machine learning & knowledge extraction

MDPI

Article

Optimal Topology of Vision Transformer for Real-Time Video Action Recognition in an End-to-End Cloud Solution

Saman Sarraf [1,2,*] and Milton Kabia [2]

[1] Santa Clara Valley Section, Institute of Electrical and Electronics Engineers, Santa Clara, CA 94085, USA
[2] School of Technology and Engineering, National University, San Diego, CA 92123, USA; mkabia@nu.edu
* Correspondence: samansarraf@ieee.org or ssarraf@nu.edu

Abstract: This study introduces an optimal topology of vision transformers for real-time video action recognition in a cloud-based solution. Although model performance is a key criterion for real-time video analysis use cases, inference latency plays a more crucial role in adopting such technology in real-world scenarios. Our objective is to reduce the inference latency of the solution while admissibly maintaining the vision transformer's performance. Thus, we employed the optimal cloud components as the foundation of our machine learning pipeline and optimized the topology of vision transformers. We utilized UCF101, including more than one million action recognition video clips. The modeling pipeline consists of a preprocessing module to extract frames from video clips, training two-dimensional (2D) vision transformer models, and deep learning baselines. The pipeline also includes a postprocessing step to aggregate the frame-level predictions to generate the video-level predictions at inference. The results demonstrate that our optimal vision transformer model with an input dimension of $56 \times 56 \times 3$ with eight attention heads produces an F1 score of 91.497% for the testing set. The optimized vision transformer reduces the inference latency by 40.70%, measured through a batch-processing approach, with a 55.63% faster training time than the baseline. Lastly, we developed an enhanced skip-frame approach to improve the inference latency by finding an optimal ratio of frames for prediction at inference, where we could further reduce the inference latency by 57.15%. This study reveals that the vision transformer model is highly optimizable for inference latency while maintaining the model performance.

Keywords: action recognition; vision transformer; cloud solution

Citation: Sarraf, S.; Kabia, M. Optimal Topology of Vision Transformer for Real-Time Video Action Recognition in an End-to-End Cloud Solution. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 1320–1339. https://doi.org/10.3390/make5040067

Academic Editors: Guoqing Chao and Xianzhi Wang

Received: 1 September 2023
Revised: 25 September 2023
Accepted: 27 September 2023
Published: 29 September 2023

1. Introduction

Among computer vision tasks, video analysis, including action recognition and event detection, has significantly progressed in recent decades. Action recognition refers to a set of algorithms that identify a specific human action or an event in a series of video frames, such as playing a musical instrument or scoring a goal in a soccer match [1,2]. This video analysis technology requires strong salient feature extractors that can easily distinguish between actions and a fast-responding system that can be deployed for real-time applications [3,4]. Nevertheless, action recognition solutions offer broad applications in annotating historical archives where the intention of digitizing such archives is to extract statistics or enhance fan engagement. In both offline and real-time applications, the inference latency plays an important role in enabling the success of this type of solution in business [5,6].

Advances in deep machine learning have enabled researchers and enterprises to explore the possibility of deploying such solutions for real-time applications in a cloud environment [7,8]. However, a cloud-based real-time data processing and predictive modeling architecture requires robust infrastructure with a fast-responsive inference and the ability to outperform deep learning models [9]. Therefore, a potential solution for real-time applications should be built using optimally selected cloud components and services from a cloud-based architecture design. Deep learning models should also be

optimized based on use cases to maintain or enhance models' performance while reducing the inference latency, referring to a prediction for real-time applications [10,11].

Among deep learning-based solutions for real-time applications, computer vision techniques such as image classification, event detection, object detection, and pose estimation in real-time scenarios have interested many researchers. Moreover, several industries have considered employing computer vision cloud-based solutions, including agriculture, transportation, broadcasting and entertainment, education, healthcare, manufacturing, mining, postal services, and telecommunication [12,13]. In such applications, practitioners seek computer vision end-to-end algorithms using imaging data, including static images or video frames, to classify images or localize objects in real-time scenarios [14].

The human brain inspires most deep learning architectures currently employed in cloud-based solutions, e.g., convolutional neural networks [15–17], recurrent neural networks, long short-term memory, known as RNN-LSTM [18,19], and attention-based networks [20,21]. For instance, a deep learning convolutional network was implemented for predicting the different stages of Alzheimer's disease, including normal aging, mild cognitive impairment, and Alzheimer's [15–17]. The researchers aimed to demonstrate that network topology optimization could produce the highest modeling performance in similar tasks while reducing the architecture's complexity [17]. Unlike sophisticated deep learning architectures that use millions of trainable parameters [17,22], the MCADNNet offered a three-layer convolution-based model through an optimal input size of imaging data, which allowed for faster prediction using the trained feedforward network [17]. This work's topology optimization concept could be examined in computer vision, medical imaging, or other machine learning applications, enabling scientists to deploy such outperforming models in a production environment [17].

A cloud-based solution for real-time machine learning applications should be evaluated based on architectural and modeling aspects. Solution architects should design a system by considering five central pillars: (a) operational excellence, (b) security, (c) reliability, (d) performance efficiency, and (e) cost optimization/frugality [23,24]. On the other hand, a deep learning or computer vision model's performance should be evaluated using various metrics such as accuracy rate, precision, recall, F1 score, and confusion matrix [25,26]. Also, the impact of lower-dimension images and fewer videos on models' performance should be measured in real-time computer vision applications [27,28].

Our work aims to demonstrate the topology optimization capabilities that should be considered when implementing computer vision solutions in a cloud environment for real-time applications. Most existing applications focus on improving the computer vision models' performance through various architectures; however, the inference latency, which can be improved through topology optimization, must be discussed more. Therefore, we use vision transformer vanilla architectures showing similar performance to stable vision models and explore simplifying this architecture while maintaining acceptable accuracy. The problem addressed in this study consists of compromised timing and machine learning (ML) model performance in real-time applications, especially for cloud-based computer vision solutions. Optimal latency and inference latency in a real-time application are achieved through a software architecture with a complex and fast-responding infrastructure or a simplified back-end machine learning model [8].

This study aims to design and implement an end-to-end cloud-based solution for optimal attention network topology in real-time applications specified for action recognition. The designed solution offers optimal software architecture to achieve high machine learning model performance while minimizing the inference latency issues in data streaming, especially for computer vision applications, including video-based action recognition. The optimization is addressed by varying the machine learning algorithm parameters after the pipeline is built. The study's deliverable is a software entity hosted in the Amazon cloud environment and developed in Python, where an optimal topology of vision transformer for video action recognition is employed. In summary, the novelty of this study includes (a) employing vision transformer models as state-of-the-art technology in

real-time video action recognition, (b) optimizing the topology of vanilla architectures by reducing the video frame dimensions as the input to the model and the number of multi-head self-attention modules, (c) selecting optimal cloud instances, and (d) proposing a novel enhanced skip-frame mechanism to reduce the inference latency further.

2. Related Work

The central pillar of this section is based on exploring the definition of attention mechanisms in human brains and artificial intelligence as it forms the core of our vision transformer models for optimization purposes. The applications of attention mechanisms have expanded into various domains, including machine learning, natural language processing, and computer vision—the main focus of this work—but have yet to be discussed for action recognition and video event detection. We also review the cloud-based solutions for real-time machine learning applications and analyze their optimizations. The major advantage of attention mechanisms and the enhanced versions known as transformers is to suppress the impact of less contributing features and intensify the most contributing attributes during the training process while considering positional information.

2.1. Transformers in Computer Vision

The human visual system inspires the core concept of using an attention mechanism in machine learning, as visual signals play a crucial role in the orienting subsystem of human attention [29]. The initial concept was developed with an attention model accumulating information across several fixed points in an image using a Boltzmann machine [30]. Enhanced feature extraction from imaging data is achieved using an attention mechanism that enables computer vision models to extract salient features associated with the type of objects' location, resulting in improved modeling performance [31]. Another brain-inspired attention mechanism extracted features from multi-scale color contrast. This attention mechanism produces a low-resolution saliency map in which salient features are highlighted [32].

Moreover, a winner-takes-all mechanism, known as WTA, was proposed. It is a location-based attention mechanism that tracks salient locations detected in consecutive frames to incorporate sequential information in the learning process [21,33,34]. Some researchers categorized the computer vision architectures with the attention mechanism into (a) CNN-based networks with recurrent connections, (b) visual transformers with tokenized inputs, (c) attention-augmented CNNs, and (d) dense layers with image transformers [35–40]. Advanced vision transformer-based architectures introduce other layers to incorporate further information extracted from data during the training process through a temporal transformer encoder in ViViT [41], a space–time attention module in TimeSformer [42], and multi-scale vision transformer (MViT) techniques [43].

Considering video as a form of 3D data, the recurrent vision transformer (RViT) framework for spatial–temporal representation offers a framework to address action recognition [44]. The novel frameworks of M2DAR and MM-ViT offer multi-view, multi-scale solutions that allow action recognition [45] using a vision transformer architecture [46]. Recently, semi-supervised computer vision models have been of interest to many scientists, and SVFormer, a transformer-based algorithm, provides this ability to recognize actions [47]. Enhancing the embedding–positioning feature of vision transformation by replacing the absolute position with a relative position improved the performance of action recognition in some studies [48]. A deep neural network model was optimized to reduce latency for edge computing through a fine-grained pipeline [49]. Researchers also showed an improved latency by optimizing graphical unit communication during computing for image classification tasks [50]. Scientists proposed a new framework using an acceleration approach to improve inference latency, known as ABM-SpConv-SIMD [51].

