Breaking the Vision: Assessing and Mitigating the Impact of Video Artifacts on ML Models in Industrial Use Cases

Marco Reisacher*, Ann-Kristin Bergmann*‡, Andreas Blenk*, Stefan Schmid†
* *Siemens AG, Munich, Germany*
† *Technical University Berlin, Berlin, Germany*
‡ *Technical University Munich, Munich, Germany*

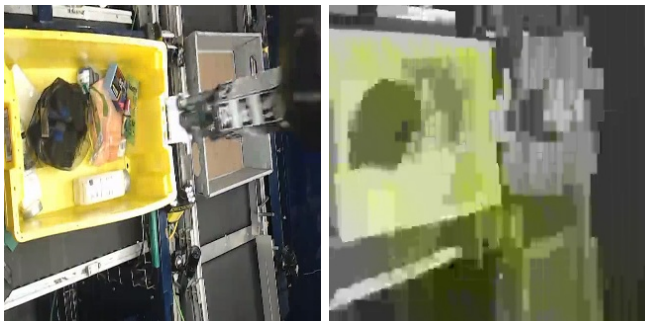
***Abstract*—Machine learning models are highly dependent on the quality of their input data. In industrial settings, where video data is transmitted to the model located in the cloud over a network, transmission artifacts (e.g., congestion or losses) can degrade model performance. These performance issues can compromise process quality and result in costly errors. Despite the increasing interest in using Machine Learning (ML) models for video tasks in industrial use cases, existing research has not adequately assessed the impact of individual video artifacts on model performance, nor has it specifically examined the performance of video-based ML models under these conditions.**

This study aims to fill this gap by considering the impact of degraded data on the model behavior. As a case study, we consider a defect detection scenario where, e.g., a manufacturing robot is monitored for defections using video. To this end, we train a Multiscale Vision Transformer and use an approach that systematically introduces various artifacts, such as bitrate reduction and pixelation. We then assess their impact on the model's performance accordingly. Hence, we provide the first insights into possible mitigations.

I. Introduction

As the Industry 4.0 paradigm drives advancements in modern manufacturing, there is an increasing reliance on intelligent sensors, devices, and machinery to gather and analyze production-related data [1]. This shift enables early error detection and improves overall process efficiency [2]. Machine Learning (ML) introduces great opportunities to deal with the growing complexity of industrial systems, coupled with the increasing volume of data. for effective data management and interpretation [3], [4]. For example, video-based defect detection in production lines is gaining traction as automatization reduces manual labor and boosts production efficiency, addressing the inconsistencies and time-consuming nature of traditional visual inspections prone to human error [5]–[7].

The Problem: A critical problem in this context is the strong dependency of ML model performance on data quality [3], [8]. As cloud manufacturing becomes prevalent, networks often transmit production-related data instead of processing it at the collection point. This process can introduce various quality issues, known as artifacts, due to, For example, limited network resources and overloaded compute units, as seen in Fig. 1, which can significantly degrade the performance of ML models [9], [10]. Despite the increasing adoption of ML in manufacturing, we currently lack a good understanding of how models are affected by such conditions. This is, however, critical: poor ML performance due to data artifacts can lead to inefficiencies in production, resulting in financial losses.



(a) Baseline Image. (b) Reduced Bitrate Image.

Fig. 1. Baseline image of a manufacturing process in good quality (a) and degraded quality (b), due to reduced bitrate.

The Challenge: The key challenge lies in accurately characterizing the impact of various artifacts on the performance of ML models. As industrial systems evolve towards build-to-order models that demand more flexible processes, and with the introduction of advanced models like Multiscale Vision Transformers (MViTs) for image and video analysis, it is crucial to understand how these artifacts impact model effectiveness [11]–[13]. Despite MViT's promising performance, this novel approach is still under evaluation for applications such as defect detection in images and videos [14].

Our Contribution: We investigate the impact of video artifacts, such as pixelation, on models. In particular, we perform an empirical study with the emerging and popular MViT models. As a case study, we consider an industrial production setting with a robot performing a pick-and-place task, which is observed by a camera to determine malfunctions in the process. We conduct this investigation by training the model on the Amazon Robotic Manipulation Benchmark Dataset [15] for defect detection tasks. Our approach introduces video artifacts resulting from camera errors or network transmission into the dataset. We then test the model under these conditions and assess its performance. We assess how these artifacts impact the model's performance, providing insights that can

guide the optimization of data acquisition and transmission processes to improve ML models in real-world industrial applications.

This paper builds upon our previous investigation into image-based machine learning behavior by extending the focus beyond static data to include video formats [16], [17].

