BenchCouncil Transactions on Benchmarks, Standards and Evaluations

TBench
Volume 3, Issue 4
2023

Original Articles

ISSN: 2772-4859

BenchCouncil Transactions on Benchmarks, Standards and Evaluations (TBench) is an open-access multi-disciplinary journal dedicated to benchmarks, standards, evaluations, optimizations, and data sets. This journal is a peer-reviewed, subsidized open access journal where The International Open Benchmark Council pays the OA fee. Authors do not have to pay any open access publication fee. However, at least one of the authors must register BenchCouncil International Symposium on Benchmarking, Measuring and Optimizing (Bench) (https://www.benchcouncil.org/bench/) and present their work. It seeks a fast-track publication with an average turnaround time of one month.

CONTENTS

BenchCouncil Transactions on Benchmarks, Standards and Evaluations 3 (2023) 100148

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/

Research article

A pluggable single-image super-resolution algorithm based on second-order gradient loss

Shuran Lin [a,b], Chunjie Zhang [a,b,*], Yanwu Yang [c]

[a] Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing Jiaotong University, Beijing, 100044, China
[b] Institute of Information Science, Beijing Jiaotong University, Beijing, 100044, China
[c] School of Management, Huazhong University of Science and Technology, Wuhan, 430074, China

ARTICLE INFO

Keywords:
Single-image super-resolution
Gradient
Texture
Loss function

ABSTRACT

Convolutional neural networks for single-image super-resolution have been widely used with great success. However, most of these methods use L1 loss to guide network optimization, resulting in blurry restored images with sharp edges smoothed. This is because L1 loss limits the optimization goal of the network to the statistical average of all solutions within the solution space of that task. To go beyond the L1 loss, this paper designs an image super-resolution algorithm based on second-order gradient loss. We impose additional constraints by considering the high-order gradient level of the image so that the network can focus on the recovery of fine details such as texture during the learning process. This helps to alleviate the problem of restored image texture over-smoothing to some extent. During network training, we extract the second-order gradient map of the generated image and the target image of the network by minimizing the distance between them, this guides the network to pay attention to the high-frequency detail information in the image and generate a high-resolution image with clearer edge and texture. Besides, the proposed loss function has good embeddability and can be easily integrated with existing image super-resolution networks. Experimental results show that the second-order gradient loss can significantly improve both Learned Perceptual Image Patch Similarity (LPIPS) and Frechet Inception Distance score (FID) performance over other image super-resolution deep learning models.

1. Introduction

As a well-known image restoration task, single-image super-resolution (SISR) aims to convert a low-resolution (LR) image into its corresponding high-resolution (HR) version. In recent years, SISR has gained significant attention from researchers owing to its practical applications in various fields, including video surveillance [1–3], medical imaging [4–6], and so on. Moreover, SISR can also be used in combination with other high-level computer vision tasks, such as object detection [7,8] and semantic segmentation [9,10], to improve their performance. However, SISR is inherently a challenging and ill-posed task, as LR images lack crucial texture details present in HR images, making it difficult to generate HR images from LR images alone. Furthermore, since a single LR image can be generated from multiple HR images that have undergone different types of degradation, the solution to the SR problem may not be unique.

Convolutional Neural Networks (CNNs) have recently shown impressive performance in information recovery due to their ability to handle complex data, and have thus been applied to the field of SISR. However, several CNN-based SISR methods currently popular prioritize high Peak Signal-to -Noise Ratio (PSNR) and Structural Similarity (SSIM) scores, which can lead to visually blurry restored images. This is because these methods often neglect the structural prior knowledge within the image and focus only on minimizing the mean absolute error between the recovered HR image and the ground truth image. Consequently, the optimization objective of the network becomes the statistical mean of all possible solutions in this one-to-many problem, resulting in blurry reconstructed images.

Images can be broken down into various frequency components, such as high-frequency and low-frequency components. The low-frequency component corresponds to its smooth regions, such as the sky, which are relatively simpler to restore. The high-frequency component pertains to its detailed regions, such as the textures of buildings, which are comparatively more challenging to restore. The human visual system is particularly sensitive to the details in images, especially the edges and textures, which play a crucial role in the perception of image quality [11]. Therefore, the accuracy of restoring the high-frequency

* Correspondence to: Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing Jiaotong University, No.3 Shangyuancun, Haidian District, Beijing, China.
E-mail address: cjzhang@bjtu.edu.cn (C. Zhang).

https://doi.org/10.1016/j.tbench.2023.100148
Received 16 August 2023; Received in revised form 8 October 2023; Accepted 8 December 2023
Available online 14 December 2023

components is essential for achieving visually pleasing results, and blurry images often occur when edge and texture details are lost due to excessive smoothing.

Numerous studies have demonstrated that incorporating prior knowledge of images, such as total variation prior [12,13], sparse prior [14–16], and gradient prior [17,18], can partially alleviate the ill-posedness of the SISR task. These prior knowledge can be viewed as supplementary constraints on the optimization objective of the network, which narrow down the solution space of the task. Among all these prior knowledge, the gradient prior is one of the most effective, as it can suppress noise and preserve edges during image reconstruction. In fact, an image can be regarded as a two-dimensional discrete function, and the gradient of the image is actually the derivative of this two-dimensional discrete function, which measures the change rate of the pixel grayscale value of the image. As the grayscale values of image pixels tend to vary greatly in edge and texture areas, the gradient map of images can accurately capture the edges and texture details of images. In the field of mathematics, the first-order derivative of a function provides information that can be utilized to describe the shape of the functional image, such as monotonicity. While the second-order derivative of the function contains more information than the first-order derivative, which has extremely important guiding significance for accurately modeling the functional image. Similarly, in the field of image processing, the second-order gradient map of images may contain more informative prior knowledge than the first-order gradient map. To validate this idea, we apply the principles of function derivation to generate the second-order gradient map of images and visualize it for a more intuitive comparison with the first-order gradient map. As shown in Fig. 1, the second-order gradient map shows more detailed information than the first-order gradient map. If fully utilized during network optimization, it can further compress the solution space of this task and reduce the difficulty of image restoration.

Based on the aforementioned discussion, this paper proposes an image super-resolution algorithm based on the second-order gradient (SG) loss. This algorithm replaces the loss function of the network with a combination of the SG loss and the L1 loss. The SG loss takes the second-order gradient map of the image as the starting point. To be specific, it first extracts the second-order gradient maps of the restored image and the HR image and then minimizes the distance between them to fully exploit the high-frequency information contained in the second-order gradient map of the image. This encourages the network to concentrate on the restoration of high-frequency components such as textures and image boundaries, improving the blurring of restored images caused by some existing methods that only use L1 loss as a constraint. The main contributions of this paper can be summarized as follows:

- We propose an image super-resolution algorithm based on the second-order gradient (SG) loss. By combining the SG loss with the L1 loss, our algorithm effectively guides the network optimization process and mitigates the problem of excessive blurring in the images restored by some existing image restoration methods to some extent. The SG loss can be easily integrated into most existing SR methods without adding extra training parameters.
- The experimental results on five widely used benchmark datasets demonstrate that the proposed SG loss can enhance high-frequency information in images and help the network recover clearer and more natural textures and edges.

2. Related works

This section provides a review of relevant image super-resolution methods from two perspectives: single-image super-resolution methods and gradient-guided super-resolution methods.

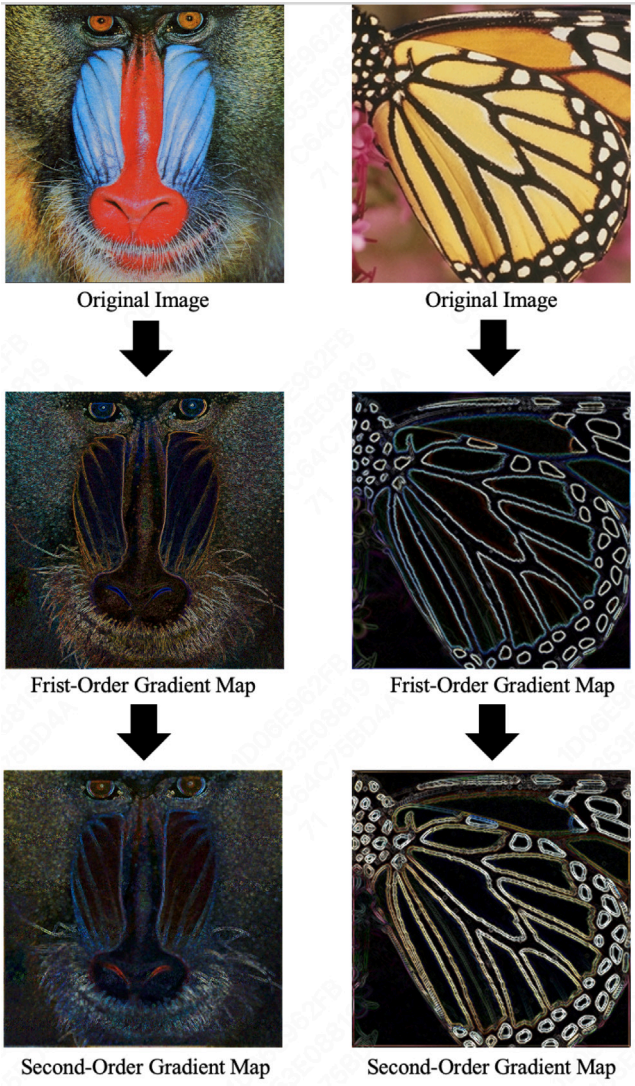


Fig. 1. Visualization of the first-order and second-order gradient maps of images.

2.1. Single image super-resolution methods

To date, numerous SISR methods have been proposed by researchers, which can be broadly classified into three categories: interpolation-based methods [19], signal processing-based methods [20–22], and deep learning-based methods [23–50].

In the initial stages of research on SISR, interpolation-based methods were commonly used. The main idea of these methods is to infer the pixel value at a specific position in the HR image by performing a weighted average of the known pixel values in the LR image surrounding that position. Different weighting schemes have been designed for image interpolation based on the fact that common pixel variations in a local region of an image can be approximated by a continuous function. For instance, bilinear interpolation, which leverages local linearity, and bicubic interpolation [19], which utilizes high-order continuity, are two examples of interpolation methods that have been proposed. Despite their simplicity and computational efficiency, these methods often result in generated images that exhibit unnatural artifacts and structural distortions. This is primarily due to the fact that the pixel variations within an image are often highly complex and cannot be accurately described by such simple predefined functions, particularly in the case of images with intricate textures. Signal processing-based SISR methods have been designed to address this issue. They apply signal

processing techniques such as sparse representation [22], local adaptive filtering [21], and wavelet transform [20] to LR images to obtain their corresponding HR images. While signal processing-based methods have shown improvements in image restoration quality compared to interpolation-based methods, they often come with high computational complexity and are susceptible to noise.

For the last few years, deep learning-based methods have revealed extraordinary capabilities in feature learning and extraction, allowing neural networks to theoretically simulate any function. Through end-to-end model training, these deep learning networks can learn the mapping relationship between LR and HR images from massive data directly. These data-driven deep learning approaches lead to momentous performance gains compared to earlier traditional approaches. As trailblazers, Dong et al. are the first to establish a connection between CNN and image SR reconstruction. They devise a super-resolution convolutional neural network composed of three convolutional layers, which lay the groundwork for deep learning-based SISR methods. Nonetheless, the limited receptive field of the three convolutional layers restricts their capacity to perfectly leverage the surrounding pixel information, leading to constrained performance enhancement. For the purpose of enlarging the receptive field, Kim et al. [27] stack more convolutional layers and integrate residual learning to tackle the problem of gradient vanishing triggered by network thickening. Given the distinct sizes of LR and HR images in SISR, the aforementioned methods typically necessitate preprocessing of LR images utilizing bicubic interpolation to upscale them to match the size of HR images before feeding them into the network for training. Nevertheless, this preprocessing is time-consuming and exacerbates the noise and blur in LR images. To deal with this issue, a deconvolution layer is appended by Dong et al. [30] at the end of the network to accomplish end-to-end mapping from LR images to HR images. Shi et al. [32] present a novel sub-pixel convolutional layer that can achieve magnification by dynamically adjusting the number of feature channels. Both of them place the upscaling operation of the LR image at the final stage of the network and make it learnable. This can not only decrease the computational burden but also enhance the precision of image restoration.

Subsequently, SR models based on neural networks have emerged continuously. For instance, Zhang et al. [34] employ a dense connection structure to augment feature propagation through feature reuse. Li et al. [35] devise a multi-scale network to selectively extract image features of varying scales to facilitate image reconstruction, which leads to further performance improvement compared to the model using only a single scale. According to Zhang et al. [36], most existing methods treat LR input features indiscriminately and disregard the correlation between low-frequency information. Consequently, they integrate the attention mechanism into the SR network to enable it to concentrate on the more critical parts of the image for restoration. Several recent studies have attempted to combine transformers from the field of natural language processing with SR networks, obtaining state-of-the-art performance.

2.2. Gradient guided super-resolution methods

By exploiting gradient prior knowledge in many traditional methods [12,51–54], the solution space can be narrowed to generate a sharper image. For example, Fattal [52] designs a method that leverages image gradient edge statistics to learn the prior correlations across different resolutions. Zhu et al. [51] introduce an innovative method that gathers a dictionary of gradient patterns and characterizes deformable gradient combinations. Yan et al. [53] propose a stochastic resonance method based on gradient contour sharpness. Motivated by the effectiveness of gradient prior in traditional methods, some recent works have also endeavored to integrate image prior knowledge with neural networks [17,18,55]. Yang et al. [17] employ a pre-trained edge detector to extract image gradients, which are subsequently utilized to guide the deep network in reconstructing SR images. Ma et al. [18] construct a dual-branch joint optimization network consisting of a main SR branch and a gradient-assisted branch, where the gradient-assisted branch takes the gradient map extracted from the LR image as input, and the optimization target becomes the gradient map of its corresponding HR image. While previous methods leverage gradient prior knowledge to enhance the visual quality of restored images, they often incorporate learnable parameters associated with gradient information into the model significantly increasing its complexity and diminishing its computational efficiency. Unlike them, the proposed method in this paper utilizes a second-order gradient prior solely during network optimization to provide supplementary supervision information, without adding any learnable parameters. Hence, the computational cost can be disregarded.

3. Second-order gradient loss guided single-image super-resolution

3.1. Problem definition

For the task of SISR, the goal is to predict a reasonable HR image I^{SR} from a LR input image I^{LR}, given its corresponding ground truth HR image I^{HR}, and ensure that the predicted HR image I^{SR} is as similar as possible to the ground truth HR image I^{HR}. Consequently, during the actual model training, it is imperative to use pre-existing paired LR and HR image pairs (I^{LR}, I^{HR}). In reality, the LR image is typically obtained from the HR image through various types of degradation, but due to the complex and diverse forms of degradation and difficult modeling, for convenience of research, most works simply model the degradation process of the image as a bicubic interpolation downsampling operation. Therefore, the corresponding LR image can be generated from the HR image by the following formula:

$$I^{LR} = (I^{HR}) \downarrow_s \tag{1}$$

where $\downarrow_s$ denotes a bicubic interpolation downsampling operation with a scaling factor of s. Typically, both LR and HR images are 3-channel RGB images, with sizes of $3\times h\times w$ and $3\times s\cdot h\times s\cdot w$, respectively, where h and w are the height and width of the LR image. If we represent the SR network as F with parameters θ, then the process of image SR can be expressed as:

$$I^{SR} = F(I^{LR}; \theta) \tag{2}$$

Assuming that the loss function L can be applied to guide the network learning. In this case, we can formulate the optimization process of the network as follows:

$$\widehat{\theta} = arg\min_{\theta} \mathbb{E}_{I^{SR}} L(F(I^{LR}; \theta), I^{HR}) \tag{3}$$

3.2. Second-order gradient loss

Most existing deep learning-based SR methods primarily rely on the L1 loss to constrain network training. The L1 loss is computed by measuring the mean absolute error between the predicted image I^{SR} generated by the network and the ground truth HR image I^{HR} at each pixel. This loss function tends to yield high Peak Signal-to-Noise Ratio (PSNR) values for the restored image. In fact, one of the limitations of using the L1 loss function in SR is that the visual results often exhibit blurriness and lack of preservation of sharp edges present in the original image. Despite the limitation aforementioned, the L1 loss function remains the most popular choice due to its effectiveness in accelerating convergence and improving the overall performance.

$$L_1 = \mathbb{E}_{I^{SR}} \| (I^{HR} - I^{SR}) \|_1 \tag{4}$$

Considering that the L1 loss treats high-frequency and low-frequency information equally, without taking into account the fact that the inherent difficulty in recovering high-frequency details, this paper proposes to utilize the second-order gradient map of the image as additional

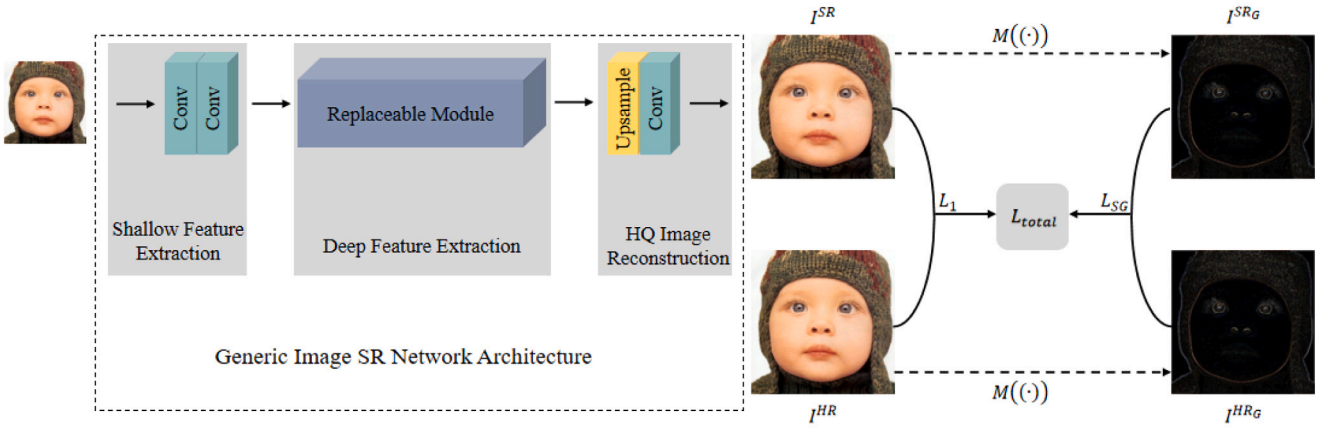


Fig. 2. Overall framework of our proposed pluggable SISR algorithm based on second-order gradient loss. The left half of this figure represents a generic deep learning-based image SR network architecture which can be easily replaced. I^{HR_G}, I^{SR_G} respectively represent the second-order gradient map extracted by using the gradient extraction function $M(\cdot)$ twice from high-resolution image I^{HR} and super-resolution image I^{SR}.

Table 1
Quantitative comparisons of cnn-based SISR models with and without second-order gradient loss on five benchmark datasets for ×4 SR. Best results are **highlighted**.

DataSet	Metric	EDSR	EDSR+SG	RDN	RDN+SG	RCAN	RCAN+SG	SwinIR	SwinIR+SG
Set5	LPIPS ↓	0.1728	**0.1446**	0.1716	**0.1560**	0.1720	**0.1401**	0.1700	**0.1412**
	FID ↓	58.86	**56.52**	57.88	**52.65**	59.74	**54.27**	58.80	**55.75**
Set14	LPIPS ↓	0.2776	**0.2353**	0.2808	**0.2564**	0.2783	**0.2268**	0.2705	**0.2262**
	FID ↓	86.45	**80.94**	88.75	**86.55**	91.95	**86.51**	89.17	**82.09**
Urban100	LPIPS ↓	0.2037	**0.1837**	0.2107	**0.1984**	0.2047	**0.1756**	0.1923	**0.1698**
	FID ↓	25.56	**23.10**	26.12	**23.85**	25.71	**22.39**	24.54	**21.63**
B100	LPIPS ↓	0.3589	**0.3018**	0.3634	**0.3274**	0.3602	**0.2906**	0.3549	**0.2894**
	FID ↓	96.08	**88.47**	96.36	**90.86**	98.15	**83.83**	95.59	**84.28**
Manga109	LPIPS ↓	0.0997	**0.0856**	0.1018	**0.0931**	0.0991	**0.0810**	0.0938	**0.0787**
	FID ↓	12.58	**10.77**	13.25	**11.39**	12.48	**10.80**	11.82	**9.97**

supervision information in the optimization process to encourage the network to pay more attention to high-frequency information during the recovery process and alleviate the problem of smoothing sharp edges. The reason why not utilizing higher-order gradient maps of the image is that studies have indicated that as the order of the gradient increases, the detail information in the gradient map becomes more intricate and complex, which may lead to instability during training and introduce additional errors. The loss function proposed in this paper involves the extraction of the second-order gradient map of the image. More precisely, to obtain the first-order gradient map of the image, we calculate the pixel-wise differences between adjacent pixels in both the horizontal and vertical directions. Subsequently, the second-order gradient map of the image is derived by calculating the pixel-wise differences between adjacent pixels of the first-order gradient map. During the actual training process, an additional constraint is imposed on the predicted high-resolution image I^{SR}. This constraint is to minimize the discrepancy between the second-order gradient maps of I^{SR} and I^{HR}. The gradient map of the image I can be generated using the following formula:

$$\begin{aligned} dx(i,j) &= I(i+1,j) - I(i-1,j) \\ dy(i,j) &= I(i,j+1) - I(i,j-1) \\ \nabla I(x,y) &= (dx(i,j), dy(i,j)) \\ M(I) &= \|\nabla I\| \end{aligned} \tag{5}$$

where (i,j) represents the coordinates of any point in the image, and $I(i,j)$ represents the pixel value of the image at (i,j). The operation $M(\cdot)$ refers to the process of extracting image gradients. It can be implemented by designing a convolution layer with fixed-weight kernels. In this paper, the weights of the convolution kernel are designed by simulating the Sobel filter. The Sobel filter is capable of detecting edge information in both the horizontal and vertical directions, making it an effective way to extract the gradient map from the image. Compared with other edge detection filters, it is not only simple to implement and fast in computation but also accurate in edge localization and good noise suppression in images. By applying the operation $M(\cdot)$ twice, we can obtain the second-order gradient map of the image. In summary, the proposed second-order gradient loss in this paper can be formulated as follows:

$$L_{SG} = \mathbb{E}_{I^{SR}} \| M(M(I^{HR})) - M(M(I^{SR})) \|_1 \tag{6}$$

Nevertheless, the second-order gradient loss primarily captures high-frequency information while lacking low-frequency information. To provide comprehensive guidance for network optimization, it is weighted and combined with the L1 loss to form the final loss function:

$$L_{total} = L_1 + \lambda L_{SG} \tag{7}$$

where λ is a hyperparameter that controls the weight of the SG loss L_{SG} in the total loss L_{total}.

To facilitate a more intuitive comprehension of the proposed second-order gradient loss, we have visualized it for illustrative purposes. As shown in Fig. 2, the left half of the figure represents a generic image SR network architecture, which consists of three parts: shallow feature extraction module, deep feature extraction module, and high-quality image reconstruction module. Typically, the shallow feature extraction module is composed of two convolutional layers that are intended to extract low-level features from the image and capture local information. The deep feature extraction module is more intricate and lacks a standardized structure, as it aims to extract high-level features from the image to capture global information. The high-quality image reconstruction module usually comprises an upsampling module and a

convolutional layer. The upsampling module is responsible for increasing the size of the features, while the convolutional layer is responsible for transforming the features into an HR image. It is important to note that the specific architecture of the deep feature extraction module varies across different super-resolution networks, contributing to the variations in their restoration performance. The right half of the figure represents our proposed SG loss, it can be easily observed that it is a plug-and-play loss function, as the left half of the figure can be substituted with any image SR network. In summary, our proposed SG loss is generalizable. For ease of understanding, we list the meanings of the symbols used in this paper in Table 4.

4. Experiments

This section commences with an overview of the datasets, evaluation metrics, and implementation details employed in the experiment. Subsequently, a comparative analysis is presented, comparing the performance and visualization of several state-of-the-art SISR networks before and after the integration of the proposed SG loss. Furthermore, we investigate the influence of the λ value on the model performance.

4.1. DataSets and metrics

All the models are trained on the 800 images from the DIV2K [57] dataset. It is a widely recognized high-quality visual dataset in the field of SISR. For testing purposes, we utilize five standard public datasets: Set5 [58], Set14 [59], Urban100 [60], B100 [61], and Manga109 [62], which contain various scenes and can comprehensively analyze the effectiveness of the proposed loss. Since paired HR and LR images are required for training, the corresponding LR images are obtained by downsampling the HR images with a scaling factor of 4 using bicubic interpolation before conducting experiments. Considering that evaluation metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) often contradict human perceptual quality, this paper adopts perceptual metrics Learned Perceptual Image Patch Similarity (LPIPS) [63] and Frechet Inception Distance score (FID) [64], which are more consistent with human perception, as evaluation metrics to quantitatively compare the restoration results of the datasets. Lower LPIPS and FID values indicate better visual quality.

4.2. Implementation details

For the purpose of conducting a fair comparison, several representative deep learning-based SR networks were retrained to establish a consistent benchmark. Specifically, during the training process, data augmentation techniques are performed on the training dataset. This includes random cropping, rotation by 90°, 180°, and 270°, as well as flipping the original images, resulting in approximately 32,000 HR images of size 480 × 480. In each training iteration, we take in 16 LR image patches with the size of 48 × 48 as input. The ADAM optimizer is employed for training, with default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1\times10^{-8}$. The learning rate is initialized as 1×10^{-4} and undergoes a halving operation every 2×10^5 iterations of back-propagation. Through empirical analysis, the hyperparameter λ is determined to be 1. Further details regarding the selection and impact of different λ values will be discussed in Section 4.5. The entire process is carried out on the PyTorch 2.0 platform, leveraging a Nvidia GeForce RTX 3090 24 GB GPU for accelerated computations.

4.3. Quantitative comparison

We select several widely recognized SR network models, including EDSR [56], RDN [34], RCAN [36], and SwinIR [37], to assess the effectiveness of our proposed SG loss function. No modifications have been made to their network architectures. Rather than using the L1 loss function alone, we augment it by adding an SG loss term to provide extra supervision information. The computational overhead incurred by this operation is negligible. The ×4 SR results on five benchmark datasets are presented in Table 1. The results marked with "+SG" indicate the outcomes obtained by adding the SG loss for auxiliary optimization to the original SR methods. The data in Table 1 demonstrates that the incorporation of the SG loss function as an auxiliary network optimization leads to lower LPIPS and FID values on all datasets for all models, compared to the original models. This observation strongly supports the belief that the second-order gradient map of the image, which contains high-frequency information, plays a crucial role in aiding the network to restore images with better perceptual quality. In particular, on the more severely degraded B100 dataset, our method achieves a substantial reduction in LPIPS scores for the SwinIR and RCAN models, with descents of 0.0655 and 0.0696, respectively. Additionally, the FID scores of these two models also exhibit notable improvements compared to the original method.

Table 2
Comparison of model restoration results trained with some training strategy and hyperparameters..

DataSet	Metric	1	2	3	4	5
Set5	LPIPS ↓	0.1446	0.1443	0.1447	0.1443	0.1441
	FID ↓	56.52	57.07	56.67	56.65	56.96
Set14	LPIPS ↓	0.2353	0.2361	0.2351	0.2355	0.2360
	FID ↓	80.94	81.54	80.89	80.05	81.36
Urban100	LPIPS ↓	0.1837	0.1837	0.1838	0.1836	0.1838
	FID ↓	23.10	22.92	23.08	22.95	22.87
B100	LPIPS ↓	0.3018	0.3018	0.3020	0.3023	0.3018
	FID ↓	88.47	88.64	88.03	87.63	87.32
Manga109	LPIPS ↓	0.0856	0.0856	0.0855	0.0856	0.0857
	FID ↓	10.77	10.67	10.65	10.70	10.80

4.4. Qualitative comparison

In order to provide further evidence of the effectiveness of our proposed SG loss, this section showcases visual results of the restored images obtained from the Set14, B100, and Urban100 datasets, with the majority of images selected from the Urban100 dataset. The Urban100 dataset was chosen for its collection of 100 images depicting buildings in urban areas. These images are rich in intricate texture details, making it an ideal dataset to demonstrate the effectiveness of the SG loss in restoring fine details. As illustrated in Fig. 3, the methods trained only using the L1 loss are capable of restoring the main contours of objects. However, they struggle to accurately restore complex image boundaries, often resulting in distorted and deformed textures. In contrast, after integrating the SG loss as supplementary supervision, the network preserves the fine details within images to a greater extent, and the reconstructed textures appear more natural and realistic.

4.5. Robustness experiment

To substantiate the robustness of the proposed SG loss function, we retrain the EDSR model multiple times using the same training strategy and hyperparameters, and the results of each training are exhibited in Table 2. We can find that although there are discrepancies in the recovery results of the models trained each time, these discrepancies are extremely minimal. This provides substantial evidence that the enhancement in model recovery performance attributed to the SG loss function is not incidental.

4.6. Ablation study of λ

To investigate the influence of different λ values on the performance of image restoration models, we train four distinct models with λ values of 0.01, 0.1, 1, and 10, respectively, employing the same training

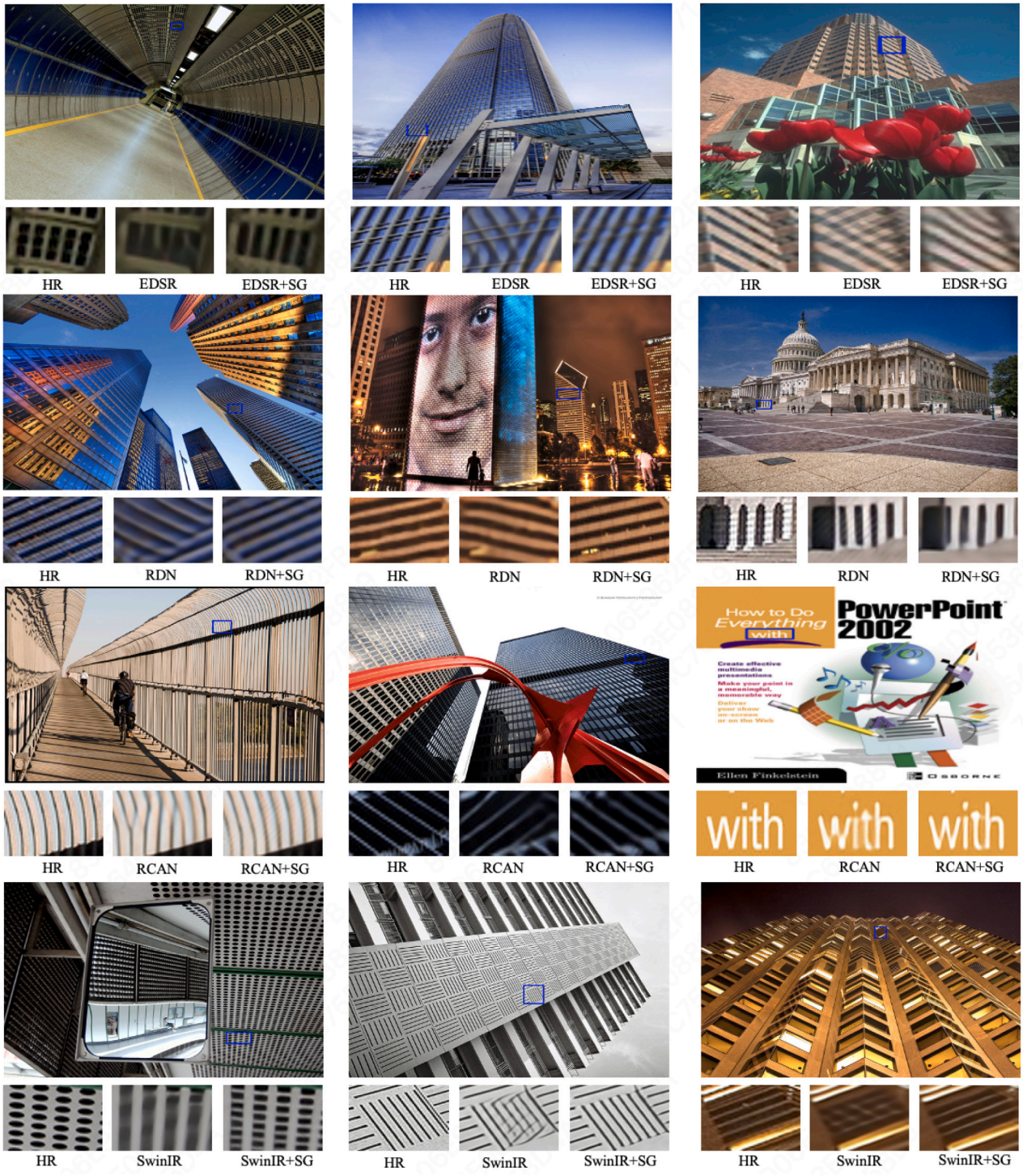


Fig. 3. Visual comparison of restoration results of different models before and after adding second gradient(SG) loss, where the first column of each image represents the GT high-resolution (HR) image, the second column represents the results recovered by EDSR [56], RDN [34], RCAN [36] and SwinIR [37], the third column represents the recovery results after integrating the SG loss by above methods.