2.2. Cloud-Based Solution for Real-Time Computer Vision Applications

Real-time computer vision applications require a fast-responding prediction module that refers to inference, in which the deployed module can accurately predict unseen data while end-to-end inference latency is low. Therefore, such real-time solutions heavily rely on infrastructure with distributed systems; a scalable, robust, and cost-effective environment such as a cloud ecosystem is considered an optimal solution [52,53]. Fast-growing machine learning (ML) applications in various industries have been investigated; such applications were transformed into cloud services known as ML-as-a-Service (MLAAS) [54]. The major issue with machine learning models was underperforming inference (high response time and long delay), which discourages artificial intelligence companies from deeply modeling products. A novel approach, MArk, was designed to offer optimal latency and inference latency for a machine learning application [55]. The proposed algorithm, built on the Amazon Web Services infrastructure, dynamically produces batches in real time based on a cost-latency-effective approach within available resources [54].

Recent advances in computer vision, such as event detection for real-time applications, have interested many commercial companies [56]. For example, real-time video analysis technology has expanded into sports to enhance fan engagement and provide game statistics or predictions [57]. Amazon Web Services, a major cloud computing service provider, has developed repeatable customer offerings regarding real-time and offline event detection for various sports such as American football, soccer, and hockey [58]. The core technology of such solutions consists of a robust deep learning-based computer vision model predicting events at a high resolution and improving the performance of models using a multimodal approach [8,58].

Another set of inference latency optimization approaches enables the incorporation of data compression such as video compression at source [59] and the use of accelerating networks in cloud environments [60]. However, the data compression [61] and accelerating techniques [62] reduce the inference latency and increase the complexity and cost of development and the maintenance of networks. Moreover, the video compression techniques degrade video frame quality before feeding the frames into the pipeline, which requires model development to be conducted on low-quality data, reducing the models' performance [63]. Therefore, applying such techniques in real-world computer vision solutions is challenging, and most use is limited to research studies.

3. Materials and Methods

3.1. Dataset

Action recognition in computer vision consists of several steps, such as video preprocessing, image classification, and a postprocessing step, including temporal modeling in real-time and offline applications. The UCF101 is a well-known, broadly used, and highly referenced action recognition dataset collected by the Center for Research in Computer Vision at the University of Central Florida [64]. The dataset includes 101 different human actions, as shown in Figure A1, collected from YouTube videos that are publicly available. The dataset can be categorized into five major classes: (a) sports, (b) playing musical instruments, (c) human–object interaction, (d) human interaction, and (e) body motion [64] (Soomro et al., 2012). UCF101 is the first action recognition dataset with the highest number of classes and ranks among the top computer vision datasets that are publicly available, focusing on action recognition and event detection. The reliability of the dataset is validated since this dataset is widely used and cited by prestigious manuscripts at top-tier computer vision conferences and in journals [65,66]. Furthermore, the dataset is trustworthy, as the raw data are videos collected through YouTube, publicly available and carefully annotated by the dataset owner, and verified by researchers over the years [67,68]. The UCF101 provides the training and testing splits of videos; this allows researchers to consistently compare their results, which is considered an additional source of validity and trustworthiness of the dataset.

3.2. Machine Learning/Computer Vision Pipeline

The pipeline implemented in this study encompassed two major components: (a) the core machine learning (vision transformer) and (b) cloud components. The machine learning pipeline included (a) preprocessing video data, (b) training vision transformer models, (c) evaluating models, (d) postprocessing the results, and (e) optimizing the models to produce competitive performance with less trainable parameters. The UCF101 (training set) used in this study included 9537 video clips split into 7721 (80%), 908 (10%), and 908 (10%) videos for training, validation, and testing purposes.

The two-dimensional (2D) frames were extracted from clips using the OpenCV Python library on a p3.8xlarge Amazon SageMaker instance, and the frames were uploaded to the Amazon Simple Storage Service (S3). SageMaker hosted the machine learning pipeline in the training and inference phases. The number of frames in each clip varied since the duration and frame per second (FPS) rate of clips differed. Figure 1 shows the number of frames extracted from the video clips, and that the dataset was slightly imbalanced.

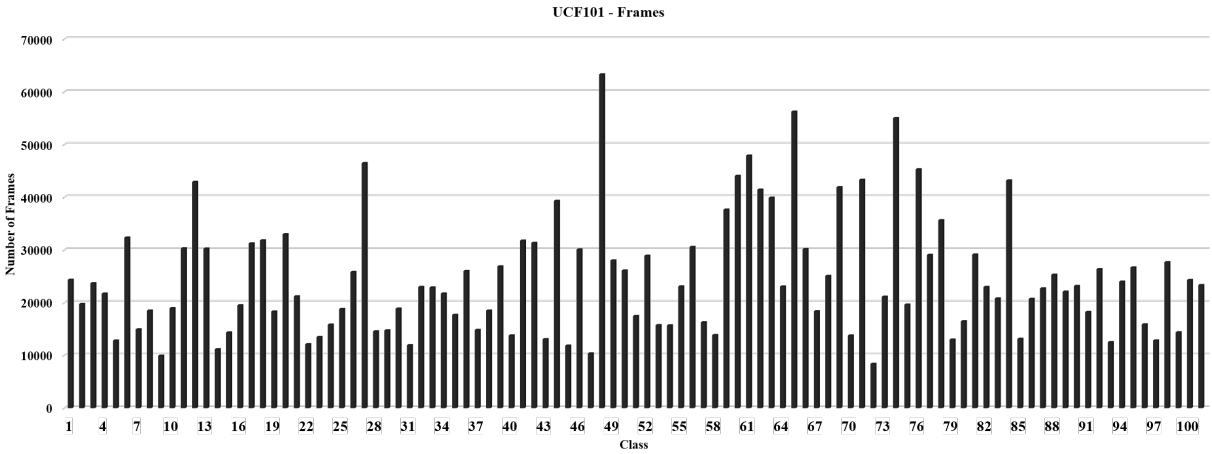

Figure 1. The figure shows that the number of frames varies per class; the distribution is slightly imbalanced.

Based on the previous step's videos, the preprocessing module produced 1,448,558, 168,890, and 168,631 frames for training, validation, and testing. The vision transformer model with a multi-head self-attention (MHSA) mechanism was considered the baseline, and this model was optimized for various heads and image sizes [37]. First, the vision transform model generated small patches of frames, transforming the patches with two axes and three channels into a vector. Next, the linear project of flattened patches was passed through a position embedding process to be processed by a multi-head transformer encoder. Finally, the features from the encoder were fed into a multi-layer perceptron, playing the role of a translator of attention-based features into a format to be processed by the cross-entropy loss. The PyTorch framework managed the hyperparameter of vision transformer vanilla mode during training, where we set batch_size to 64, epochs to 30, learning rate to 3×10^{-5}, dropout to 0.1, and gamma to 0.7 as the initial values. The other hyperparameters for optimization were explored by modifying MHSA and image inputs, as explained later.

Figure 2 illustrates the architecture of the vision transformer for a given input frame where the patches were generated for the training process. The implementation of an end-to-end cloud-based pipeline was performed using a proposed solution by system-deep learning architects at Amazon, considering the vanilla architecture for optimization. Amazon S3, Kinesis, Gateway, and SageMaker Endpoint were the AWS services used to build the end-to-end pipeline. The optimization objective from the solution perspective was to reduce the inference latency based on the vanilla architecture without onboarding further services such as AWS outputs and Wavelength.

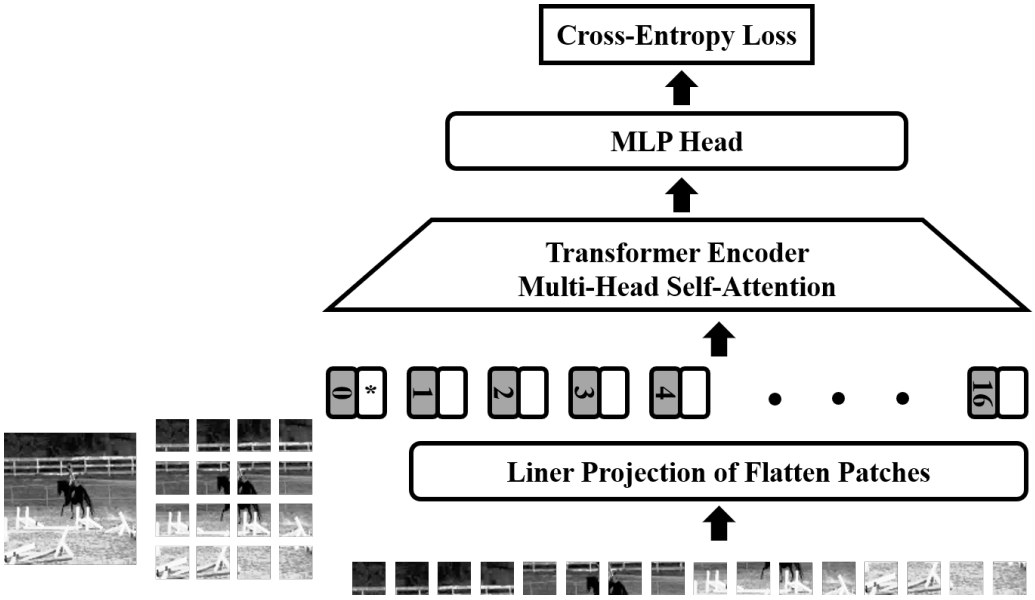

Figure 2. The vision transformer model, with a multi-head self-attention mechanism encapsulated in the transformer encoder, was optimized for the different numbers of heads and image sizes. However, the vision transformer required a significant amount of memory and computation, and the optimization focused on exploring the optimal number of layers and input dimensions to improve the inference latency for real-time applications.