Paper Organization: The remainder of this paper is organized as follows. In Sec. II, we introduce the evaluation methodology. Sec. III shows the evaluation results and discusses key findings. Finally, Sec. V summarizes the observations and gives some outlooks on future research.

II. Methodology

This section presents the architecture of the MViT, the dataset and the metrics characterizing the ML model.

A. Dataset

As a benchmark dataset, we chose the Armbench dataset, published by Amazon in 2023 [15]. The dataset consists of 104,075 videos with an average duration of 5.93 seconds and an average size of 2.51 MB. The videos were recorded at $720p$ resolution, 30 FPS, and are stored in (IPPP) frame pattern. In the videos, a robotic manipulation work cell is featured during a pick-and-place process on a mixed collection of items. The task of the robotic arm is to pick one item at a time and transfer it to moving trays. In the dataset, three robot-induced defect categories are defined for this process: The robot picks several items at once, the item gets opened, and the item gets disassembled. In the following, we will refer to these categories as *nominal*, *open* and *deconstruction*. In the context of video-based defect detection, the focus will be on the defect categories *open* and *deconstruction*, as these types of errors can occur at any point in the process and are, therefore, best detected on videos. The dataset is naturally unbalanced regarding the occurrences of defective and non-defective cases, as the defect occurrences are relatively rare in real-world scenarios. Since not detecting an error is more costly than falsely marking a nominal case as a defect, we downsampled the nominal cases to 50% and kept the split of 11% of nominal cases in validation data and test data. Tab. I shows the train, validation, and test split. Additionally, we used a split of $70:15:15$ for the defect cases.

TABLE I
Absolute Amount of Labels in Different Data Splits

Label	Train	Validation	Test
nominal	2919	5000	5000
open	1520	323	322
deconstruction	1399	303	303

No other sizable video dataset currently exists that captures a comparable range of items and setups in robotic manipulation, as most other real-world datasets rely on a closed-set approach with a constrained number of object types. This limitation potentially hinders algorithms from adapting to new objects, which presents a risk to large-scale processes [15]. Most other research for defect detection focuses on anomaly detection or surface defect identification, with relevant datasets available in [18]–[20].

B. Preprocessing of the Dataset

In the data preprocessing, the video frames are extracted and stored as a three-dimensional array with a 24-bit representation per pixel. Subsequently, the individual frames are cropped to a defined region of interest and resized to a consistent to quadratic shape of 320×320. We then store the frames as individual image files for further analysis.

During training, the model further transforms data before processing. It flips the frames horizontally with a probability of 0.5, normalizes the pixel values, and crops each frame randomly from 90% and 100% of the original frame size. The model then rescales the crop to a standardized size of 224×224 pixels. Finally, the model uniformly selects 32 frames from the video sequence as input.

C. MViT Architecture

Multiscale Vision Transformers are an advanced adaptation of Vision Transformers (ViTs) designed to enhance multiscale feature representation and improve efficiency in video-based computer vision tasks.

Instead of processing videos at a single resolution or scale, MViTs introduce hierarchical structures that dynamically adjust the level of detail at different stages of processing. This allows them to maintain fine-grained local details while still benefiting from global context awareness. The key innovations in MViTs include:

1) **Hierarchical Tokenization** – Images are progressively downsampled, with features extracted at multiple scales.
2) **Multiscale Attention Mechanisms** – Self-attention operates at various levels of abstraction, capturing both local and global features efficiently.
3) **Hybrid Architectures** – Many MViT models combine CNN-like inductive biases with transformer-based architectures for better performance.

Unlike traditional ViTs, which rely on fixed-size patches, MViTs use hierarchical tokenization and multiscale attention mechanisms. These techniques enable effective capturing of local and global image contexts.

MViTs are used in various domains, such as medical imaging, remote sensing, and object detection. Integration with convolutional neural networks (CNNs) and other hybrid architectures further enhance their performance [21], [22].

D. MViT Evaluation Setup

We train our model for up to 30 epochs or until convergence, using a batch size of 4 and a depth of 16 layers. The learning rate starts at 0.00003 and decreases to a minimum of 0.0000015 during training. For optimization, we employ the AdamW optimizer along with a CosineAnnealingLR scheduler. Additionally, we configure the model to double both the embedding dimension and the attention head dimension at layers 1, 3, and 14. We assign the stride sizes for pooling the query (q) as $[1, 2, 2]$ and determine the stride values for

TABLE II
PERFORMANCE METRICS FOR DIFFERENT APPROACHES TO HANDLING CLASS IMBALANCE, EVALUATED AT THE EPOCH WITH THE HIGHEST WEIGHTED AVERAGE F1-SCORE.