Table 3
Comparison of model restoration results trained with different λ values.

Metrics	$\lambda = 0$	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1$	$\lambda = 10$
LPIPS ↓	0.2037	0.2013	0.1948	**0.1837**	0.1946
FID ↓	25.56	25.07	24.31	**23.10**	36.31

strategy. Subsequently, we conduct a comprehensive evaluation of the 4× SR performance of these four models on the Urban100 dataset. The results are presented in Table 3, with the highlighted numbers indicating the lowest LPIPS and FID scores in each row. To guarantee a fair comparison, all four models adopt the EDSR network structure. Fig. 4 is provided to offer a more intuitive observation of the differences in visual. The reference model, which does not incorporate the SG loss (i.e. $\lambda = 0$), exhibits the highest LPIPS and FID scores compared to the other models. As we increase the value of λ from 0 to 1, the

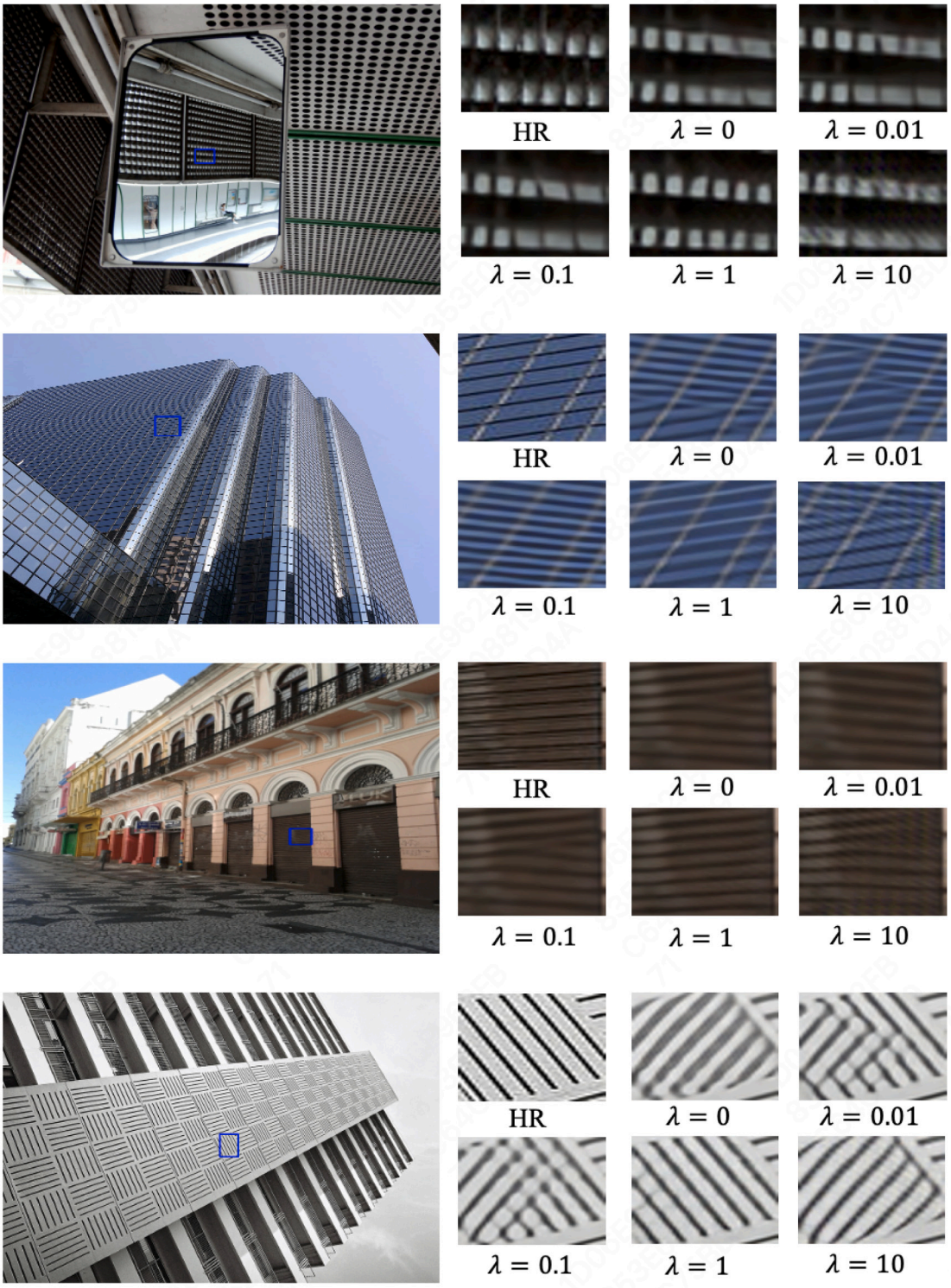


Fig. 4. Visual comparison of restoration results of models which trained with different λ values.

importance of the SG loss supervision gradually grows, resulting in a positive impact on the quality of the SR results. The best performance is attained when $\lambda = 1$. In this case, the model reaches a better balance between emphasizing image boundaries and preserving smooth regions. Nonetheless, if the λ value is excessively large, the model may overemphasize the textures and edges within images, and may even introduce unnatural artifacts in the smooth areas, resulting in a decline in model performance. Summarizing the above analysis, we recommend setting λ to 1 when using the SG loss.

4.7. Ablation study of different loss functions

We compare several loss functions commonly used in this field with our proposed SG loss function to further demonstrate its effectiveness, EDSR is still selected as the baseline for a fair comparison. Here, l2 loss [65] is a commonly used loss in the early days, charbonnier loss [66] is a variant of the l1 loss, which can better handle outliers and enhance model robustness, and ssim loss [67] can better simulate the perception of images by the human eyes. As shown in Table 5, when

Table 4
The meanings of symbols used in this paper..

Symbols	Meanings
I^{LR}	Low-Resolution Image.
I^{HR}	High-Resolution Image.
I^{SR}	Super-Resolution Image generated by our method.
$\downarrow_s$	Bicubic interpolation downsampling operation with a scaling factor of s.
I^{HR_G}	The second-order gradient map of the high-resolution image.
L^{SR_G}	The second-order gradient map of the super-resolution image generated by our method.
$dx(i,j)$	Horizontal gradient at the point (i, j).
$dy(i,j)$	Vertical gradient at the point (i, j).
$\nabla I(x,y)$	Horizontal and vertical gradient of image I.
$M(I)$	The first-order gradient map of image I.
$M(M(I))$	The second-order gradient map of image I.
L_1	L1 loss.
L_{SG}	Second-Order Gradient Loss.
L_{total}	The total loss used in this paper.
λ	The hyperparameter to balance the L1 loss and SG loss.

Table 5
Comparison of model restoration results trained with different loss functions.

Loss	LPIPS↓	FID↓
L1 loss	0.2037	25.56
L2 Loss	0.2064	25.04
SSIM loss	0.2057	24.98
Charbonnier Loss	0.2026	25.37
L1 loss + SG loss	**0.1837**	**23.10**

combined with the l1 loss function, our proposed SG loss can help the model achieve the best performance in all the quality metrics.

5. Conclusion

In this paper, an innovative high-frequency texture detail enhancement loss, referred to as the second-order gradient loss, is proposed to alleviate the problem of blurry high-resolution images generated by most existing SISR methods trained only with L1 loss. More specifically, the proposed second-order gradient loss function offers supplementary supervision for network optimization so that the solution space is compressed. This is accomplished by minimizing the discrepancy between the second-order gradient maps of the restored image and the high-resolution image. Furthermore, it can be seamlessly integrated with existing deep learning-based SISR methods without the need for introducing extra training parameters. This makes it a practical and convenient solution for enhancing the performance of SISR models without significant modifications to their existing architectures. The evaluation conducted on five public benchmark datasets indicate that the integration of this loss function significantly enhances the quality and fidelity of the restored images.

CRediT authorship contribution statement

Shuran Lin: Methodology, Software, Validation, Writing – original draft. **Chunjie Zhang:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Resources, Supervision, Validation, Writing – original draft, Writing – review & editing. **Yanwu Yang:** Data curation, Methodology, Resources, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by National Natural Science Foundation of China: 62072026, 72171093; Beijing Natural Science Foundation: JQ20022.

References

[1] L. Zhang, H. Zhang, H. Shen, P. Li, A super-resolution reconstruction algorithm for surveillance images, Signal Process. 90 (3) (2010) 848–859.
[2] Y. Pang, J. Cao, J. Wang, J. Han, JCS-net: Joint classification and super-resolution network for small-scale pedestrian detection in surveillance images, IEEE Trans. Inf. Forensics Secur. 14 (12) (2019) 3322–3331, http://dx.doi.org/10.1109/TIFS.2019.2916592.
[3] P. Shamsolmoali, M. Zareapoor, D.K. Jain, V.K. Jain, J. Yang, Deep convolution network for surveillance records super-resolution, Multimedia Tools Appl. 78 (2019) 23815–23829.
[4] Y. Huang, L. Shao, A.F. Frangi, Simultaneous super-resolution and cross-modality synthesis of 3D medical images using weakly-supervised joint convolutional sparse coding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6070–6079.
[5] Y. Li, B. Sixou, F. Peyrin, A review of the deep learning methods for medical images super resolution problems, Irbm 42 (2) (2021) 120–133.
[6] D. Mahapatra, B. Bozorgtabar, R. Garnavi, Image super-resolution using progressive generative adversarial networks for medical image analysis, Comput. Med. Imaging Graph. 71 (2019) 30–39.
[7] Y. Bai, Y. Zhang, M. Ding, B. Ghanem, Sod-mtgan: Small object detection via multi-task generative adversarial network, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 206–221.
[8] J. Shermeyer, A. Van Etten, The effects of super-resolution on object detection performance in satellite imagery, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019.
[9] L. Wang, D. Li, Y. Zhu, L. Tian, Y. Shan, Dual super-resolution learning for semantic segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3774–3783.
[10] A. Aakerberg, A.S. Johansen, K. Nasrollahi, T.B. Moeslund, Semantic segmentation guided real-world super-resolution, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2022, pp. 449–458.
[11] S. Mandal, A.K. Sao, Edge preserving single image super resolution in sparse environment, in: 2013 IEEE International Conference on Image Processing, IEEE, 2013, pp. 967–971.
[12] M.K. Ng, H. Shen, S. Chaudhuri, A.C. Yau, Zoom-based super-resolution reconstruction approach using prior total variation, Opt. Eng. 46 (12) (2007) 127003.
[13] M.K. Ng, H. Shen, E.Y. Lam, L. Zhang, A total variation regularization based super-resolution reconstruction algorithm for digital video, EURASIP J. Adv. Signal Process. 2007 (2007) 1–16.
[14] J. Yang, J. Wright, T.S. Huang, Y. Ma, Image super-resolution via sparse representation, IEEE Trans. Image Process. 19 (11) (2010) 2861–2873.
[15] J. Yang, J. Wright, T. Huang, Y. Ma, Image super-resolution as sparse representation of raw image patches, in: 2008 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2008, pp. 1–8.
[16] N. Akhtar, F. Shafait, A. Mian, Bayesian sparse representation for hyperspectral image super resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3631–3640.
[17] W. Yang, J. Feng, J. Yang, F. Zhao, J. Liu, Z. Guo, S. Yan, Deep edge guided recurrent residual learning for image super-resolution, IEEE Trans. Image Process. 26 (12) (2017) 5895–5907.

[18] C. Ma, Y. Rao, Y. Cheng, C. Chen, J. Lu, J. Zhou, Structure-preserving super resolution with gradient guidance, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7769–7778.
[19] R. Keys, Cubic convolution interpolation for digital image processing, IEEE Trans. Acoust. Speech Signal Process. 29 (6) (1981) 1153–1160.
[20] D.K. Shin, Y.S. Moon, Super-resolution image reconstruction using wavelet based patch and discrete wavelet transform, J. Signal Process. Syst. 81 (2015) 71–81.
[21] A. Danielyan, A. Foi, V. Katkovnik, K. Egiazarian, P. Milanfar, Spatially adaptive filtering as regularization in inverse imaging: Compressive sensing super-resolution and upsampling, Super-Resolut. Imag. (2010) 123–154.
[22] K.I. Kim, Y. Kwon, Single-image super-resolution using sparse regression and natural image prior, IEEE Trans. Pattern Anal. Mach. Intell. 32 (6) (2010) 1127–1133.
[23] C. Dong, C.C. Loy, K. He, X. Tang, Image super-resolution using deep convolutional networks, IEEE Trans. Pattern Anal. Mach. Intell. 38 (2) (2015) 295–307.
[24] J. Li, F. Fang, J. Li, K. Mei, G. Zhang, MDCN: Multi-scale dense cross network for image super-resolution, IEEE Trans. Circuits Syst. Video Technol. 31 (7) (2021) 2547–2561, http://dx.doi.org/10.1109/TCSVT.2020.3027732.
[25] N. Ahn, B. Kang, K.-A. Sohn, Fast, accurate, and lightweight super-resolution with cascading residual network, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 252–268.
[26] M.S. Sajjadi, B. Scholkopf, M. Hirsch, Enhancenet: Single image super-resolution through automated texture synthesis, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4491–4500.
[27] J. Kim, J.K. Lee, K.M. Lee, Accurate image super-resolution using very deep convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1646–1654.
[28] Z. Zhang, Z. Wang, Z. Lin, H. Qi, Image super-resolution by neural texture transfer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 7982–7991.
[29] J. Liu, W. Zhang, Y. Tang, J. Tang, G. Wu, Residual feature aggregation network for image super-resolution, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 2359–2368.
[30] C. Dong, C.C. Loy, X. Tang, Accelerating the super-resolution convolutional neural network, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, October 11-14, 2016, Proceedings, Part II 14, Springer, 2016, pp. 391–407.
[31] B. Niu, W. Wen, W. Ren, X. Zhang, L. Yang, S. Wang, K. Zhang, X. Cao, H. Shen, Single image super-resolution via a holistic attention network, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16, Springer, 2020, pp. 191–207.
[32] W. Shi, J. Caballero, F. Huszár, J. Totz, A.P. Aitken, R. Bishop, D. Rueckert, Z. Wang, Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1874–1883.
[33] Y. Yan, W. Ren, X. Hu, K. Li, H. Shen, X. Cao, SRGAT: Single image super-resolution with graph attention network, IEEE Trans. Image Process. 30 (2021) 4905–4918, http://dx.doi.org/10.1109/TIP.2021.3077135.
[34] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, Y. Fu, Residual dense network for image super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2472–2481.
[35] J. Li, F. Fang, K. Mei, G. Zhang, Multi-scale residual network for image super-resolution, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 517–532.
[36] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, Y. Fu, Image super-resolution using very deep residual channel attention networks, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 286–301.
[37] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, R. Timofte, Swinir: Image restoration using swin transformer, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 1833–1844.
[38] W.-S. Lai, J.-B. Huang, N. Ahuja, M.-H. Yang, Deep laplacian pyramid networks for fast and accurate super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 624–632.
[39] K.-W. Hung, K. Wang, J. Jiang, Image interpolation using convolutional neural networks with deep recursive residual learning, Multimedia Tools Appl. 78 (2019) 22813–22831.
[40] T. Tong, G. Li, X. Liu, Q. Gao, Image super-resolution using dense skip connections, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4799–4807.
[41] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, L. Zhang, Second-order attention network for single image super-resolution, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11065–11074.
[42] Y. Mei, Y. Fan, Y. Zhou, L. Huang, T.S. Huang, H. Shi, Image super-resolution with cross-scale non-local attention and exhaustive self-exemplars mining, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 5690–5699.
[43] Y. Zhang, D. Wei, C. Qin, H. Wang, H. Pfister, Y. Fu, Context reasoning attention network for image super-resolution, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 4278–4287.
[44] H. Chen, Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu, W. Gao, Pre-trained image processing transformer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 12299–12310.
[45] Y. Mei, Y. Fan, Y. Zhou, Image super-resolution with non-local sparse attention, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3517–3526.
[46] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., Photo-realistic single image super-resolution using a generative adversarial network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4681–4690.
[47] K. Zhang, W. Zuo, S. Gu, L. Zhang, Learning deep CNN denoiser prior for image restoration, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3929–3938.
[48] J. Yu, Y. Fan, J. Yang, N. Xu, Z. Wang, X. Wang, T. Huang, Wide activation for efficient and accurate image super-resolution, 2018, arXiv preprint arXiv:1808.08718.
[49] M. Haris, G. Shakhnarovich, N. Ukita, Deep back-projection networks for super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1664–1673.
[50] Z. Hui, X. Wang, X. Gao, Fast and accurate single image super-resolution via information distillation network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 723–731.
[51] Y. Zhu, Y. Zhang, B. Bonev, A.L. Yuille, Modeling deformable gradient compositions for single-image super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5417–5425.
[52] R. Fattal, Image upsampling via imposed edge statistics, in: ACM SIGGRAPH 2007 Papers, 2007, pp. 95–es.
[53] Q. Yan, Y. Xu, X. Yang, T.Q. Nguyen, Single image superresolution based on gradient profile sharpness, IEEE Trans. Image Process. 24 (10) (2015) 3187–3202.
[54] W. Dong, L. Zhang, G. Shi, X. Wu, Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization, IEEE Trans. Image Process. 20 (7) (2011) 1838–1857.
[55] F. Fang, J. Li, T. Zeng, Soft-edge assisted network for single image super-resolution, IEEE Trans. Image Process. 29 (2020) 4656–4668.
[56] B. Lim, S. Son, H. Kim, S. Nah, K. Mu Lee, Enhanced deep residual networks for single image super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 136–144.
[57] E. Agustsson, R. Timofte, Ntire 2017 challenge on single image super-resolution: Dataset and study, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 126–135.
[58] M. Bevilacqua, A. Roumy, C. Guillemot, M.L. Alberi-Morel, Low-Complexity Single-Image Super-Resolution Based on Nonnegative Neighbor Embedding, BMVA Press, 2012.
[59] R. Zeyde, M. Elad, M. Protter, On single image scale-up using sparse-representations, in: Curves and Surfaces: 7th International Conference, Avignon, France, June 24-30, 2010, Revised Selected Papers 7, Springer, 2012, pp. 711–730.
[60] D.R. Martin, C.C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: Proceedings of the Eighth International Conference on Computer Vision, Vol. 2, ICCV-01, Vancouver, British Columbia, Canada, July 7-14, 2001, IEEE Computer Society, 2001, pp. 416–425, http://dx.doi.org/10.1109/ICCV.2001.937655.
[61] J.-B. Huang, A. Singh, N. Ahuja, Single image super-resolution from transformed self-exemplars, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5197–5206.
[62] Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, K. Aizawa, Sketch-based manga retrieval using manga109 dataset, Multimedia Tools Appl. 76 (2017) 21811–21838.
[63] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 586–595.
[64] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, Gans trained by a two time-scale update rule converge to a local nash equilibrium, in: Advances in Neural Information Processing Systems, vol. 30, 2017.
[65] Z. Zhang, Parameter estimation techniques: A tutorial with application to conic fitting, Image Vision Comput. 15 (1) (1997) 59–76.
[66] P. Charbonnier, L. Blanc-Feraud, G. Aubert, M. Barlaud, Two deterministic half-quadratic regularization algorithms for computed imaging, in: Proceedings of 1st International Conference on Image Processing, Vol. 2, IEEE, 1994, pp. 168–172.
[67] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: From error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612, http://dx.doi.org/10.1109/TIP.2003.819861.

BenchCouncil Transactions on Benchmarks, Standards and Evaluations 3 (2023) 100150

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/

Full length article

CloudAISim: A toolkit for modelling and simulation of modern applications in AI-driven cloud computing environments

Abhimanyu Bhowmik [a], Madhushree Sannigrahi [a], Deepraj Chowdhury [b,c], Ajoy Dey [d], Sukhpal Singh Gill [e,*]

[a] SeaTech School of Engineering, University of Toulon, Toulon, France
[b] Center for Application Research in India (CARIn), Carl Zeiss (Bangalore) India Pvt Ltd., Bangalore, India
[c] Dept of Electronics and Communication Engineering, IIIT Naya Raipur, Naya Raipur, India
[d] TCS Research and Innovation, Kolkata, India
[e] School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK

ARTICLE INFO

Keywords:
Artificial intelligence
Explainable AI
Cloud computing
Machine learning
Healthcare
CloudAISim
Simulation

ABSTRACT

There is a very significant knowledge gap between Artificial Intelligence (AI) and a multitude of industries that exist in today's modern world. This is primarily attributable to the limited availability of resources and technical expertise. However, a major obstacle is that AI needs to be flexible enough to work in many different applications, utilising a wide variety of datasets through cloud computing. As a result, we developed a benchmark toolkit called CloudAISim to make use of the power of AI and cloud computing in order to satisfy the requirements of modern applications. The goal of this study is to come up with a strategy for building a bridge so that AI can be utilised in order to assist those who are not very knowledgeable about technological advancements. In addition, we modelled a healthcare application as a case study in order to verify the scientific reliability of the CloudAISim toolkit and simulated it in a cloud computing environment using Google Cloud Functions to increase its real-time efficiency. A non-expert-friendly interface built with an interactive web app has also been developed. Any user without any technical knowledge can operate the entire model, which has a 98% accuracy rate. The proposed use case is designed to put AI to work in the healthcare industry, but CloudAISim would be useful and adaptable for other applications in the future.

1. Introduction

A set of practices aiming to base decisions on the analysis of data instead of intuitive insights can be used to define data-driven decision-making. When compared to conventional ones, businesses that implement data-driven decision-making processes are more financially beneficial and productive [1]. The outcomes of recent Artificial Intelligence (AI) research projects serve as the foundation for many decision-making tools [2]. The development of Machine Learning (ML) techniques is largely responsible for the success of AI-based tools [3]. The availability of sizable datasets on various real-world features as well as the rise in computational gains, which are typically attributable to the powerful Graphics Processing Unit (GPU) cards [4], are particularly encouraging in this regard.

The need to create sophisticated AI models with previously unheard-of performance levels has progressively given way to a rising interest in alternative design elements that would improve the usability of emerging products [5]. Complex AI models lose a part of their practical effectiveness in a wide range of application domains [6]. The main cause is that AI models are frequently created with a performance-focused approach, neglecting other significant – and occasionally crucial – aspects like accountability, transparency, and justice [7]. The AI models are typically "black boxes" since there is no explanation provided for the elements that are projected to perform well; as a result, they simply allow for the prominent display of input and output parameters while hiding the visibility of the intrinsic relationships between those parameters [8]. It is advantageous to have some explanations of individual predictions that are recognised using an AI system, more specifically in an automated environment, because these applications may include crucial decision-making [9].

This research aims to develop a transparent and self-explanatory system using AI, especially Automated Machine Learning (AutoML)

* Corresponding author.
E-mail addresses: abhimanyu-bhowmik@etud.univ-tln.fr (A. Bhowmik), madhushree-sannigrahi@etud.univ-tln.fr (M. Sannigrahi), deepraj.chowdhury@zeiss.com (D. Chowdhury), deyajoy80@gmail.com (A. Dey), s.s.gill@qmul.ac.uk (S.S. Gill).

https://doi.org/10.1016/j.tbench.2024.100150
Received 10 August 2023; Received in revised form 3 January 2024; Accepted 6 January 2024
Available online 9 January 2024

systems [10], that uses cloud computing, particularly serverless computing to propose the best machine-learning configuration for a particular issue and trace the reasoning behind a recommendation. It could also make it possible to interpretably and credibly examine the predicted outcomes.

1.1. Motivation

Even if AI/ML tools are open sources and widely available, creating a data processing pipeline and generating fine-tuned models for specific domains requires knowledge and skills in data science and AI, which are not always present in public sector industries like hospitals and nursing homes [11]. Therefore, we designed a toolkit called CloudAISim by utilising AI and cloud computing for the modelling and simulation of modern applications to bridge the gaps between non-professionals and AI expertise, starting with the healthcare application as a case study that would be useful and adaptable for other applications in the future.

The healthcare industry produces enormous amounts of data, using various sensors and health-monitoring devices to collect data [12]. The availability of data about the health of millions of patients may make it possible to create AI-based processes and models that give healthcare professionals useful insights [13]. This research also looks beyond the healthcare domain to design a benchmark model that can be useful for other applications from different domains. The framework and insights presented in this article can serve as a blueprint for extending AutoML and explainability to various other domains, from finance and retail to agriculture and manufacturing. The scalability and flexibility offered by cloud-based solutions make it feasible to democratise these AI technologies and accelerate innovation across industries [14]. This research mainly aims to provide the power of AI to non-experts without writing a single line of code. The proposed CloudAISim toolkit will do all the technical steps like Explanatory Data Analysis (EDA), feature engineering, choosing the best algorithm, and explaining the results predicted by the framework for better understanding by the user. In this paper, we have considered the applications/dataset of the healthcare domain, but it can be used for any domain where the dataset is in CSV format.

1.2. Contributions

The main contributions of this work are:

1. Proposing a toolkit called CloudAISim for efficient explainable machine learning technique modelling and implementation in the healthcare domain.
2. Finding the most accurate and responsive machine learning model for chronic as well as infectious diseases like diabetes, heart disease, breast cancer and COVID-19 in the healthcare domain.
3. Simulating a prototype web application for the validation of CloudAISim to provide a visual display for data, models and the explainability of results.
4. Implementing the CloudAISim in a cloud computing environment using Google Cloud Functions to increase real-time efficiency.
5. Highlighting the promising future directions.

The rest of the paper is structured as follows: The relevant related works regarding ML-based data analytics solutions and the requirement for transparency to build confidence in AI models are covered in Section 2. The proposed framework is described in Section 3 along with how its various elements work together to accomplish the desired outcomes. The results obtained using a few test cases are discussed in Section 4. Section 5 demonstrates the proposed application and Section 6 discusses the important findings of this research. Finally, Section 7 concludes the paper and offers future directions.

2. Related works

Various AutoML systems have been developed in recent years that offer partial or full ML automation, including systems like Auto-sklearn [15], tree-based pipeline optimisation tool (TPOT) [16], Auto-WEKA [17], and ATM [18] and commercialised software like Google AutoML,[1] RapidMiner,[2] and DarwinAI.[3] These methods include automatic feature engineering [19,20], automatic model selection [21,22], automatic hyperparameter tuning [23], and automatic data preparation [24,25]. Some methods make an effort to automatically select an ML algorithm while also optimising its hyper-parameters.

Many of the AutoML solutions, some of which are the results of contests from 2015 to 2018, were developed in the last few years. The ChaLearn AutoML Challenges[4] primarily focused on automating the solution of supervised machine learning tasks under certain computing restrictions. These computational restrictions varied significantly across tasks, but they were typically time (about 20 min for training and assessment) and memory consumption restrictions. Guyon et al. [26] review a thorough examination of the AutoML difficulties from 2015 to 2018. In essence, neural architecture may be considered a specific kind of indifferentiable hyperparameter. Hyperparameter optimisation is one of these activities that most directly relates to the approach we suggest in this study. Grid search and random search [27] are two of the simplest techniques to find a suitable configuration of hyperparameters from a list of options without considering past results. There are various sets of approaches that are often used in hyperparameter optimisation. Being one of the most well-known Sequential Model-based Optimisation (SMBO) [28] techniques that can take advantage of historical data, Bayesian optimisation [29] uses the Gaussian method for prototyping the surrogate function that roughly imitates the relationships between hyperparameters and their desired outputs. All of these techniques are, however, black-box optimised. The single study on AutoML for graph representation [30] employs the Gaussian Process to determine the performance of the hyperparameters, but it scarcely explains how individual hyperparameter affects the performance of the model or why a specific value is picked for a hyperparameter to execute the subsequent assessment trial.

AI systems that can give human-understandable explanations for their activities and output are referred to as Explainable AI (XAI) [6]. By their very nature, end users may be curious as to how and why systems reach any conclusion [31]. They are seen as "black boxes" when the sophistication of AI algorithms and systems increases [32]. Growing complexity may lead to a lack of openness that makes it difficult to comprehend these systems' logic, which has a detrimental impact on users' faith in them.

2.1. Critical analysis

Table 1 compares the proposed CloudAISim with existing frameworks based on important parameters. The model accuracy of the aforementioned studies is pretty high; however, the generalisation of the studies is limited. Only 3 of the studies have formalised feature extraction procedures that can be generalised. Two of the research have implemented explainability, which can be critical for many industries such as healthcare. None of the mentioned studies have implemented their framework in a serverless cloud environment. A novel CloudAISim framework that has high customisation and consists of a web interface that can be very easily used by non-technical users. This will enable the non-expert to unleash the power of AI and can be immensely helpful for any industry such as the healthcare industry. Moreover, the CloudAISim

[1] https://cloud.google.com/automl
[2] https://rapidminer.com
[3] https://darwinai.com/
[4] http://automl.chalearn.org/

Table 1
Comparison of proposed CloudAISim framework with existing solutions.