A stable cloud architecture used for offline and real-time applications with a batch-processing approach was considered, as illustrated in Figure 3. This architecture was carefully designed by consulting two reference architectures published by AWS Machine Learning Blog (publicly available), which are (a) Machine Learning Inference at Scale Using AWS Serverless (https://aws.amazon.com/blogs/machine-learning/machine-learning-inference-at-scale-using-aws-serverless/, accessed on 30 August 2023) and (b) Building a Data Pipeline for Tracking Sporting Events Using AWS Services (https://aws.amazon.com/blogs/architecture/building-a-data-pipeline-for-tracking-sporting-events-using-aws-services/, accessed on 30 August 2023). We used various AWS instances to explore the training duration and inference latency to discover the most optimal instance for deployment purposes.

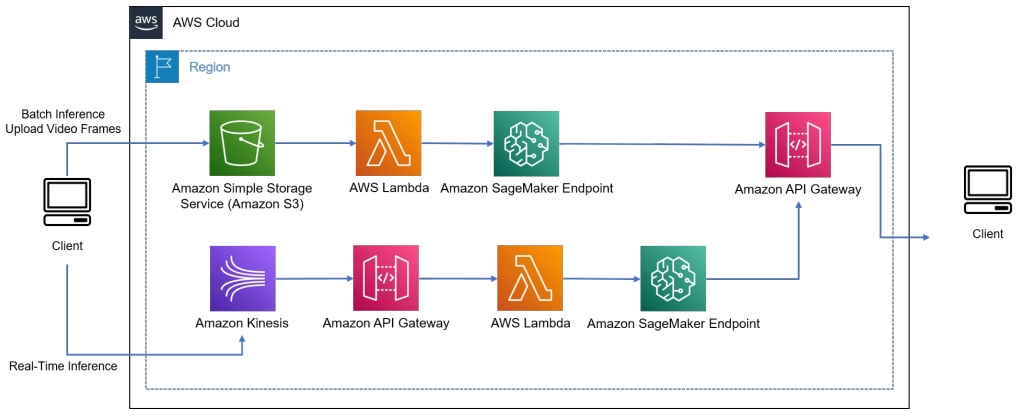

Figure 3. In this solution architecture, based on AWS's two architectures, the client triggers the real-time inference to stream videos from an external device where Amazon Kinesis receives the videos and calls the Amazon API Gateway. This service triggers an AWS Lambda function, which invokes Amazon SageMaker Endpoint, offering access to the trained and optimized attention-based model for computer vision applications. Upon completion of inference and analytics readiness, the results are provided to the client for broadcasting or other usages.

4. Results

In the first step, we explored to what extent our cloud-based solution for real-time applications could be optimized for inference latency by enhancing architecture components, compared to vanilla architecture, for the same use cases. Table 1 compares the inference latency for vanilla and optimized vision transformer models against an identical testing set performed using four popular GPU-based instances for computer vision applications. The batch size in the prediction module was set to 64 images and was kept consistent across the study. The instances' specifications, including memory, bandwidth, and cost per hour, were also provided for comparison purposes. The latency of each instance was measured in the inference against comparable testing data; the results were then compared in terms of the latency, memory, network bandwidth, and cost per hour extracted from the AWS website in 2022.

Table 1. Comparison of three popular instances used for computer vision with GPU and parallel computing capabilities.

Model		**ViT Vanilla**	**ViT Optimized**
Multi-Head Self-Attention (Number of Heads)		**16**	**8**
Image Size (x, y, Channel)		**224 × 224 × 3**	**56 × 56 × 3**
p3.16xlarge	Latency (sec)	1474.0831	856.8517
	Test Samples (PNG)	168,631	
	Memory (GB)	488	
	GPU Memory (GB)	128	
	Bandwidth	10	
	Cost USD/Hour	USD24.48	
p3.8xlarge	Latency (sec)	1434.5108	853.8936
	Test Samples (PNG)	168,631	
	Memory (GB)	244	
	GPU Memory (GB)	64	
	Bandwidth	Up to 10	
	Cost USD/Hour	USD12.24	
p3.2xlarge	Latency (sec)	1420.8712	851.4978
	Test Samples (PNG)	168,631	
	Memory (GB)	61	
	GPU Memory GB)	16	
	Bandwidth	10	
	Cost USD/Hour	USD3.06	

In the second step, we explored the possibility of reproducing identical, similar, or improved performance using an optimized vision transformer model for real-time application compared to a deeper model used for offline application. We posed the question of under what conditions an optimized architecture for real-time computer vision applications can produce equal and improved performance compared to a similar architecture designed for offline applications. The optimization strategy centered on modifying the number of multi-head self-attention layers (8, 16) and reducing the input size for frames (224 × 224, 112 × 112, 56 × 56) fed into the model during implementation. We only refer to (224, 112, 56) for simplification as the input size. The assumption was that an optimized model would produce a similar or improved performance for real-time application represented by a lightweight model (ViT_56_8) compared to an offline application represented by a complex and heavy model (ViT_224_16). The results in Table 2 indicate that the ViT_224_16

offline model outperformed the ViT_56_8 real-time model by 3.75% and 3.23% in terms of the macro averaged F1 score at the frame level for validation and testing sets, respectively. Table A1 shows the full classification report for the frame-level analysis, and the support column refers to the number of frames used for testing and validation purposes (the success criteria are stated in the discussion section).

Table 2. The frame-level performance of vision transformer models optimized for the number of heads and input size. The optimization of the number of MHSA layers and input showed that these two parameters affected the performance of models at the frame level.

Model	F1 Score Macro Avg	
	Validation	Test
ViT_56_8	0.868368201	0.871495053
ViT_56_16	0.871914203	0.875206396
ViT_112_8	0.891644812	0.890590870
ViT_112_16	0.895766029	0.891423734
ViT_224_8	0.908368265	0.908376098
ViT_224_16	0.905895462	0.903802112

Since the original data included the video clips, the performance of video-level models was calculated to address the optimization question properly. First, the predicted labels from the frames were collected and aggregated based on the video clip identification (ID) numbers. Next, a postprocessing step was applied to the frames of each clip by calculating the probability of each class in a given video clip, followed by selecting the top class based on the concept of voting by the majority. The postprocessing module mapped 2D predicted labels (frames) into 3D space (videos). We aggregated the performance per second at inference for visualization and anchoring purposes. The video-level results shown in Table 3 indicate that the offline heavy ViT_224_16 model outperformed the real-time light ViT_56_8 vision transformer model by 3.22% and 2.62% in terms of the macro averaged F1 score at the video level for the validation and testing sets, respectively (confusion matrix shown in Figure A2). Moreover, the F1 scores of video-level evaluations were improved by 0.53% and 0.59% compared to the frame-level analysis. The video-level analysis aligned better with the business concept of real-time applications. Table A2 shows the full classification report for the video-level analysis.

Table 3. The performance of vision transformer models at the video level is shown in this figure as the video data are given in the original data format. The results show that the the majority considered the postprocessing step to improve the performance of models from the frame level to video level.

Model	F1 Score Macro Avg	
	Validation	Test
ViT_56_8	0.905055103	0.912229318
ViT_56_16	0.910509808	0.921660936
ViT_112_8	0.930591619	0.924362794
ViT_112_16	0.923940036	0.928365605
ViT_224_8	0.946523733	0.936052540
ViT_224_16	0.937209031	0.938384157

The analysis of the physical properties of the vision transformer, ResNet, Inception, and VGG models showed that the optimized vision transformer model for real-time applications (ViT_56_8) offered a significantly lower training time and memory volume occupation during the training process, as shown in Table 4. Therefore, the optimized model training time on the same server instance was 2.25 times faster, the equivalent of 3:29:30 hours per epoch, than the offline ViT_224_16 model. Furthermore, since all of the models were

trained for 30 epochs (Figure A3), the optimized model saved 104:45:00 hours and the corresponding cost of server usage compared to the heavy model. Furthermore, the results show that the optimized model occupied slightly less space in the disk compared to other vision transformer models; however, the investigation indicated that the model size comparison with the benchmark model, such as ResNet families, should be ignored as those models were specifically optimized to reduce the model size with a different approach. In summary, the results in Table 4 show that the optimized vision transformer with an input size of 224 and eight heads occupied the least amount of memory. Since two different Python packages are used to implement vision transformers and ResNet models, the volume of saved models on the disk is incomparable. The last column includes the training time of each model, showing that the vision transformers are faster to train. The time in the column refers to the second epoch of each model, which was consistent across the models.

Table 4. Comparison of vision transformer and deep learning ResNet, Inception, and VGG family models. This table compares the vision transformer models and ResNet family regarding memory occupation in the training process, and the volume of saved models on the disk.