Model \ Metrics	**Weighted Mean F1-Score**	**Deconstruction F1-Score**	**Open F1-Score**	**Nominal F1-Score**
BCE_5000 (1)	0.924266	0.739726	0.351852	0.936066
WBCE_5000 (2)	0.902162	0.733229	0.346549	0.946711
BCE_Balanced (3)	0.925107	0.754472	0.451613	0.964789

the temporal, height, and width dimensions at these layers. Furthermore, we initialize the key-value (kv) pooling stride at $[1, 8, 8]$ and adapt it further based on the query pooling ratio. Finally, we configure the kernel size for pooling operations across the query, key, and value streams as $[3, 3, 3]$. We refer to the default settings outlined in [23] for more details.

The model is initialized with pre-trained weights from the Kinetics-400 dataset [24], as provided by PyTorchVideo [23]. In our set-up, the classification head outputs a two-channel vector representing two categories: *open* and *deconstruction*. Each channel in this vector represents the probability of the input belonging to one of these two classes. The video is classified as non-defective if both predictions are below a threshold. This binary classification approach allows the model to predict binary classifications and utilizes Binary Cross-Entropy (BCE) as the loss function. To address data imbalance, we explore three distinct approaches. The approaches differ in the data split and the loss function used:

1) BCE loss with 5000 nominal videos, as described in [15]
2) Weighted BCE loss with the same split
3) BCE loss with a balanced train set with an equal number of defective and no-defective samples

We compare the average weighted F1-Scores of various approaches, as presented in Tab. II, and select the model with the highest average weighted F1-Score. In this instance, the model trained on the balanced dataset performs the best. Additionally, it achieves the best balance between the F1-Scores for the *open*, *deconstruction*, and *nominal* classes.

E. Evaluation Metrics

We chose the F1-Score (1) as the metric to evaluate the model's performance on various video modifications. Since the dataset is imbalanced and the nominal class significantly outnumbers the defective classes, accuracy can be misleading as it might reflect a bias towards the majority class. The F1-Score, however, considers both precision (the proportion of true positives among all positive predictions) and recall (the proportion of true positives among all actual positives), providing a more balanced assessment of the model's performance. This is particularly valuable because misclassifications often lead to changes in precision and recall metrics. A class with increased recall may experience a rise in false positives, causing precision to decrease. Conversely, high precision in a class may result from having a small number of true positives.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{1}$$

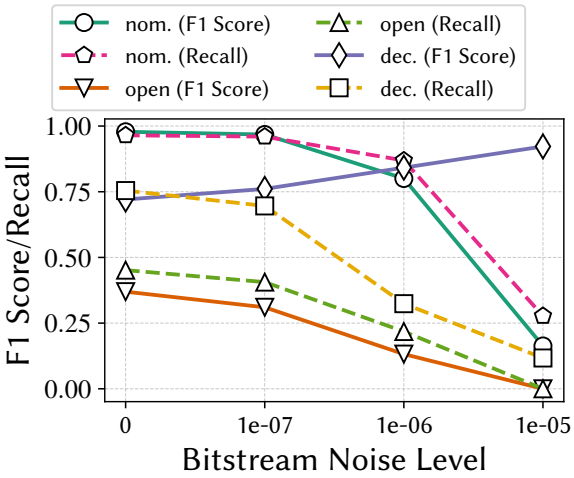


Fig. 2. Performance evaluation based on recall and F1-score for videos of the nominal (nom.), open, and deconstruction (dec.) classes that have been altered using a bitstream noise filter.

In some cases, we include the recall (2) in the evaluation as it helps provide insights into which class the model assigns the misclassified samples. Additionally, recall effectively captures the most costly scenario, where the model incorrectly classifies a defect as non-defective.

$$recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2}$$

A weighted average shows the ML model's overall performance, taking the number of tested occurrences per class into account.

III. EVALUATION

In this section we evaluate the impact of different modifications on the model's performance. We apply all video modifications before preprocessing, encode the videos using H.264 in MPEG-4 format and process them in the same manner as the benchmark dataset. For each modification, we test the model and analyze its performance.