Related works	Dataset	Feature extraction	AutoML	Cloud computing (Serverless)	XAI
Ferreira et al. [33]	Pneumonia dataset	✗	WebDL	✗	✗
Shawi et al. [34]	Breast Cancer Wisconsin	✓	NNI	✗	✗
Alaa et al. [35]	Risk factors Cystic Fibrosis	✓	AutoPrognosis	✗	✓
Alnegheimish et al. [36]	MIMIC-III	✓	MLBlocks	✗	✗
Garouani et al. [13]	Big Industrial Data	✗	AMLBID	✗	✓
CloudAISim (Proposed work)	**Breast Cancer Wisconsin, Covid, Heart Disease Cleveland Dataset**	✓	AutoKeras	✓	✓

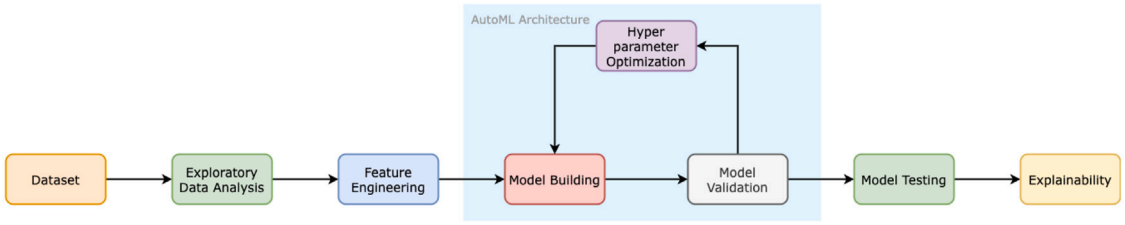


Fig. 1. The Abstract View of Proposed Framework.

framework enhances the scalability of the environment and can be incorporated into a large-scale scenario. To the best of our knowledge, the proposed solution is the first to use generic explanation techniques of Auto ML systems as decision support systems in a serverless setting.

3. CloudAISim framework

In this section, the description of the proposed approach used to achieve the objectives of the work has been discussed. Fig. 1 shows the abstract view or high-level view of the proposed CloudAISim framework. As shown in Fig. 1, the framework accepts the dataset from the user's device, like a smart phone or laptop. Then it is passed to the Automated EDA tool for explanatory data analysis, and then feature engineering is done on the dataset by the feature tool, the best machine learning model is generated, and hyperparameter tuning is done. After that, the explanation of the prediction is shown using lime. The entire execution is conducted on a serverless platform.

3.1. Architecture

Fig. 2 shows the main architecture of CloudAISim, which includes AutoML models, and the usage of Explainable AI (XAI) to demonstrate the working of the ML models, as well as the serverless architecture of the prototype. The main components of the proposed architecture are discussed further in Sections 3.2, 3.3, 3.4, 3.5, 3.6, and 3.7.

3.2. Dataset

Firstly, the "Breast Cancer Wisconsin (Diagnostic) Data Set" by "UCI ML Repository" is implemented on the novel methodology for the paper [37]. The dataset contains tabular data with 32 features and over 569 data points. A fine needle aspirate (FNA) of a breast lump is used to generate the features from a digital image in 3-dimensional space as described by Bannett et al. [38]. They characterise the properties of all the observable cell nuclei in the image. Every data point is classified into either Benign (B) or Malignant (M) class. Secondly, the architecture is applied to the "Heart Disease Cleveland dataset" Dataset by "UCI ML Repository" [39]. The dataset constitutes over 300 patients' data with 75 attributes. However, only 14 of the features are taken into consideration for determining whether a patient has heart disease or not. Thirdly, the "Diabetes dataset", originally from the National Institute of Diabetes and Digestive and Kidney Diseases, is used in this work [40]. The goal is to determine if a patient has diabetes based on diagnostic parameters. The implemented Diabetes dataset is a subset of an enormous dataset with 10 attributes and 768 instances. All patients are Pima Indian females who are at least 21 years old. Finally the "Covid-19" is a dataset, used in the paper [41] which contains data from 800 people and 26 attributes such as their profession, health parameters and lifestyle parameter, and the risk factor of getting infection by covid is mentioned. The higher the risk factor the higher chance of getting infected by Covid. So we classified the person with a risk factor of more than 0.5 as high (1) and less than 0.5 as low (0).

3.3. Serverless cloud

Serverless computing is a method for providing backend services on an "as-needed" basis. The cloud provider controls the servers on behalf of their customers while expanding and maintaining the system as necessary [42]. This is the cloud computing execution paradigm. Since any device, regardless of its specifications, may access an application, it becomes a resource independently [43]. This allows for greater scalability and flexibility as the application can automatically scale up or down based on demand [44]. Additionally, serverless computing eliminates the need for developers to worry about server management and infrastructure maintenance, allowing them to focus solely on writing and deploying code.

The experimental architecture was developed on Google Cloud Platform (GCP), a serverless solution enabling efficient data storage and analysis. The proposed experiment was conducted using cloud functions in the Python 3.9 runtime. Google Cloud Storage is also used for storing the data and its respective output. GCP also allows seamless interactions with other Google Cloud services, like Cloud Storage events, which are used as triggers for the respective cloud functions. This architecture also offers built-in security features and automatic scaling capabilities, ensuring optimal performance and cost efficiency for the application.

In the proposed architecture, three subsequent cloud functions were used as shown in Fig. 2:

1. To perform automated feature selection and feature engineering using the open-source Feature tools;
2. To generate an AutoML model and predict the results using the Auto-Keras Python library; and
3. To explain the predicted results.

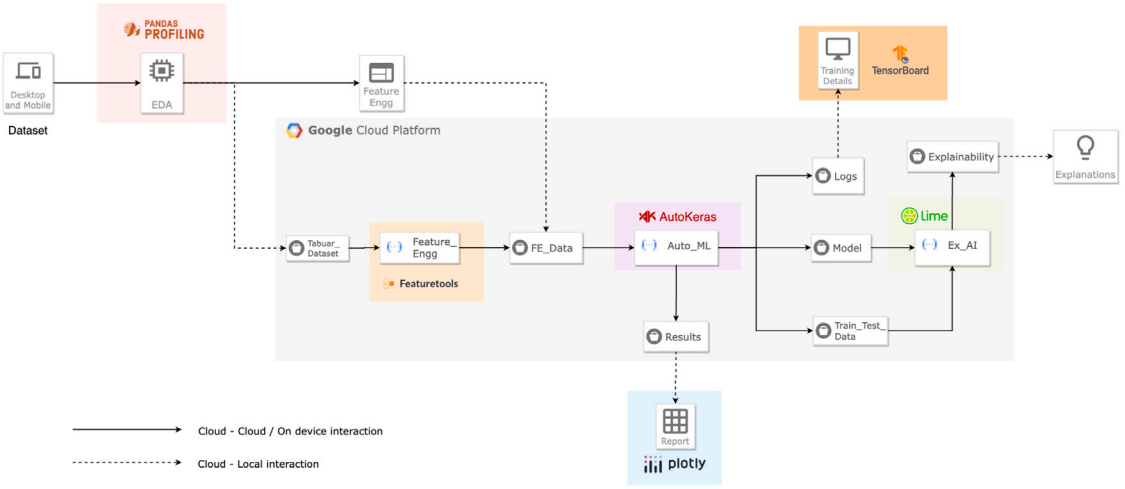


Fig. 2. CloudAISim Architecture.

The model explanation is completed using the Lime library. In place of standard Lime explanations, SP-Lime (Submodular Pick Locally Interpretable Model-Agnostic Explanations) is used to explain the model's global decision boundary over a sample set of observations.

3.4. Data preprocessing

Case-specific and automated data pre-processing was a considerable challenge while implementing the system in a cloud platform. This section discusses the implementation of Pandas Profiling[5] and Feature Tools[6] for automated and rapid EDA and Feature Engineering respectively.

3.4.1. Automated EDA using Pandas profiling

Exploratory data analysis (EDA) is a vital stage in constructing any impressive model. EDA involves finding outliers, spotting missing values, figuring out how skewed the datasets are, converting categorical variables, and overall understanding the underlying characteristics and ways to apply them in models.

Pandas Profiling[6] is a user-friendly open-source Python tool for automated exploratory data analysis. It generates a data frame report in a range of different formats. Although the Pandas' df.describe the operation can demonstrate basic information, it does not give a full data frame report. Pandas profiling was implemented in the system architecture for automated and rapid analysis of data.

3.4.2. Automated feature engineering using feature tools

Feature engineering is the process of creating and adding new features, or variables, to the dataset to enhance the effectiveness and precision of the machine learning model. Case-based knowledge and accessible data sources serve as the foundation for the most efficient feature engineering. Without requiring any human input, automated feature extraction employs deep networks or specialised algorithms to automatically extract characteristics from images or signals.

Featuretools[7] is a free and open-source Python architecture for automated feature engineering. It generates features automatically from relational and temporal information. Deep Feature Synthesis (DFS) is utilised for implementing automated feature engineering. Featuretools gives users the ability to perform feature selection by (1) removing null values; (2) removing single-value features; and (3) removing highly correlated attributes. For machine learning and predictive modelling, one can construct useful features by combining the raw data with information about the data.

3.5. AutoML

The time-consuming and iterative activities required in developing a machine learning model may be automated using Automated Machine Learning or AutoML. It provides a diverse range of approaches to help those with little background in machine learning access this technology. It attempts to minimise the requirement for experienced individuals to create the ML model. Additionally, it helps to increase productivity and promote machine learning research [13].

To properly comprehend automated machine learning, we must first understand the life cycle of a data science or machine learning project. A data science project's lifetime typically includes many stages, including data cleaning, feature engineering, model selection, parameter optimisation, and model validation. Even though technology has improved so much, all of these procedures still involve manual labour, which takes time and calls for several data scientists with the necessary skills. For non-ML professionals, it is quite challenging to do these jobs because of their intricacies. The demand for automating these activities has increased because of the rapid development of ML apps, which will make it easier for those without technical expertise to utilise them. Consequently, automated machine learning was developed in order to fully automate the process, from data cleansing to parameter optimisation. Not only does it save time, but it also performs fantastically.

In this paper, AutoML is considered the first mandatory cloud function for the framework. It is launched when the clean data is uploaded to the designated bucket. This will create five different models from the training set (which makes up 70% of the total data), train them for 100 iterations, and select the model with the highest accuracy for the given dataset. The performance matrices (Confusion Matrix, Classification Report) will be generated after the chosen model has been evaluated on test data (30% of the supplied data). The model itself, all training and testing data, performance matrices, and more will be exported into separate cloud buckets as in Fig. 1. The TensorFlow-Keras standard format for exporting ML models, *.h5*, is used to export the architecture. There are multiple models to perform AutoML operations. Some of them are given in Table 2.

3.5.1. AutoKeras

A Keras-based AutoML system is called AutoKerac.[7] It was created by Texas A & M University's DATA Lab. Making machine learning accessible to everyone is the aim of AutoKeras. It searches using Neural Architecture Search (NAS) algorithms to eventually eliminate the

[5] https://pandas-profiling.ydata.ai/docs/master/index.html
[6] https://www.featuretools.com
[7] https://autokeras.com/

requirement for deep learning engineers. AutoKeras takes advantage of Keras to conduct a potent neural network search for model parameters. It offers modular building pieces to carry out architectural searches as well as high-level end-to-end APIs like ImageClassifier or TextClassifier to address machine learning challenges in a few lines.

3.5.2. Auto-sklearn

The detailed description of the Auto-sklearn approach by Feurer et al. [15] demonstrates how it empowers the machine learning user from algorithm selection and hyperparameter setting. Meta-learning, Bayesian optimisation (BO), and ensemble construction make up its three main elements. Auto-sklearn uses traditional ML algorithms provided by the sklearn library. The optimal hyperparameters may be found using Bayesian optimisation, which is data-efficient. However, Auto-Keras performs better than Auto-sklearn when utilising data that is suitable for deep neural networks. Moreover, the recent version of Auto-Sklearn has compatibility issues with various Operating systems. Hence, it was not considered for this research work.

3.5.3. TPOT

The tree-based pipeline optimisation tool (TPOT) [16], an AutoML framework, is based on an evolutionary method that uses genetic programming to select a framework for implementation. To provide the optimal Python code, it can evaluate hundreds of pipelines. It has built-in classification and regression algorithms. It combines several pipeline operators to create flexible tree-based pipelines. ML models from the Sklearn library or different data transformation operators make up these operators. However, it cannot handle categorical data and language processing power. TPOT mainly employs ML pipelines but with complex datasets, deep learning implementation is required which is provided by AutoKeras.

3.5.4. AutoWEKA

AutoWEKA [17] is an automated machine learning tool, which has been designed to find the best machine learning algorithm automatically for any dataset. It uses the WEKA framework which is a meta-learning approach for finding the best algorithm for a specific problem by comparing the performance of a lot of machine learning algorithms on the given dataset. AutoWEKA also tunes the hyperparameters of the best algorithm selected to improve its performance. It is a user-friendly tool that requires very limited user input and is suitable for both experts and non-experts. AutoWEKA is an open-source software tool that can be used freely.

3.5.5. Google AutoML

Google AutoML[2] is an attempt by Google to empower professionals with limited knowledge in Machine Learning to generate models based on their specific needs. They use techniques like evolutionary algorithms, and neural architectures to build a deep learning model using data and prompts by the user. However, it is only restricted to Google Clouds and it is very difficult to transfer enormous amounts of data from existing systems to a cloud network.

3.5.6. DarwinAI

DarwinAI[4] is an AI solutions provider that provides a range of AI tools and technologies to develop and deploy Neural Network models. This allows users to quickly and easily develop highly optimised neural network architectures without requiring a deep understanding of neural network architecture design. The platform also includes a range of tools for data preparation, model training, and model deployment, making it a comprehensive end-to-end AI tool.

3.5.7. Lale

Lale [45] is a semi-automated approach using scikit-learn, for tuning hyperparameters and selecting algorithms. It mainly aims at users with some knowledge of data science and machine learning, providing a high-level interface to experiment with different neural structures with their data. Lale uses popular existing tools like GridSearchCV, XGBoost, etc. for automation and interoperability. Lale is available as an off-the-shelf tool in Python. Nevertheless, it is only limited to professionals and is difficult to extend it to other domains.

3.5.8. Auto Pytorch

Auto Pytorch [46] is an automl framework based Pytorch. It was developed by AutoML Groups Freiburg and Hannover in the year 2021 with close collaboration with the University of Freiburg. This automl architecture is specialised to solve tabular data and time series datasets. This framework combines neural architecture search with ML hyperparameter tuning. It gives a developer-friendly API to interact with the model similar to AUtoKeras. However it does not optimise for Image and Text data as input format, so it is less generalised compared to AutoKeras.

3.5.9. Online AutoML (OAML)

The Online AutoML framework (OMAL) is an extension of an open-source General Automated Machine Learning Assistant (GAMA) framework. It has been developed by Pieter Gijsbers [47] in 2019. This framework generates an ideal machine-learning model depending upon the dataset and resource limitation provided to the framework. GAMA uses several search processes to find some implacable machine-learning models and combine them into one ensemble pipeline. OMAL extends GAMA's ability to handle online learning.

3.6. Result visualisation

Once AutoKeras creates the best neural network based on the given data, it is very important for the user to know and evaluate its performance. Hence, it is very important to visualise the result. We have used Plotly to satisfy this requirement.

3.6.1. Plotly

Plotly[8] is a graphing tool used to communicate data with customised visualisation and interactive graphs. It is available as a free-for-use library available for many coding languages like Python and R languages. Plotly provides an extensive range of charts and graphs that can be easily embedded in web applications. Thus, Plotly was used for most of the visualisations in the application to make it user-friendly and interactive.

3.7. Explainable AI (XAI)

In the field of machine learning, "explanations" at different levels offer insights into various elements of the model, from knowledge of the learnt representations to the identification of various prediction techniques, general trends and patterns, as well as the evaluation of the general model behaviour [41]. The two types of model explainability are global explainability and local explainability. When a model is globally explainable, users may infer its meaning from its general organisation. Local explainability only takes into account a single input and seeks to understand why the model chooses a certain course of action.

[8] https://plotly.com

Table 2
Comparison of different AutoML models.

Tool	Framework	Auto feature extraction	User interface	Explainability
Auto-Sklearn [15]	Scikit-learn	Only Missing Values	✗	✗
AutoKeras[a]	Keras	✗	✗	✗
TPOT [16]	Scikit-learn	✗	✗	✗
AutoWeka [17]	Weka	✗	✗	✗
Google AutoML[b]	TensorFlow,SparkML	Only Missing Values	✓	✓
DarwinAI[c]	TensorFlow	✓	✓	✓
Lale [45]	Scikit-learn	✗	✗	✗
Auto-PyTorch [46]	PyTorch	✗	✗	✗
Online AutoML (OAML) [48]	GAMA	✗	✗	✗

[a] https://plotly.com
[b] https://rapidminer.com
[c] http://automl.chalearn.org/

3.7.1. Local interpretable model-agnostic explanations (LIME)

The LIME methodology proposed by Ribeiro et al. [49] generates local explanations of classifier f predictions by fitting a simpler, interpretable explanation model g locally around the data point x to be explained. To maintain interpretability in the generated explanations, LIME represents the data in a way that is comprehensible, locally accurate, and model-neutral. These explanations are simpler because they demonstrate a closer relationship between the input and prediction [50].

For instance, let the grey-scale value vector of pixels in an image be $x \in \mathbb{R}^d$. A comprehensible representation of the initial dataset is used to fit the XAI model. The presence or absence of pixels in the picture might therefore be represented by a binary value vector as an interpretable representation of $x' \in \{0,1\}^{d'}$. (absence refers to having the value of a background colour, e.g., white). As a result of resolving the optimisation issue, the LIME explanation $\hat{g}$ is generated (1).

$$\hat{g} = \underset{g \in G}{argmin} \mathscr{L}(f, g, \pi_x) + \Omega(g) \tag{1}$$

Here,
G = the explanation model family,
$\mathscr{L}$ = loss function,
π_x = the locality around x,
Ω = complexity penalty.

Practically, G is the collection of linear regression models, with Ω limiting the expanse of explanatory features that can possess regression weights other than zero (even if several explanatory models may be employed). The weighted L2 distance is assumed to represent the loss function as in Eqs. (2).

$$\mathscr{L}(f, g, \pi_x) = \sum_i \pi_x(z_i)(f(z_i) - g(z_i'))^2 \tag{2}$$

where the sum passes over a collection of selected perturbed points around $x, (z_i, z_i')i = 1, \ldots, m$, where,
z_i = a disturbed data point in the initial dataset,
z_i' = the corresponding explainable version;

Here, $\pi_x(z_i)$ assigns a weight to each sample according to how similar they are to the point x, which is used to explain the classification result.

The second cloud function receives the model along with the training and test data when they are generated and uploaded to their respective cloud storage. The function generates SP-Lime explanations. For generating the explanations, a random 20 test data samples were chosen, among them, 5 results were generated. The combined graphs were then uploaded to a cloud storage bucket as a single HTML file. Any client-side web or mobile application can use the Google Cloud SDK to retrieve data, upload predictions, and explain them.

4. Validation of CloudAISim toolkit: Modelling of healthcare application

We used a healthcare application as a case study in order to verify the scientific reliability of this proposed CloudAISim framework. So, this section discusses the experimental process, datasets, and data preparation with all the relevant details on several use cases as in Table 7. The results on different datasets are compared against each other visually and through various metrics. In the healthcare domain, we have considered four different datasets such as Breast Cancer Wisconsin Diagnosis, Heart Disease Cleveland Dataset, Diabetes Dataset and COVID-19 Dataset.

Table 3
Classification report for 75:25 train–validation ratio of Breast Cancer dataset.

Class	Accuracy	Precision	Recall	f1-score
0	0.97	0.99	0.98	0.98
1	0.99	0.96	0.98	0.97

Table 4
Classification report for 75:25 train–validation ratio for Heart Disease Cleveland dataset.

Classes	Accuracy	Precision	Recall	f1-score
0	0.95	0.98	0.94	0.96
1	0.95	0.92	0.98	0.95

4.1. Case I: Breast cancer Wisconsin diagnosis

The greatest cause of cancer-related death among women worldwide and the malignancy with the highest rate of diagnosis is breast cancer [51]. According to Mubarak's epidemiological research, breast cancer, a particularly deadly kind of cancer, can cause death and mortality in women if it is not recognised in its early stages [52]. It can be found using a variety of techniques, including X-ray mammography, 3-D ultrasound, computed tomography, positron emission tomography, magnetic resonance imaging (MRI), and breast temperature monitoring, although a pathology diagnosis is the most reliable [53]. Sex, age, oestrogen, family history, gene abnormalities, an unhealthy lifestyle, and other variables linked to the development of the illness are only a few of the many risk factors for breast cancer [54].

Our dataset contains tabular data with 32 features and over 569 data points. A fine needle aspirate (FNA) of a breast lump is used to generate the features from a digital image in 3-dimension. The model is trained-tested with a 75:25 ratio of this dataset as given in Table 3. With such specifications, The proposed application produced an accuracy of 98% which is much better than existing cloud-based models. Further, the Confusion matrix and the ROC curve are also shown in Fig. 3.

4.2. Case II: Heart Disease Cleveland Dataset

Several different cardiac disorders are referred to as "heart disease". Coronary Artery Disease (CAD), which disrupts the blood flow to the heart, is the most typical kind of heart disease in the United States. A heart attack may result from reduced blood flow. Heart illness can

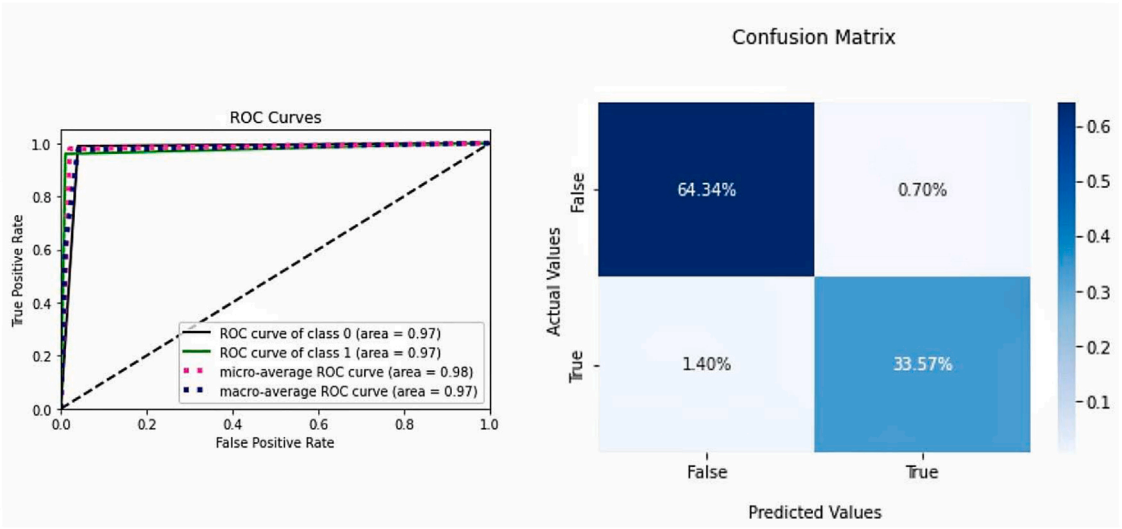


Fig. 3. ROC curve and Confusion matrix for Breast Cancer Dataset.

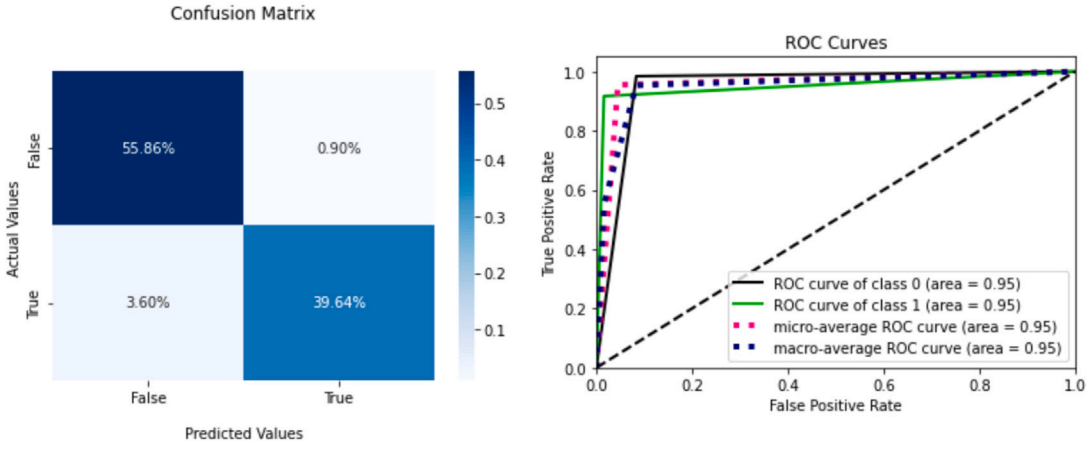


Fig. 4. ROC curve and Confusion matrix for Heart Disease Cleveland Dataset.

Table 5
Classification report for 75:25 train–validation ratio for Diabetes dataset.

Classes	Accuracy	Precision	Recall	f1-score
0	0.99	0.99	0.95	0.97
1	0.9	0.91	0.99	0.95

Table 6
Classification report for 75:25 train–validation ratio of Covid-19 dataset.

Classes	Accuracy	Precision	Recall	f1-score
0	0.96	0.97	0.98	0.97
1	0.94	0.93	0.92	0.93

sometimes go unnoticed until a person exhibits the early symptoms or signs of a cardiac arrest, heart failure, or an arrhythmia [55].

The dataset implemented in the proposed model constitutes over 300 patients' data with 75 attributes. With a ratio of 75:25 from this dataset, the model is trained and validated. With these conditions, the proposed application obtained an accuracy of 95% with a precision and recall of 0.95 and 0.96 respectively (as shown in Table 4). In addition, Fig. 4 also displays the ROC curve and the Confusion matrix.

4.3. Case III: Diabetes dataset

Diabetes is a chronic condition brought on by either insufficient insulin production by the pancreas or inefficient insulin use by the body. The hormone called insulin controls blood sugar levels. Uncontrolled diabetes frequently results in hyperglycemia, or elevated blood sugar, which over time causes substantial harm to many different bodily systems, including the neurons and blood vessels. A total of 1.5 million fatalities were directly related to diabetes in 2019, and 48% of these deaths occurred in those under the age of 70. Diabetes contributed to an additional 460,000 renal disease deaths, and high blood glucose is responsible for 20% of cardiovascular fatalities around the world [56].

To test the proposed application, the Diabetes dataset from the National Institute of Diabetes and Digestive and Kidney Disease is taken into consideration. The dataset constituted 10 characteristics and 768 instances with all patients being Pima Indian females who are at least 21 years old. The model is trained-tested on the 75:25 ratio of the dataset and achieved an overall accuracy of 96%. The Classification report is provided in Table 5. Further, the ROC-AUC curve and confusion matrix are given in Fig. 5.

4.4. Case IV: COVID-19 dataset

In general, human life and health have been profoundly damaged by the SARS-Cov2-led COVID-19 pandemic [57]. The majority

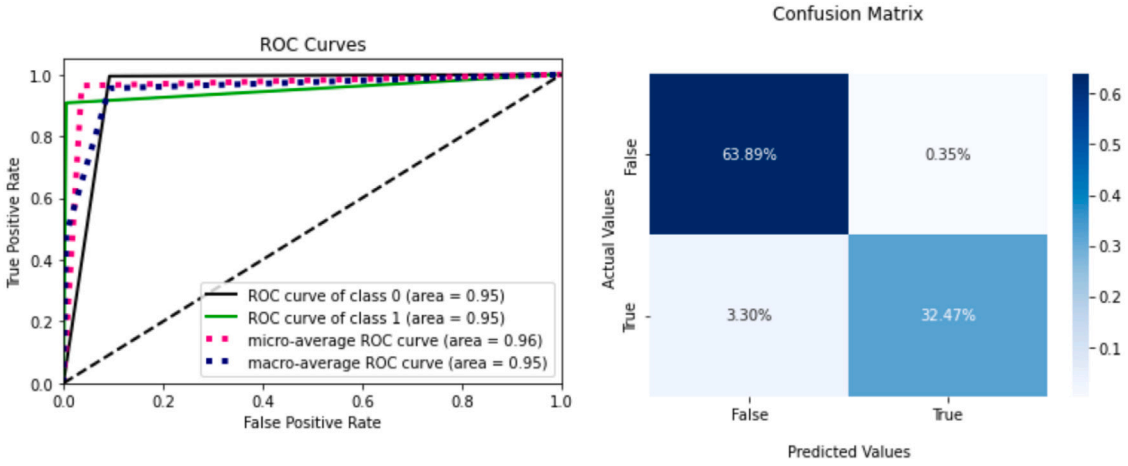


Fig. 5. ROC curve and Confusion matrix for Diabetes Dataset.

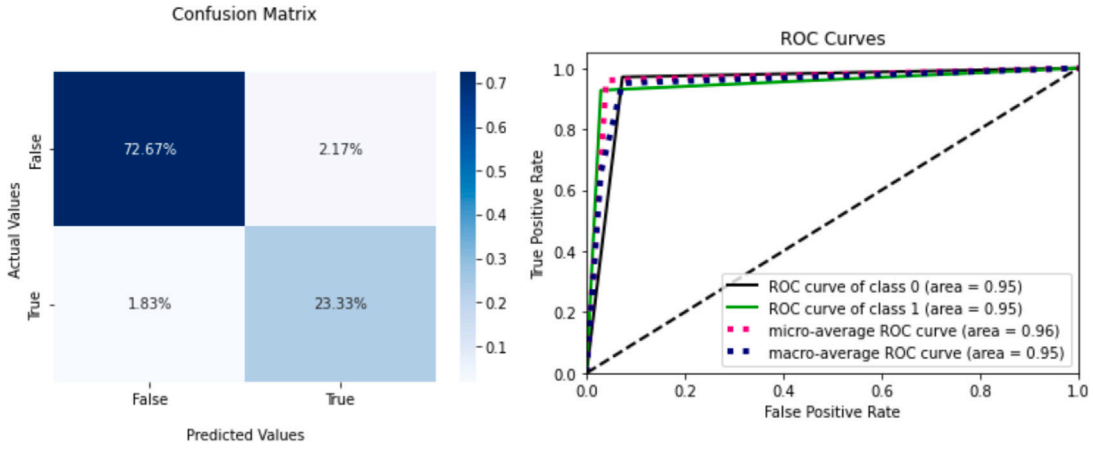


Fig. 6. ROC curve and Confusion matrix for COVID-19 Dataset.

of COVID-19 patients have mild to moderate symptoms. However, this devastating outbreak caused suffering and death in people. COVID-19 has the propensity to target and harm lung tissue [58]. This devastating outbreak caused death in 6,657,706 people around the world. Because of its fast-spreading ability, the World Health Organisation (WHO) designated COVID-19 a Public Health Emergency.

For Covid-19 dataset, tabular data is used with about 800 patient data. It contains 26 attributes such as age, heart conditions, smoking, pregnancy, etc. With a ratio of 75:25 from this dataset, the model is trained and validated. With these conditions, the proposed application has obtained an accuracy of 96% as shown in Table 6. In addition, Fig. 6 also displays the ROC curve and the Confusion matrix.

5. Simulation of proposed healthcare application

The aim of developing this application is to make AI usable for healthcare professionals and normal users, without any technical knowledge. An interactive web application is developed using StreamLit Framework to make it possible. The Application is deployed on StreamLit cloud and Google App Engine.

5.1. Implementation details

This section describes the working of the proposed model and its implementation details. The proposed model uses AutoML to automate the task of recognising and classifying different diseases and verifying the diagnosis.