Model	Input	Head Layer	Memory in Training MB	Volume on Disk MB	Training Time (H:M:S)
ViT	224	16	11,967	196	6:16:37
ViT	224	8	8703	148	6:08:14
ViT	112	16	4027	196	4:48:15
ViT	112	8	3397	148	4:23:27
ViT	56	16	2453	195	2:47:22
ViT	56	8	2125	147	2:47:07
ResNet	224	101	13,983	164	10:53:12
ResNet	224	50	10,521	91	10:12:27
ResNet	224	18	5275	41	7:09:29
Inception	224	48	13,865	529	5:11:05
VGG	224	16	13,863	530	5:12:45

We aimed to further optimize the inference latency of the optimal vision transformer topology using an enhanced in-house version of the skip-frame approach [69]. We leveraged the skip-frame approach and developed a two-stage algorithm to find an optimal ratio of frames within one second concerning the number of frames in the given second (FPS) across our testing set. In the first step, we randomly shuffled the frames' indices in each second of videos; then, we started from 10% to 100% of frames with a step of 10%. We repeated this process five times and measured the average and standard deviation of F1 scores against the testing test. In the second step, we selected the video frames in each second using an ordinal approach instead of randomly shuffling samples. For example, 50% of frames in this approach were selected by skipping one frame at a time. Next, we measured the F1 scores against the testing set. Lastly, we introduced an error metric to measure the distance of corresponding experiments in the first and second steps. The objective was to find a ratio with a minimum error using Equation (1). A ratio with a minimum error rate could represent a stable threshold for maintaining the performance while reducing the number of frames, resulting in lower inference latency in our real-time batch processing approach.

$$Error_{SkipFrame} = |Random - Ordinal| \tag{1}$$

The results depicted in Figures 4 and 5 reveal that the optimal ratio of frames in our study is 40%, where the error rate is 0.0001562, which allows us to batch process 60% fewer frames, resulting in a significant improvement in the inference latency.

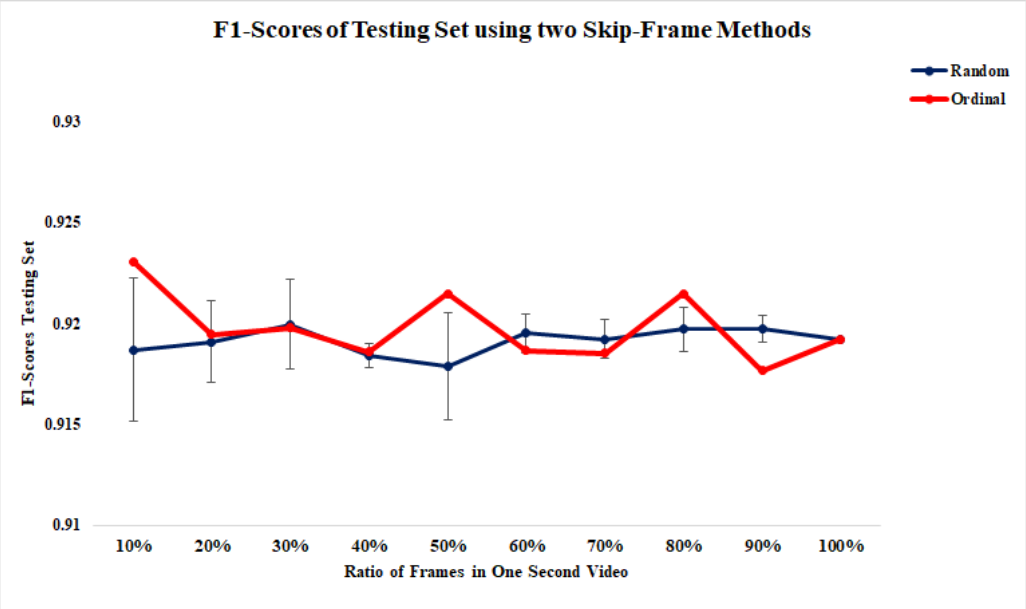

Figure 4. The F1 scores for 10 ratios across two experiments including 5 repetitions of the random approach and one-time ordinal. The standard deviation of 5 times the random method is illustrated as the blue line.

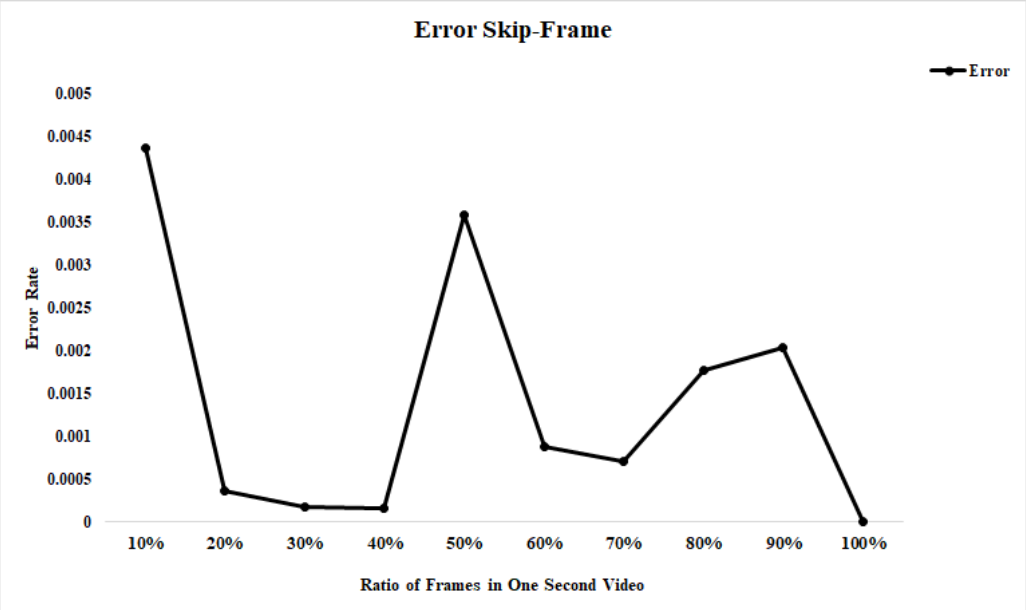

Figure 5. The error rates defined in Equation (1) across the analysis of ten ratios show that both skip-frame methods converge at a 40% frame ratio, with the lowest error rate and standard deviation in the random method. This enhanced approach should be considered a hyperparameter in each relevant study as it can reduce the number of frames to predict at inference.

5. Discussion

We aimed to address the issues with inference latency in the real-time computer vision applications where video data must be analyzed at the frame level. High-resolution video frames with high FPS require time-consuming and costly computation, which increases the inference latency and cost of development. Most existing solutions aim to improve the model performance by employing sophisticated algorithms and expensive cloud infrastructure. In such solutions, limited techniques such as video data compression are considered, which usually increase the cost and complexity of the solution. Furthermore, incorporating data compression techniques and enhanced network bandwidth is rarely

feasible. Therefore, we proposed our end-to-end solution to address the current gaps in the literature by optimizing the vision transformer's topology followed by an enhanced and fast skip-frame mechanism to reduce the inference latency.

In this study, we developed an end-to-end solution that reduces the complexity of the vision transformer model as a state-of-the-art technique, resulting in lower inference latency and deployment costs. The first research question in this study outlined the optimization of cloud-based architecture for real-time computer vision applications from the cloud service or component perspective. Existing AWS cloud services, including storage, streaming services, and step functions, were investigated to address this question. The research indicated that solution architects have already implemented several vanilla solutions by piping the cloud components, and the architecture shown in Figure 3 is among the most-used solutions. Further investigation showed that cloud users and practitioners were not given access to modify the core of cloud services, such as streaming services. Therefore, the end-to-end optimization centered on implementing an attention model (vision transformer) to produce consistent performance compared to the vanilla attention architecture and existing state-of-the-art models while significantly improving the inference latency.

The optimization of vision transformer models was conducted against the number of heads in the attention layer and input dimension. The optimal model was considered with an 8-head attention layer and input dimensions of $56 \times 56 \times 3$ (ViT_8_56) against the vanilla transformer with a 16-head attention layer and dimensions of $224 \times 224 \times 3$ (ViT_16_224). The prediction module was executed against the testing dataset with 168,631 frames, and the inference latency was measured for three GPU-based instances. The results shown in Table 1 demonstrate a consistent pattern of significantly lower latency for ViT_8_56 than the original model across all instances, which was aligned with the literature [70,71], where the optimized model predicted the testing set faster than the original model by a factor of 1.67.

The second research question focused on comparing the performance of vanilla vision transformer models and optimized topology. This question sought a condition where the optimized vision transformation produced similar or improved performance (±3% and ±5% confidence intervals) compared to the original architecture. These success criteria were inspired by a real-world business use case where the name of our stakeholder is confidential, as per their request. The vision transformer architecture was modified by varying the number of heads in the attention layer for two values of 8 and 16 and by exploring the network's input dimension for 224, 112, and 56 input sizes. This strategy built six vision transformer models from scratch using identical training, validation, and testing sets at the frame level.

A postprocessing step was applied to the predicted frames to obtain the video-clip-level results. Finally, the full classification reports were generated to validate and test the (unseen) data, including precision, recall, F1 score, and accuracy. The frame-level and video-level results depicted in Tables 2 and 3 show that the performance of ViT_56_8 (optimized topology) competed with ViT_224_16 (vanilla transformer) in the validation and testing experiments, with a 3.2153% and 2.6154% difference in F1 score at the video level. Therefore, the performance of ViT_56_8 fell into the predefined confidence interval. The performance of vision transformers against the UCF101 action recognition dataset was on par with existing state-of-the-art models [72,73].

The third research question addressed benchmarking vision transformer models against existing state-of-the-art models such as ResNet18, ResNet50, ResNet101 (the ResNet family), VGG-16, and Inception-V3. Again, a training strategy identical to the vision transformer models was employed to implement the baseline models. Finally, the performance of baselines was evaluated to answer the research question of whether an optimized vision transformer produced a similar or improved performance, with ±3% and ±5% confidence intervals compared to the baselines.

The results depicted in Tables A1 and A2 show that the optimized vision transformer (ViT_56_8) produced F1 scores of 90.50% and 91.22% for validation and testing sets. Although the optimized model underperformed compared to the best-performing baseline

(ResNet18) by 4.96% and 4.31%, the difference in the model's performance fell into the ±5% confidence interval, which met the criteria of marginal acceptance of the optimization objective. Furthermore, as discussed, the inference latency of ViT_56_8 was significantly better than that of ResNet18 and the rest of the models.

