolute error (MAE) between degraded/generated images and the ground-truth across 30 web samples.

Data availability

The images used in this study are subject to copyright restrictions held by a private entity. As such, these images are not publicly available. However, access to the images may be granted upon reasonable request and after a review process to ensure compliance with the applicable copyright restrictions. Please direct all requests to the corresponding author, Luca Martini (email: luca.martini@edu.unige.it), who will provide further instructions.

Received: 31 October 2025; Accepted: 3 December 2025
Published online: 15 December 2025

References

1. Hugging Face. Diffusers Documentation. https://huggingface.co/docs/diffusers/en (2025). **Accessed 2025-01-17**.
2. BlackForestLabs. Announcing Black Forest Labs. https://blackforestlabs.ai/announcing-black-forest-labs/ (2025). **Accessed 2025-01-20**.
3. Jasperai. Flux. https://huggingface.co/jasperai/Flux.1-dev-Controlnet-Upscaler (2024). **Accessed 2025-01-16**.
4. Hu, E. J. et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations* (2022).
5. Ho, J., Jain, A. & Abbeel, P. Denoising diffusion probabilistic models. In *Neural Information Processing Systems* (2020).
6. Ledig, C., Theis, L., Huszár, F., Caballero, J. & Others. Photo-realistic single image super-resolution using a generative adversarial network. In *Conference on Computer Vision and Pattern Recognition* (2017).
7. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations* (2014).
8. Wang, X. et al. Esrgan: Enhanced super-resolution generative adversarial networks. In *Computer Vision – ECCV 2018 Workshops* (2019).
9. Wang, X., Xie, L., Dong, C. & Shan, Y. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *IEEE/CVF International Conference on Computer Vision Workshops* (2021).
10. Chauhan, K. et al. Deep learning-based single-image super-resolution: A comprehensive review. *IEEE Access* **11**, 21811–21830 (2023).