The entire system is a hybrid architecture of cloud-based platforms and physical servers. The user end has a simple easy-to-read interface to access the proposed framework.

The dataset is uploaded by the user into a cloud storage bucket using the web interface as shown in Fig. 7. After the dataset is entered into the system by the user, the basic information of the dataset like datatype, class distribution, data correlation, etc. is available on the interface. The next Exploratory Data Analysis tab allows the user to select the data to be visualised and the attributes for analysis which will then generate a detailed summary with the help of Pandas Profiling, as shown in Fig. 8. All these computations are carried out on-device. For feature engineering, the user has the choice to make the entire process either manual or automated. In case of manual feature selection, all computations are carried out in the local device and will then be uploaded into the feature engineering cloud bucket (FE_Data) in GCP server. This gives the user freedom to select attributes, impute missing values, perform feature transformation, and remove outliers using different methods like Z-score, inter-quartile range, and so on. When opting for automated feature engineering, the tabular dataset is uploaded into a cloud bucket and the entire process is carried out in the serverless platform using Featuretools (Feature_Engg function) and is uploaded into the FE_Data bucket, as shown in Fig. 9.

As soon as any dataset is uploaded into the FE_Data bucket, the Auto_ML function automatically gets triggered in the cloud. This function has 3 main tasks (1) To generate the best possible model using AutoKeras (default hyperparameters: 200 epochs and 50 max trials); (2) To generate a Train–Test dataset for the upcoming explainability

Table 7
Comparison of proposed model with various train–test ratios for each case study.

Type of datasets	Train–Test ratio	Precision	Recall	Accuracy	f1-score
Case I: Breast Cancer Wisconsin Diagnosis	90:10	0.98	0.98	0.98	0.98
	80:20	0.97	0.99	0.98	0.98
	70:30	0.97	0.98	0.98	0.98
	60:40	0.97	0.96	0.97	0.97
Case II: Heart Disease Cleveland dataset	90:10	0.97	0.97	0.97	0.97
	80:20	0.95	0.96	0.95	0.96
	70:30	0.95	0.96	0.95	0.95
	60:40	0.94	0.94	0.94	0.94
Case III: Diabetes dataset	90:10	0.97	0.96	0.97	0.96
	80:20	0.96	0.96	0.97	0.96
	70:30	0.95	0.96	0.96	0.96
	60:40	0.91	0.92	0.92	0.92
Case IV: COVID-19 dataset	90:10	0.97	0.97	0.98	0.97
	80:20	0.97	0.97	0.98	0.97
	70:30	0.95	0.95	0.95	0.95
	60:40	0.94	0.95	0.96	0.95

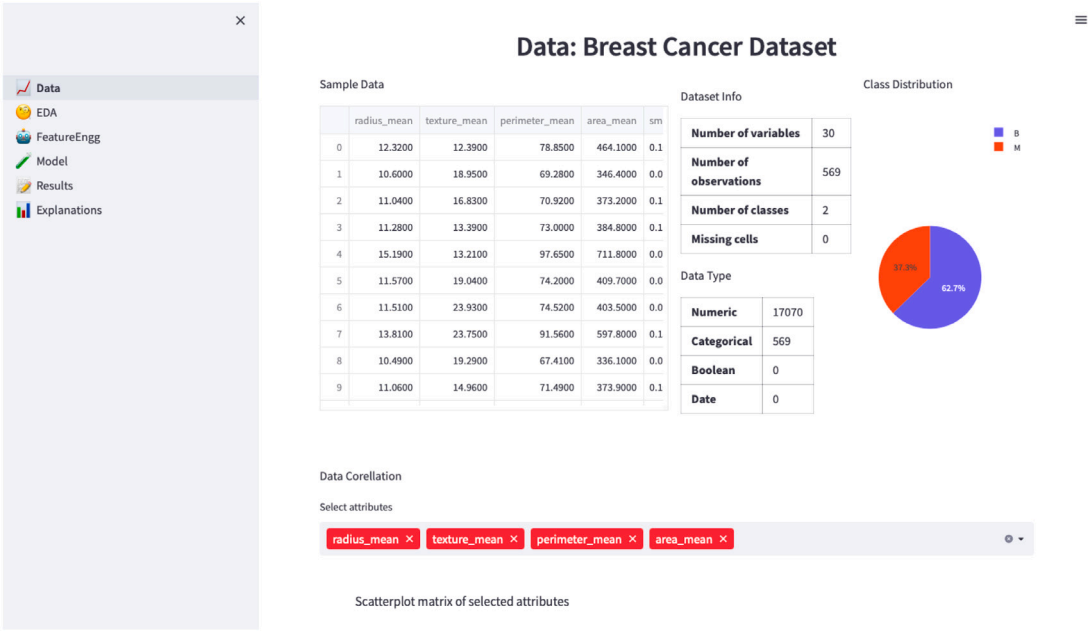


Fig. 7. Web Interface Showing Dataset Page with Breast Cancer Dataset.

function (default Train–Test split: 70-30); (3) To generate results in form of confusion matrics, ROC-AUC curve, PR-curve and classification reports which are plotted using Plotly to make the graphs interactive, as shown in Fig. 10; (4) To generate Tensorflow Logs of the AutoKeras model in the .zip format to the cloud bucket. This data can be accessed by the client-side application using TensorBoard as demonstrated in Fig. 11. Lastly, the LIME explainer cloud function(Ex_AI) is initiated which accesses this model file along with the Train–Test dataset file from the respective cloud bucket and generates 5 sample explanations which are then displayed on the user screen in an HTML format, as shown in Fig. 12. The Lime Explainer displays a feature value table and a plot showing which feature contributed to a particular decision and how much the contribution concerning other features. This means more the value in the feature table, the higher the impact of the feature in the predicted outcome by the model.

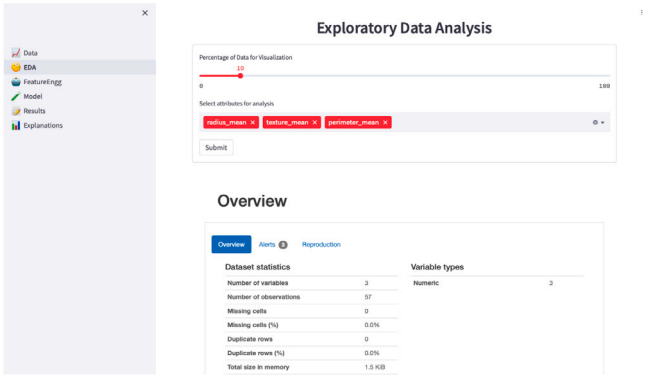


Fig. 8. The EDA page with Breast Cancer Dataset.

6. Discussion

By testing several instances of the function in different conditions, we have realised that the execution lasted almost 2 min on average. Although there are instances when the 'auto_ml' cloud function executes for up to 4 min, it is evident that the large dataset requires more memory and hardware capacity. On the other hand, we are also able to understand from Fig. 13 that many of the function instances have crashed due to several reasons. Two major reasons are the incompatibility of the dataset and the long execution time. As GCP has a limit on the

Fig. 9. The Feature Engineering page with Breast Cancer Dataset.

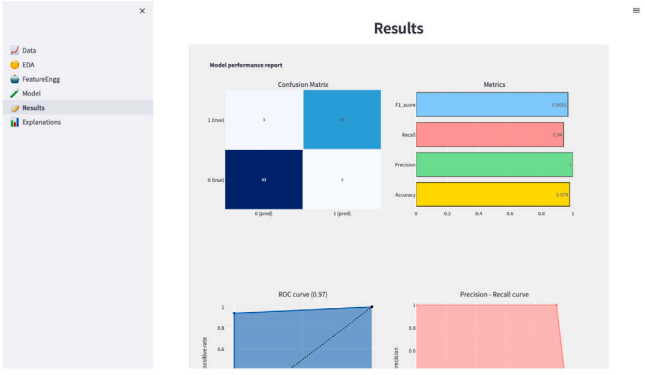


Fig. 10. The Results page with Breast Cancer Dataset.

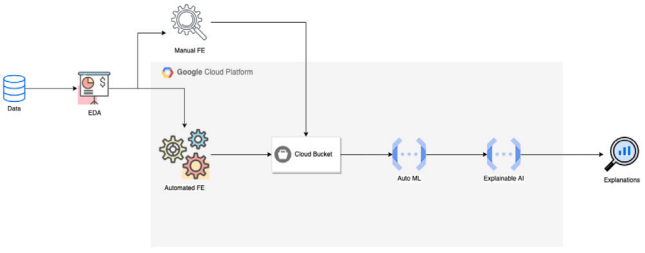


Fig. 11. The Model page with Breast Cancer Dataset.

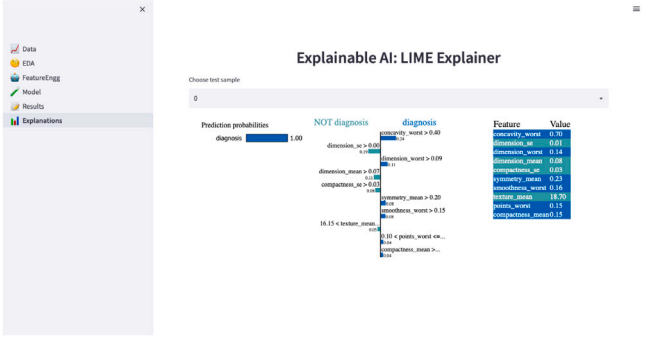


Fig. 12. The LIME Explainer page with Breast Cancer Dataset.

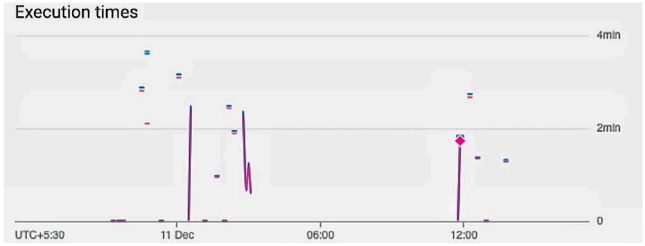


Fig. 13. Execution Time Graph of 'auto_ml' cloud function.

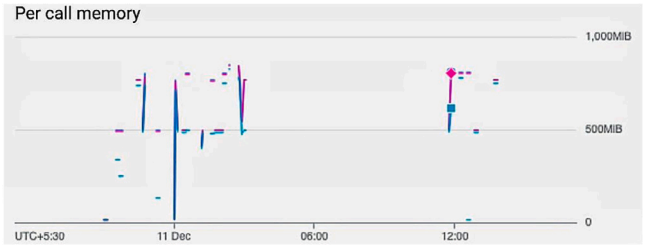


Fig. 14. Per call memory utilisation Graph of 'auto_ml' cloud function.

execution time of up to 5 min for cloud functions, the implementation of a large ML model with a long training time might be problematic. There are multiple ways to fix this issue, which include using multiple functions parallelly or consecutively for a long training period.

Not only does execution time matter for cloud functions, but the memory capacity of the functions also plays a very important role in a cloud environment. According to Fig. 14, most of the function execution took 500 MB to 1000 MB of memory bandwidth. The right balance of memory and processing capability is crucial for smooth function execution. As these are tabular datasets, this execution requires less amount of memory bandwidth in comparison to the image and time series datasets. That means while dealing with image and time-series data we must increase the function memory capacity. The memory bandwidth of the 'auto_ml' function is set at 4 GB at max so that any interruption can be prevented.

It has been concluded that the CloudAISim framework only needs its users to upload the dataset. After uploading the dataset, the CloudAISim framework performs tasks such as EDA, feature engineering, selection of the best machine learning/deep learning model, hyperparameter optimisation, result prediction, and explainability of the result. This makes the CloudAISim framework suitable for all non-experts and non-coders to use the power of AI without writing a single line of code. The CloudAISim framework has achieved a better accuracy of 98% in comparison with existing work [34], which has achieved 85.75% while considering the breast cancer Wisconsin dataset only.

7. Conclusions and future work

There has been substantial progress in globalizing the use of ML to non-experts in data analysis. However, these robust support systems behave like highly efficient black boxes since they do not offer comprehensive information on the recommendations and the internal workings of these models. Traditional ML methods do not always cater to the diverse nature of the datasets, and it is very difficult and tedious for a non-professional to design models specifically for specific datasets. Additionally, these powerful systems are mostly resource-intensive models, which is a big obstacle for the healthcare industry. Moreover, conventional machine learning approaches may not consistently address the varied characteristics of datasets, making it challenging and laborious for individuals without the expertise to create models tailored to particular datasets. In this paper, we have presented

a novel transparent serverless and self-explanatory AutoML framework called CloudAISim to overcome these issues. The proposed framework possesses the ability to autonomously select models in accordance with the given dataset and the task. Further, we designed a healthcare application as a case study in order to verify the scientific reliability of this proposed CloudAISim toolkit. The proposed healthcare application is promising for automated machine learning that has the potential to make AI accessible to non-technical individuals and healthcare professionals. An interactive web application that is user-friendly and effective has been created using the StreamLit Framework and deployed on both the Google App Engine and the StreamLit cloud. The model's maximum accuracy of 98% demonstrates that it is successful in achieving its objective. By providing a more effective and precise way to analyse medical data, the application has the potential to help both patients and healthcare professionals significantly.

7.1. Possible extensions of CloudAISim toolkit

In the future, the CloudAISim can be further extended in the following ways:

1. **Regression Model:** The model is primarily addressing the classification data problem as it is the most prominent use case in the healthcare domain. It can be extended into regression problems as well.
2. **IoT Applications:** The application of this model can be extended to further domains from agriculture to manufacturing and finance [44].
3. **Training:** The model can also be trained for incorporating different types of inputs like images, audio files, text data etc.
4. **Time-series Data:** StrutureDataClassifier and StrutureDataRegressor deal with tabular data and it does not count on the other forms of data such as time-series data [59]. Although time-series data are less frequent than image data in the context of health care, it is useful while measuring real-time patient activity.
5. **Data Variability**: Different forms of data like images and videos need more processing capability, which can be implemented using edge architecture and extended using a cloud model training schedule [60].
6. **Edge Computing**: Real-time disease detection using on-device model prediction can be implemented in edge-fog and cloud models.
7. **Privacy**: Federated learning [61] can be implemented for the privacy protection of the patients by which the learning model can improve from individual patients' model feedback.

Software availability

All the data, material and code involved in this research work are available for public use in Github at abhimanyubhowmik/CloudAISim.

CRediT authorship contribution statement

Abhimanyu Bhowmik: Writing – original draft, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Madhushree Sannigrahi:** Writing – original draft, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Deepraj Chowdhury:** Writing – original draft, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ajoy Dey:** Writing – original draft, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Sukhpal Singh Gill:** Writing – original draft, Supervision, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] E. Brynjolfsson, L.M. Hitt, H.H. Kim, Strength in numbers: How does data-driven decisionmaking affect firm performance? 2011, Available at SSRN 1819486.
[2] W. Samek, K.-R. Müller, Towards explainable artificial intelligence, in: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, Springer, 2019, pp. 5–22.
[3] Y. Bengio, Y. LeCun, et al., Scaling learning algorithms towards AI, Large-Scale Kernel Mach. 34 (5) (2007) 1–41.
[4] E. Lindholm, J. Nickolls, S. Oberman, J. Montrym, NVIDIA Tesla: A unified graphics and computing architecture, IEEE Micro 28 (2) (2008) 39–55.
[5] S.S. Gill, M. Xu, C. Ottaviani, P. Patros, R. Bahsoon, A. Shaghaghi, M. Golec, V. Stankovski, H. Wu, A. Abraham, et al., AI for next generation computing: Emerging trends and future directions, Internet Things 19 (2022) 100514.
[6] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, J. Zhu, Explainable AI: A brief survey on history, research areas, approaches and challenges, in: CCF International Conference on Natural Language Processing and Chinese Computing, Springer, 2019, pp. 563–574.
[7] R. Singh, et al., Edge AI: a survey, Internet Things Cyber-Phys. Syst. (2023).
[8] S. Iftikhar, et al., AI-based fog and edge computing: A systematic review, taxonomy and future directions, Internet Things (2022) 100674.
[9] G.K. Walia, et al., AI-empowered fog/edge resource management for IoT applications: A comprehensive review, research challenges and future perspectives, IEEE Commun. Surv. Tutor. (2023).
[10] A. Bhowmik, M. Sannigrahi, D. Chowdhury, A.D. Dwivedi, R.R. Mukkamala, DBNex: Deep belief network and explainable AI based financial fraud detection, in: 2022 IEEE International Conference on Big Data, Big Data, IEEE, 2022, pp. 3033–3042.
[11] S. Tuli, et al., Predicting the growth and trend of COVID-19 pandemic using machine learning and cloud computing, Internet Things 11 (2020) 100222.
[12] F. Desai, et al., HealthCloud: A system for monitoring health status of heart patients using machine learning and cloud computing, Internet Things 17 (2022) 100485.
[13] M. Garouani, A. Ahmad, M. Bouneffa, M. Hamlich, G. Bourguin, A. Lewandowski, Towards big industrial data mining through explainable automated machine learning, Int. J. Adv. Manuf. Technol. 120 (1) (2022) 1169–1188.
[14] M. Golec, et al., HealthFaaS: AI based smart healthcare system for heart patients using serverless computing, IEEE Internet Things J. (2023).
[15] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, F. Hutter, Auto-sklearn 2.0: Hands-free automl via meta-learning, 2020, arXiv:2007.04074 [cs.LG].
[16] R.S. Olson, J.H. Moore, TPOT: A tree-based pipeline optimization tool for automating machine learning, in: Workshop on Automatic Machine Learning, PMLR, 2016, pp. 66–74.
[17] L. Kotthoff, C. Thornton, H.H. Hoos, F. Hutter, K. Leyton-Brown, Auto-WEKA: Automatic model selection and hyperparameter optimization in WEKA, in: Automated Machine Learning, Springer, Cham, 2019, pp. 81–95.
[18] T. Swearingen, W. Drevo, B. Cyphers, A. Cuesta-Infante, A. Ross, K. Veeramachaneni, ATM: A distributed, collaborative, scalable system for automated machine learning, in: 2017 IEEE International Conference on Big Data, Big Data, IEEE, 2017, pp. 151–162.
[19] U. Khurana, H. Samulowitz, D. Turaga, Feature engineering for predictive modeling using reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, No. 1, 2018.
[20] F. Nargesian, H. Samulowitz, U. Khurana, E.B. Khalil, D.S. Turaga, Learning feature engineering for classification, in: Ijcai, 2017, pp. 2529–2535.
[21] M. Reif, F. Shafait, M. Goldstein, T. Breuel, A. Dengel, Automatic classifier selection for non-experts, Pattern Anal. Appl. 17 (1) (2014) 83–96.
[22] R. Vainshtein, A. Greenstein-Messica, G. Katz, B. Shapira, L. Rokach, A hybrid approach for automatic model recommendation, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 1623–1626.
[23] M. Feurer, J. Springenberg, F. Hutter, Initializing bayesian hyperparameter optimization via meta-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 29, No. 1, 2015.
[24] B. Bilalli, A. Abelló, T. Aluja-Banet, R. Wrembel, Automated data pre-processing via meta-learning, in: International Conference on Model and Data Engineering, Springer, 2016, pp. 194–208.
[25] B. Bilalli, A. Abelló, T. Aluja-Banet, R.F. Munir, R. Wrembel, Presistant: data pre-processing assistant, in: International Conference on Advanced Information Systems Engineering, Springer, 2018, pp. 57–65.
[26] I. Guyon, L. Sun-Hosoya, M. Boullé, H.J. Escalante, S. Escalera, Z. Liu, D. Jajetic, B. Ray, M. Saeed, M. Sebag, et al., Analysis of the automl challenge series, Autom. Mach. Learn. (2019) 177.

[27] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2) (2012).
[28] F. Hutter, H.H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in: International Conference on Learning and Intelligent Optimization, Springer, 2011, pp. 507–523.
[29] J. Snoek, H. Larochelle, R.P. Adams, Practical bayesian optimization of machine learning algorithms, Adv. Neural Inf. Process. Syst. 25 (2012).
[30] K. Tu, J. Ma, P. Cui, J. Pei, W. Zhu, Autone: Hyperparameter optimization for massive network embedding, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 216–225.
[31] B. Hall, D.D. Henningsen, Social facilitation and human–computer interaction, Comput. Hum. Behav. 24 (6) (2008) 2965–2971.
[32] M. Garouani, A. Ahmad, M. Bouneffa, A. Lewandowski, G. Bourguin, M. Hamlich, Towards the automation of industrial data science: A meta-learning based approach, in: ICEIS (1), 2021, pp. 709–716.
[33] J. Ferreira, C. Costa, Web platform for medical deep learning services, in: 2021 IEEE International Conference on Bioinformatics and Biomedicine, BIBM, IEEE, 2021, pp. 1727–1732.
[34] R.E. Shawi, K. Kilanava, S. Sakr, An interpretable semi-supervised framework for patch-based classification of breast cancer, Sci. Rep. 12 (1) (2022) 1–15.
[35] A.M. Alaa, M. van der Schaar, Prognostication and risk factors for cystic fibrosis via automated machine learning, Sci. Rep. 8 (1) (2018) 1–19.
[36] S. Alnegheimish, N. Alrashed, F. Aleissa, S. Althobaiti, D. Liu, M. Alsaleh, K. Veeramachaneni, Cardea: An open automated machine learning framework for electronic health records, in: 2020 IEEE 7th International Conference on Data Science and Advanced Analytics, DSAA, IEEE, 2020, pp. 536–545.
[37] Breast cancer wisconsin (diagnostic) data set, 1995, URL: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29.
[38] K.P. Bennett, O.L. Mangasarian, Robust linear programming discrimination of two linearly inseparable sets, Optim. Methods Softw. 1 (1) (1992) 23–34.
[39] UCI Machine Learning Repository, Heart disease data set, 1998, URL: https://archive.ics.uci.edu/ml/datasets/heart+Disease.
[40] National Institute of Diabetes and Digestive and Kidney Diseases, Diabetes dataset, 2022, URL: https://www.kaggle.com/datasets/mathchi/diabetes-data-set.
[41] D. Chowdhury, S. Poddar, S. Banarjee, R. Pal, A. Gani, C. Ellis, R.C. Arya, S.S. Gill, S. Uhlig, CovidXAI: Explainable AI assisted web application for COVID-19 vaccine prioritisation, Internet Technol. Lett. (2022) e381.
[42] S.S. Gill, Quantum and blockchain based serverless edge computing: A vision, model, new trends and future directions, Internet Technol. Lett. (2021) e275.
[43] A. Bhowmik, M. Sannigrahi, D. Chowdhury, D. Das, RiceCloud: A cloud integrated ensemble learning based rice leaf diseases prediction system, in: 2022 IEEE 19th India Council International Conference, INDICON, IEEE, 2022, pp. 1–6.
[44] S.S. Gill, et al., Modern computing: Vision and challenges, Telematics Inform. Rep. (2024) 1–46.
[45] M. Hirzel, K. Kate, P. Ram, A. Shinnar, J. Tsay, Gradual automl using lale, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 4794–4795.
[46] L. Zimmer, M. Lindauer, F. Hutter, Auto-pytorch tabular: Multi-fidelity MetaLearning for efficient and robust AutoDLL. arXiv 2020, 2000, arXiv preprint arXiv:2006.13799.
[47] P. Gijsbers, J. Vanschoren, GAMA: Genetic automated machine learning assistant, J. Open Source Softw. 4 (33) (2019) 1132, http://dx.doi.org/10.21105/joss.01132.
[48] B. Celik, P. Singh, J. Vanschoren, Online automl: An adaptive automl framework for online learning, Mach. Learn. 112 (6) (2023) 1897–1921.
[49] M.T. Ribeiro, S. Singh, C. Guestrin, Model-agnostic interpretability of machine learning, 2016, arXiv preprint arXiv:1606.05386.
[50] T. Peltola, Local interpretable model-agnostic explanations of Bayesian predictive models via Kullback-Leibler projections, 2018, arXiv preprint arXiv:1810.02678.
[51] L. Wilkinson, T. Gathani, Understanding breast cancer as a global health concern, Br. J. Radiol. 95 (1130) (2022) 20211033.
[52] S. Mubarik, Y. Yu, F. Wang, S.S. Malik, X. Liu, M. Fawad, F. Shi, C. Yu, Epidemiological and sociodemographic transitions of female breast cancer incidence, death, case fatality and DALYs in 21 world regions and globally, from 1990 to 2017: an age-period-cohort analysis, J. Adv. Res. 37 (2022) 185–196.
[53] X. Zhou, C. Li, M.M. Rahaman, Y. Yao, S. Ai, C. Sun, Q. Wang, Y. Zhang, M. Li, X. Li, et al., A comprehensive review for breast histopathology image analysis using classical and deep neural networks, IEEE Access 8 (2020) 90931–90956.
[54] W. Majeed, B. Aslam, I. Javed, T. Khaliq, F. Muhammad, A. Ali, A. Raza, Breast cancer: major risk factors and recent developments in treatment, Asian Pac. J. Cancer Prev. 15 (8) (2014) 3353–3358.
[55] About heart disease, 2022, URL: https://www.cdc.gov/heartdisease/about.htm.
[56] Diabetes, 2022, URL: https://www.who.int/news-room/fact-sheets/detail/diabetes.
[57] M. Singh, et al., Quantifying COVID-19 enforced global changes in atmospheric pollutants using cloud computing based remote sensing, Remote Sens. Appl.: Soc. Environ. 22 (2021) 100489.
[58] D. Chowdhury, A. Das, A. Dey, S. Banerjee, M. Golec, D. Kollias, M. Kumar, G. Kaur, R. Kaur, R.C. Arya, et al., CoviDetector: A transfer learning-based semi supervised approach to detect Covid-19 using CXR images, BenchCouncil Trans. Benchmarks Stand. Eval. 3 (2) (2023) 100119.
[59] A. Bhowmik, et al., DYNAMITE: Dynamic aggregation of mutually-connected points based clustering algorithm for time series data, Internet Technol. Lett. (2022) e395.
[60] A. Bhowmik, M. Sannigrahi, P.K. Dutta, S. Bandyopadhyay, Using edge computing framework with the internet of things for intelligent vertical gardening, in: 2023 1st International Conference on Advanced Innovations in Smart Cities, ICAISC, IEEE, 2023, pp. 1–6.
[61] D. Chowdhury, S. Banerjee, M. Sannigrahi, A. Chakraborty, A. Das, A. Dey, A.D. Dwivedi, Federated learning based Covid-19 detection, Expert Syst. 40 (5) (2023) e13173.

BenchCouncil Transactions on Benchmarks, Standards and Evaluations 3 (2023) 100151

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/

Full length article

Characterizing and understanding deep neural network batching systems on GPUs

Feng Yu [a,b], Hao Zhang [c], Ao Chen [a,b], Xueying Wang [d], Xiaoxia Liang [e], Sheng Wang [c], Guangli Li [a,b,f,*], Huimin Cui [a,b], Xiaobing Feng [a,b]

[a] State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, China
[b] University of Chinese Academy of Sciences, China
[c] China Mobile Research Institute, China
[d] Beijing University of Posts and Telecommunications, China
[e] Intel Corporation, China
[f] UNSW Sydney, Australia

ARTICLE INFO

Keywords:
Deep learning systems
Dynamic batching
Neural networks
Performance characterization

ABSTRACT

As neural network inference demands are ever-increasing in intelligent applications, the performance optimization of model serving becomes a challenging problem. Dynamic batching is an important feature of contemporary deep learning serving systems, which combines multiple requests of model inference and executes them together to improve the system's throughput. However, the behavior characteristics of each part in deep neural network batching systems as well as their performance impact on different model structures are still unknown. In this paper, we characterize the batching system by leveraging three representative deep neural networks on GPUs, performing a systematic analysis of the performance effects from the request batching module, model slicing module, and stage reorchestrating module. Based on experimental results, several insights and recommendations are offered to facilitate the system design and optimization for deep learning serving.

1. Introduction

As the demand for deep learning algorithms based on deep neural networks (DNNs) continues to increase, serving systems [1–3], which provide DNN training and inference as services to users on computing platforms, are sparking interest in both academia and industry. Given the user's real-time response desire, achieving low-latency inference becomes a fundamental prerequisite in these serving systems. To effectively handle model inference requests, dynamic batching plays a crucial role in existing serving systems for improving the system throughput by leveraging parallelism and locality between batched inputs. Unlike training, where all training inputs are available before training starts, inference presents a different challenge as input arrives at the serving system over time, and its arrival rate depends on the popularity of the deployed models. Therefore, inference batching must carefully balance the trade-off between latency and throughput. For instance, larger batch sizes may improve throughput but introduce longer waits for the scheduler to accumulate a sufficiently large input batch and thus increase latency, whereas smaller batch sizes may reduce latency but at the cost of lower throughput.

Traditional deep learning serving systems represented by Triton [1] and TensorFlow-Serving [2] relied on configuring the model-allowed maximum batch size (MAX-BS), which limits the input that can be batched, and the batching time window (TW), which indicates the longest wait time for inputs for combining a batch, as hyper-parameters. Unfortunately, these statically configured serving systems lack the flexibility to dynamically adjust server traffic to accommodate varying loads, leading to sub-optimal performance. For instance, during periods of low-load inference request traffic, employing a large time window results in over-provisioning, as queued requests within the window increase the average response time. Conversely, in server congestion scenarios, larger batch time windows and batch sizes may prove advantageous. Traditional serving systems lack the capability of interrupting ongoing batches to serve new arriving requests. Recently, multi-entry multi-exit batching systems, e.g., DVABatch [3], have arisen, which adopt sub-graphs as the scheduling granularity and introduce several meta-operations to improve the system throughput.

* Correspondence to: Institute of Computing Technology, Chinese Academy of Sciences, 100190 Beijing, China.
E-mail addresses: yufeng@ict.ac.cn (F. Yu), zhanghao@chinamobile.com (H. Zhang), chenao23s@ict.ac.cn (A. Chen), wangxueying@bupt.edu.cn (X. Wang), xiaoxia.liang@intel.com (X. Liang), wangshengyjy@chinamobile.com (S. Wang), liguangli@ict.ac.cn (G. Li), cuihm@ict.ac.cn (H. Cui), fxb@ict.ac.cn (X. Feng).

https://doi.org/10.1016/j.tbench.2024.100151
Received 2 November 2023; Received in revised form 3 January 2024; Accepted 6 January 2024
Available online 13 January 2024

DNN serving systems are intrinsically intricate, influenced by numerous factors encompassing neural network models, load levels, and model slicing patterns, among others. However, existing studies predominantly focus on a localized perspective of batching systems, without providing a comprehensive characterization and understanding of batch behavior. As such, this paper aims to reveal the intricacies of batching behavior in DNN serving systems, offering valuable insights into resource management and system design, particularly concerning typical neural network models and workloads encountered by service providers. Meanwhile, we underscore the limitations of existing serving system batching techniques while presenting innovative optimization avenues for serving system developers.