The trade-off between inference latency and performance was the focus of the fourth question for the optimization task. Varying the number of heads in the attention layer of the vision transformer and input dimension were considered the two key parameters of optimization. This research question investigated whether a trade-off between the models' performance and latency across the attention-based models was achievable. It explored whether the optimized vision transformer referencing ViT_56_8 met the trade-off criteria. The success criteria for the optimization task were set for an optimized model with performance falling into a confidence interval of ±3% as acceptable performance and ±5% as marginally acceptable performance. Moreover, the criterion of inference latency improvement was set to an improvement of at least +5% against the original model. Therefore, correlation analyses were conducted between the inference latency, the number of heads, and the input dimension for three different servers.

The results in Table 4 indicate a strong correlation between the inference latency and input dimension. However, a naïve correlation was found between all models' inference latency and the number of heads. For example, a significant statistical difference was found between vision transformers with a constant input dimension of 56 but a different number of heads of 8 and 16. The results show that the number of heads impacted the inference latency. Table A2 shows that the performance of the optimized vision transformer (ViT_56_8) for the testing set was an F1 score of 91.22%, which was 2.62% and 4.32% lower than the original vision transformer (ViT_224_16) and best-performing CNN-based baseline (ResNet18). The results fell into the predefined confidence intervals; the optimized model met the accepted optimization criteria and was marginally acceptable from the performance perspective. Tables 4 and A3 indicate that the optimized model (ViT_56_8) was a factor, with factors of 2.25 (55.63%) and 1.67 (40.70%) in training step and prediction (inference) compared to the original model (ViT_224_16); therefore, the optimization criterion was met from the inference latency perspective.

In the last step of this optimization pipeline, we employed an enhanced in-house skip-frame technique to explore the optimal ratio of frames at a one-second level to reduce the inference latency during real-time batch processing. The testing set included 168,631 video frames, and we initially achieved the best performance of 851.4978 seconds for this set. The skip-frame approach showed that the optimal ratio of frames for our testing dataset is 40%. Thus, we modified the prediction module to only process a batch of 40% of frames in each second of videos regardless of FPS based on the randomly selected frames explained above. We reduced the batch processing time to 364.8635843 sec, equivalent to a 57.1504% decrease in the inference latency where our optimal vision transformer's performance remained approximately at the same value. The proposed skip-frame method allows practitioners to reduce the inference latency; however, it should be considered a hyperparameter for tuning, as the ratio of frames might vary for different video datasets.

We further analyzed the confusion matrix of the testing set to explore the impact of a slightly imbalanced data situation, as shown in Figure 1. We extracted the diagonal values of the confusion matrix normalized by true (labels) values. Then, we sorted the normalized diagonal values and video sample distribution in the testing set per class in ascending order. We selected the first ten low-performing classes (values < 75%) and low-sample classes in the distribution. Next, we obtained the intersection between those two lists and noticed that BlowingCandles (class 13), ParallelBars (class 56), and StillRings (class 85), with accuracies of 0.25%, 0.6% and 0.67%, are common classes between the two lists. This analysis shows that a potential data augmentation, including over- and under-sampling, might help to improve the low-performing classes in future work.

Our objective was to reduce the complexity of end-to-end machine learning algorithms, ultimately reducing the inference latency. The original vision transformer vanilla

model with three channels, an input size of 224 × 224, and 16 attention heads included 53,532,675 trainable parameters, whereas our optimized model with three channels, an input size of 56 × 56, and eight attention heads had 38,406,147 (the report was generated using PyTorch). After model training, we implemented the two extra steps of postprocessing and the enhanced skip-frame mechanism, the algorithm complexity of which is O(n(log(n))) due to being grouped by function to convert frames to video clips and O(n) because of the single for-loop in the optimization of frame ratio. The algorithm complexity at inference is only impacted by the optimized vision transformer model, as the steps mentioned above are excluded during inference.

6. Limitations

We identified two main limitations of our study, which include (a) the dataset and (b) business-driven success criteria. Although UCF101 is a popular action recognition dataset broadly used to calculate the baseline performance in machine learning and computer vision studies, we could use other publicly available datasets such as Kinetics (Kinetics Human Action Video Dataset) [74] and HMDB51 [75] to expand this study in the next iteration. As described in the manuscript, the objective of this study was to explore the optimization of an end-to-end cloud solution and the optimal topology of a vision transformer for real-time application, inspired by a real-world business use case. Therefore, we defined our optimization success criteria based on the business's needs. Other optimization criteria could be revisited; however, they will be less likely to impact the results. We will also explore a broader range of AWS instances for inference latency investigation, in addition to the GPU-based servers.

7. Conclusions

The objective of this study was to explore the optimization of an end-to-end cloud-based real-time action recognition solution using the optimal topology of a vision transformer. Our optimized vision transformer with eight attention heads and an input size of 56 × 56 × 3 produced an F1 score of 91.497% for a consistent testing set at the video level across the study, meeting our success criteria in terms of performance. The analysis of inference latency for the models on three GPU-based server instances revealed that the prediction by ViT_56_8 was 40.70% faster than with the vanilla vision transformer. Another analysis of the training time showed that the optimized model offered a shorter training time by 55.63% compared to the original model. We also introduce a novel enhanced skip-frame mechanism to discover the optimal ratio of frames; the mechanism further improved the inference latency by 57.15%. We conclude that vision transformers are highly optimizable for real-time action recognition applications, producing results that are able to compete with state-of-the-art algorithms with less trainable parameters. In the future, we will explore various video datasets, various splits of data, and data augmentation for imbalanced data situations and aim to explore other vision transformers. We will consider other techniques such as network optimization and data compression to improve the inference latency.

Author Contributions: S.S. designed, implemented, and executed the end-to-end project including data processing, modeling, and writing. M.K. reviewed and proofread the project and manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A

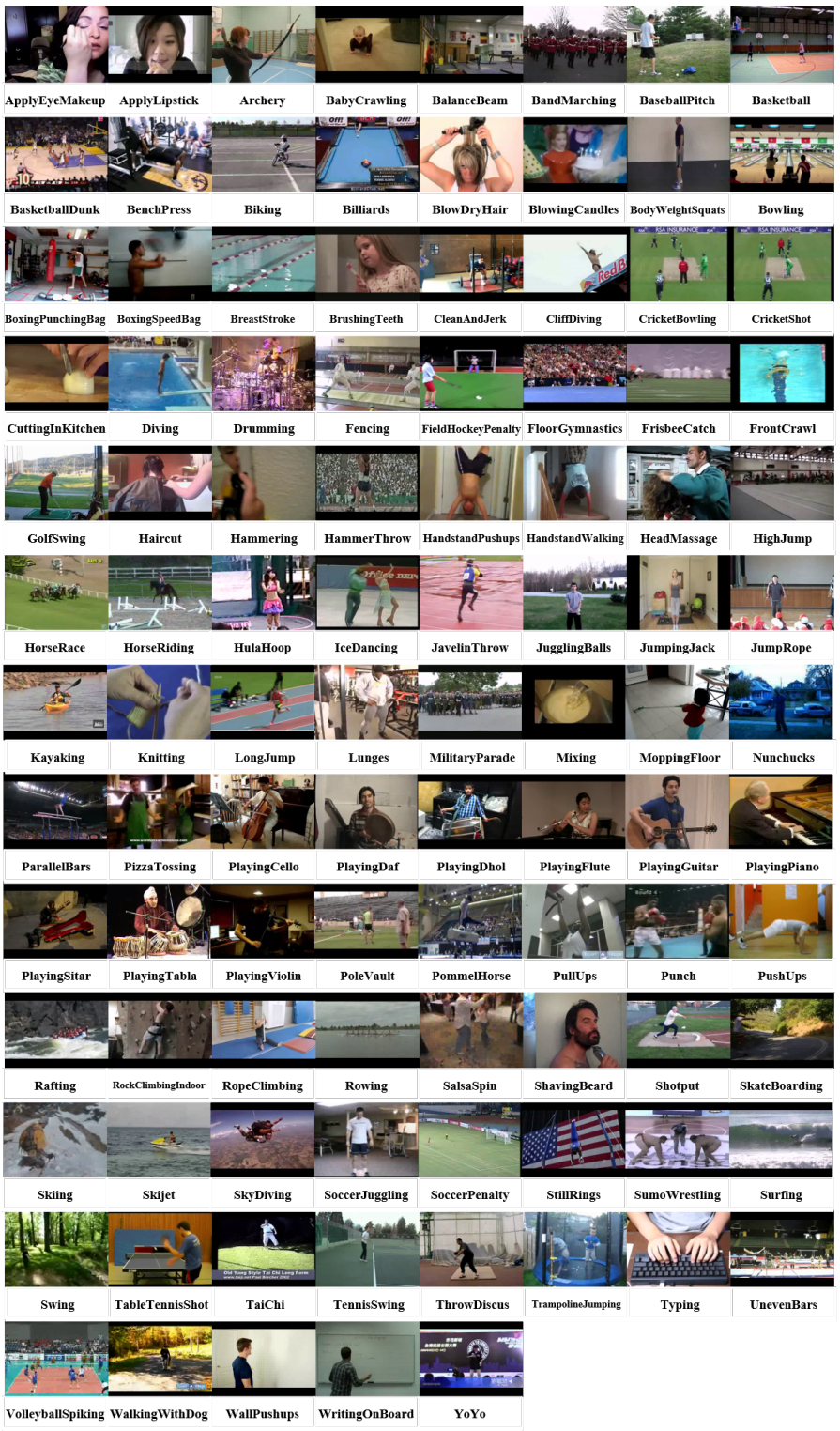

Figure A1. A total of 101 human actions (classes) available in the UCF101 dataset were used for model development.