A. Bitstream Noise

With the bitstream noise filter available in FFmpeg [25], we introduce noise into the test data. After applying the noise filter, we encode the transformed data and store it in the Huffyuv codec format, a lossless video codec, ensuring the preservation of the noisy data in its transformed state. From a technical perspective, the filter processes each byte of the video data

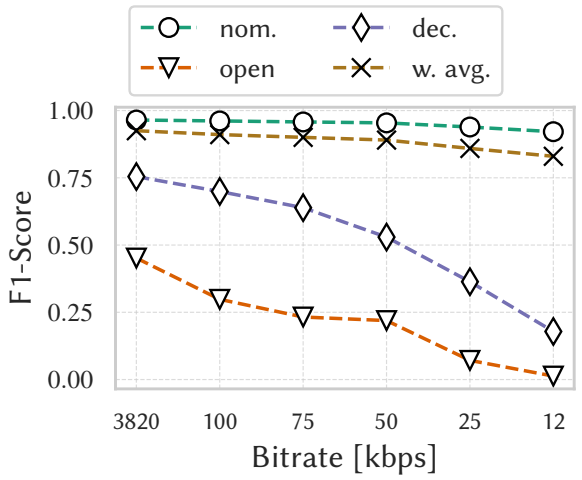


Fig. 3. Performance evaluation based on the F1-Score for videos of the nominal (nom.), open, and deconstruction (dec.) classes with altered bitrate. A weighted average (w. avg.) shows the average score over all classes.

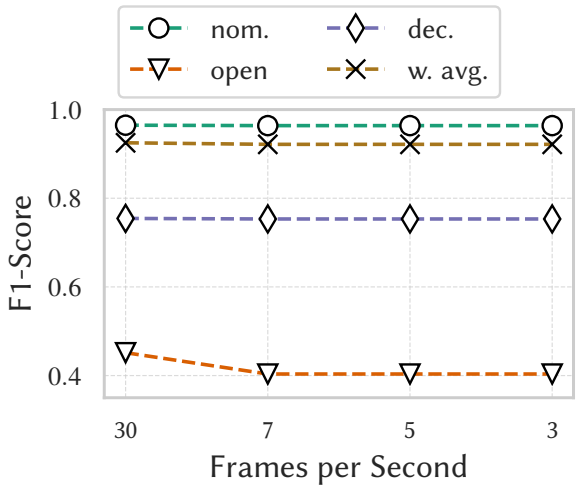


Fig. 4. Performance evaluation based on the F1-Score for videos of the nominal (nom.), open, and deconstruction (dec.) classes with reduced frames per second. A weighted average (w. avg.) shows the average score over all classes.

sequentially. The filter updates an internal state variable for each byte by adding the current byte value, incremented by one. This updated state then determines whether the byte should be modified. Specifically, if the updated state modulo the specified noise level equals zero, the byte is replaced with the current value of the state. The bitstream filter damages the content of the byte or drops it entirely while leaving the video intact. This filter emulates corruption during video transmission caused by network errors such as packet loss or jitter. Finally, the H.264 codec encodes and processes the video like the benchmark data. We evaluate the data for noise levels of 1×10^{-5}, 1×10^{-6} and 1×10^{-7}.

As the amount of bitstream noise increases, the recall for both the *open* and *nominal* classes declines significantly, as shown in Fig. 2. This indicates that the model correctly identifies only 16.3% of the true positive cases for the *nominal* class and fails to detect any true positive cases for the *open* class. Given that the recall for the *deconstruction* class increases while its F1 score decreases, this implies that precision decreases more significantly than recall increases. Consequently, it suggests that the model incorrectly labels many falsely classified cases as deconstruction defects. This outcome is particularly interesting because it contrasts with other modifications, where the model tends to classify most cases as non-defective as the degradation increases. In this case, however, non-defective cases are classified as defective.

- *Takeaway:* Increasing bitstream noise reduces recall for the *open* and *nominal* classes, leading to a significant drop in true positive detections while also increasing false classifications as *deconstruction* defects.

B. Bitrate Reduction

We adjust the bitrate with the FFmpeg bitrate option [26], reducing the average bitrate from 3820 kbps across all videos to a range of values between 100 and 12 kbps. This range was chosen because wired fieldbus systems typically offer bitrates from hundreds of kilobits to tens of megabits per second, and wireless fieldbus systems are expected to have comparable bitrates [27]. With decreasing bitrate, the defect classes *open* and *deconstruction* experience substantial declines in the F1-Score, with decreases up to 98.21% and 74.14%, respectively, as seen in Fig. 3. This suggests that lower bitrates severely degrade the reliability of detecting these defect types. In contrast, the *nominal* class shows minimal impact, with only a 3.29% reduction, indicating that nominal conditions are less sensitive to bitrate reductions. The F1-Score of the nominal class decreases as defects are misclassified as non-defective.

- *Takeaway:* Decreasing bitrate significantly reduces F1-Score for the *open* and *deconstruction* defect classes, following a linear trend. In contrast, the *nominal* class remains largely unaffected, suggesting that lower bitrates severely impact defect detection reliability.