To characterize the batch behavior within DNN serving systems, we perform a comprehensive systematic evaluation on a GPU platform. We conducted experimental evaluations using representative DNN models from three different domains, including ResNet [4] for image classification, BERT [5] for natural language processing (NLP), and LinkNet [6] for image segmentation. As described in Section 3.2, ResNet has low utilization of computing resources, BERT can saturate system resources even with small batches, and LinkNet is memory-bounded. Leveraging these three representative models, we conduct an in-depth investigation into the behaviors of the request batching, model slicing, and stage reorchestrating within batched serving systems. Regarding the request batching module, we initially examine the impact of batch size on system throughput and request average latency (Section 4.1), followed by a comprehensive exploration of hyperparameters, specifically the MAX-BS and TW, and their relationship with system throughput (Sections 4.2 and 4.3). For the model slicing module, we discuss the influence of slicing positions and the number of stages on system throughput (Sections 5.1 and 5.2), respectively. For the stage reorchestrating module, we analyze the correlation between reorchestrating strategies and system throughput across varying workloads and network models (Section 6.1). Subsequently, we conduct a comprehensive analysis of meta-operations in multi-entry multi-exit systems, including split and stretch operations (Sections 6.2 and 6.3). Based on observations, we present potential application scenarios, along with insights for various research directions (Section 7). Our contribution can be summarized as follows:

- We perform a comprehensive analysis of batching behavior within deep learning serving systems on GPUs by leveraging three representative neural network models from different application scenarios.
- We characterize the effects of batch sizes and hyperparameters on the behavior of the request batching module, explore different slicing patterns associated with batching within the model slicing module and analyze the influence of stage reorchestrating strategies and meta-operations on the behavior of the reorchestrating module.
- Based on experimental studies, we provide several insights and recommendations to facilitate the system design and optimization for deep learning serving. We hope that these observations could pave the road for developing high-efficiency deep neural network batching systems.

2. DNN batching serving systems

2.1. Meta-operations

In traditional DNN serving systems, such as Triton, the batch size remains constant until the inference is completed, as depicted in the upper part of Fig. 1. In such a design, the next batch can only be launched for execution after the ongoing batch inference is completed, and the requests in the batch cannot be exited early, resulting in longer response latency [7]. To support requests being able to exit or join the serving system, DVABatch abstract the two actions of request exit and join into meta-operations, namely split and stretch operations. Fig. 1 shows how meta-operations can reduce average latency. To simplify the explanation, we assume that each operator completes in 1 time unit (T) and the MAX-BS is 4. In this case, once 4 requests are received or the batching time window ends, the received requests will be batched and issued for execution.

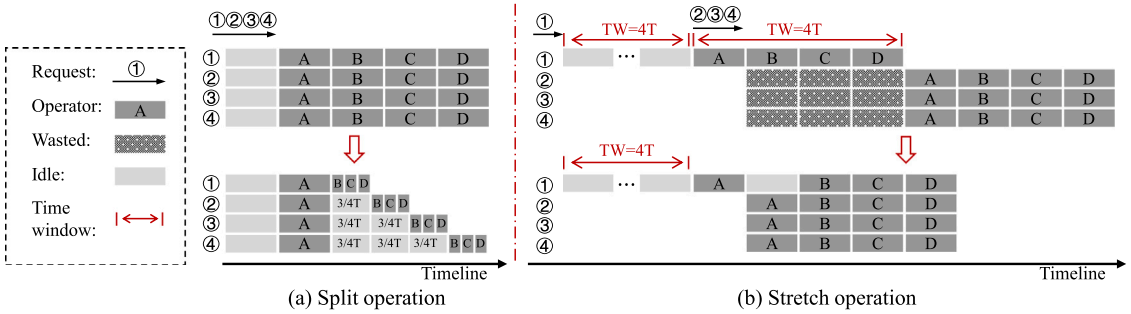
Through the split operation, a large ongoing batch is split into several smaller batches for individual processing, which makes it easier for some queries in the batch to exit early. Fig. 2 shows the execution time of two convolution operations in Resnet under different batch sizes. Convolution operations dominate DNNs (accounting for 86% of the computation time) [8]. As shown in the figure, the preferred batch sizes of Convolution-A and Convolution-B are 4 and 1, respectively. For Convolution-A, using a batch size smaller than 4 cannot fully utilize the GPU (the processing time starts to increase only when the batch size is greater than 4). For Convolution-B, batching will only increase its execution time without improving processing throughput. Fig. 1(a) shows how the split operation can reduce average latency, where operator A has a preferred batch size of 4, operators B, C, and D have a preferred batch size of 1, and the received requests have been batched and are ready to be issued for execution. In Triton, (upper half of Fig. 1(a)), the requests in the batch start processing at the same time and end at the same time. The lower half of Fig. 1(a) shows the split operation, i.e., operator A executes the full batch, then splits the batch into four smaller batches with a batch size of 1 at operator B, and executes these small batches in sequence. In this way, Requests ①, ②, and ③ can exit earlier. The average latency can be reduced by 28.1% (from 4 T to 2.875 T).

Through the stretch operation, new incoming queries are added to the ongoing batch to form a larger batch, thereby utilizing the hardware computing power. Fig. 1(b) shows how the stretch operation can reduce average latency, where the batching time window is 4 T and the operator preferred batch size is 4. In Triton (i.e., the upper half of Fig. 1(b)), Request ① starts running individually after waiting for a time window, leaving the GPU underutilized. During the processing of Request ①, Requests ②, ③, and ④ arrive, but they must wait to be executed in the next batch. The lower half of Fig. 1(b) shows the stretch operation, where the first batch (containing only Request ①) waits for the second batch after completing operator A, and then the two insufficient batches are merged into a new large batch to fully utilize the hardware. In this way, the average latency can be reduced by 34.4% (from 8 T to 5.25 T).

2.2. Major components

In this section, we introduce three major components of contemporary DNN batching serving systems, including a request batching module, a model slicing module, and a stage reorchestrating module. These three components are ubiquitously present in DNN batching serving systems, such as Triton [1], DVABatch [3], Ebird [9], and LazyBatching [7], among others.

Request Batching Module (RBM). The serving system initiates by placing end-users' requests into a request queue. The RBM subsequently organizes these requests into batches, based on two hyperparameters: MAX-BS and TW. These formed batches are placed in a batch queue for the request processing module to utilize. Fig. 3 illustrates the behavior of different configurations of RBM under the circumstance where request R_i enters the request queue at time t_i. RBM with configuration 0 efficiently aggregates two requests within a specified time window to form a batch. Conversely, RBM with configuration 1 can accumulate three requests during the same time window, resulting in a batch of size equal to MAX-BS. However, RBM with configuration 2, although also capable of collecting three requests within the designated time window, is constrained by MAX-BS, leading to the creation of a reduced-size batch of 2.



(a) Split operation (b) Stretch operation

Fig. 1. Illustration of how meta-operations address the long latency problem of user requests. Split operation enables requests to exit early when encountering operators with high parallelism. Stretch operation enables the merging of multiple insufficient batches to reduce waiting time and fully utilize hardware.

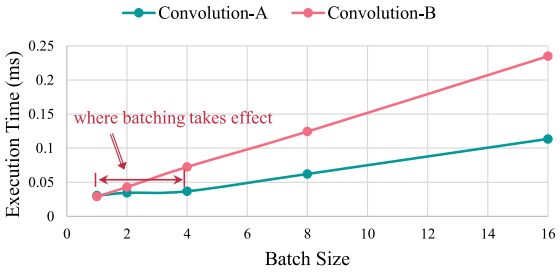


Fig. 2. Execution time of two convolution operators from ResNet with different batch sizes on A100.

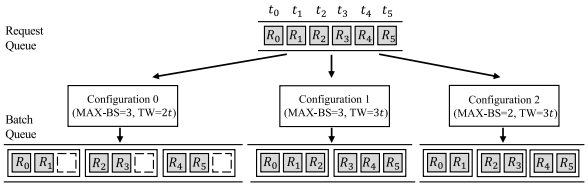


Fig. 3. The behavior of the request batching module under various parameter configurations, namely, the maximum allowed batch size (MAX-BS) and the time window (TW).

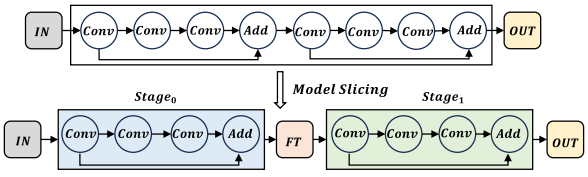


Fig. 4. Diagram of model slicing, where IN/OUT are input/output tensors, and FT are feature tensors.

Model Slicing Module (MSM). To support interruptible batch execution, the serving system needs to slice the models during deployment, which includes determining the slice positions and the number of stages formed after slicing [3]. Fig. 4 provides an example of graph slicing, where slicing occurs after the first Add operation and only once, resulting in two stages with identical graph structures. Batching serving systems frequently employ stage performance models to guide meta-operation decisions, making model slicing critical for system throughput due to its direct influence on stage determination.

Stage Reorchestrating Module (SRM). The stage reorchestrating module typically employs a reorchestrating strategy involving split and stretch operations to control batch and stage execution. The split operation is employed to split large batches into multiple smaller sub-batches, enabling the early completion of smaller sub-batches without waiting for the entire large batch, thus reducing the average request latency. As illustrated in Fig. 5, the split operation divides a batch of size 64 from the $Stage_0$ output into two sub-batches of size 32 each, which are then processed sequentially by two stage instances ($Stage_{1a}$ and $Stage_{1b}$). On the other hand, the stretch operation is used to merge multiple small sub-batches into a larger batch, harnessing hardware parallelism to enhance throughput. As shown in Fig. 5, when a new batch arrives, the current batch is undergoing inference in $Stage_{0a}$. Once the current batch completes the inference in $Stage_{0a}$, SRM passes the new batch to $Stage_{0b}$ for processing. Subsequently, the stretch operation increases the batch size from 16 to a larger batch of size 32, combining the outputs from these two stage instances for $Stage_1$ inference.

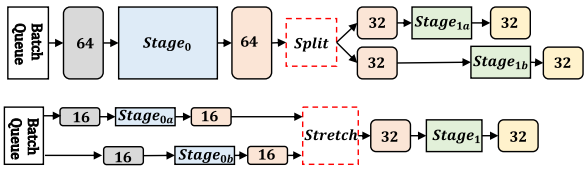


Fig. 5. Diagram of split and stretch operations, where the numbers inside the rounded rectangles represent batch sizes.

3. Experimental setup

3.1. Hardware and software setting

Table 1 lists the setups of the experiments. In this paper, we characterize and analyze the dynamic batching with two serving systems, Triton Inference Server (version 22.05) [1] and DVABatch (main branch) [3], on a high-performance platform that integrates Intel Xeon CPUs and an NVIDIA A100 GPU. As the latency of a DNN model/operator varies with DNN frameworks or compilers [10–12], we employ TensorRT (version 8.2.3) [13] as the inference engine for both of these serving systems to provide SOTA operator performance. Additionally, We use the NVIDIA Triton client [14], which employs an approach similar to MLPerf [15] for generating workloads with arrival times that conform to a uniform distribution. The client uses the HTTP protocol to send requests and sets the QoS target to 200 ms. Regarding DVABatch, we set request rates corresponding to 1/4, 3/5, and 9/10 of the peak throughput as low, medium, and high loads. For ease of experimentation, we align the request rate with the number of client threads, which is 64. Leveraging NVIDIA's Model Analyzer tool [16], we ascertain the maximum throughput attainable by the serving system for specific models. Specifically, the Model Analyzer indicates peak throughputs for ResNet, BERT, and LinkNet as 4288, 1088, and 3264 in DVABatch, respectively.

Table 1
Evaluation specifications.

Hardware		CPU: Intel Xeon Gold 6248 GPU: NVIDIA A100
OS & Driver		Ubuntu: 18.04.2 (kernel 5.4.0-72) GPU Driver: 515.43.04
Software	Client	NVIDIA Triton Client: v22.05
	Server	NVIDIA Triton inference server: v22.05 DVABatch: main branch
	Inference engine	TensorRT: v8.2.3

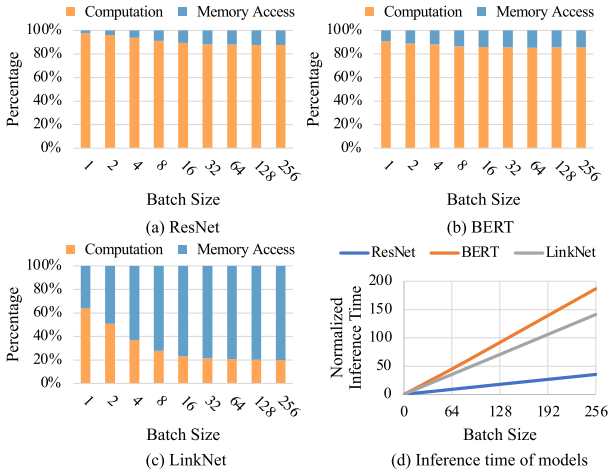


Fig. 6. Performance of three benchmarked models.

3.2. Benchmarked deep neural networks

Incumbent Internet giants have been offering services for tasks such as image classification, natural language processing, and image segmentation, exemplified by Google Cloud Vision AI [17,18], Microsoft Azure Text Analytics [19,20], and Amazon Rekognition [21,22]. This study focuses on these AI domains, employing benchmark network architectures: ResNet [4], BERT [5], and LinkNet [6] for experimental evaluation. Specifically, the study utilizes Torchvision's resnet152 [23], HuggingFace's bert-base-uncased [24], and LinkNet from Purdue University's e-Lab project.

Fig. 6 visually illustrates the end-to-end inference latency of these neural network models for different batch sizes, while also presenting a detailed breakdown of time allocation for computation and memory access. We observe that for the ResNet and BERT models, computation time takes the lead (see Fig. 6(a) and (b)), whereas in the LinkNet model, memory access time predominates (see Fig. 6(c)). Furthermore, Fig. 6(d) illustrates the relationship between inference time and batch size. It is evident from the figure that as the batch size increases, the time of the ResNet model increases relatively slowly, whereas the time of the BERT model experiences a sharp rise. In other words, under small batch sizes, ResNet exhibits lower resource utilization, while BERT potentially leads to system resource saturation.

In summary, ResNet and BERT primarily emphasize computational resources, with ResNet demonstrating efficient resource utilization under small batch sizes, while BERT's resource demands quickly saturate the system. In contrast, LinkNet places a stronger focus on memory access, making it more memory-bound compared to the other models.

3.3. Evaluation metrics

The evaluation metrics include:

- Latency, defined as the average time taken by the serving system to process a query, encompassing both the waiting time and the inference time for the query.
- Throughput, defined as the average number of queries processed by the system per second.
- Inference time, defined as the time required for a DNN model to perform inferences on input data. Unlike request latency, inference time does not encompass the waiting time associated with the request.

Since batching is a technique employed to enhance the throughput of the serving system, in this paper, we will use "system performance" interchangeably with system throughput.

4. Analysis of request batching

4.1. Performance of different batch sizes

Popular DNN serving systems such as Triton support batch execution of multiple requests. In this experiment, as the serving system receives batched inputs that are already formed, it does not wait for them to be collected; thus, we set the time window to 0. Fig. 7 presents the throughput of batching for three typical networks across various batch sizes. Additionally, we demonstrate the benefits of batching in reducing request latency, indicated by the blue line in the corresponding figure.

Finding 1. *In general, the system's throughput can be enhanced by increasing the batch size while meeting QoS requirements.* Observing Fig. 7, it becomes apparent that as the batch size increases, effective throughput rapidly rises, amortizing the inference cost and significantly reducing the request latency. This phenomenon occurs because larger batch sizes increase the computational workload required for inference, allowing better saturation of the GPU's computational resources, thereby achieving higher throughput.

Finding 2. *Enlarging the batch size does not always lead to an improvement for the system throughput.* Once a specific threshold for batch size is exceeded, GPU resources are fully utilized, and further increasing the batch size may lead to request latency exceeding users' expected response time, without yielding additional enhancements in throughput.

4.2. Effects of max batch size settings

We evaluate Triton's performance across various workloads by running three typical neural networks under different MAX-BS configurations. In this experiment, for ResNet, BERT, and LinkNet, the time windows are set to 500 μs, 10 μs, and 10 μs, respectively, aligning with the observations presented in Section 4.3. Our corresponding results are presented in Tables 2, 3, and 4. In these tables, "Collected-BS" represents the batch size formed by the batcher, and "Latency" denotes the average request latency (in milliseconds).

Finding 3. *For ResNet and BERT models, enlarging the MAX-BS parameter has the potential to improve the system throughput.* We observe that as MAX-BS gradually increases, the batch size formed by the RBM also increases correspondingly. For both ResNet and BERT models, Triton's throughput steadily increases with the increasing values of MAX-BS, eventually plateauing, regardless of the workload. In situations where MAX-BS is configured with a smaller value, such as 1, it may lead to a substantial number of requests being blocked in the queue. This occurrence stems from Triton's operational design, where a new batch will initiate execution only upon the completion of the preceding batch. To mitigate this, we can increase MAX-BS to maximize the batch size per execution, thereby reducing the average wait time for requests. However, as MAX-BS increases to a certain extent, although the batch scheduler can form larger batches to amortize inference overhead, it also leads to longer waiting times for requests in the queue.

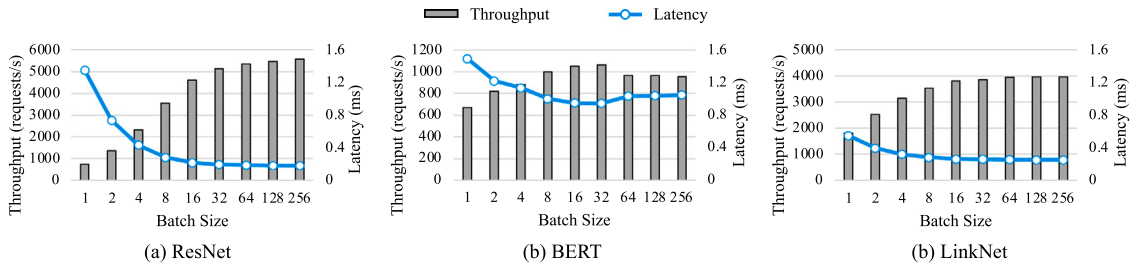


Fig. 7. Effect of batching on throughput and latency of batched execution as a function of batch size. For this experiment, we assume that the batched inputs are already formed, without waiting for them to be collected.

Table 2
Performance of Triton with varying MAX-BS for ResNet model across three workloads.

MAX-BS	Low load			Medium load			High load		
	Throughput	Latency	Collected-BS	Throughput	Latency	Collected-BS	Throughput	Latency	Collected-BS
1	211.40	406.73	1	212.76	410.08	1	212.37	411.79	1
2	414.65	214.52	2	415.43	218.64	2	413.89	220.13	2
4	798.72	109.98	4	810.27	114.45	4	801.31	116.01	4
8	1082.94	8.33	6	1475.22	63.29	8	1476.97	63.67	8
16	1083.26	8.32	6	2356.09	38.30	16	2485.64	37.93	16
32	1082.79	8.34	6	2604.12	13.35	22	3458.38	27.19	32
64	1083.20	8.38	6	2606.72	13.63	22	3463.90	27.28	47

Table 3
Performance of Triton with varying MAX-BS for the BERT model across three workloads.

MAX-BS	Low load			Medium load			High load		
	Throughput	Latency	Collected-BS	Throughput	Latency	Collected-BS	Throughput	Latency	Collected-BS
1	319.33	2.83	1	664.29	64.48	1	663.88	91.08	1
2	319.37	2.95	1	702.70	4.17	1	797.73	74.48	2
4	319.33	2.83	1	700.26	7.56	2	889.94	65.50	4
8	319.32	2.75	1	700.13	13.19	4	1002.18	37.82	8
16	319.30	2.93	1	696.00	22.00	3	1000.02	50.39	13
32	319.32	2.83	1	695.00	22.00	3	989.57	53.73	18
64	319.32	2.84	1	697.06	22.41	5	1003.20	48.88	23

Table 4
Performance of Triton with varying MAX-BS for the LinkNet model across three workloads.

MAX-BS	Low load			Medium load			High load		
	Throughput	Latency	Collected-BS	Throughput	Latency	Collected-BS	Throughput	Latency	Collected-BS
1	830.63	1.37	1	1977.77	1.80	1	2044.28	30.63	1
2	830.34	1.47	1	1976.69	2.52	2	2453.50	25.37	2
4	830.27	1.45	1	1975.77	4.15	3	2541.31	24.58	4
8	830.18	1.55	1	1974.40	8.74	5	2570.41	24.44	8
16	830.31	1.61	1	1967.48	16.67	10	2195.27	28.68	13
32	830.31	1.75	1	1964.68	24.64	16	2006.76	31.41	21
64	829.64	2.26	1	1966.96	23.27	16	1947.94	32.37	20

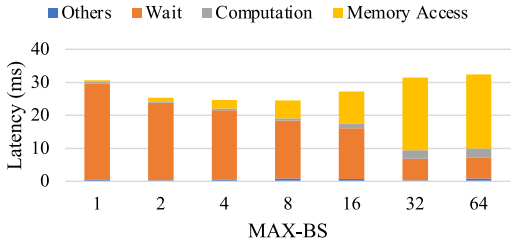


Fig. 8. Performance breakdown with different MAX-BS for the LinkNet model.

Finding 4. *MAX-BS mainly influences the queue wait time, the data transmission time, and the computation time.* For the LinkNet model, Triton exhibits similar behavior to ResNet and BERT models under medium to low workloads. However, under high workloads, Triton's observed throughput initially increases but then decreases as MAX-BS values continue to grow. To further analyze this behavior, we provide a decomposition graph of request latency for different MAX-BS values, as shown in Fig. 8. Fig. 8 shows that as the batch size gradually increases, the waiting time decreases, but memory access time increases due to the growing data volume. In Triton, excessive batch sizes cause a significant increase in request latency, as the memory access time for a request equals that of the entire batch.

4.3. Effects of batching time window

Fig. 9 depicts the influence of time windows on system throughput. In this experiment, for ResNet, BERT, and LinkNet, we configure MAX-BS as 64, 16, and 8, respectively, based on the observations in Section 4.2. The x-axis represents the request rate (the number of client requests sent per second), while the y-axis signifies the system throughput. In addition, the positions of symbols L, M, and H correspond to low, medium, and high rates, respectively.

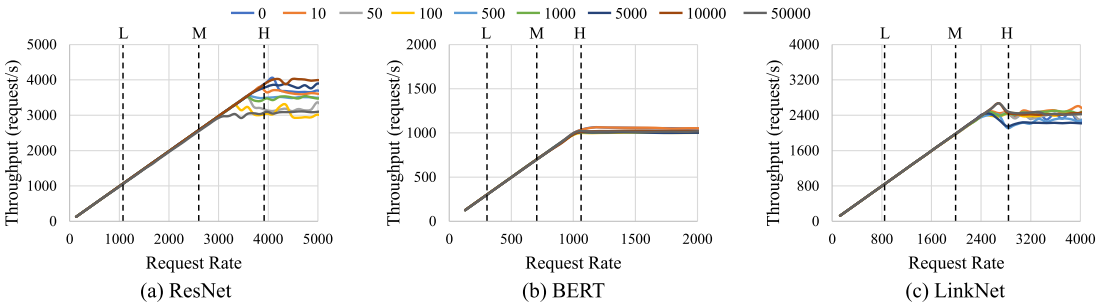


Fig. 9. Effects of time windows on Triton throughput with different request rates. At low and medium loads, the impact of the time window on system throughput is limited, indicating a linear correlation between system throughput and request rate. At high loads, BERT exhibits lower sensitivity to the time window compared to ResNet and LinkNet.

Finding 5. *Under medium to low workloads, the effect of time windows on the system throughput is limited.* As the request rate increases, system throughput exhibits linear growth. This behavior stems from the fact that, under medium to low workloads, the RBM collects a relatively small number of requests. Elevating the request rate can augment the quantity of requests gathered by the RBM, consequently amplifying the system throughput. Nonetheless, once the request rate surpasses a certain threshold, further increases do not contribute to enhanced throughput. This is because, when the request rate surpasses the system's processing capability, requests will be blocked in the queue, resulting in heightened latency.

Finding 6. *Under high workloads, for models that do not fully utilize the resources, time windows impact the system throughput through waiting times and batch sizes.* When subjected to high workloads, Triton's peak throughput for the BERT model exhibits minimal variance across different time windows. Since the BERT model saturates the system's resources with small batches, necessitating requests to wait in the batch queue until resources become available. The time window serves as a parameter for regulating the waiting time of requests in the request queue and the size of batches formed. Selecting an appropriate time window size can enhance the overall throughput of the system. A shorter window reduces queue wait times but limits batch size, underutilizing hardware. Conversely, a longer window extends waits but yields larger batches, maximizing hardware utilization. Therefore, compared to models like BERT, the impact of the time window is more pronounced for models that underutilize resources, such as ResNet and LinkNet.

5. Analysis of model slicing

5.1. Effects of slice positions

In this section, we slice the model into two subgraphs (i.e., stages) and investigate the impact of varying the slicing position on system throughput. We use a slicing ratio to denote the slicing position, specifically, the percentage of the total network compute time allocated to $Stage_0$, that is, $\frac{Stage_0}{Stage_0+Stage_1}$. The evaluation results are presented in Fig. 10, with the x-axis denoting the slicing ratio and the y-axis representing throughput.

Finding 7. *The choice of slice positions has an impact on both the model's computation time and memory access time.* Fig. 11 illustrates the breakdown of end-to-end model inference time at various slice positions. In Fig. 11, there are slight variations in computation time at different slice positions. This phenomenon is attributed to the fact that model slice disrupts operator fusion and other optimizations within the graph. Furthermore, we observe that memory access time at different slice positions is closely related to the model's architecture, specifically, it is influenced by the volume of data exchanged between stages. In addition, although Fig. 11 shows that the model inference time is the lowest when the model is not sliced (i.e., the slicing ratio is 100%), this also implies that meta-operations cannot be applied, so the system throughput is not necessarily optimal, as shown in Table 5.

Finding 8. *When selecting slice points, the computation time, the model structure, and the memory access time are important factors that need to be considered.* Fig. 10 demonstrates the impact of slice points on system throughput. "Naive Batching" refers to a system devoid of meta-operations and pipelined execution. The x-axis represents the slice ratio, and the y-axis represents throughput. Fig. 10 clearly indicates that slice points significantly influence the performance of pipelining execution systems by affecting pipeline balance. We also note that for ResNet, optimal throughput is achieved when slicing occurs in the model's middle, while for LinkNet, it is more advantageous towards the model's end. This variation is attributed to data transfer costs between stages, as depicted in Fig. 11.

5.2. Effects of stage counts

In this experiment, we employ the PipeDream [25] tool to slice the model into several stages with approximately equal execution time, aligning with the experimental methodology of the DVABatch. Fig. 12 illustrates the impact of the number of stages on system throughput, where the x-axis represents the number of stages, and the y-axis represents throughput.

Finding 9. *The optimal number of stages is typically small and, in most cases is not equal to 1.* As the number of stages increases, system throughput experiences a brief increase followed by a gradual decline. In contrast to schemes without model slicing, multi-stage designs support batch interruptions to leverage meta-operations for enhanced system throughput. However, increasing the number of stages introduces additional system overhead, such as synchronization costs between stages, resulting in finer scheduling granularity that undermines graph optimizations like layout selection and operator fusion, subsequently reducing system throughput. Additionally, we observe that pipelined execution is highly sensitive to the number of stages; as the stage count increases, system throughput deteriorates rapidly. Increasing the number of pipelined stages can enhance system throughput, but surpassing a specific threshold may reduce throughput due to resource contention.

6. Analysis of stage reorchestrating

6.1. Effects of reorchestrating strategies

Reorchestrating strategy is a method that prescribes execution in batches or stages, aimed at enhancing system throughput. In batching serving systems, reorchestrating strategy manages batch execution through meta-operations while also determining whether pipelining execution of stage instances is permissible. Stretch, split and pipeline execution are mutually independent, thereby allowing users to configure strategies to determine how these three operations are employed. Table 5 presents the system throughput of eight strategies for the model under various workloads. It can be observed from this table that the impact of strategies on system performance is limited in medium and

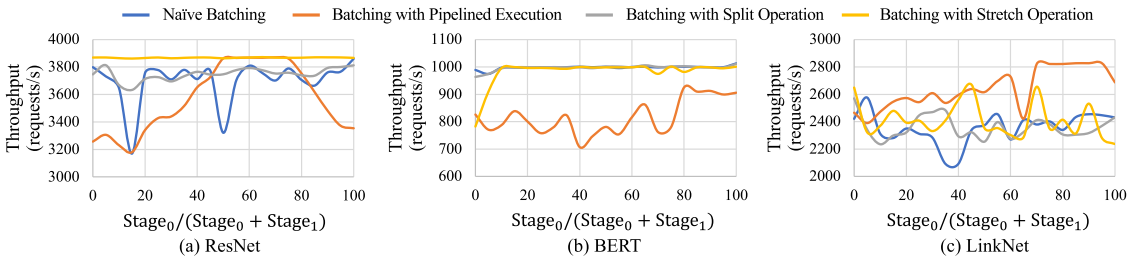
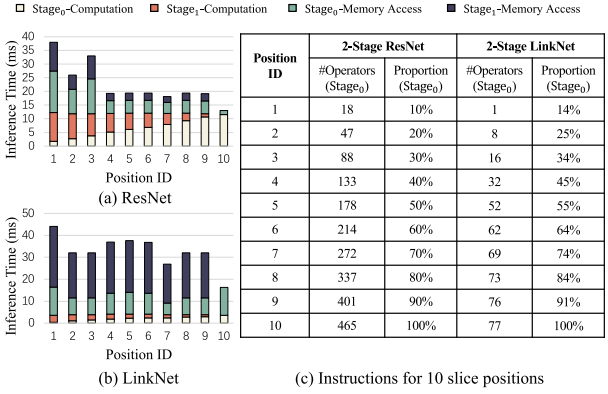


Fig. 10. Effects of slice position on the system's throughput. The optimal slicing position is related to the model structure. For example, the optimal slicing position for ResNet models is in the middle, while the optimal slicing position for LinkNet models is at the end. Pipeline parallel execution is highly sensitive to the selection of the slicing position. BERT models are not suitable for pipeline parallel execution.

Table 5
Effects of reorchestrating strategies on the system's throughput. At low and medium loads, the system throughput is hardly affected by variations with different reorchestrating strategy. Under high loads, however, the performance of the reorchestrating strategy differs among different types of models.

Strategy	Stretch	Split	Pipeline	ResNet			BERT			LinkNet		
				Low	Medium	High	Low	Medium	High	Low	Medium	High
I	0	0	0	1083.14	2608.62	3455.97	319.32	698.61	999.95	830.10	1974.27	2315.73
II	0	0	1	1081.07	2603.97	3867.68	319.33	692.98	749.00	830.02	1974.22	2774.40
III	0	1	0	1083.07	2607.60	3087.69	319.34	697.34	999.47	829.97	1974.36	2409.07
IV	0	1	1	1081.04	2604.87	3740.58	319.35	695.63	997.73	829.98	1974.86	2351.10
V	1	0	0	1081.12	2605.03	3854.02	319.30	696.19	995.43	830.03	1974.49	2368.43
VI	1	0	1	1080.92	2602.08	3866.82	319.33	686.51	752.64	829.91	1973.39	2764.35
VII	1	1	0	1080.89	2604.82	3756.22	319.34	695.99	993.90	829.94	1974.31	2322.81
VIII	1	1	1	1080.88	2602.44	3867.37	319.33	681.77	737.95	829.99	1974.06	2746.51



Position ID	2-Stage ResNet		2-Stage LinkNet	
	#Operators ($Stage_0$)	Proportion ($Stage_0$)	#Operators ($Stage_0$)	Proportion ($Stage_0$)
1	18	10%	1	14%
2	47	20%	8	25%
3	88	30%	16	34%
4	133	40%	32	45%
5	178	50%	52	55%
6	214	60%	62	64%
7	272	70%	69	74%
8	337	80%	73	84%
9	401	90%	76	91%
10	465	100%	77	100%

(c) Instructions for 10 slice positions

Fig. 11. Performance breakdown of 2-stage models with different slice positions. The impact of slicing positions on memory access time is notably significant and correlates with the model structure.

low-load scenarios. Under high load, the impact of strategies varies depending on the model type.