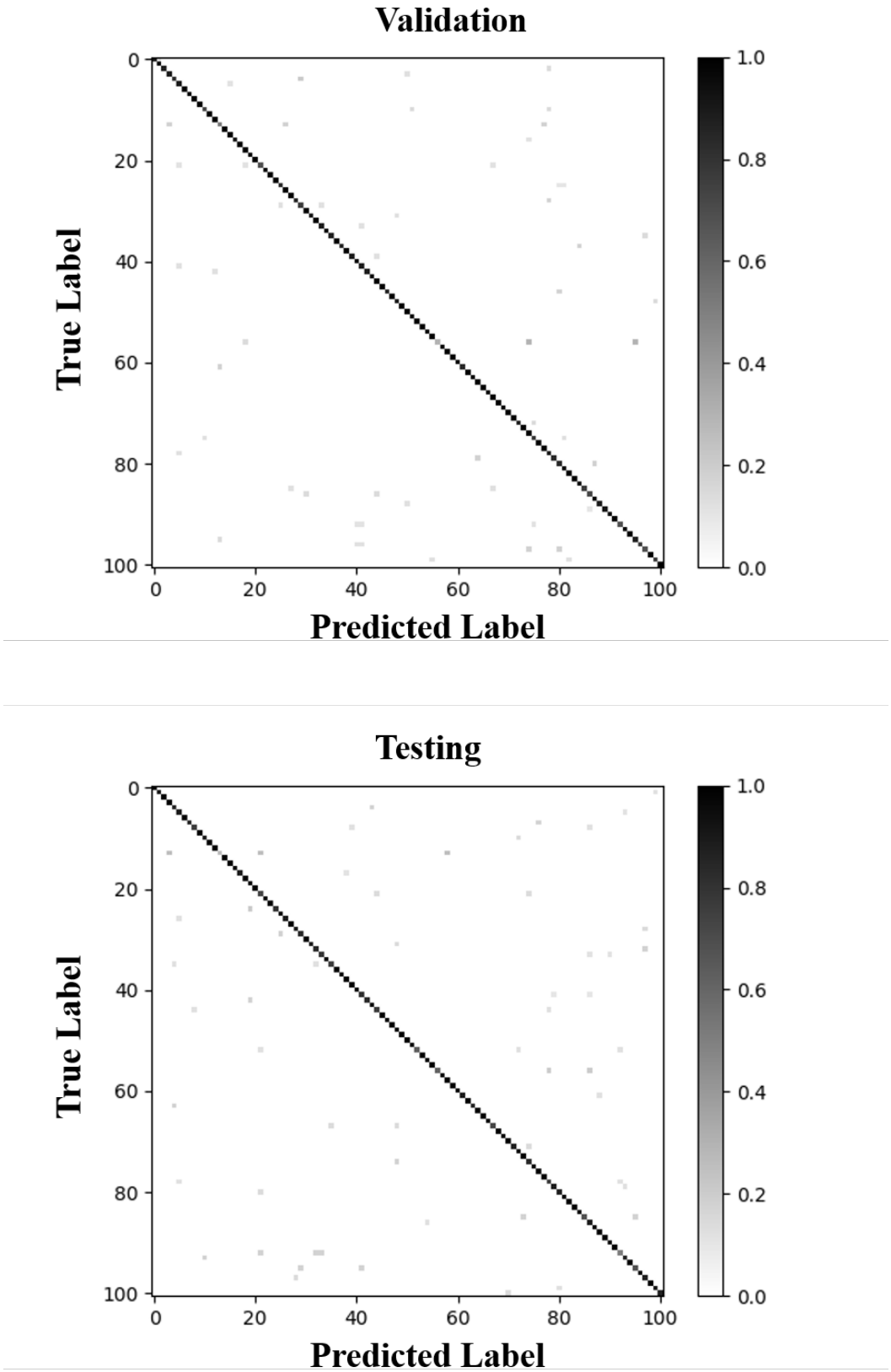

Figure A2. The confusion matrix of ViT_56_8 (optimized model) for the validation (**top**) and testing (**bottom**) sets reveals the correct and mis-prediction of the optimized ViT model.

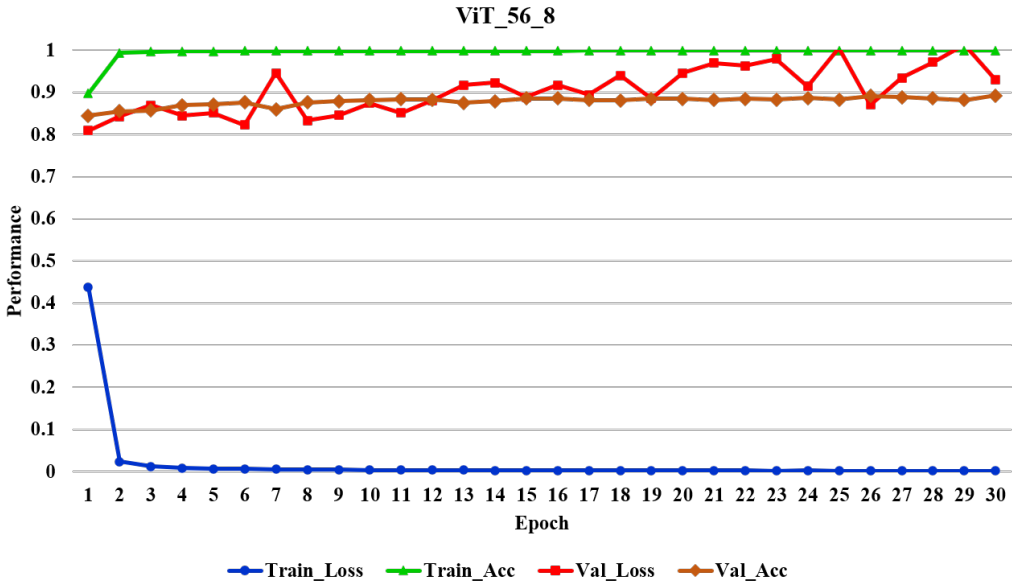

Figure A3. Evaluation metrics across 30 epochs of training for ViT_56_8 (optimized model).

Table A1. The frame-level classification report for all the ViT, ResNet, Inception, and VGG models. The macro average (MA) and weighted average (WA) were used to generate the full classification report for all the models developed in this study.

Model	Accuracy	Precision MA.	Precision WA.	Recall MA.	Recall WA.	F1 Score M.A.	F1 Score W.A.	Support
ViT_224_16_test	0.92406	0.91014	0.9303	0.90779	0.92406	0.9038	0.92344	168631
ViT_224_16_val	0.9236	0.91243	0.92668	0.90819	0.9236	0.9059	0.92167	168890
ViT_224_8_test	0.92561	0.91287	0.93024	0.91162	0.92561	0.90838	0.92503	168631
ViT_224_8_val	0.92574	0.91191	0.93025	0.91245	0.92574	0.90837	0.92527	168890
ViT_112_16_test	0.91277	0.89425	0.91914	0.89688	0.91277	0.89142	0.91306	168631
ViT_112_16_val	0.91599	0.89986	0.91962	0.90056	0.91599	0.89577	0.91454	168890
ViT_112_8_test	0.91181	0.896	0.91706	0.89369	0.91181	0.89059	0.91121	168631
ViT_112_8_val	0.91229	0.89606	0.91667	0.89547	0.91229	0.89164	0.91156	168890
ViT_56_16_test	0.89653	0.87922	0.9011	0.87842	0.89653	0.87521	0.89612	168631
ViT_56_16_val	0.89541	0.87908	0.89915	0.87414	0.89541	0.87191	0.89397	168890
ViT_56_8_test	0.89516	0.87321	0.89695	0.87512	0.89516	0.8715	0.89411	168631
ViT_56_8_val	0.89312	0.87408	0.89781	0.87235	0.89312	0.86837	0.89208	168890
ResNet101_test	0.94119	0.92764	0.94566	0.92693	0.94119	0.92412	0.94091	168631
ResNet101_val	0.9429	0.93211	0.94742	0.93141	0.9429	0.92829	0.94264	168890
ResNet18_test	0.95343	0.9413	0.95546	0.94159	0.95343	0.93891	0.95283	168631
ResNet18_val	0.94915	0.94355	0.95212	0.93969	0.94915	0.93889	0.94852	168890
ResNet50_test	0.94518	0.93282	0.94718	0.93126	0.94518	0.92892	0.94393	168631
ResNet50_val	0.94115	0.9304	0.94383	0.92689	0.94115	0.92606	0.94057	168890
Inception_v3_test	0.89637	0.8845	0.90527	0.88079	0.89637	0.87737	0.89653	168631
Inception_v3_val	0.89762	0.89083	0.9113	0.88162	0.89762	0.87914	0.89884	168890
VGG_16_test	0.90065	0.88897	0.90662	0.88417	0.90065	0.88147	0.89972	168631
VGG_16_val	0.90009	0.88997	0.90706	0.88288	0.90009	0.88224	0.90026	168890

Table A2. The video-clip-level classification report for all the ViT, ResNet, Inception, and VGG models. We translated the frame-level predictions to video-clip predictions using the postprocessing module.