C. Framerate Reduction

To determine if reducing the frames per second (FPS) could serve as an alternative to transmitting lower-quality frames, we evaluated the model at FPS values ranging from 7 to 3. The results, depicted in Fig. 4, show no decline in F1-Score for the classes *deconstruction* and *nominal*. For the open class, the F1-Score decreases by 16.66%. This result can be partially explained by the model's preprocessing, which involves sampling and then uniformly selecting a subset of frames. It is particularly noteworthy that the model consistently classifies the same subset of classes across all tested reduction values. We can conclude that lowering the frames per second provides a viable alternative to reducing video resolution for network transmission, as the defect classes remain reliably detectable.

- *Takeaway:* Reducing FPS does not significantly impact defect detection performance, making it a viable alternative to lowering video resolution for network transmission while preserving model reliability.

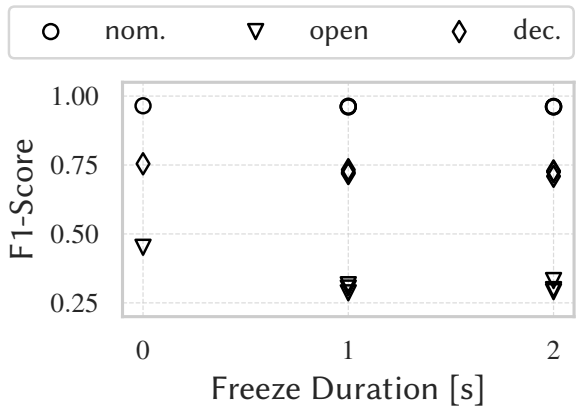


Fig. 5. Performance evaluation based on the F1-Score for videos of the nominal (nom.), open, and deconstruction (dec.) classes with freezing periods.

D. Freezing

To introduce a freezing effect, we determine a random start time within the video and then trim the video into three segments: before, during, and after. The effect repeats the freeze frame for one second in the first trial and two seconds in the second trial. We repeat this process for each duration until the model's performance converges. The results show a small decrease in correct detections for the deconstruction class of 2.59% on average and a stronger decrease of $43,31\%$ for the open class. This confirms that, as with other modifications, the model is most vulnerable in detecting the open class. Despite the model technically having the same information, an explanation for the decrease in correct detection lies in the strong temporal bias exhibited by Multi-Scale Vision Transformers [13]. Additionally, it is notable that the model performance shows almost no difference between the different durations of freezing, as visualized in Fig. 5.

- *Takeaway:* The freezing effect significantly reduces detection performance for the *open* class while having minimal impact on other classes, highlighting the model's vulnerability to temporal disruptions.

E. Comparative Overview

Tab. III provides a comparative analysis of the F1-Scores for various video modifications, providing insights into how each modification impacts the detection performance of the three classes: nominal, open, and deconstruction. The data indicates the modifications of pixelation, bitstream noise, and bitrate reduction severely degrade the detection performance of defect classes, with F1-Scores approaching zero. The open class is notably the most vulnerable across different modifications, as it shows the most significant decline in detection performance and is more likely to be misclassified as non-defective. This finding coincides with the results from the ARMBench evaluations [15]. Boxblur and Freezing primarily affect the detection performance of the open class. Both noise and bufsize modifications exhibit relatively stable performance with minimal impact on defect detection. Additionally, changes in the frame rate (FPS) do not significantly alter the F1-Scores, suggesting that adjustments to the frame rate are not crucial for maintaining model performance. We observe that artifacts that are more likely to occur due to network transmission, such as limited bitrate and pixelation, generally cause more damage than artifacts that may result from camera quality.

IV. Related Work

This section first reviews fault detection techniques in smart factories, highlighting the shift from sensor-based to vision-based methods. Next, we discuss the impact of network-induced corruption on video quality and ML performance. We then examine research on file corruption in action recognition models, followed by studies on lossy compression and its effects on ML. Finally, we outline how our work builds on these findings by leveraging MViTs.

Fault detection in smart factories has been widely studied, particularly in robotic systems where failures can lead to significant inefficiencies. Neural networks have been applied to minimize actuator faults, such as in a LEGO Mindstorms NXT pick-and-place robot [6]. Traditional sensor-based methods, however, face challenges due to environmental and system constraints, prompting researchers to explore vision-based approaches [28]. These techniques leverage image segmentation and hashing to identify anomalies in robotic movements. Some studies combine vision and sensor data for more robust fault detection [29], [30]. In contrast, others focus on specialized techniques such as frame difference methods for filament defect detection [31] or deep-learning-based approaches like RetinaNet with Feature Pyramid Networks (FPN) for steel defect identification [32]. Our work employs MViTs to address sensor limitations and enhance dataset balance, ensuring more reliable defect detection [28], [33].