Finding 10. *Pipelined parallel execution is suitable for models with unsaturated computational resources or those encountering memory access bottlenecks. Stretch operations enhance the utilization of system computational resources, and split operations are effective for models bottlenecked by memory access.* For the ResNet model, strategies involving pipelined execution or stretch operations effectively improve resource utilization, thereby enhancing system throughput. However, for strategies that only involve split operations, performance decreases due to the sequential execution of the sub-batches, which prolongs request completion times. For models with saturated system resources, such as BERT, pipelined execution exacerbates resource contention and leads to performance degradation. In contrast, strategies incorporating meta-operations prevent performance degradation because the timing of operations is based on stage-specific performance models. For models with memory access bottlenecks, such as LinkNet, strategies involving pipelined execution effectively hide memory latency and enhance the system's throughput, while stretch operations only marginally reduce average computational time. Furthermore, split operations enable requests to finish in advance, enhancing system throughput by eliminating the need to wait for the entire batch to complete.

6.2. Performance analysis on split operations

The split operation allows requests to exit early, reducing average latency, but the resulting sub-batches may suffer from lower resource utilization, potentially reducing throughput. Therefore, the timing of split operations is a critical factor affecting system throughput. In this section, we explore the impact of the slice position, initial batch size for split, and the number of final sub-batches on the effectiveness of split operations. The evaluation results are presented in Figs. 13 and 14, where the x-axis represents the slice ratio, and the y-axis represents the speedup achieved by split operations compared to naive batching. We divide the model into two stages, $Stage_0$ and $Stage_1$, and the slice ratio refers to the percentage of the total network compute time allocated to $Stage_0$.

Finding 11. *Split operations yield more pronounced acceleration when occurring earlier (i.e., with lower slice ratios).* Observing Figs. 13 and 14, it is evident that split operations achieve their optimal effects with lower slice ratios. As the slice ratio increases, split operations gradually degrade into graph batching. This is because the benefits of split operations stem from the reduced average latency during the execution of sub-batches in the $Stage_1$ sequence. Therefore, a higher percentage of time allocated to $Stage_1$, the primary contributor to performance gains, implies greater potential benefits from split operations.

Finding 12. *Split operations are effective for the system under large batches, and the larger the batch to be divided, the greater the performance gain of split operations achieved.* Examining Fig. 13, it becomes apparent that, given a fixed slice ratio (e.g., 5%), split operations yield higher benefits as the batch size to be divided increases. Large batches may lead to resource contention due to the system's limited resources. Split operations mitigate resource competition by subdividing large batches into smaller sub-batches, thereby reducing average latency. Smaller sub-batches, on the other hand, are often unable to fully utilize hardware resources, and split operations further decrease hardware resource utilization, consequently reducing system throughput. Additionally,

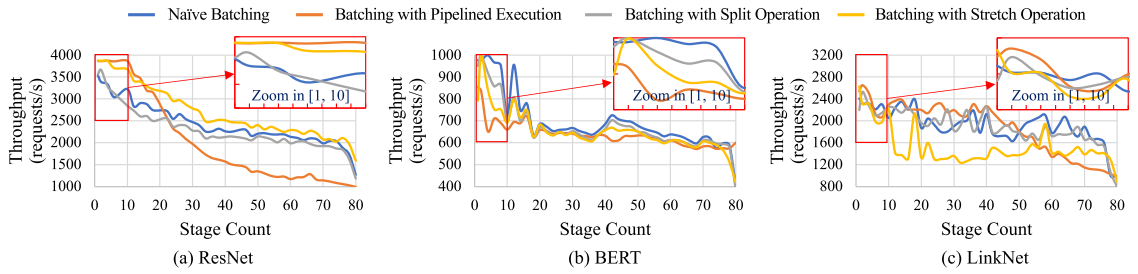


Fig. 12. Effects of stage counts on the system's throughput. As the number of stages increases, the system throughput initially experiences a transient increase, followed by a gradual decline.

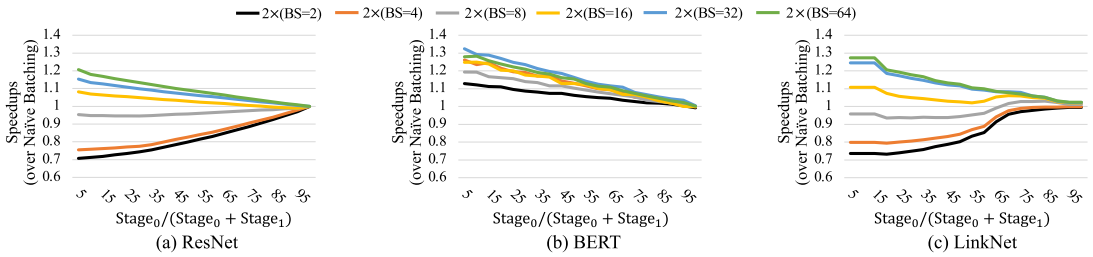


Fig. 13. Effects of split operations on batching process across varying batch sizes, where the legend "2×(BS = N)" represents dividing a batch of size N into two sub-batches, each of size $N/2$.

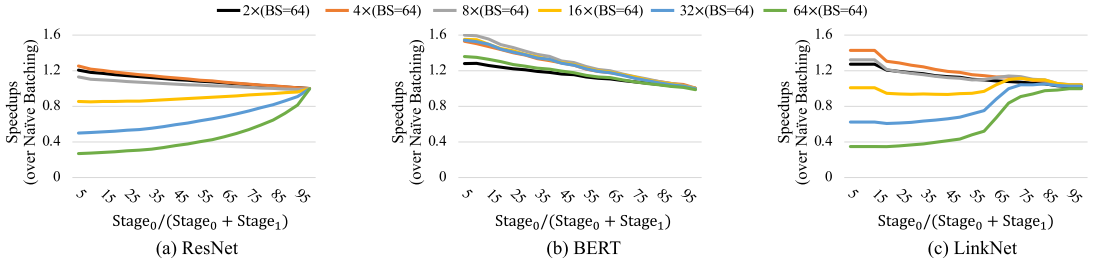


Fig. 14. Effects of split operations on batching process across varying sub-batch counts, where the legend "n×(BS = 64)" represents dividing a batch of size 64 into n sub-batches, each of size $64/n$.

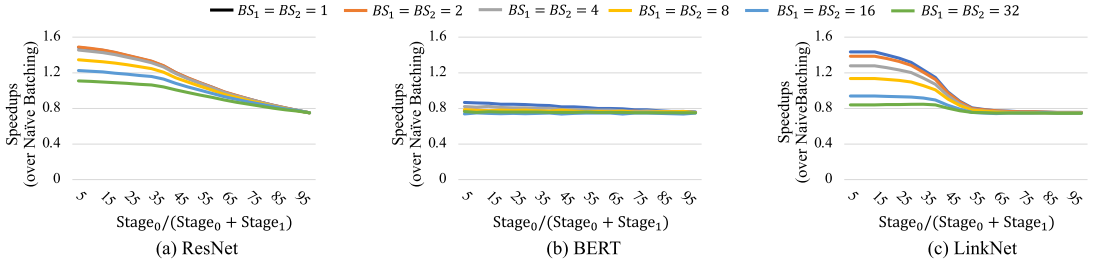


Fig. 15. Effects of stretch operations on the batching process across varying batch sizes.

since the BERT model saturates system resources with small batches, split operations result in acceleration across various batch sizes.

Finding 13. *The optimal number of sub-batches could be guided by the stage's performance model and does not follow the "more is better" principle.* Split operation is applicable to the scenario where resource utilization is saturated, that is, batching only increases its inference time without improving the processing throughput, such as Convolution-B in Fig. 2. Consequently, the split operation can split the original batch into several sequentially executed sub-batches to reduce the average latency, as shown in Fig. 1 (a). Observing the speedups of the split operation for the slice ratio of 5% in Fig. 14, we can find that the optimal number of sub-batches is not 64 (i.e., the green line), that is, the number of sub-batches is not the more the better. This is because when the sub-batch size reaches a certain threshold, further reducing the batch size will lead to insufficient hardware resource utilization due to the small batch size, which does not meet the premise of using the split operation, and thus leads to the ineffectiveness of the split operation or even negative effects. For this reason, we recommend that the timing of using the split operation should be referenced to the curve of the execution time of the stage with the batch size (such as Fig. 2), that is, the performance model.

6.3. Performance analysis on stretch operations

The stretch operation enhances system throughput by consolidating multiple small batches into a larger batch to fully exploit hardware resources. Fig. 5 illustrates the stretch operation process: when a new

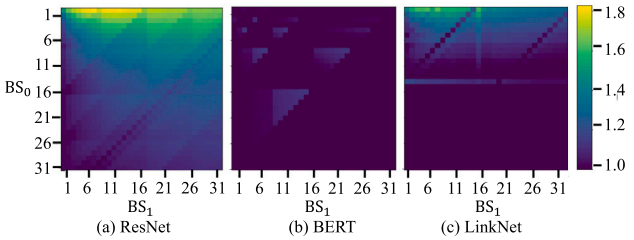


Fig. 16. Effects of various batch combinations on the effectiveness of stretch operations, where BS_0 and BS_1 denote two batches arriving subsequently. The intensity of color shading indicates the acceleration effect of stretch operations. For clarity, we also use 1 to represent negative effects. Stretch operations have a significant effect on ResNet models.

batch (BS_1) arrives, the current batch (BS_0) is in the middle of inference at $Stage_{0a}$. The stretch operation first completes the inference of BS_0 at $Stage_{0a}$, then proceeds to perform inference on BS_1 at $Stage_{0b}$, and finally merges them into a larger batch for $Stage_1$ inference. While stretch operations maximize computational resources by forming larger batches, they introduce waiting time during the batching process. This section analyzes the impact of the sizes and combinations of BS_0 and BS_1, as well as slice positions, on the effectiveness of stretch operations. For ease of analysis, we consider the scenario where BS_0 has just started execution at stage 0 and BS_1 arrives as our target scenario. Fig. 15 illustrates the impact of slice positions on the effectiveness of stretch operations, where the x-axis represents the slice ratio, and the y-axis represents the speedups over not using stretch operations. Fig. 16 illustrates the influence of various combinations of BS_0 and BS_1 on stretch operation efficacy. The brightness of the color signifies speedup levels relative to non-stretch operation scenarios, with instances of negative effect (speedups less than 1) marked as 1 for clarity.

Finding 14. *Stretch operations yield more significant acceleration when performed earlier.* As seen in Fig. 15, stretch operations are more likely to achieve noticeable acceleration when the slice ratio is low. This is because a lower slice ratio implies less waiting time, and with a higher proportion in stage 1 when system resources are not saturated, batch processing benefits more. As the slice ratio increases, the acceleration ratio of stretch operations tends to converge to 0.75. This is because when the slice ratio approaches 100, the proportion of stage 1 becomes nearly zero. Due to the necessity to wait for BS_1 to finish at stage 0, BS_0 cannot exit prematurely, resulting in an average delay of approximately 1.75 times that of batch processing.

Finding 15. *Applying stretch operations can usually enhance the system throughput under small batches.* As shown in Fig. 15, stretch operations exhibit noticeable acceleration when merging small batches, as the system cannot efficiently utilize hardware resources with small batches. However, for models like BERT, the system resources are already saturated with small batches, making stretch operations unsuitable for such models.

Finding 16. *Stretch operations are more suitable for ResNet-like models when computational resources are not saturated, and they are influenced by waiting time and batch processing gains.* Fig. 16 shows that in ResNet models, stretch operations generally have an acceleration effect, particularly when BS_0 is small, as it reduces the waiting time for BS_1 execution and still improves resource utilization after merging. However, for BERT models, stretch operations have minimal acceleration as the system resources are already saturated with small batches. For LinkNet models, stretch operations produce acceleration only with specific batch combinations. Thus, systems should decide whether to adopt stretch operations based on stage-specific performance models.

7. Discussion

In this work, an in-depth analysis and appraisal of the DNN batching serving system were undertaken, offering significant findings. This section gives the application scenarios and potential inspirations based on these findings.

7.1. Serving system configuration

In existing serving systems, the configuration of hyperparameters is a critical factor that affects the effectiveness of batching. However, there is a lack of comprehensive analysis and guidance on the configuration of these hyperparameters. This work fills this gap by providing insights into the impact of hyperparameters on batching effectiveness. Model deployment personnel can use the Finding 3 to configure MAX-BS to a larger value when deploying computationally intensive models. This will help to improve hardware resource utilization by forming larger batches. Furthermore, Finding 5 suggests that deployment personnel need not overly focus on the time window under medium to low loads. Serving system developers can adhere to the recommendations in Finding 9 for setting the number of stages. With these findings and suggestions in place, users of the serving system can more easily obtain appropriate parameter settings without undergoing complex, tedious, and time-consuming experiments and adjustments, thereby accelerating the application of the serving system.

7.2. DNN system optimization

We explore the potential directions outlined by the findings in this paper for promoting optimization and design of DNN serving systems. We first analyzed the relationship between the hyperparameters in the request batching module and the batching effect, and revealed the constituents of request latency (Findings 3–6). This provides a foundation for researchers to design adaptive parameter tuning systems for serving systems. Considering that workloads in practical scenarios often exhibit burstiness [26], and the inference serving time is deterministic [27], we can fit the collected request arrival traces to a Markov arrival process [28] at runtime to capture the burstiness. Based on the components of latency and deterministic inference time, we design a parameter tuner. The tuner determines optimal hyperparameter configurations based on the arrival process and QoS, maximizing throughput while meeting the QoS. Furthermore, we discover that in the model slicing module, the selection of slicing positions should consider computation time, model structure, and memory access time. Additionally, the number of stages correlates with runtime synchronization overhead. The aforementioned analysis offers possibilities for researchers to automatically determine optimal slicing positions and the number of stages. This inspires researchers to design an profiler to obtain computation time and access time under different slicing locations. Then, they can model the inference process under different stage reorchestrating strategies and query arrival processes, subsequently automatically determining the optimal slicing positions and stage numbers based on the performance model. Lastly, we examine the stage reorchestrating module and find that the conditions for utilizing pipelined execution and meta-operations should consider model characteristics and stage performance models. This insight guides researchers designing multi-tenant serving systems to execute computation- intensive stages and memory access-intensive stages in a pipelined manner to fully utilize hardware resources. Concurrently, performance models of stages inform the execution of meta-operations and resource allocation for the stages.

7.3. DNN application development

The findings in this paper also have implications for neural network application developers. Findings 1 and 2 indicate that the performance improvements achieved through batching techniques primarily arise from the efficient utilization of hardware computing resources, particularly when larger batch sizes are employed. Therefore, in the design of neural network models, efforts should be made to reduce the proportion

of memory access time. This hints at the importance for application developers to use lightweight operators whenever possible, such as employing depth-wise convolution operations in place of naive convolutions, and adopting quantization techniques to reduce memory access time. Finding 8 indicates that the position of model slices affects data flow and tensor lifecycle management. Long-lived tensors occupy memory resources for extended periods, increasing memory consumption and limiting the number of batching requests. Thus, DNN application developers should avoid designing long-lived tensors. Finding 9 suggests that model slicing may impact graph optimization techniques like operator fusion. Therefore, our advice to model designers is to construct network models using small, reusable blocks as much as possible to minimize the impact on graph optimization techniques such as operator fusion.

7.4. Impact on large language models

In various applications, the significance of language generation tasks has escalated, sparking heightened interest in optimizing serving systems via batching techniques. Orca represents the inaugural adaptation of DVABatch tailored for Large Language Models (LLMs). A pivotal insight of Orca posits that Transformer-based generative models function iteratively, so the batching should focus on iterations rather than individual requests. Consequently, Orca aligns DVABatch stages with LLM iterations and supports batching arbitrary requests by executing the iterations in a batch that are in prefill and decode states separately. In this study, we conducted a comprehensive evaluation of the DVABatch system, yielding several critical insights.

BERT and Transformer models differ in terms of task objectives and output layers. Transformer is a sequence processing model that uses SoftMax for probability distribution computation at the output layer, while BERT focuses on learning language representations from text data, which is typically used to generate context-related word embeddings. However, they are both implemented based on multiple stacked transformer layers (i.e., including attention layers and forward feedback layers). In this paper, we discover that BERT can saturate hardware resources even with small batch sizes. Furthermore, serving systems utilizing pipeline parallelism exhibit lower throughput when confronted with the BERT model compared to naive serving systems. Consequently, this insight suggests that designers of LLM serving systems should refrain from employing pipeline parallelism on a single GPU platform.

Given that LLMs typically operate iteratively, and the behavioral characteristics during the prefill and decode phases exhibit significant differences [29,30], this constitutes the most prominent distinction between LLM and BERT. Researchers can leverage the findings of this paper and integrate the unique features of LLM to design serving systems effectively. In this context, we propose two potential research directions and offer possible solutions to stimulate further scholarly discourse. Findings 4 and 6 elucidate that the queue's waiting time markedly impacts the serving system, primarily due to the unpredictable request distribution. In LLM, the arrival time distribution and iteration count remain indeterminate. Hence, researchers may formulate a multi-feedback queue scheduler for handling unknown arrival times [31] and develop a compact model consistent with LLM to forecast request iteration counts [32], facilitating batch processing of requests with analogous iteration counts to minimize latency. Finding 10 suggests that the design and use of meta-operations should align with model characteristics, offering insights for researchers in designing new meta-operations for LLM serving systems. This prompted researchers to develop new meta-operations that couple multiple iterations in decoding states with a iteration in prefill states, utilizing weight data reuse to reduce memory access and thereby improve system throughput [33].

In future work, we will augment the characterization of batching behavior within the LLM serving system and undertake a more profound exploration based on the aforementioned two research directions.

7.5. Multi-GPU platforms

In existing DNN serving system designs, the batching module and the inference engine module are independently designed, encompassing serving systems such as Triton, DVABatch, and Orca. In contemporary DNN serving systems, tensor parallelism and pipeline parallelism are commonly utilized for inference services across multiple GPUs [34], primarily within the confines of the inference engine module. While this paper focuses on the batching system, insights into the design of the inference engine remain beneficial. For instance, in Section 5, we highlight that the selection of model slicing positions is associated with the model structure, which can provide guidance for the design of the pipeline stages of the pipeline parallelism paradigm in the execution engine layer. Furthermore, this work clarifies existing DNN serving system designs, laying the groundwork for future collaborative designs between the batching system and the inference engine. For example, considering a machine equipped with two GPU cards (GPU_1 and GPU_2) using the pipeline parallelism paradigm—where GPU_1 handles the front portion of the model and GPU_2 manages the rear portion. Assuming two batches of varying sizes, A and B (with A having a larger batch size than B), arrive sequentially. Orca would first execute A on GPU_1 (front portion of the model) followed by A on GPU_2 (rear portion) while simultaneously processing B on GPU_1 (front portion). Due to A's larger batch size compared to B, a bubble occurs on GPU_1. If the batching system layer can perceive that the execution engine layer uses the pipeline parallelism paradigm, it can reduce the occurrence of bubbles by dividing the requests into finer granularities.

8. Related work

Dynamic Batching. In the realm of model training, researchers focused on adjusting batch sizes to strike a balance between training efficiency and model generalization [35–37]. In the training phase, all input data is available, allowing for the efficient collection of multiple samples without latency. However, in the inference phase, since the ML serving systems receive input at different times, and batching system needs to balance latency and throughput, which poses challenges. Therefore, our paper focuses on analyzing batching techniques in the inference phase.

Regarding batching techniques during model inference, there are three primary types, as delineated in prior studies [7,38]: static batching, dynamic batching, and application-specific batching. Static batching, as exemplified by systems such as Triton and TensorFlow-Serving, relied on two critical hyperparameters: the model-allowed maximum batch size and the time window, which govern request batching behavior. In a static batching system, new batches can only be executed after the current batch inference is done, causing longer request wait times.

Therefore, researchers have proposed dynamic batching, allowing batch size modification during the inference process, with some typical serving systems including LazyBatching [7] and DVAbatch [3]. In dynamic batching techniques, models are sliced into different subgraphs to support the addition of new requests and the early exit of old requests. LazyBatching slices the model at the granularity of operators and employs a QoS-aware slack time prediction algorithm to delay request processing, creating larger batches. DVABatch, built upon LazyBatching, uses subgraphs as the slice granularity and introduces stretch and split operations to adapt to different application scenarios.

Furthermore, there have been batching techniques tailored for specific applications. As the number of iterations varies for different requests in the generation model, Orca [39] introduces iteration-level batching, i.e., considering whether to incorporate new iterations or early exit the iteration from the batch. In applications involving diverse sequence lengths, researchers explored strategies for concatenating requests into larger inputs [40] or adopting finer-grained grouping techniques [41] to improve performance.

Despite numerous DNN serving system batching techniques, their applicability and operational contexts remain unclear. Additionally, these methods often target specific modules, such as static batching for request batching module and dynamic batching for stage reorchestrating modules. This work delivers a holistic assessment of the influence of parameter configurations, model slicing strategies, and stage reorchestrating strategies on batching serving systems across diverse models and workloads. To the best of our knowledge, this is the first study that comprehensively evaluates and analyzes DNN batching serving system.

Serving Systems. In serving systems, batch processing was often considered in conjunction with factors such as resource allocation and QoS. Various approaches were devised to employ adaptive strategies, enhancing efficiency and ensuring equitable resource distribution to fulfill users' inference demands. DyBatch [42] adjusted batch sizes based on device workloads and task requisites to uphold fairness. Nanily [43] dynamically allocated computational resources, aiming to meet QoS requirements while optimizing resource utilization. Ebird [9, 44] excelled in performance maximization across fluctuating workloads. In the design of serving systems, batching techniques typically need to be collaboratively designed with other optimization techniques. This study contributes to a better understanding of batching techniques for developers and lays the foundation for designing superior serving systems.

9. Conclusion

Optimizing and deploying DNN serving systems lay in understanding the behavior of batching throughout the entire system. In this paper, we characterized the behavior of the request batching module, model slicing module, and stage reorchestrating module, in deep neural network batching systems on GPUs, by using three representative models. Based on experimental results, several meaningful insights and findings are provided for future research to further enhance deep learning serving systems.

CRediT authorship contribution statement

Feng Yu: Writing – original draft, Methodology, Conceptualization. **Hao Zhang:** Software, Formal analysis. **Ao Chen:** Validation, Investigation. **Xueying Wang:** Validation, Formal analysis. **Xiaoxia Liang:** Software, Investigation. **Sheng Wang:** Validation, Investigation. **Guangli Li:** Project administration, Methodology, Conceptualization. **Huimin Cui:** Supervision, Methodology. **Xiaobing Feng:** Supervision, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (62232015, 62090024, 62302479), the China Postdoctoral Science Foundation (2023M733566), and the Innovation Funding of ICT, CAS, China (E361010).

References

[1] N. Inc., NVIDIA triton inference server, 2023, URL https://docs.nvidia.com/deeplearning/triton-inference-server/, Accessed: August, 2023.

[2] C. Olston, N. Fiedel, K. Gorovoy, J. Harmsen, L. Lao, F. Li, V. Rajashekhar, S. Ramesh, J. Soyke, Tensorflow-serving: Flexible, high-performance ml serving, 2017, arXiv preprint arXiv:1712.06139.

[3] W. Cui, H. Zhao, Q. Chen, H. Wei, Z. Li, D. Zeng, C. Li, M. Guo, DVABatch: Diversity-aware Multi-Entry Multi-Exit batching for efficient processing of DNN services on GPUs, in: 2022 USENIX Annual Technical Conference, 2022, pp. 183–198.

[4] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[5] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.

[6] A. Chaurasia, E. Culurciello, Linknet: Exploiting encoder representations for efficient semantic segmentation, 2017, arXiv preprint arXiv:1707.03718.

[7] Y. Choi, Y. Kim, M. Rhu, Lazy batching: An SLA-aware batching system for cloud machine learning inference, in: International Symposium on High-Performance Computer Architecture, 2021, pp. 493–506.

[8] X. Li, G. Zhang, H.H. Huang, Z. Wang, W. Zheng, Performance analysis of GPU-based convolutional neural networks, in: 2016 45th International Conference on Parallel Processing, ICPP, IEEE, 2016, pp. 67–76.

[9] W. Cui, M. Wei, Q. Chen, X. Tang, J. Leng, L. Li, M. Guo, Ebird: Elastic batch for improving responsiveness and throughput of deep learning services, in: International Conference on Computer Design, 2019, pp. 497–505.

[10] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze, et al., TVM: An automated End-to-End optimizing compiler for deep learning, in: 13th USENIX Symposium on Operating Systems Design and Implementation, 2018, pp. 578–594.

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019).

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM International Conference on Multimedia, 2014, pp. 675–678.

[13] N. Inc., NVIDIA tensorRT, 2021, URL: https://developer.nvidia.com/tensorrt, Accessed on 2023-09-04.

[14] N. inc, Triton client libraries and examples, 2021, URL: https://github.com/triton-inference-server/client, Accessed on 2023-09-04.

[15] V.J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, et al., Mlperf inference benchmark, in: International Symposium on Computer Architecture, 2020, pp. 446–459.

[16] D. Yastremsky, Maximizing deep learning infer-ence performance with NVIDIA model analyzer, 2020, URL https://developer.nvidia.com/blog/maximizing-deep-learning-inference-performance-with-nvidia-model-analyzer, Accessed: August, 2023.

[17] S.V. Saavedra, A.L. Uribe, Google cloud vision and its application in image processing using a raspberry Pi, in: Colombian Conference on Computing, Springer, 2022, pp. 102–113.

[18] D. Avinash, J.A. Kumar, R. Chandansingh, Use of AI in cloud-based certificate authentication for travel concession, in: Mobile Computing and Sustainable Informatics: Proceedings of ICMCSI 2023, Springer, 2023, pp. 349–361.

[19] A. Satapathi, A. Mishra, Build a multilanguage text translator using azure cognitive services, in: Developing Cloud-Native Solutions with Microsoft Azure and. NET: Build Highly Scalable Solutions for the Enterprise, Springer, 2022, pp. 231–248.

[20] H.-M. Sormunen, Enhancing customer feedback processing with machine learning in Microsoft Azure, 2022.

[21] M. Singh, Single stage facial recognition based on YOLOv5, in: 2022 International Conference on INnovations in Intelligent SysTems and Applications, INISTA, IEEE, 2022, pp. 1–6.

[22] T. Leonor Estévez Dorantes, D. Bertani Hernández, A. León Reyes, C. Elena Miranda Medina, Development of a powerful facial recognition system through an API using ESP32-Cam and amazon rekognition service as tools offered by industry 5.0, in: 2022 the 5th International Conference on Machine Vision and Applications, ICMVA, 2022, pp. 76–81.

[23] S. Marcel, Y. Rodriguez, Torchvision the machine-vision package of torch, in: Proceedings of the 18th ACM International Conference on Multimedia, 2010, pp. 1485–1488.

[24] S.M. Jain, Hugging face, in: Introduction to Transformers for NLP: With the Hugging Face Library and Models to Solve Problems, Springer, 2022, pp. 51–67.

[25] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N.R. Devanur, G.R. Ganger, P.B. Gibbons, M. Zaharia, PipeDream: Generalized pipeline parallelism for DNN training, in: Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP '19, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450368735, 2019, pp. 1–15, http://dx.doi.org/10.1145/3341301.3359646.

[26] A. Ali, R. Pinciroli, F. Yan, E. Smirni, Batch: machine learning inference serving on serverless platforms with adaptive batching, in: International Conference for High Performance Computing, Networking, Storage and Analysis, 2020, pp. 1–15.

[27] F. Yan, O. Ruwase, Y. He, E. Smirni, SERF: Efficient scheduling for fast deep neural network serving via judicious parallelism, in: SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2016, pp. 300–311.

[28] M.F. Neuts, A versatile Markovian point process, J. Appl. Probab. 16 (4) (1979) 764–779.
[29] K. Hong, G. Dai, J. Xu, Q. Mao, X. Li, J. Liu, K. Chen, H. Dong, Y. Wang, FlashDecoding++: Faster large language model inference on GPUs, 2023, arXiv preprint arXiv:2311.01282.
[30] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C.H. Yu, J. Gonzalez, H. Zhang, I. Stoica, Efficient memory management for large language model serving with pagedattention, in: Proceedings of the 29th Symposium on Operating Systems Principles, 2023, pp. 611–626.
[31] B. Wu, Y. Zhong, Z. Zhang, G. Huang, X. Liu, X. Jin, Fast distributed inference serving for large language models, 2023, arXiv preprint arXiv:2305.05920.
[32] Q. Su, C. Giannoula, G. Pekhimenko, The synergy of speculative decoding and batching in serving large language models, 2023, arXiv preprint arXiv:2310.18813.
[33] A. Agrawal, A. Panwar, J. Mohan, N. Kwatra, B.S. Gulavani, R. Ramjee, SARATHI: Efficient LLM inference by piggybacking decodes with chunked prefills, 2023, arXiv preprint arXiv:2308.16369.
[34] X. Miao, C. Shi, J. Duan, X. Xi, D. Lin, B. Cui, Z. Jia, SpotServe: Serving generative large language models on preemptible instances, 2023, arXiv preprint arXiv:2311.15566.
[35] A. Devarakonda, M. Naumov, M. Garland, Adabatch: Adaptive batch sizes for training deep neural networks, 2017, arXiv preprint arXiv:1712.02029.
[36] A. Lydia, S. Francis, Adagrad—an optimizer for stochastic gradient descent, Int. J. Inf. Comput. Sci. 6 (5) (2019) 566–568.
[37] M. Zaheer, S. Reddi, D. Sachan, S. Kale, S. Kumar, Adaptive methods for nonconvex optimization, Adv. Neural Inf. Process. Syst. 31 (2018).
[38] E.L. Cade Daniel, R. Liaw, How continuous batching enables 23x throughput in LLM inference while reducing p50 latency, 2023, URL: https://www.anyscale.com/blog/continuous-batching-llm-inference, Accessed on 2023-09-04.
[39] G.-I. Yu, J.S. Jeong, G.-W. Kim, S. Kim, B.-G. Chun, Orca: A distributed serving system for transformer-based generative models, in: USENIX Symposium on Operating Systems Design and Implementation, 2022, pp. 521–538.
[40] B. Fu, F. Chen, P. Li, D. Zeng, TCB: Accelerating transformer inference services with request concatenation, in: Proceedings of the 51st International Conference on Parallel Processing, 2022, pp. 1–11.
[41] Y. Zhai, C. Jiang, L. Wang, X. Jia, S. Zhang, Z. Chen, X. Liu, Y. Zhu, ByteTransformer: A high-performance transformer boosted for variable-length inputs, in: International Parallel and Distributed Processing Symposium, 2023, pp. 344–355.
[42] S. Zhang, W. Li, C. Wang, Z. Tari, A.Y. Zomaya, DyBatch: Efficient batching and fair scheduling for deep learning inference on time-sharing devices, in: International Symposium on Cluster, Cloud and Internet Computing, 2020, pp. 609–618.
[43] X. Tang, P. Wang, Q. Liu, W. Wang, J. Han, Nanily: A qos-aware scheduling for dnn inference workload in clouds, in: International Conference on High Performance Computing and Communications, 2019, pp. 2395–2402.
[44] W. Cui, Q. Chen, H. Zhao, M. Wei, X. Tang, M. Guo, E2bird: Enhanced elastic batch for improving responsiveness and throughput of deep learning services, IEEE Trans. Parallel Distrib. Syst. 32 (6) (2020) 1307–1321.