Model	Accuracy	Precision MA.	Precision WA.	Recall MA.	Recall WA.	F1 Score M.A.	F1 Score W.A.	Support
ViT_224_16_test	0.94273	0.94935	0.95087	0.93892	0.94273	0.93838	0.94152	908
ViT_224_16_val	0.94053	0.94920	0.94937	0.93688	0.94053	0.93721	0.93971	908
ViT_224_8_test	0.93943	0.94383	0.94493	0.93647	0.93943	0.93605	0.93818	908
ViT_224_8_val	0.94934	0.95250	0.95389	0.94615	0.94934	0.94652	0.94899	908
ViT_112_16_test	0.93172	0.93804	0.94039	0.92838	0.93172	0.92837	0.93144	908
ViT_112_16_val	0.92952	0.93339	0.93572	0.92459	0.92952	0.92394	0.92805	908
ViT_112_8_test	0.92952	0.93530	0.93703	0.92470	0.92952	0.92436	0.92797	908
ViT_112_8_val	0.93502	0.93951	0.94165	0.93124	0.93502	0.93059	0.93400	908
ViT_56_16_test	0.92511	0.93299	0.93386	0.92160	0.92511	0.92166	0.92419	908
ViT_56_16_val	0.91740	0.92264	0.92434	0.91182	0.91740	0.91051	0.91490	908
ViT_56_8_test	0.91630	0.92018	0.92173	0.91263	0.91630	0.91223	0.91497	908
ViT_56_8_val	0.90969	0.91827	0.92007	0.90580	0.90969	0.90506	0.90870	908
ResNet101_test	0.94383	0.94689	0.94900	0.94037	0.94383	0.93972	0.94261	908
ResNet101_val	0.94824	0.94979	0.95327	0.94511	0.94824	0.94430	0.94776	908
ResNet18_test	0.95815	0.96032	0.96200	0.95593	0.95815	0.95539	0.95749	908
ResNet18_val	0.95705	0.95977	0.96071	0.95459	0.95705	0.95472	0.95653	908
ResNet50_test	0.94383	0.94932	0.94923	0.94123	0.94383	0.94039	0.94206	908
ResNet50_val	0.94273	0.94390	0.94744	0.93946	0.94273	0.93849	0.94206	908
Inception_v3_test	0.92731	0.93923	0.94104	0.92461	0.92731	0.92501	0.92762	908
Inception_v3_val	0.92841	0.94126	0.94231	0.92586	0.92841	0.92772	0.92975	908
VGG_16_test	0.93833	0.94885	0.94887	0.93611	0.93833	0.93636	0.93759	908
VGG_16_val	0.93282	0.93872	0.94062	0.93020	0.93282	0.93018	0.93249	908

Table A3. The inference latency (sec) for testing sets (168631frames) using three SageMaker instances measured using the identical machine learning and Python libraries.

Model	Head	Input Size	p3.16xlarge	p3.8xlarge	p3.2xlarge
ViT_224_16	16	224	1474.0831	1434.5108	1420.8712
ViT_224_8	8	224	1343.1789	1300.7208	1304.822
ViT_112_16	16	112	1016.7971	1279.6002	1241.0731
ViT_112_8	8	112	995.6327	977.1326	962.3076
ViT_56_16	16	56	861.1608	859.4907	862.2177
ViT_56_8	8	56	856.8517	853.8936	851.4978

References

1. Rahmani, H.; Bennamoun, M.; Ke, Q. Human Action Recognition from Various Data Modalities: A Review. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *45*, 3200–3225. [CrossRef]
2. Pareek, P.; Thakkar, A. A survey on video-based Human Action Recognition: Recent updates, datasets, challenges, and applications. *Artif. Intell. Rev.* **2020**, *54*, 2259–2322. [CrossRef]
3. Ahn, D.; Kim, S.; Hong, H.; Ko, B.C. STAR-Transformer: A Spatio-temporal Cross Attention Transformer for Human Action Recognition. In Proceedings of the 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 2–7 January 2023. [CrossRef]
4. Morshed, M.G.; Sultana, T.; Alam, A.; Lee, Y.K. Human Action Recognition: A Taxonomy-Based Survey, Updates, and Opportunities. *Sensors* **2023**, *23*, 2182. [CrossRef] [PubMed]
5. Zhang, H.B.; Zhang, Y.X.; Zhong, B.; Lei, Q.; Yang, L.; Du, J.X.; Chen, D.S. A Comprehensive Survey of Vision-Based Human Action Recognition Methods. *Sensors* **2019**, *19*, 1005. [CrossRef] [PubMed]
6. Johnson, W.R.; Mian, A.; Donnelly, C.J.; Lloyd, D.; Alderson, J. Predicting athlete ground reaction forces and moments from motion capture. *Med. Biol. Eng. Comput.* **2018**, *56*, 1781–1792. [CrossRef]
7. Lee, E.J.; Kim, Y.H.; Kim, N.; Kang, D.W. Deep into the Brain: Artificial Intelligence in Stroke Imaging. *J. Stroke* **2017**, *19*, 277–285. [CrossRef]
8. Yu, M.; Huang, Q.; Qin, H.; Scheele, C.; Yang, C. Deep learning for real-time social media text classification for situation awareness—Using Hurricanes Sandy, Harvey, and Irma as case studies. *Int. J. Digit. Earth* **2019**, *12*, 1230–1247. [CrossRef]

9. Jayakodi, N.K.; Chatterjee, A.; Choi, W.; Doppa, J.R.; Pande, P.P. Trading-Off Accuracy and Energy of Deep Inference on Embedded Systems: A Co-Design Approach. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2018**, *37*, 2881–2893. [CrossRef]
10. Jiang, Z.; Chen, T.; Li, M. Efficient Deep Learning Inference on Edge Devices. 2018. Available online: https://www.amazon.science/publications/efficient-deep-learning-inference-on-edge-devices (accessed on 30 August 2023).
11. Li, Y.; Han, Z.; Zhang, Q.; Li, Z.; Tan, H. Automating Cloud Deployment for Deep Learning Inference of Real-time Online Services. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020. [CrossRef]
12. Cipriani, G.; Bottin, M.; Rosati, G. Applications of Learning Algorithms to Industrial Robotics. In *Mechanisms and Machine Science*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 260–268. [CrossRef]
13. Gheisari, M.; Wang, G.; Bhuiyan, M.Z.A. A Survey on Deep Learning in Big Data. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017. [CrossRef]
14. Aslam, A.; Curry, E. A Survey on Object Detection for the Internet of Multimedia Things (IoMT) using Deep Learning and Event-based Middleware: Approaches, Challenges, and Future Directions. *Image Vis. Comput.* **2021**, *106*, 104095. [CrossRef]
15. Sarraf, S.; Tofighi, G. Deep learning-based pipeline to recognize Alzheimer's disease using fMRI data. In Proceedings of the 2016 Future Technologies Conference (FTC), San Francisco, CA, USA, 6–7 December 2016; pp. 816–820.
16. Sarraf, S.; DeSouza, D.D.; Anderson, J.; Tofighi, G. DeepAD: Alzheimer's Disease Classification via Deep Convolutional Neural Networks using MRI and fMRI. *BioRxiv* **2016**. [CrossRef]
17. Sarraf, S.; Desouza, D.D.; Anderson, J.A.E.; Saverino, C. MCADNNet: Recognizing Stages of Cognitive Impairment Through Efficient Convolutional fMRI and MRI Neural Network Topology Models. *IEEE Access* **2019**, *7*, 155584–155600. [CrossRef] [PubMed]
18. Graves, A.; Liwicki, M.; Fernandez, S.; Bertolami, R.; Bunke, H.; Schmidhuber, J. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 855–868. [CrossRef] [PubMed]
19. Graves, A.; Bellemare, M.G.; Menick, J.; Munos, R.; Kavukcuoglu, K. Automated curriculum learning for neural networks. In Proceedings of the International Conference on Machine Learning. Pmlr, Sydney, NSW, Australia, 6–11 August 2017; pp. 1311–1320.
20. Sun, X.; Lu, W. Understanding Attention for Text Classification. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020. [CrossRef]
21. Zanca, D.; Gori, M.; Melacci, S.; Rufa, A. Gravitational models explain shifts on human visual attention. *Sci. Rep.* **2020**, *10*, 16335. [CrossRef]
22. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015. [CrossRef]
23. Bahsoon, R.; Ali, N.; Heisel, M.; Maxim, B.; Mistrik, I. Introduction. Software Architecture for Cloud and Big Data: An Open Quest for the Architecturally Significant Requirements. In *Software Architecture for Big Data and the Cloud*; Elsevier: Amsterdam, The Netherlands, 2017; pp. 1–10. [CrossRef]
24. Seda, P.; Masek, P.; Sedova, J.; Seda, M.; Krejci, J.; Hosek, J. Efficient Architecture Design for Software as a Service in Cloud Environments. In Proceedings of the 2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Moscow, Russia, 5–9 November 2018. [CrossRef]
25. Vikash.; Mishra, L.; Varma, S. Performance evaluation of real-time stream processing systems for Internet of Things applications. *Future Gener. Comput. Syst.* **2020**, *113*, 207–217. [CrossRef]
26. Needham, C.J.; Boyle, R.D. Performance Evaluation Metrics and Statistics for Positional Tracker Evaluation. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 278–289. [CrossRef]
27. Bhardwaj, S.; Srinivasan, M.; Khapra, M.M. Efficient Video Classification Using Fewer Frames. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019. [CrossRef]
28. Hu, D.; Krishnamachari, B. Fast and Accurate Streaming CNN Inference via Communication Compression on the Edge. In Proceedings of the 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI), Sydney, Australia, 21–24 April 2020. [CrossRef]
29. Geva, R.; Zivan, M.; Warsha, A.; Olchik, D. Alerting, orienting or executive attention networks: Differential patters of pupil dilations. *Front. Behav. Neurosci.* **2013**, *7*, 145. [CrossRef] [PubMed]
30. Larochelle, H.; Hinton, G.E. Learning to combine foveal glimpses with a third-order Boltzmann machine. *Adv. Neural Inf. Process. Syst.* **2010**, *23*, 1–9.
31. Borji, A.; Cheng, M.M.; Hou, Q.; Jiang, H.; Li, J. Salient object detection: A survey. *Comput. Vis. Media* **2019**, *5*, 117–150. [CrossRef]
32. Itti, L.; Koch, C.; Niebur, E. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **1998**, *20*, 1254–1259. [CrossRef]
33. Gosmann, J.; Voelker, A.; Eliasmith, C. A Spiking Independent Accumulator Model for Winner-Take-All Computation. In Proceedings of the CogSci, London, UK, 26–29 July 2017.