Beyond fault detection, another area of research examines how network conditions impact video quality and, subsequently, ML model performance. Studies indicate that packet loss and jitter in real-time video streaming over IP networks can cause visual degradation, with delays exceeding 20 ms leading to blurring and reduced sharpness [4]. The H.264 codec is particularly sensitive to packet loss, especially in high-motion, complex scenes, whereas resolution changes have minimal impact. These insights are crucial in assessing how video degradation affects video-based ML models.

Related work also investigates the impact of file corruption on action recognition models [10]. Corruption is broadly categorized into random bit flips, which stem from hardware faults, malware, or cosmic rays, and contiguous corruptions resulting from temperature fluctuations or firmware issues. These errors introduce visual distortions such as freezing and smearing [34]–[36]. Studies using a pre-trained 3D-ResNet18 on HMDB51 and UCF101 reveal that severe corruption renders videos unplayable—up to 90% for random flips and 46% for contiguous errors. Performance degradation is significant, with accuracy dropping by 68.9% on HMDB51 and 77.1% on UCF101. Notably, random flips cause an exponential decline in performance, whereas contiguous corruption results in a

TABLE III
COMPARISON OF THE F1-SCORES ACROSS DIFFERENT VIDEO MODIFICATIONS.

	Highest F1-Score			Lowest F1-Score		
	nominal	open	deconstruction	nominal	open	deconstruction
Benchmark	0.9648	0.4516	0.7545	-	-	-
Pixelation	0.9632	0.3747	0.7254	0.9412	**0.00**	**0.00**
Boxblur	0.9634	0.377	0.7291	0.9511	**0.00**	0.5095
Bitstream Noise	0.9598	0.406	0.696	**0.2773**	**0.00**	0.1172
Noise	0.965	0.4637	0.7521	0.9587	0.3317	0.6252
FPS	0.9638	0.4035	0.7532	0.9638	0.4035	0.7532
Bitrate	0.9614	0.2982	0.6991	0.9214	0.0129	0.1786
CRF	0.9648	0.4525	0.7524	0.9626	0.3433	0.7077
Bufsize	0.965	0.4388	0.7547	0.9639	0.408	0.7422
Freezing	0.9627	0.331	0.7329	0.9605	0.2864	0.7093

linear decline. Additionally, misclassified videos exhibit 1.57× more pixel-space perturbations than correctly classified ones.

Extending this research, Chang and Fu [9] explored network-induced corruption in video-based ML models. Their experiments, conducted over emulated UDP links using Wi-Fi, cellular, and wired networks, showed that Wi-Fi produces the most severe artifacts, while cellular and wired networks primarily introduce truncation. The study analyzed the effects of corruption on action recognition, object tracking, segmentation, and video captioning. Results showed that visual artifacts consistently degrade model performance, whereas temporal truncation mainly impacts tasks requiring a single prediction per video, particularly in short-video datasets. In contrast, object tracking and segmentation remained largely unaffected.

While prior research has extensively examined the effects of file corruption and network transmission errors on ML models [9], [10], our work takes a different approach by evaluating the impact of specific artifacts from diverse sources on model performance. By employing MViTs, we aim to provide a more nuanced understanding of how these artifacts influence video-based ML models, distinguishing our work.

Another critical area of related research focuses on the trade-offs between lossy image compression and ML performance. Prior work has shown that high JPEG compression levels can significantly degrade CNN performance, particularly in thermal imagery, where accuracy drops sharply at high compression rates [37]. However, retraining models on compressed images have been found to recover up to 76% of lost accuracy. Similarly, researchers have studied the effects of compression on action classification and found no clear correlation between compression and CNN performance [38].

Our research extends this analysis to MViTs, investigating whether these newer architectures exhibit similar degradation patterns under compression. Furthermore, we examine the influence of additional bitrate-affecting factors, such as constant rate factor (CRF) and frame rate, to gain deeper insights into the robustness of modern video-based ML models.

V. CONCLUSIONS AND OUTLOOK

Machine learning performance depends heavily on data quality, which can degrade due to artifacts introduced during acquisition and transmission in factory settings. Accurately assessing the impact of these distortions is crucial for optimizing data handling.

This study evaluates the effects of nine artifacts on a Multi-Scale Vision Transformer for defect detection in a robotic pick-and-place scenario. Pixelation, bitstream noise, and bitrate reduction significantly impair performance, with network-related artifacts causing more damage than camera-related ones. The defect class open is particularly prone to misclassifications. However, reducing the frame rate while maintaining a stable bitrate has minimal impact and poses a valid alternative to reduce resource utilization.