BenchCouncil Transactions on Benchmarks, Standards and Evaluations 3 (2023) 100152

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/

Full length article

AIGCBench: Comprehensive evaluation of image-to-video content generated by AI

Fanda Fan [a,b], Chunjie Luo [a], Wanling Gao [a], Jianfeng Zhan [a,b,*]

[a] Research Center for Advanced Computer Systems, State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, China
[b] University of Chinese Academy of Sciences, China

ARTICLE INFO

Keywords:
Artificial intelligence generated content
Video generation
Image-to-video benchmark
Diffusion model
Multimodal AI

ABSTRACT

The burgeoning field of Artificial Intelligence Generated Content (AIGC) is witnessing rapid advancements, particularly in video generation. This paper introduces AIGCBench, a pioneering comprehensive and scalable benchmark designed to evaluate a variety of video generation tasks, with a primary focus on Image-to-Video (I2V) generation. AIGCBench tackles the limitations of existing benchmarks, which suffer from a lack of diverse datasets, by including a varied and open-domain image–text dataset that evaluates different state-of-the-art algorithms under equivalent conditions. We employ a novel text combiner and GPT-4 to create rich text prompts, which are then used to generate images via advanced Text-to-Image models. To establish a unified evaluation framework for video generation tasks, our benchmark includes 11 metrics spanning four dimensions to assess algorithm performance. These dimensions are control-video alignment, motion effects, temporal consistency, and video quality. These metrics are both reference video-based and video-free, ensuring a comprehensive evaluation strategy. The evaluation standard proposed correlates well with human judgment, providing insights into the strengths and weaknesses of current I2V algorithms. The findings from our extensive experiments aim to stimulate further research and development in the I2V field. AIGCBench represents a significant step toward creating standardized benchmarks for the broader AIGC landscape, proposing an adaptable and equitable framework for future assessments of video generation tasks. We have open-sourced the dataset and evaluation code on the project website: https://www.benchcouncil.org/AIGCBench.

1. Introduction

Artificial Intelligence Generated Content (AIGC) encompasses a wide array of applications that leverage AI technologies to automate the creation or editing of content across different media types, such as text, images, audio, and video. With the rapid advancement of diffusion models [1–5] and multimodal AI technologies [6], the AIGC field is experiencing considerable and rapid progress. The explosive growth of AIGC has made its evaluation and benchmarking an urgent task.

A representative application of AIGC is video generation [7–11]. Current video generation includes Text-to-Video (T2V), Image-to-Video (I2V), Video-to-Video (V2V), as well as a few other works that utilize additional information such as depth [10], pose [12], trajectory [13], and frequency [14] to generate videos. Among these, T2V and I2V are the two most mainstream tasks at present. Early video generation primarily used text prompts to generate videos and achieved good results [7,8,15–19]. However, using text alone makes it difficult to depict the specific scenes that users want. Recently, I2V has ignited the AIGC community. The I2V task refers to the generation of a dynamic, moving video sequence based on a static input image and is usually accompanied by a text prompt.[1] Compared to T2V, I2V can better define the content of video generation, achieving excellent results in many scenarios such as film, e-commerce advertising, and micro-animation effects.

While benchmarks for the T2V task have seen notable progress [20–22], benchmarks for the I2V task have scarcely advanced. Previous efforts like Latent Flow Diffusion Models (LFDM) [23] and CATER-GEN [24] were tested under domain-specific video scenarios. VideoCrafter [25] and I2VGen-XL [26] only utilized visual comparisons for the I2V task. Seer [27] and Stable Video Diffusion (SVD) [28] employed video–text datasets and utilized a few metrics that require reference videos. Existing I2V benchmarks suffer from (1) a lack of

* Corresponding author at: Research Center for Advanced Computer Systems, State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, China.
E-mail addresses: fanfanda@ict.ac.cn (F. Fan), zhanjianfeng@ict.ac.cn (J. Zhan).

[1] However, the community often refers to it as Image-to-Video, rather than Text-Image-to-Video.
[2] Open-domain images refer to images that cover a wide variety of subjects or topics without specific restrictions on the content or category.
[3] Here, equivalent conditions refer to using the same evaluation dataset and assessment dimensions for all video generation tasks.

https://doi.org/10.1016/j.tbench.2024.100152
Received 3 January 2024; Received in revised form 17 January 2024; Accepted 23 January 2024
Available online 26 January 2024

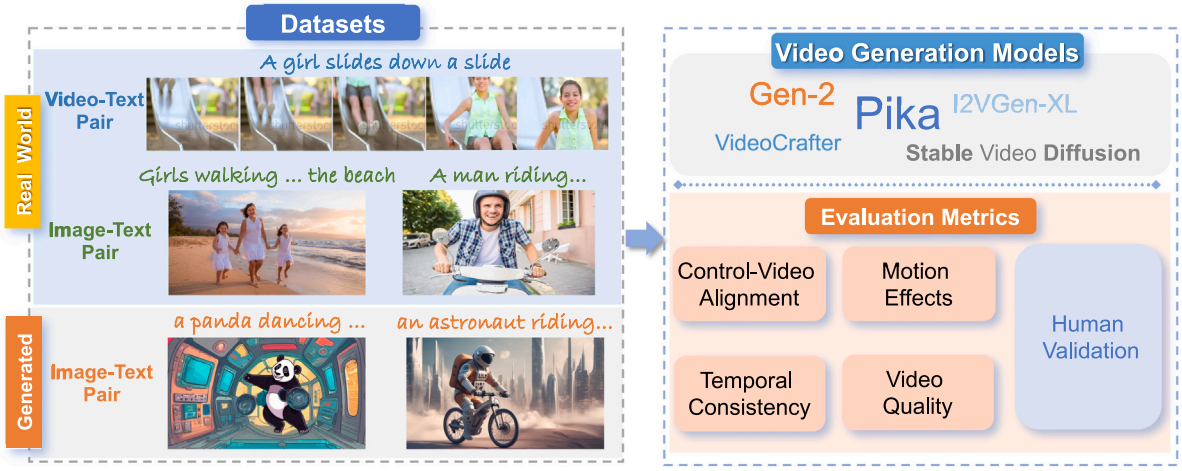


Fig. 1. Illustration of our AIGCBench. Our AIGCBench is divided into three modules: the evaluation dataset, the evaluation metrics, and the video generation models to be assessed. Our benchmark encompasses two types of datasets: video–text and image–text datasets. To construct a more comprehensive evaluation dataset, we expand the image–text dataset by our generation pipeline. Additionally, for a thorough evaluation of video generation models, we introduce a set of evaluation metrics comprising 11 metrics across four dimensions. These metrics include both reference video-based and reference video-free metrics, making full use of the benchmark we propose. We also adopted human validation to confirm the rationality of the evaluation standards we proposed.

diverse, open-domain images[2] with various subjects and styles to test the efficacy of different state-of-the-art algorithms; (2) an absence of a unified consensus on which evaluation metrics should be used to assess the final generated results. From the perspective of [29], these two shortcomings hinder the capability of capturing stakeholders' concerns and interests, while also failing to construct equivalent evaluation conditions.[3]

To address this gap, we present AIGCBench, a unified benchmark for video generation tasks. AIGCBench aims to encapsulate all mainstream video generation tasks, such as T2V, I2V, V2V, and the synthesis of video from additional modalities like depth, pose, trajectory, and frequency. We present an overview of AIGCBench in Fig. 1. Our AIGCBench is divided into three modules: the evaluation dataset, the evaluation metrics, and the video generation models to be assessed. Considering the high relevance and interconnectivity[4] of video generation tasks, our AIGCBench can enable the comparison of different algorithms under equivalent evaluation conditions. This allows for an analysis of the strengths and weaknesses of different state-of-the-art video generation algorithms, thereby aiding progress in the field of video generation. In the first version of our AIGCBench, we address the current lack of a reasonable benchmark for I2V tasks by providing a thorough evaluation for them. In subsequent versions, we plan to include more video generation tasks and place them under equivalent evaluation conditions for a fair comparison.

Recognizing the limitations of existing benchmarks, AIGCBench is engineered to meet the diverse demands of users looking to animate a broad array of static images. Where previous benchmarks have fallen short, not fully accommodating the expansive range of images users might choose to animate – such as a blue dragon skateboarding in Times Square – AIGCBench rises to the challenge. We address this by deploying a text combiner to generate a rich assortment of text prompts that span a multitude of subjects, behaviors, backgrounds, and artistic styles. Further refining the creative process, we employ the advanced capabilities of GPT-4 [30] to enhance the text prompts, rendering them more vivid and intricate. These detailed prompts then guide the generation of images through state-of-the-art Text-to-Image diffusion models.

[4] The interconnectivity arises because some algorithms have the capability to perform multiple types of video generation tasks.

By judiciously blending video–text and image–text datasets, along with our generated image–text pairs, AIGCBench ensures a robust and comprehensive evaluation of an array of I2V algorithms, thus addressing the first major shortcoming identified in existing benchmarks.

To establish a comprehensive and standardized set of evaluation metrics for video generation tasks that cater to mainstream tasks such as T2V and I2V, our AIGCBench evaluates four critical dimensions: control-video alignment, motion effects, temporal consistency, and video quality, thereby capturing every aspect of video generation. This integrated framework combines metrics that are both reference video-based and video-free metrics, enhancing the benchmark's rigor without exclusively relying on video–text datasets or image–text datasets alone. We strengthen this approach by incorporating image–text datasets into our evaluations, which allows us to assess content beyond the scope of existing video–text datasets and add reference video-free metrics for assessment. Considering the complexity and diversity of tasks, we believe that the evaluation metrics should cover at least these four aspects. For each aspect, we aim to use both reference video-based and video-free metrics. After satisfying these categorizations, the benefits of increasing the number of metrics become marginal, while it is insufficient without covering these aspects. The experimental results demonstrate that our evaluation standard correlates well with human ratings, confirming its effectiveness. Following a thorough evaluation, we present the strengths and weaknesses of each model, alongside several insightful findings, in hopes of spurring discussions that advance the I2V field.

Our contributions are as follows:

1. We introduce AIGCBench, a benchmark for comprehensive evaluation of diverse video generation tasks, with an initial focus on Image-to-Video (I2V) generation and a commitment to placing these models under equivalent evaluation conditions for fair comparison.
2. We extend our image–text dataset using a text combiner and GPT-4, complemented by state-of-the-art Text-to-Image models to generate high-quality images, enabling a deeper evaluation of I2V algorithm performance;
3. We evaluate I2V algorithms comprehensively using both reference video-based and video-free metrics across four aspects and verify the validity of our proposed evaluation standard with human judgment;

Table 1
Compare the features of our AIGCBench with other I2V benchmarks. ✗ and ✓ indicate whether the benchmark includes the features listed in the respective columns. Video-based metrics, which use reference videos, contrast with video-free metrics that do not. Considering the difficulty of the evaluation, we are not counting the sample numbers for domain-specific benchmarks [23,24,27].

Benchmark	Open-Domain	Video-Text Pairs	Image-Text Pairs	Generated Dataset	#Samples	Metric Types	# Metrics
LFDM Eval [23]	✗	✓	✗	✗	-	Video-based	3
CATER-GEN [24]	✗	✓	✓	✓	-	Video-based & Video-free	7
Seer Eval [27]	✗	✓	✗	✗	-	Video-based	2
VideoCrafter Eval [25]	✓	✓	✓	✗	-	-	-
I2VGen-XL Eval [26]	✓	✓	✓	✗	-	-	-
SVD Eval [28]	✓	✓	✓	✗	900	Video-based	5
AnimateBench [31]	✓	✗	✗	✓	105	Video-free	2
AIGCBench (Ours)	✓	✓	✓	✓	3928	Video-based & Video-free	11

4. We offer several insightful findings to aid the better development of the I2V community.

2. Background and related work

Current video generation primarily encompasses two major tasks: Text-to-Video (T2V) and Image-to-Video (I2V). Given the high relevance of T2V tasks to I2V tasks, we discuss video generation models, with a particular focus on I2V models, in Section 2.1. We will introduce related benchmarks for T2V in Section 2.2 and describe the existing benchmarks for I2V in Section 2.3.

2.1. Video generation models

Thanks to the development of diffusion models [1–5] and multimodal techniques [6], video generation algorithms are becoming increasingly sophisticated. Early video generation was primarily based on text-to-video approaches [7,8,10,15,17–19,32–36]. Most of the work is based on diffusion models [2–5,16], with some being transformer-based [15,33]. They all rely on extensive video–text or image–text datasets to train scalable models. However, considering that using only text can make it challenging to intuitively depict the video scenes users want to generate, image-to-video has started to gain popularity in the video generation community.

Seer [27] introduced an approach for I2V tasks that combines the conditional image latent with a noisy latent, utilizing causal attention within the temporal component of a 3D U-Net [37]. VideoComposer [38] concatenated image embedding with image style embedding to preserve the initial image information. Recently, VideoCrafter [25] encoded the image prompt through a lightweight image encoder and fed it into the cross-attention layer. Similarly, I2VGen-XL [26] not only merges the image latent with the noisy latent at the input layer but also employs a global encoder that extracts the image CLIP feature into the video latent diffusion model (VLDM). Stable video diffusion [28] is an extension of a pretrained image-based diffusion model [39]. It is trained through three stages: text-to-image pretraining, video pretraining, and high-quality video fine-tuning. Emu Video [40] identified critical design decisions, such as adjusted noise schedules for diffusion and multi-stage training, which enabled the generation of high-quality videos without requiring a deep cascade of models as in prior work. Beyond academic research, the video generation results from industry players like Pika [9] and Gen2 [10] are also quite impressive. All of these I2V algorithms are based on video diffusion models, and the majority leverage the parameter priors from image diffusion models to aid in the convergence of video models.

To evaluate state-of-the-art I2V models, we have reviewed three open-source works in this paper: VideoCrafter [25], I2VGen-XL [26], and Stable Video Diffusion [28], as well as two closed-source industry efforts, Pika [9] and Gen2 [10]. These currently represent the five most influential works in the video generation community, and we will briefly introduce their experimental parameters in Section 5.1.

2.2. Benchmarks for text-to-video generation

The FETV benchmark [20] conducts a comprehensive manual evaluation of representative T2V models and reveals their strengths and weaknesses in handling a diverse range of text prompts from multiple perspectives. EvalCrafter [21] starts by creating a new set of prompts for T2V generation with the assistance of a large language model, ensuring that the prompts are representative of actual user queries. EvalCrafter's benchmarks [21] are meticulously designed to evaluate generated videos from several critical dimensions: visual quality, content accuracy, motion dynamics, and the alignment between generated video content and the original text captions. VBench [22] has created 16 distinct evaluation dimensions, each with specialized prompts for precise assessment.

The task of T2V differs from I2V, as videos generated from the same text can vary widely, making it less suitable for evaluation metrics that require a reference video. For T2V tasks, the results generated by different models for the same text prompt can be quite dissimilar. However, for I2V tasks, since the image imposes certain constraints, the variation in results produced by different models is generally not as pronounced. This allows us to conduct a comprehensive evaluation of different Image-to-Video (I2V) algorithms on video–text datasets using evaluation metrics that are based on reference videos. Our AIGCBench draws on these T2V benchmarks but differs from them in several respects: (1). We need to collect or construct images for the I2V model's input, which requires considering the comprehensiveness of both the text prompt set and the image set. (2). Although our evaluations are similar to those of the T2V task in terms of the dimensions assessed, we need to employ new evaluation standards due to the differences between T2V and I2V tasks.

2.3. Benchmarks for image-to-video generation

Domain-specific I2V benchmark. LFDM Eval [23] is evaluated on facial expression and human action datasets, employing just a few evaluation metrics to gauge the quality of video generation. The CATER-GEN [24] benchmark uses predefined 3D objects and specific initial images for testing the quality of videos that depict the motion of 3D objects. Nonetheless, neither LFDM Eval [23] nor the CATER-GEN [24] benchmark is appropriate for evaluating video generation in open-domain scenarios.

Open-domain I2V benchmark. The open-domain I2V benchmark is currently based on two main types of evaluation data: video–text and image–text datasets. Seer [27] and SVD [28] have utilized video–text datasets and employed a limited number of metrics that require reference videos for evaluation. VideoCrafter [25] and I2VGen-XL [26]

have used image–text datasets and relied solely on visual comparisons. Very recently, AnimateBench [31] was released for the purpose of evaluating I2V tasks. They also generated images using Text-to-Image models. However, they were limited by a small number of text prompts and a limited collection of images. At the same time, there is a lack of comprehensive evaluation metrics. Both are constrained by limited evaluation datasets and an incomplete set of assessment metrics. This leads to the evaluation datasets not being representative of all stakeholders' concerns and interests, and there is also a lack of a unified and comprehensive consensus on evaluation. In this paper, we expand the image–text dataset using state-of-the-art Text-to-Image models. To ensure the complexity of the generated text prompts, we generate prompts through the combinatorial traversal of four metatypes and enhance them with the capabilities of large language models. We compare our AIGCBench with other I2V benchmarks in Table 1.

Generating image–text dataset. While most benchmarks gather datasets from the real world, CATER-GEN [24] constructs datasets using a limited set of text prompts for specific object movement scenarios. Very recently, AnimateBench [31] utilized a limited number of manually designed text prompts and also employed Text-to-Image models to generate images. However, this approach is constrained by the simplicity of the text combinations and the limited diversity of the images. Our generation pipeline uses a text combiner to randomly generate text prompts and incorporates GPT-4 [30] to enrich the content. Simultaneously, we filter the generated results to select high-quality image–text pairs.

3. AIGCBench: Establishing the image-to-video generation benchmark

The framework of our AIGCBench is shown in Fig. 1. Our AIGCBench framework comprises three components: the evaluation dataset, the video generation models to be assessed, and the evaluation metrics. To construct a comprehensive benchmark, we evaluate I2V models using two types of datasets: video–text and image–text. For the image–text dataset, we utilize evaluation metrics that do not require reference videos. In this section, we will introduce how we collected the evaluation datasets, in Section 4 we present the evaluation criteria we have established, and in Section 5.1 we provide a brief introduction to the video generation models to be evaluated.

3.1. Collect dataset from real-world

Video–text pairs. The WebVid-10M [41] dataset is a substantial collection specifically designed to aid in the development and training of AI models for video understanding tasks. It consists of approximately 10 million video–text pairs, making it one of the larger datasets available for this type of research. Considering that video generation is time-consuming, we have sampled 1000 videos from the validation set of the WebVid10M [41] dataset based on subtype for evaluation purposes.

Image–text pairs. The LAION-5B [42] dataset is a large-scale, open dataset consisting of around 5,85 billion image–text pairs. It was created to facilitate research in computer vision and machine learning, specifically in areas such as multi-modal language-vision models, Text-to-Image generation, and more (e.g. CLIP [6], DALL-E [43]). LAION-Aesthetics is a subset from LAION-5B [42] with high visual quality. We randomly sampled 925 image–text pairs from the LAION-Aesthetics dataset to serve as a reference for video-free evaluation metrics.

3.2. Generated image–text pairs

Using only real-world datasets is insufficient. Users often input images and text generated by designers or T2I (Text-to-Image) models to create videos. This includes certain image–text pairs that cannot be sampled in the real world. To bridge this gap, we propose a T2I generation pipeline. As shown in Fig. 2, we provide an overview of our generation pipeline above and present some generated cases below.

3.2.1. Text combiner

To generate as diverse text prompts as possible, we construct text templates based on four types: subject, behavior, background, and image style. We then generate a list of 3000 text prompts randomly by following the template: **subject** + **behavior** + **background**, in the **image style** style. We have listed some examples:

1. Subject: *a dragon, a knight, an alien, a robot, a panda, a nymph*;
2. Behavior: *riding a bike, fight a monster, searching for a treasure, dancing, solving a puzzle*;
3. Background: *in a forest, in a futuristic city, in a space station, in an old western town at high noon*;
4. Image style: *oil painting, water color, cartoon, realistic, Van Gogh, Picasso*.

We have compiled our text corpus from high-frequency words often entered by users in the T2I community of Civit AI [44], along with some potentially valuable text prompts. Considering the flexibility of our generation pipeline, our benchmark is scalable. Subsequently, we can update and iterate on the versions of our text corpus.

3.2.2. Optimizing text prompts

Although utilizing text templates with various text corpora can generate reasonable images, it might lead to poor diversity in the generated images, which is not conducive to evaluating I2V tasks. We leverage the capabilities of the GPT-4 model [30], using the prompt "*make the content more vivid and rich*" to optimize the texts generated from templates.

3.2.3. Generate images and filter

To generate high-quality images based on the generated texts, we have employed the best Text-to-Image (T2I) model available to date — the Stable Diffusion model [39]. The Stable Diffusion model [39] is particularly notable for its ability to create high-quality and coherent images that closely match the style and content described by the input text prompts. We utilized the latest xl-base T2I model released by their community. Considering that the I2V model is primarily trained with an aspect ratio of 16:9, we used a height of 720 and a width of 1280 to generate images.

In order to select high-quality image–text pairs, we filtered out the top 2003 high-quality image–text pairs based on the automatic metrics from the T2I-CompBench [45]. Some examples generated by our pipeline can be seen in the lower half of Fig. 2.

4. Evaluation metrics

Our evaluation dataset includes both video–text and image–text datasets. To conduct a comprehensive evaluation, we employ two types of assessment metrics: one that requires reference videos and another that does not. In addition, we considered previous Text-to-Video benchmarks [20–22] and have integrated to propose an evaluation standard suitable for the Image-to-Video (I2V) task, covering both types of dataset. We assess the performance of different I2V models from four aspects: control-video alignment.[5] motion effects, temporal consistency, and overall video quality[6]. Considering that videos generated by different algorithms have varying numbers of frames, for a standardized evaluation, we adopt the approach of extracting the first 16 frames, unless otherwise specified.

[5] "control" refers to the input signals from the user, such as text, images, and other forms of control signals.

[6] The code is available at https://github.com/BenchCouncil/AIGCBench.

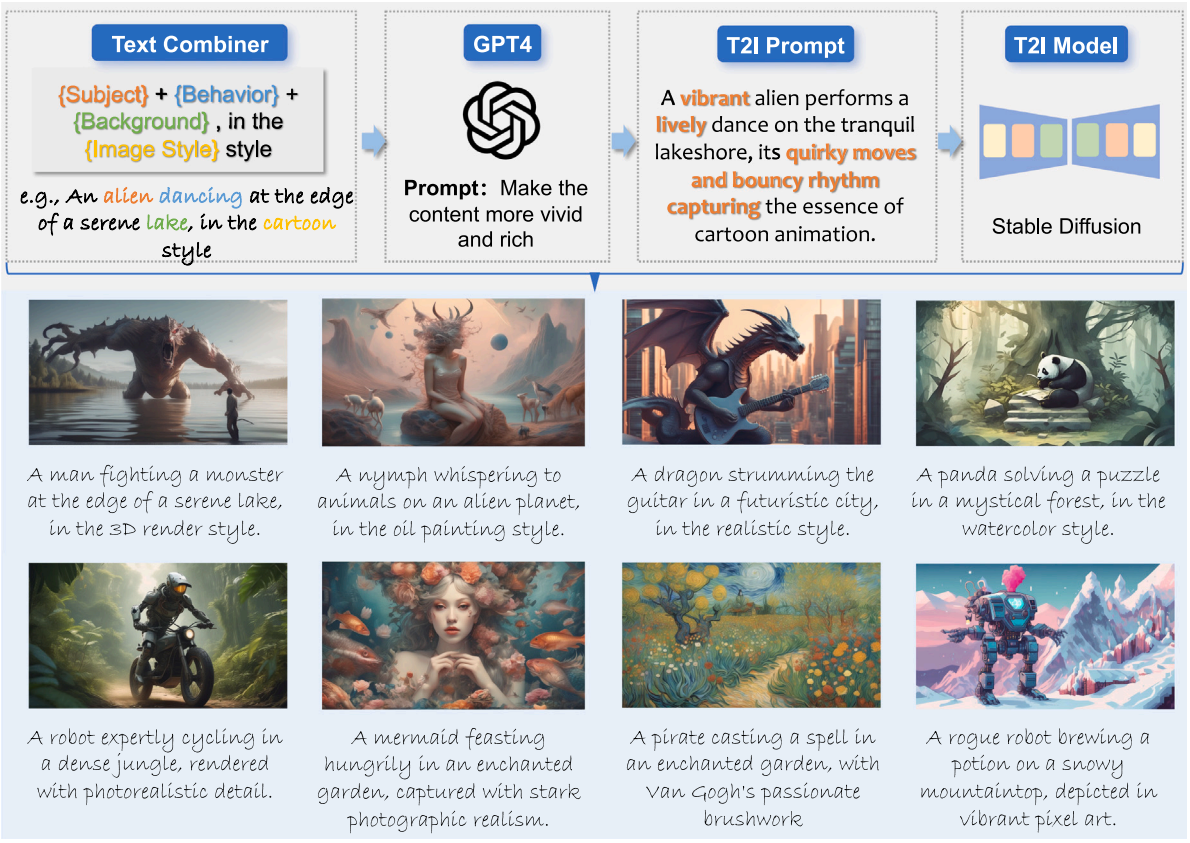


Fig. 2. Image–text dataset generation pipeline and results. Above: An overview of our T2I generation pipeline is presented. Below: Eight generated cases are showcased, with the original text produced by the text combiner displayed beneath each image.

4.1. Control-video alignment

The control-video alignment measures the degree of alignment between the user's input control signals, such as text and images, and the generated video. Considering that current video generation tasks primarily involve two types of inputs—a starting image and a text prompt—we introduce two evaluation metrics in the first version of our benchmark: **image fidelity** and **text-video alignment**. The image fidelity metric evaluates how similar the generated video frames are to the image input into the I2V model, especially the first frame. To assess fidelity, for the first frame of the generated video, we use metrics such as Mean Squared Error (MSE) and Structural Similarity Index Measure (SSIM) [46] to calculate the degree of preservation of the first frame. For all frames of the video, we calculate the similarity of CLIP (Contrastive Language–Image Pre-training) [6] embeddings between the input image and each frame of the generated video. We use MSE (First), SSIM (First), and Image-GenVideo CLIP to represent these three evaluation metrics, respectively.

Considering that the I2V models we evaluate also take text as input, we need to assess whether the generated videos are relevant to the input text. For the generated videos, we use CLIP [6] to calculate the similarity between the input text and the generated video results. We assume that the videos in the video–text dataset are consistent with the textual descriptions. For the video–text dataset, we use the keyframes from the reference videos and the generated videos to compute the CLIP [6] similarity. Considering that the text typically describes high-level semantics and that the generated videos may not correspond perfectly with the original videos, we uniformly sample four keyframes for comparison. We use GenVideo-Text Clip and GenVideo-RefVideo CLIP (Keyframes) to represent these two evaluation metrics, respectively.

4.2. Motion effects

Motion effects primarily evaluate whether the amplitude of the motion in the generated video is significant and whether the movements are reasonable. As for the amplitude of the motion, we follow the [21, 22] and use a pretrained optical flow estimation method, RAFT [47], to calculate the flow score between adjacent frames of the generated video, with the final average value representing the magnitude of the motion effects. We use the square average of the predicted values from adjacent frames to represent the motion dynamics of the video, with higher values indicating stronger motion effects. Considering that there are some bad cases in video generation, we set a threshold where the square average value must be less than 10 to filter out these bad cases. For the video–text dataset, we have real videos corresponding to the text. We measure the reasonableness of the generated motion effects by calculating the similarity between each frame of the generated video and each frame of the reference video, and then taking the average. For robustness, we use the image CLIP [6] metric to calculate the similarity between frames. We use Flow-Square-Mean and GenVideo-RefVideo CLIP (Corresponding frames) to represent these two evaluation metrics, respectively.

4.3. Temporal consistency

Temporal consistency measures whether the generated video frames are consistent and coherent with each other. We calculate the image CLIP [6] similarity between every two adjacent frames in the generated video and take the average as an indicator of the temporal consistency of the generated video. We use GenVideo Clip (Adjacent frames) to represent this evaluation metric. In addition, we also use GenVideo-RefVideo (Corresponding frames) from Section 4.2 to represent temporal consistency.

Table 2
Quantitative analysis for different Image-to-Video algorithms. An upward arrow indicates that higher values are better, while a downward arrow means lower values are preferable.

Dimensions	Metrics	VideoCrafter [25]	I2VGen-XL [26]	SVD [28]	Pika [9]	Gen2 [10]
Control-video alignment	MSE (First) ↓	3929.65	4491.90	640.75	**155.30**	235.53
	SSIM (First) ↑	0.300	0.354	0.612	0.800	**0.803**
	Image-GenVideo Clip ↑	0.830	0.832	0.919	0.930	**0.939**
	GenVideo-Text Clip ↑	0.23	0.24	–	**0.271**	0.270
	GenVideo-RefVideo CliP (Keyframes) ↑	0.763	0.764	–	**0.824**	0.820
Motion effects	Flow-Square-Mean	1.24	1.80	2.52	0.281	1.18
	GenVideo-RefVideo CliP (Corresponding frames) ↑	0.764	0.764	0.796	**0.823**	0.818
Temporal consistency	GenVideo Clip (Adjacent frames) ↑	0.980	0.971	0.974	**0.996**	0.995
	GenVideo-RefVideo CliP (Corresponding frames) ↑	0.764	0.764	0.796	**0.823**	0.818
Video quality	Frame Count ↑	16	32	25	72	**96**
	DOVER ↑	0.518	0.510	0.623	0.715	**0.775**
	GenVideo-RefVideo SSIM ↑	0.367	0.304	0.507	**0.560**	0.504

4.4. Video quality

Video quality is a relatively subjective dimension, measuring the overall quality of video production. We first use the number of frames generated by videos to gauge the ability of different algorithms to generate long videos. We utilize disentangled objective video quality evaluator (DOVER) [48], a no-reference video quality assessment metric. DOVER [48] comprehensively rates videos from both aesthetic and technical perspectives, using the collected DIVIDE-3k dataset. Experimental results show that the DOVER [48] metric highly correlates with human opinions in both aesthetic and technical perspectives. For the DOVER evaluation metric, we calculate it using all frames generated by their respective algorithms. For the video–text dataset, since we have reference videos available, we measure the spatial structural similarity of the generated videos to the reference videos by calculating the SSIM (Structural Similarity Index Measure) between the corresponding frames of the generated and reference videos. We denote this evaluation metric as GenVideo-RefVideo SSIM.