34. Li, S.; Zhou, M.; Luo, X.; You, Z.H. Distributed Winner-Take-All in Dynamic Networks. *IEEE Trans. Autom. Control* **2017**, *62*, 577–589. [CrossRef]
35. Bello, I.; Zoph, B.; Le, Q.; Vaswani, A.; Shlens, J. Attention Augmented Convolutional Networks. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019. [CrossRef]
36. Wu, B.; Xu, C.; Dai, X.; Wan, A.; Zhang, P.; Yan, Z.; Tomizuka, M.; Gonzalez, J.; Keutzer, K.; Vajda, P. Visual transformers: Token-based image representation and processing for computer vision. *arXiv* **2020**, arXiv:2006.03677.
37. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
38. Fu, J.; Liu, J.; Tian, H.; Li, Y.; Bao, Y.; Fang, Z.; Lu, H. Dual Attention Network for Scene Segmentation. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019. [CrossRef]
39. Tay, C.P.; Roy, S.; Yap, K.H. AANet: Attribute Attention Network for Person Re-Identifications. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019. [CrossRef]
40. Zhao, T.; Wu, X. Pyramid Feature Attention Network for Saliency Detection. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019. [CrossRef]
41. Arnab, A.; Dehghani, M.; Heigold, G.; Sun, C.; Lučić, M.; Schmid, C. Vivit: A video vision transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Virtual, 11–17 October 2021; pp. 6836–6846.
42. Bertasius, G.; Wang, H.; Torresani, L. Is space-time attention all you need for video understanding? In Proceedings of the ICML, Virtual Event, 18–24 July 2021; Volume 2, p. 4.
43. Fan, H.; Xiong, B.; Mangalam, K.; Li, Y.; Yan, Z.; Malik, J.; Feichtenhofer, C. Multiscale vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Virtual, 11–17 October 2021; pp. 6824–6835.
44. Yang, J.; Dong, X.; Liu, L.; Zhang, C.; Shen, J.; Yu, D. Recurring the Transformer for Video Action Recognition. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022. [CrossRef]
45. Chen, J.; Ho, C.M. MM-ViT: Multi-Modal Video Transformer for Compressed Video Action Recognition. In Proceedings of the 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 3–8 January 2022. [CrossRef]
46. Ma, Y.; Yuan, L.; Abdelraouf, A.; Han, K.; Gupta, R.; Li, Z.; Wang, Z. M2DAR: Multi-View Multi-Scale Driver Action Recognition with Vision Transformer. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Vancouver, BC, Canada, 17–24 June 2023. [CrossRef]
47. Xing, Z.; Dai, Q.; Hu, H.; Chen, J.; Wu, Z.; Jiang, Y.G. SVFormer: Semi-supervised Video Transformer for Action Recognition. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023. [CrossRef]
48. Ma, Y.; Wang, R. Relative-position embedding based spatially and temporally decoupled Transformer for action recognition. *Pattern Recognit.* **2023**, 109905. [CrossRef]
49. Mu, L.; Li, Z.; Xiao, W.; Zhang, R.; Wang, P.; Liu, T.; Min, G.; Li, K. A Fine-Grained End-to-End Latency Optimization Framework for Wireless Collaborative Inference. *IEEE Internet Things J.* **2023**. [CrossRef]
50. Zhang, Y.; Jiang, H.; Zhu, Y.; Zhang, R.; Cao, Y.; Zhu, C.; Wang, W.; Dong, D.; Li, X. LOCP: Latency-optimized channel pruning for CNN inference acceleration on GPUs. *J. Supercomput.* **2023**, *79*, 14313–14341. [CrossRef]
51. Li, X.; Gong, X.; Wang, D.; Zhang, J.; Baker, T.; Zhou, J.; Lu, T. ABM-SpConv-SIMD: Accelerating Convolutional Neural Network Inference for Industrial IoT Applications on Edge Devices. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 3071–3085. [CrossRef]
52. Gannon, D.; Barga, R.; Sundaresan, N. Cloud-Native Applications. *IEEE Cloud Comput.* **2017**, *4*, 16–21. [CrossRef]
53. Sether, A. Cloud Computing Benefits. *SSRN Electron. J.* **2016**. [CrossRef]
54. Zhang, C.; Yu, M.; Wang, W.; Yan, F. {MArk}: Exploiting Cloud Services for {Cost-Effective},{SLO-Aware} Machine Learning Inference Serving. In Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC 19), Renton, WA, USA, 10–12 July 2019; pp. 1049–1062.
55. Zhang, R. Making convolutional networks shift-invariant again. In Proceedings of the International Conference on Machine Learning (PMLR), Beach, CA, USA, 9–15 June 2019; pp. 7324–7334.
56. Tsagkatakis, G.; Jaber, M.; Tsakalides, P. Goal!! Event detection in sports video. *Electron. Imaging* **2017**, *29*, 15–20. [CrossRef]
57. Khan, A.; Lazzerini, B.; Calabrese, G.; Serafini, L. Soccer Event Detection. In Proceedings of the Computer Science & Information Technology (CS & IT), Dubai, UAE, 1–2 July 2018; Academy & Industry Research Collaboration Center (AIRCC); 2018. [CrossRef]
58. Sarraf, S.; Noori, M. Multimodal deep learning approach for event detection in sports using Amazon SageMaker. *AWS Mach. Learn. Blog* **2021**, *1*, 1–12.
59. Pandit, S.; Shukla, P.K.; Tiwari, A.; Shukla, P.K.; Maheshwari, M.; Dubey, R. Review of video compression techniques based on fractal transform function and swarm intelligence. *Int. J. Mod. Phys. B* **2020**, *34*, 2050061. [CrossRef]
60. Mohammed, T.; Joe-Wong, C.; Babbar, R.; Francesco, M.D. Distributed Inference Acceleration with Adaptive DNN Partitioning and Offloading. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020. [CrossRef]

61. Sengar, S.S.; Mukhopadhyay, S. Motion segmentation-based surveillance video compression using adaptive particle swarm optimization. *Neural Comput. Appl.* **2019**, *32*, 11443–11457. [CrossRef]
62. Dong, F.; Wang, H.; Shen, D.; Huang, Z.; He, Q.; Zhang, J.; Wen, L.; Zhang, T. Multi-exit DNN Inference Acceleration based on Multi-Dimensional Optimization for Edge Intelligence. *IEEE Trans. Mob. Comput.* **2022**. [CrossRef]
63. Uy, W.I.T.; Hartmann, D.; Peherstorfer, B. Operator inference with roll outs for learning reduced models from scarce and low-quality data. *Comput. Math. Appl.* **2023**, *145*, 224–239. [CrossRef]
64. Soomro, K.; Zamir, A.R.; Shah, M. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv* **2012**, arXiv:1212.0402.
65. Ma, S.; Bargal, S.A.; Zhang, J.; Sigal, L.; Sclaroff, S. Do less and achieve more: Training CNNs for action recognition utilizing action images from the Web. *Pattern Recognit.* **2017**, *68*, 334–345. [CrossRef]
66. Zhu, W.; Hu, J.; Sun, G.; Cao, X.; Qiao, Y. A Key Volume Mining Deep Framework for Action Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
67. Janghel, R.; Rathore, Y. Deep convolution neural network based system for early diagnosis of Alzheimer's disease. *Irbm* **2021**, *42*, 258–267. [CrossRef]
68. Wang, L.; Qiao, Y.; Tang, X. Action recognition with trajectory-pooled deep-convolutional descriptors. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015. [CrossRef]
69. Potdar, A.; Barbhaya, P.; Nagpure, S. Face Recognition for Attendance System using CNN based Liveliness Detection. In Proceedings of the 2022 International Conference on Advances in Computing, Communication and Materials (ICACCM), Dehradun, India, 10–11 November 2022. [CrossRef]
70. Jin, Y.; Qian, Z.; Sun, G. A real-time multimedia streaming transmission control mechanism based on edge cloud computing and opportunistic approximation optimization. *Multimed. Tools Appl.* **2018**, *78*, 8911–8926. [CrossRef]
71. Zhang, J.; Wang, D.; Yu, D. TLSAN: Time-aware long- and short-term attention network for next-item recommendation. *Neurocomputing* **2021**, *441*, 179–191. [CrossRef]
72. Duan, H.; Zhao, Y.; Xiong, Y.; Liu, W.; Lin, D. Omni-Sourced Webly Supervised Learning for Video Recognition. In *Computer Vision—ECCV 2020*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 670–688. [CrossRef]
73. Qiu, Z.; Yao, T.; Ngo, C.W.; Tian, X.; Mei, T. Learning Spatio-Temporal Representation With Local and Global Diffusion. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seoul, Republic of Korea, 27 October–2 November 2019. [CrossRef]
74. Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al. The kinetics human action video dataset. *arXiv* **2017**, arXiv:1705.06950.
75. Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; Serre, T. HMDB: A large video database for human motion recognition. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2556–2563. [CrossRef]