In future work, it will be interesting to explore training models with degraded data to evaluate a possible performance increase or use Generative Adversarial Networks (GANs) to extend datasets. Additionally, testing more vision models and industrial datasets will strengthen the findings' generalizability and applicability. We also plan to perform further testing in a live environment to refine our insights. Furthermore, adversarial attacks and new corruptions, as shown by researchers in [39], [40], will extend the the gathered knowledge about the ML models and their robustness to such attacks.

ACKNOWLEDGEMENT

This work was partially funded by the German Federal Ministry of Education and Research (BMBF) project 6G-ANNA (16KISK098).

REFERENCES

[1] D. I. Rahul Rai, Manoj Kumar Tiwari and A. Dolgui, "Machine learning in manufacturing and industry 4.0 applications," *Int. J. Prod. Res.*, vol. 59, no. 16, pp. 4773–4778, 2021.

[2] L. Monostori *et al.*, "Cyber-physical systems in manufacturing," *CIRP Annals*, vol. 65, pp. 621–641, 2016.

[3] I. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, p. 160, 2021. [Online]. Available: https://doi.org/10.1007/s42979-021-00592-x

[4] J. Frnda, M. Voznak, and L. Sevcik, "Impact of packet loss and delay variation on the quality of real-time video streaming," *Telecommun. Syst.*, vol. 62, no. 2, pp. 265–275, 2016.

[5] A. Rasheed, B. Zafar, A. Rasheed, N. Ali, M. Sajid, S. H. Dar, U. Habib, T. Shehryar, and M. T. Mahmood, "Fabric defect detection using computer vision techniques: A comprehensive review," *Mathematical Problems in Engineering*, vol. 2020, p. 24, 2020.

[6] P. Jadhav and I.-G. Chun, "Fault detection and fault tolerant system for smart factories," in *2017 19th Int. Conf. Adv. Commun. Technol. (ICACT)*, 2017, pp. 765–772.

[7] Y. Tang, K. Sun, D. Zhao, Y. Lu, J. Jiang, and H. Chen, "Industrial defect detection through computer vision: A survey," in *2022 7th IEEE Int. Conf. Data Sci. Cyberspace (DSC)*, 2022, pp. 605–610.

[8] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S. Guttula, S. Mujumdar, S. Afzal, R. S. Mittal, and V. Munigala, "Overview and importance of data quality for machine learning tasks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery & Data Mining*, ser. KDD '20. Association for Computing Machinery, 2020, pp. 3561–3562.

[9] T. Chang and D. Y. Fu, "Lost in transmission: On the impact of networking corruptions on video machine learning models," 2022. [Online]. Available: https://arxiv.org/abs/2206.05252

[10] T. Chang, D. Y. Fu, C. Ré, and Y. Li, "Beyond the pixels: Exploring the effects of video file corruptions on model robustness." [Online]. Available: https://pages.cs.wisc.edu/~sharonli/publications/video-corruption.pdf

[11] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[12] A. Reichhart and M. Holweg, *Build-to-Order: Impacts, Trends and Open Issues*. London: Springer London, 2008, pp. 35–53.

[13] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer, "Multiscale vision transformers," *CoRR*, vol. abs/2104.11227, 2021. [Online]. Available: https://arxiv.org/abs/2104.11227

[14] G. Bai, H. Guo, and C. Xiao, "Research on the application of transformer in computer vision," *Journal of Physics: Conference Series*, vol. 2649, no. 1, p. 012033, 2023. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/2649/1/012033

[15] C. Mitash, F. Wang, S. Lu, V. Terhuja, T. Garaas, F. Polido, and M. Nambi, "Armbench: An object-centric benchmark dataset for robotic manipulation," 2023. [Online]. Available: https://arxiv.org/abs/2303.16382

[16] M. Reisacher, R. Frank, and A. Blenk, "Factorydc: network and resource planning for emerging applications in future factories," in *Proc. 1st Workshop Enhanced Netw. Techn. Technol. Ind. IoT to Cloud Continuum*, New York, NY, USA, 2023, pp. 1–7.

[17] M. Reisacher, N. Mitsakis, and A. Blenk, "Inout optima: Trading off machine learning prediction quality with data quantity for network optimization," in *NOMS 2025-2025 IEEE Network Operations and Management Symposium*, 2025, accepted for publication.

[18] K.-C. Song, H. Shaopeng, and Y. Yan, "Automatic recognition of surface defects on hot-rolled steel strip using scattering convolution network," *J. Comput. Inf. Syst.*, vol. 10, pp. 3049–3055, 2014.