11. Zhang, L., Rao, A. & Agrawala, M. Adding conditional control to text-to-image diffusion models. In *IEEE/CVF International Conference on Computer Vision* (2023).
12. Cuenca, P. and Paul, S. Using LoRA for Efficient Stable Diffusion Fine-Tuning. https://huggingface.co/blog/lora (2023). **Accessed 2025-01-17**.
13. Martini, L., Iacono, S., Zolezzi, D. & Vercelli, G. V. Advancing persistent character generation: Comparative analysis of fine-tuning techniques for diffusion models. *AI* **5**, 1779–1792 (2024).
14. Peebles, W. & Xie, S. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision* (2023).
15. Yan, Y. & Guo, Y. Partial label unsupervised domain adaptation with class-prototype alignment. In *International Conference on Learning Representations* (2023).
16. Xiao, H., Wang, X., Jun, W. & Cai, J. Single image super-resolution with denoising diffusion gans. *Sci. Rep.* **14**, https://doi.org/10.1038/s41598-024-52370-3 (2024).
17. Shi, Y., Zhang, X., Jia, Y. & Zhao, J. Multi-scale adversarial diffusion network for image super-resolution. *Sci. Rep.* **15**, https://doi.org/10.1038/s41598-025-96185-2 (2025).
18. Wang, Y. et al. SinSR: Diffusion-based image super-resolution in a single step. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024).
19. Chen, Z. et al. BI-DiffSR: Binarized diffusion model for image super-resolution. In *Advances in Neural Information Processing Systems* (2024).
20. Zhou, S., Yang, P., Wang, J., Luo, Y. & Loy, C. Upscale-a-video: Temporal-consistent diffusion model for real-world video super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024).
21. Comfyanonymous. ComfyUI. https://github.com/comfyanonymous/ComfyUI (2025). **Accessed 2025-01-20**.
22. NVIDIACorporation. nvidia-ml-py: NVIDIA Management Library Python Bindings. https://pypi.org/project/nvidia-ml-py/ (2025). **Accessed 2025-01-20**.
23. PillowDevelopers. Concept: Filters. https://pillow.readthedocs.io/en/stable/handbook/concepts.html#concept-filters (2025). **Accessed 2025-01-20**.
24. Upscayl, AI Corp. Upscayl: Free and Open-Source AI Image Upscaler. https://upscayl.org/ (2025). **Accessed 2025-01-20**.
25. StabilityAI. Stable Diffusion X4 Upscaler. https://huggingface.co/stabilityai/stable-diffusion-x4-upscaler (2025). **Accessed 2025-01-20**.
26. HuggingFace. Image-to-Image with Diffusers. https://huggingface.co/docs/diffusers/v0.32.2/en/using-diffusers/img2img (2023). **Accessed 2025-01-20**.
27. StabilityAI. Stable Diffusion XL Base 1.0. https://huggingface.co/stabilityai/stable-diffusion-xl-base-1.0 (2025). Accessed 2025-01-20.
28. StabilityAI. Stable Diffusion 3.5 Medium. https://huggingface.co/stabilityai/stable-diffusion-3.5-medium (2025). **Accessed 2025-01-20**.
29. StabilityAI. Stable Diffusion 3.5 Large. https://huggingface.co/stabilityai/stable-diffusion-3.5-large (2025). **Accessed 2025-01-20**.
30. Civitai. Civitai - A Hub for AI Models. https://civitai.com/ (2025). **Accessed 2025-01-20**.
31. SG161222. RealVisXL V4.0. https://huggingface.co/SG161222/RealVisXL_V4.0 (2025). **Accessed 2025-01-20**.
32. Black Forest Labs. FLUX.1 [dev]. https://huggingface.co/black-forest-labs/FLUX.1-dev (2025). **Accessed 2025-01-17**.
33. OpenAI. CLIP ViT-Large Patch14. https://huggingface.co/openai/clip-vit-large-patch14 (2025). **Accessed 2025-01-20**.
34. Google. T5 v1.1 XXL. https://huggingface.co/google/t5-v1_1-xxl (2025). **Accessed 2025-01-20**.
35. HuggingFace. Flux Pipeline API Documentation. https://huggingface.co/docs/diffusers/v0.32.2/api/pipelines/flux (2023). **Accessed 2025-01-20**.
36. HuggingFace. Memory Optimizations for Stable Diffusion 3. https://huggingface.co/blog/sd3##memory-optimizations-for-sd3 (2023). **Accessed 2025-01-20**.
37. HuggingFace. Fast Diffusion Tutorial. https://huggingface.co/docs/diffusers/v0.32.1/en/tutorials/fast_diffusion (2023). **Accessed 2025-01-20**.
38. HuggingFace. ControlNet API Documentation. https://huggingface.co/docs/diffusers/v0.16.0/en/api/pipelines/stable_diffusion/controlnet (2023). **Accessed 2025-01-20**.
39. KohyaSS. SD Scripts. https://github.com/kohya-ss/sd-scripts (2025). **Accessed 2025-01-20**.
40. OpenAI. OpenAI products. https://openai.com/ (2025). **Accessed 2025-01-20**.
41. KohyaSS. SD Scripts Configuration Documentation. https://github.com/kohya-ss/sd-scripts/blob/sd3/docs/config_README-en.md (2025). **Accessed 2025-01-20**.
42. HuggingFace. Accelerate Documentation. https://huggingface.co/docs/accelerate/en/index (2025). **Accessed 2025-01-20**.
43. Microsoft. DeepSpeed. https://github.com/microsoft/DeepSpeed (2025). **Accessed 2025-01-20**.
44. PyTorchDevelopers. Torch Compiler: Dynamo Overview. https://pytorch.org/docs/stable/torch.compiler_dynamo_overview.html (2025). **Accessed 2025-01-20**.
45. Bmaltais. LoRA Training Parameters. https://github.com/bmaltais/kohya_ss/wiki/LoRA-training-parameters (2025). **Accessed 2025-01-20**.
46. HuggingFace. Optimization with Torch 2.0. https://huggingface.co/docs/diffusers/en/optimization/torch2.0 (2025). **Accessed 2025-01-20**.
47. HuggingFace. Loading Adapters in Diffusers. https://huggingface.co/docs/diffusers/en/using-diffusers/loading_adapters (2025). **Accessed 2025-01-20**.

Author contributions

L.M. developed the methodology, developed the code, performed the experiments, and wrote the paper. D.Z. and S.I. created the dataset, developed the methodology, and revised the paper. L.O. and G.V.V. developed the methodology and revised the paper.

Funding

This work was partially supported by the Ecosystem "RAISE–Robotics and Artificial Intelligence for Socio-Economic Empowerment" (NextGenerationEU code ECS 00000035) PNRR-M4C2-I1.5.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to L.M.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.