[19] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Mvtec ad – a comprehensive real-world dataset for unsupervised anomaly detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019.

[20] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection - a new baseline," in *2018 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 6536–6545.

[21] M. Q. Alkhatib, "Polsar image classification using complex-valued multiscale attention vision transformer (cv-msatvit)," *Int. J. Appl. Earth Observ. Geoinf.*, vol. 137, p. 104412, 2025.

[22] M. El Akrouchi, M. Mhada, M. Bayad, M. J. Hawkesford, and B. Gérard, "Ai-based framework for early detection and segmentation of green citrus fruits in orchards," *Smart Agricultural Technol.*, vol. 10, p. 100834, 2025.

[23] F. AI, "Pytorchvideo: A deep learning library for video understanding," https://pytorchvideo.readthedocs.io/en/latest/model_zoo.html, 2021, accessed: 2024-08-12.

[24] W. Kay *et al.*, "The kinetics human action video dataset," 2017. [Online]. Available: https://arxiv.org/abs/1705.06950

[25] FFmpeg Developers, "Ffmpeg bitstream filters - noise," https://www.ffmpeg.org/ffmpeg-bitstream-filters.html#noise, 2024, accessed: 2024-07-27.

[26] "Ffmpeg documentation: Main options," https://www.ffmpeg.org/ffmpeg.html#toc-Main-options, accessed: 2024-07-28.

[27] R. Zurawski, *Industrial Communication Technology Handbook*. CRC Press, 2015. [Online]. Available: https://learning.oreilly.com/library/view/industrial-communication-technology/9781482207323/xhtml/C36_chapter.xhtml#sec30_2

[28] Y. Peng and D. Chen, "Vision-based fault action detection for industrial robots," *Open Access Library J.*, vol. 8, pp. 1–10, 2021.

[29] L. M. Capisani, A. Ferrara, and P. Pisu, "Sliding mode observers for vision-based fault detection, isolation and identification in robot manipulators," in *Proc. 2010 Amer. Control Conf.*, 2010, pp. 4540–4545.

[30] V. Filaretov, A. Zuev, and A. Protsenko, "Synthesis and experimental research of fault detection system for actuators of robot manipulator by technical vision," in *2024 Int. Russian Smart Industry Conf. (SmartIndustryCon)*, 2024, pp. 650–655.

[31] X. Ren and L. Cai, "Video-based broken filaments automatic detection and counting," in *2017 3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, 2017, pp. 2116–2119.

[32] F. Akhyar, C.-Y. Lin, K. Muchtar, T.-Y. Wu, and H.-F. Ng, "High efficient single-stage steel surface defect detection," in *2019 16th IEEE Int. Conf. Adv. Video and Signal Based Surveillance (AVSS)*, 2019, pp. 1–4.

[33] L. Malburg, M.-P. Rieder, R. Seiger, P. Klein, and R. Bergmann, "Object detection for smart factory processes by machine learning," *Procedia Comput. Sci.*, vol. 184, pp. 581–588, 2021.

[34] Y. Zhang, A. Rajimwale, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "End-to-end data integrity for file systems: A ZFS case study," in *Proc. 8th USENIX Conf. on File and Storage Technol.*, 2010, p. 3.

[35] G. Sivathanu, C. P. Wright, and E. Zadok, "Ensuring data integrity in storage: Techniques and applications," in *Proc. 2005 ACM Workshop Storage Secur. and Survivability*, 2005, pp. 26–36.

[36] T. J. O'Gorman, J. M. Ross, A. H. Taber, J. F. Ziegler, H. P. Muhlfeld, C. J. Montrose, H. W. Curtis, and J. L. Walsh, "Field testing for cosmic ray soft errors in semiconductor memories," *IBM Journal Res. Develop.*, vol. 40, no. 1, pp. 41–50, 1996.

[37] N. Bhowmik, J. W. Barker, Y. F. A. Gaus, and T. P. Breckon, "Lost in compression: the impact of lossy image compression on variable size object detection within infrared imagery," 2022. [Online]. Available: https://arxiv.org/abs/2205.08002

[38] V. S. S. Gowrisetty and A. Fernando, "Static video compression's influence on neural network performance," *Electronics*, vol. 12, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/1/8

[39] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," 2019. [Online]. Available: https://arxiv.org/abs/1903.12261

[40] F. Brau, G. Rossolini, A. Biondi, and G. Buttazzo, "On the minimal adversarial perturbation for deep neural networks with provable estimation error," *IEEE Trans. Pattern Anal. Mach. Learn.*, vol. 45, no. 4, pp. 5038–5052, 2023.