5. Experiments

5.1. Evaluated models

5.1.1. Open-source project

VideoCrafter. VideoCrafter [25] is an open-source video generation and editing toolbox for crafting video content. It supports the generation of videos from images. We use a guidance scale of 12 and ddim steps of 25. For videos with an aspect ratio of 1, we employ a resolution of 512 * 512, while for videos with an aspect ratio of 0.5625, we use a resolution of 512 * 320, and then uniformly resize to align with the resolutions used by other methods.

I2vgen-XL. I2VGen-XL [26] is an open-source video synthesis codebase developed by Tongyi Lab at Alibaba Group, which features state-of-the-art video generative models. We use a guide scale of 9 and infer with fp16 precision.

Stable video diffusion. Stable Video Diffusion (SVD) [28] is an expansion of the model based on Image Stable Diffusion [39]. We use the 25-frame version of Stable Video Diffusion. It is worth noting that the current model does not support text input temporarily, hence we did not calculate the text-video alignment for this model.

5.1.2. Closed-source project

Pika. Pika [9] is a technology company revolutionizing video creation by making it effortless and accessible for everyone. In just six months, Pika has built a community of half a million users producing millions of videos per week. The company recently launched Pika 1.0, a significant upgrade featuring a new AI model that supports various video styles, including 3D animation, anime, cartoons, and cinematic, coupled with an improved web experience. Considering that Pika [9] does not have open-source code, we manually tested 60 cases on the Discord platform (30 from the WebVid dataset and 30 from our own generated dataset). We used the default parameters of motion set to 1 and the guidance scale set to 12.

Gen2. Gen2 [10] is a multimodal AI system that can generate novel videos with text, images, or video clips. We used the default motion setting of 5 from the demo and did not employ the camera movement parameter to generate videos.

5.2. Comprehensive results analysis

Table 2 presents the evaluation of five state-of-the-art (SOTA) I2V algorithms across five dimensions: image fidelity, motion effects, text-video alignment, temporal consistency, and video quality. We present the qualitative results of different I2V algorithms in Fig. 3. We find that VideoCrafter and I2VGen-xl struggle to preserve the original image. I2VGen-xl maintains relatively good semantics, but the spatial structure of the initial image is mostly not preserved. VideoCrafter can approximate the spatial structure of the initial image to some extent, but the preservation of details is generally mediocre. SVD, Pika, and Gen2 preserve the original image quite well, with Gen2 achieving the best preservation effect. As for the aspect of Text-video alignment, Gen2 and Pika are nearly on par with each other and both outperform the open-source algorithms. However, existing algorithms and evaluation metrics do not effectively capture fine-grained textual changes. In terms of motion effects, VideoCrafter tends to remain static. I2VGen-xl and SVD lean towards camera movement rather than subject motion, which is why they score high on the flow-square-mean but obtain low GenVideo-RefVideo Clip scores. Pika tends to favor both local and subject movement, thus achieving high GenVideo-RefVideo Clip scores and low flow-square-mean scores. Gen2, on the other hand, favors movement in both the foreground and background, but the background movement is not as pronounced as with SVD.

In the aspect of temporal Consistency, VideoCrafter, due to its poorer motion effects, does not perform poorly in terms of temporal consistency. Considering that SVD has stronger motion effects and still maintains good temporal consistency, it has achieved the best performance among open-source I2V algorithms. Similarly, Pika, because of its tendency for local movement, has achieved the highest score in overall temporal consistency. As for video quality, Gen2 is capable of generating the longest videos of up to 96 frames, with the highest levels of aesthetics and clarity. Pika, due to its tendency for local movement, has achieved the highest similarity in the GenVideo-RefVideo SSIM metric. SVD benefits from the priors of the image stable diffusion model, resulting in videos that reach the best performance among open-source I2V algorithms. In summary, the two closed-source projects, Pika

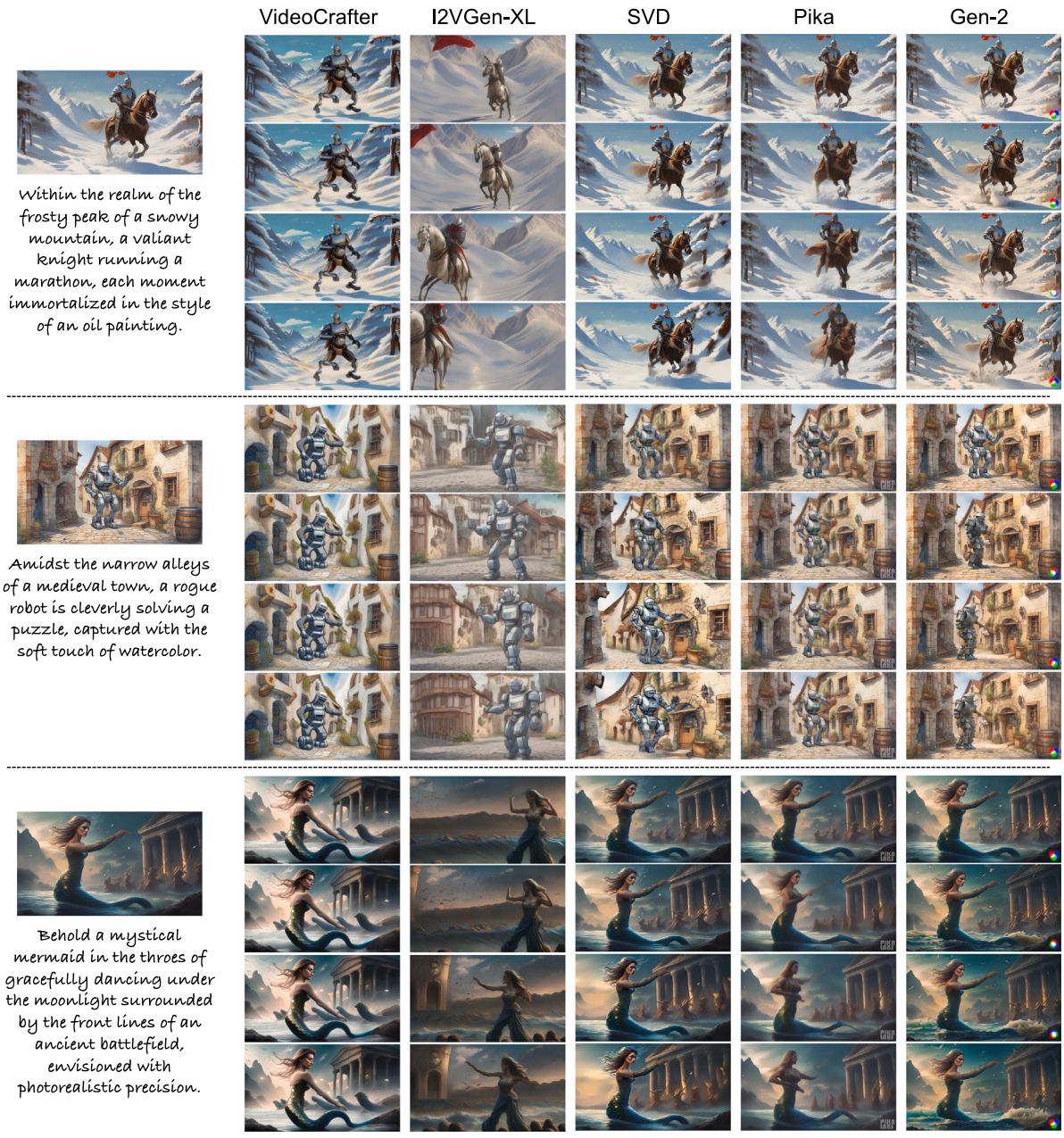


Fig. 3. We present three I2V cases utilizing five state-of-the-art algorithms, among which VideoCrafter, I2VGen-XL, and SVD are open-source research, while Pika and Gen2 are closed-source project. For additional videos, please refer to our supplementary materials.

and Gen2, achieved the most optimal generation effects, capable of producing long videos. Pika excels in generating local motion, while Gen2 tends to prefer global motion. SVD achieved the best results among the open-source options, demonstrating outcomes that were close to those of the two closed-source projects.

5.3. User study

To validate whether the proposed evaluation standards are aligned with human preference, we randomly sampled 30 generated results from each of the five methods and tallied the best algorithm outcomes in each of the four dimensions (Image Fidelity, Motion Effects, Temporal Consistency, Video Quality) through human voting. We have tallied the votes of a total of 42 individuals, with the specific results presented in Fig. 4. We discovered that Gen2's performance is on par with Pika, both achieving optimal results. Pika excelled in temporal consistency and motion effects, while Gen2 came out on top in terms of image fidelity and video quality. SVD showed a balanced performance across all areas, securing the best results among the open-source options. We found that the users' votes are relatively consistent with the results evaluated by our assessment criteria.

5.4. Findings and discussions

Despite the notable achievements of I2V and the rapid updates of new algorithms, there is still significant room for improvement in existing solutions. Utilizing AIGCBench, we have conducted a detailed survey and evaluation of the five most advanced I2V algorithms from both academia and industry. The comprehensive analysis facilitated by AIGCBench has led us to make the following discoveries:

Lacking fine-grained control. AIGCBench's testing methodology has revealed that current I2V tasks often fall short in allowing users to generate content with precise textual descriptions. The benchmark's

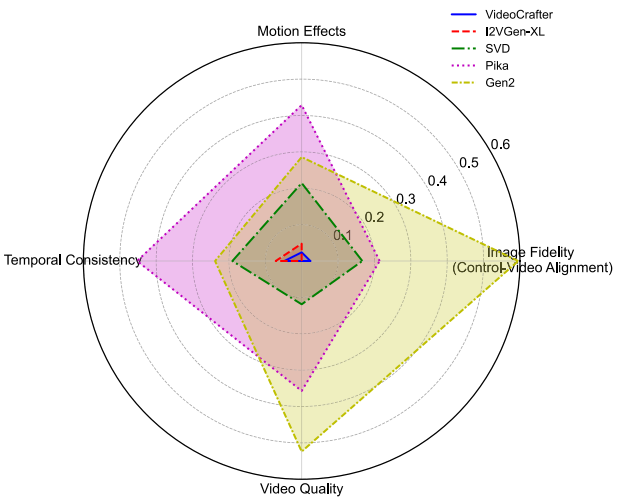


Fig. 4. We tallied the votes of 42 individuals, evaluating five state-of-the-art I2V algorithms from four aspects. The numerical values in the radar chart represent the proportion of users who voted for each algorithm as being the best performer in that aspect.

diverse datasets and nuanced evaluation metrics have shown that while solutions based on CLIP [6] and large language models [49] are a step in the right direction, they are not fully capturing the fine-grained details that yield a high degree of control over video content. AIGCBench's detailed feedback on these algorithms has underscored the need for a model that is specifically trained for video contexts to improve text-video alignment and to provide users with that control. Our findings suggest that integrating AIGCBench's evaluation criteria into the development process could lead to algorithms that better align with human preferences.

Long video generation. The current I2V algorithms can generate up to 96 frames in a single inference, which is far from satisfying users' needs for longer video production. Considering that video scenes typically have a frame rate of 24 fps, the basic generation capability of mainstream algorithms is around 3 s. There are mainly two approaches to address this limitation. One is to use multiple inferences, where most adopt a coarse-to-fine generation pipeline—first generating diluted keyframes, then densely producing all frames. The challenge of this method lies in maintaining temporal consistency across multiple inferences. The other approach is to use multi-GPU training and inference with a single model, which currently struggles to guarantee satisfactory results. How to generate longer videos should be an urgent issue for the AI-generated content (AIGC) community to address next.

Inference speed. Currently, the speed of video generation is relatively slow. For a 3-second video, mainstream algorithms generally require about 1 min on a V100 graphics card. Considering that video generation scenarios are based on diffusion models [1–5], there are currently two main routes for speeding up the process. One is to reduce the dimensionality of the video in the latent space. For example, Stable Diffusion [28] maps the video into a latent space, roughly decreasing the size of the video by about 8 times, with only a minimal loss of video quality. The other is to improve the inference speed of the diffusion model, which is also a hot research topic in the AIGC community.

In light of these findings from AIGCBench, it becomes evident that the benchmark not only aids in the identification of current shortcomings but also offers a structure for addressing them. AIGCBench's comprehensive framework for evaluation, encompassing a rich variety of datasets and multi-dimensional metrics, provides a roadmap for advancing the state of I2V algorithms. By exposing algorithmic weaknesses and offering a standardized platform for comparison, AIGCBench guides researchers towards developing solutions that overcome the challenges of fine-grained control, video length, and inference speed. The insights gained from AIGCBench evaluations are instrumental in pushing the boundaries of what is possible in the field of I2V generation.

6. Conclusion

In this work, we have introduced AIGCBench, a comprehensive and scalable benchmark tailored for the evaluation of Image-to-Video (I2V) generation tasks. AIGCBench provides a much-needed framework to assess the performance of various state-of-the-art I2V algorithms under equivalent evaluation conditions. Our benchmark stands out by incorporating a diverse set of real-world video–text and image–text datasets, as well as a novel dataset produced through our proprietary generation pipeline. We have also proposed a novel set of evaluation metrics that span across four critical dimensions: control-video alignment, motion effects, temporal consistency, and video quality. These metrics have been validated against human judgment to ensure their alignment with human preferences. Our extensive evaluation of leading I2V models has not only highlighted their strengths and weaknesses but also unearthed significant insights that will guide the future development of the I2V domain.

AIGCBench marks a foundational step in benchmarking for AIGC, pushing the frontier of I2V technology evaluation. By offering a scalable and precise assessment methodology, we set the stage for continuous enhancements and innovations in this rapidly evolving research field. As we progress, we plan to expand AIGCBench to encompass a broader range of video generation tasks, creating a unified and extensive benchmark that reflects the multifaceted nature of AIGC.

7. Limitations and future work

Due to the slow inference speed of video generation by I2V models and the fact that some works are not open-sourced (e.g., Pika [9], Gen2 [10]), our benchmark only evaluated 3928 test cases. Considering the complexity of video generation tasks, we believe this number is insufficient. Furthermore, given the lack of fine-grained video recognition models currently available, our evaluation system is unable to accurately judge whether the direction of object movement in the generated videos matches the text description. For instance, whether water flows from left to right or from right to left, we are currently unable to determine through automated evaluation metrics if the direction of the water flow in the generated video is consistent with the textual description.

Moving forward, we will integrate tasks related to T2V and new video generation tasks into a large-scale video generation benchmark. Additionally, to address the issues mentioned above, we may train a fine-grained video representation model aligned with text, which will be utilized for fine-grained alignment of video and text scenes.

CRediT authorship contribution statement

Fanda Fan: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Chunjie Luo:** Writing – review & editing, Writing – original draft, Resources, Investigation, Conceptualization. **Wanling Gao:** Writing – review & editing, Writing – original draft, Methodology, Investigation. **Jianfeng Zhan:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

Fanda Fan, Chunjie Luo, Wanling Gao and Jianfeng Zhan are faculty at Institute of Computing Technology, Chinese Academy of Sciences University of Chinese Academy of Sciences.

Acknowledgments

We want to thank the reviewers and editors for their constructive comments and suggestions. This research is partly supported by the Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No. XDA0320000 and XDA0320300. I would like to extend my heartfelt thanks to Professor Lei Wang for his insightful discussions and valuable revisions to this manuscript. I am also grateful to Mengya He for her contributions to the discussions of this paper. Furthermore, I wish to acknowledge Litong Gong, Weijie Li, Yiran Zhu, and Biao Wang from Alibaba Company for their support in the experimental aspects of this research.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.tbench.2024.100152.

References

[1] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, S. Ganguli, Deep unsupervised learning using nonequilibrium thermodynamics, in: International Conference on Machine Learning, PMLR, 2015, pp. 2256–2265.

[2] Y. Song, S. Ermon, Generative modeling by estimating gradients of the data distribution, in: Advances in Neural Information Processing Systems, vol. 32, 2019.

[3] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 6840–6851.

[4] A.Q. Nichol, P. Dhariwal, Improved denoising diffusion probabilistic models, in: International Conference on Machine Learning, PMLR, 2021, pp. 8162–8171.

[5] P. Dhariwal, A. Nichol, Diffusion models beat gans on image synthesis, in: Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 8780–8794.

[6] A. Radford, J.W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al., Learning transferable visual models from natural language supervision, in: International Conference on Machine Learning, PMLR, 2021, pp. 8748–8763.

[7] U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni, et al., Make-a-video: Text-to-video generation without text-video data, 2022, arXiv preprint arXiv:2209.14792.

[8] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D.P. Kingma, B. Poole, M. Norouzi, D.J. Fleet, et al., Imagen video: High definition video generation with diffusion models, 2022, arXiv preprint arXiv:2210.02303.

[9] I. Pika, Pika lab discord server, 2023, https://www.pika.art/. (Accessed: 30 August 2023).

[10] P. Esser, J. Chiu, P. Atighehchian, J. Granskog, A. Germanidis, Structure and content-guided video synthesis with diffusion models, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 7346–7356.

[11] Q. Sun, Y. Cui, X. Zhang, F. Zhang, Q. Yu, Z. Luo, Y. Wang, Y. Rao, J. Liu, T. Huang, X. Wang, Generative multimodal models are in-context learners, 2023, arXiv:2312.13286.

[12] J. Karras, A. Holynski, T.-C. Wang, I. Kemelmacher-Shlizerman, Dreampose: Fashion image-to-video synthesis via stable diffusion, 2023, arXiv preprint arXiv:2304.06025.

[13] S. Yin, C. Wu, J. Liang, J. Shi, H. Li, G. Ming, N. Duan, Dragnuwa: Fine-grained control in video generation by integrating text, image, and trajectory, 2023, arXiv preprint arXiv:2308.08089.

[14] Z. Li, R. Tucker, N. Snavely, A. Holynski, Generative image dynamics, 2023, arXiv preprint arXiv:2309.07906.

[15] W. Hong, M. Ding, W. Zheng, X. Liu, J. Tang, Cogvideo: Large-scale pretraining for text-to-video generation via transformers, 2022, arXiv preprint arXiv:2205.15868.

[16] Y. He, T. Yang, Y. Zhang, Y. Shan, Q. Chen, Latent video diffusion models for high-fidelity video generation with arbitrary lengths, 2022, arXiv preprint arXiv:2211.13221.

[17] J.Z. Wu, Y. Ge, X. Wang, S.W. Lei, Y. Gu, Y. Shi, W. Hsu, Y. Shan, X. Qie, M.Z. Shou, Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 7623–7633.

[18] Z. Luo, D. Chen, Y. Zhang, Y. Huang, L. Wang, Y. Shen, D. Zhao, J. Zhou, T. Tan, VideoFusion: Decomposed diffusion models for high-quality video generation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 10209–10218.

[19] Y. Guo, C. Yang, A. Rao, Y. Wang, Y. Qiao, D. Lin, B. Dai, Animatediff: Animate your personalized text-to-image diffusion models without specific tuning, 2023, arXiv preprint arXiv:2307.04725.

[20] Y. Liu, L. Li, S. Ren, R. Gao, S. Li, S. Chen, X. Sun, L. Hou, FETV: A benchmark for fine-grained evaluation of open-domain text-to-video generation, 2023, arXiv preprint arXiv:2311.01813.

[21] Y. Liu, X. Cun, X. Liu, X. Wang, Y. Zhang, H. Chen, Y. Liu, T. Zeng, R. Chan, Y. Shan, EvalCrafter: Benchmarking and evaluating large video generation models, 2023, arXiv preprint arXiv:2310.11440.

[22] Z. Huang, Y. He, J. Yu, F. Zhang, C. Si, Y. Jiang, Y. Zhang, T. Wu, Q. Jin, N. Chanpaisit, et al., VBench: Comprehensive benchmark suite for video generative models, 2023, arXiv preprint arXiv:2311.17982.

[23] H. Ni, C. Shi, K. Li, S.X. Huang, M.R. Min, Conditional image-to-video generation with latent flow diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 18444–18455.

[24] Y. Hu, C. Luo, Z. Chen, A benchmark for controllable text-image-to-video generation, IEEE Trans. Multimed. (2023).

[25] H. Chen, M. Xia, Y. He, Y. Zhang, X. Cun, S. Yang, J. Xing, Y. Liu, Q. Chen, X. Wang, et al., Videocrafter1: Open diffusion models for high-quality video generation, 2023, arXiv preprint arXiv:2310.19512.

[26] S. Zhang, J. Wang, Y. Zhang, K. Zhao, H. Yuan, Z. Qin, X. Wang, D. Zhao, J. Zhou, I2vgen-xl: High-quality image-to-video synthesis via cascaded diffusion models, 2023, arXiv preprint arXiv:2311.04145.

[27] X. Gu, C. Wen, J. Song, Y. Gao, Seer: Language instructed video prediction with latent diffusion models, 2023, arXiv preprint arXiv:2303.14897.

[28] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, et al., Stable video diffusion: Scaling latent video diffusion models to large datasets, 2023, arXiv preprint arXiv:2311.15127.

[29] J. Zhan, L. Wang, W. Gao, H. Li, Y. Huang, C. Wang, Y. Li, Z. Yang, G. Kang, C. Luo, H. Ye, S. Dai, Z. Zhang, Evaluatology: The Science and Engineering of Evaluation, Technical Report, Institute of Computing Technology Chinese Academy of Sciences, 2024.

[30] OpenAI, GPT-4 technical report, 2023, arXiv:2303.08774.

[31] Y. Zhang, Z. Xing, Y. Zeng, Y. Fang, K. Chen, PIA: Your personalized image animator via plug-and-play modules in text-to-image models, 2023, arXiv preprint arXiv:2312.13964.

[32] A. Blattmann, R. Rombach, H. Ling, T. Dockhorn, S.W. Kim, S. Fidler, K. Kreis, Align your latents: High-resolution video synthesis with latent diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 22563–22575.

[33] L. Yu, Y. Cheng, K. Sohn, J. Lezama, H. Zhang, H. Chang, A.G. Hauptmann, M.-H. Yang, Y. Hao, I. Essa, L. Jiang, MAGVIT: Masked generative video transformer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.

[34] S. Ge, S. Nah, G. Liu, T. Poon, A. Tao, B. Catanzaro, D. Jacobs, J.-B. Huang, M.-Y. Liu, Y. Balaji, Preserve your own correlation: A noise prior for video diffusion models, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 22930–22941.

[35] L. Khachatryan, A. Movsisyan, V. Tadevosyan, R. Henschel, Z. Wang, S. Navasardyan, H. Shi, Text2video-zero: Text-to-image diffusion models are zero-shot video generators, 2023, arXiv preprint arXiv:2303.13439.

[36] J. Wang, H. Yuan, D. Chen, Y. Zhang, X. Wang, S. Zhang, Modelscope text-to-video technical report, 2023, arXiv preprint arXiv:2308.06571.

[37] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, Springer, 2015, pp. 234–241.

[38] X. Wang, H. Yuan, S. Zhang, D. Chen, J. Wang, Y. Zhang, Y. Shen, D. Zhao, J. Zhou, VideoComposer: Compositional video synthesis with motion controllability, 2023, arXiv preprint arXiv:2306.02018.

[39] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, High-resolution image synthesis with latent diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10684–10695.

[40] R. Girdhar, M. Singh, A. Brown, Q. Duval, S. Azadi, S.S. Rambhatla, A. Shah, X. Yin, D. Parikh, I. Misra, Emu video: Factorizing text-to-video generation by explicit image conditioning, 2023, arXiv preprint arXiv:2311.10709.

[41] M. Bain, A. Nagrani, G. Varol, A. Zisserman, Frozen in time: A joint video and image encoder for end-to-end retrieval, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 1728–1738.

[42] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, J. Jitsev, LAION-5B: An open large-scale dataset for training next generation image-text models, 2022, arXiv:2210.08402.

[43] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, I. Sutskever, Zero-shot text-to-image generation, in: International Conference on Machine Learning, PMLR, 2021, pp. 8821–8831.

[44] I. Civit AI, CivitAI, 2022, https://civitai.com/ (Accessed: (2022)).
[45] K. Huang, K. Sun, E. Xie, Z. Li, X. Liu, T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation, 2023, arXiv preprint arXiv:2307.06350.
[46] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: From error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[47] Z. Teed, J. Deng, Raft: Recurrent all-pairs field transforms for optical flow, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16, Springer, 2020, pp. 402–419.
[48] H. Wu, E. Zhang, L. Liao, C. Chen, J. Hou, A. Wang, W. Sun, Q. Yan, W. Lin, Exploring video quality assessment on user generated contents from aesthetic and technical perspectives, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 20144–20154.
[49] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P.J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, J. Mach. Learn. Res. 21 (1) (2020) 5485–5551.

BenchCouncil Transactions on Benchmarks, Standards and Evaluations 3 (2023) 100153

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.elsevier.com/locate/tbench

Full Length Article

Benchmarking ChatGPT for prototyping theories: Experimental studies using the technology acceptance model

Tiong-Thye Goh [a], Xin Dai [b], Yanwu Yang [b,*]

[a] School of Information Management, Victoria University of Wellington, Address: 23 Lambton Quay, Wellington 6011, New Zealand
[b] School of Management, Huazhong University of Science and Technology, Address: Luoyu Road, Wuhan 430074, China

ARTICLE INFO

Keywords:
ChatGPT
Large language model
Technology acceptance model
Prototyping Theory

ABSTRACT

This paper explores the paradigm of leveraging ChatGPT as a benchmark tool for theory prototyping in conceptual research. Specifically, we conducted two experimental studies using the classical technology acceptance model (TAM) to demonstrate and evaluate ChatGPT's capability of comprehending theoretical concepts, discriminating between constructs, and generating meaningful responses. Results of the two studies indicate that ChatGPT can generate responses aligned with the TAM theory and constructs. Key metrics including the factors loading, internal consistency reliability, and convergence reliability of the measurement model surpass the minimum threshold, thus confirming the validity of TAM constructs. Moreover, supported hypotheses provide an evidence for the nomological validity of TAM constructs. However, both of the two studies show a high Heterotrait–Monotrait ratio of correlations (HTMT) among TAM constructs, suggesting a concern about discriminant validity. Furthermore, high duplicated response rates were identified and potential biases regarding gender, usage experiences, perceived usefulness, and behavioural intention were revealed in ChatGPT-generated samples. Therefore, it calls for additional efforts in LLM to address performance metrics related to duplicated responses, the strength of discriminant validity, the impact of prompt design, and the generalizability of findings across contexts.

Introduction

ChatGPT (Generative Pretrained Transformer), powered by the generative large language model, possesses remarkable capabilities in generating human-like responses and engaging in naturalistic conversations across diverse topics. It is a state-of-the-art natural language processing model developed by OpenAI, trained on an extensive dataset. ChatGPT has been leveraged for a wide range of applications, ranging from aiding in creative writing and content generation to providing customer support and answering user queries [42,43].

Recent works have focused on profiling ChatGPT to explore its gender, personality, and political inclinations [34,39,49]. By prompting ChatGPT with specific instructions, researchers have investigated how these factors influence the model's responses, e.g., whether ChatGPT exhibits gender or political biases in its generated content and perception. Prompting ChatGPT with 630 political statements from voting advice applications and a political compass test, Hartmann et al. [20] uncovered ChatGPT's pro-environmental, left-libertarian ideology. In a study by Wong and Kim [49], 501 participants were recruited from Prolific to examine biases in perceiving ChatGPT's gender, where participants watched videos showcasing ChatGPT's capabilities and then provided gender ratings using an 8-point scale or a binary choice. Their results revealed a consistent tendency to perceive ChatGPT as more male than female, regardless of the response scale. These studies uncovered potential biases in AI systems, raised awareness about societal impacts of such biases, and developed methods to mitigate them [5,35]. Understanding the capabilities and limitations of ChatGPT in terms of profiling is crucial for responsibly deploying large-scale AI systems in real-world applications. By scrutinizing ChatGPT's responses, researchers can devote to developing more robust, inclusive, and unbiased AI technologies that can positively contribute to various domains and empower users with reliable and fair interactions [25].

Profiling ChatGPT assumes that ChatGPT represents a single user with stochastic attitudes and behaviours [15], thereby limiting its breadth of applications. In effect, it is possible to induce ChatGPT to simulate a population of different individual profiles, which significantly expands its capabilities. Jiang et al. [22] devised a method called chain prompting, which enables the language model to exhibit specific personalities and diverse behaviours in a controlled manner. This approach allows ChatGPT to cater to a broader range of user needs and preferences by considering different communication styles, cultural backgrounds, and domain-specific knowledge. Individual profiles enable personalized recommendations and guidance by capturing the unique characteristics of each simulated user. With ChatGPT capable of simulating a

* Corresponding author.
E-mail addresses: tiong.goh@vuw.ac.nz (T.-T. Goh), daixin@mail.hust.edu.cn (X. Dai), yangyanwu@hust.edu.cn (Y. Yang).

https://doi.org/10.1016/j.tbench.2024.100153
Received 30 October 2023; Received in revised form 25 January 2024; Accepted 31 January 2024
Available online 6 February 2024

population of individual profiles, it opens a potential avenue in conceptual understanding and theory prototyping which have not been thoroughly explored, to the best of our knowledge. By employing ChatGPT as a platform for prototyping a theory and evaluating its comprehension of related concepts, researchers can assess its ability to grasp and manipulate abstract ideas, as well as the interconnections between concepts.

The purpose of this study is twofold. First, it aims to explore the use of ChatGPT to respond to conceptual theories and assess its ability of comprehending constructs. Second, this study seeks to evaluate the validity of conceptual theories by examining relationships among different constructs with participants and survey responses generated by ChatGPT. The research questions are:

(1) How well does ChatGPT process various constructs within the context of provided conceptual theories?
(2) How valid are relationships between different constructs with survey responses generated by ChatGPT when being evaluated through a structural equation model?

This study has several potential contributions. First, it addresses a design shortcoming between engineering science prototyping and conceptual development in social science. While engineering commonly utilizes design tools for product prototyping, the realm of social science traditionally relied heavily on extensive human participation for conceptual theory development. The advent of Large Language Models (LLMs) like ChatGPT transforms this landscape by providing a design platform analogous to an engineering design tool. Software and system designers can integrate LLMs, such as ChatGPT, with other simulation tools for theoretical development. This would enable a novel and efficient approach to theory design in social science, offering a platform where human input and machine-generated insights collaborate seamlessly. The result is a more cohesive and comprehensive research methodology that amalgamates the strengths of human knowledge with the analytical capabilities of LLMs, fostering a dynamic and iterative process in social science research.

Second, researchers can follow the methodology used in this study to assess their theories' understanding, validate theoretical frameworks, and iteratively refine their theories through the analysis of responses generated by ChatGPT. This not only aids in enhancing the robustness of theoretical foundations but also provides a valuable tool for continuous improvement.

Third, the introduced research paradigm facilitates rapid exploration, fostering collaboration, and aiding in hypothesis generation. The interactive nature of ChatGPT allows for the swift identification of errors or inconsistencies, enabling researchers to promptly refine their theories. This accelerates the research prototyping process, saving valuable time and resources.

Lastly, the scalability of ChatGPT permits the testing of theories across a broad spectrum of contexts. This scalability, combined with interactive capabilities, promotes efficient theory development by allowing researchers to explore diverse scenarios and adapt their hypotheses accordingly.

In essence, the contributions of this study put forward the idea, drawing on two studies, that Large Language Models (LLMs) potentially play an important role in enhancing the efficiency, collaboration, and adaptability of research processes such as theory development in the fields of business, education, and social science using an interdisciplinary paradigm [50] which might not be possible in the past.

Related work

Large language model (LLM) - ChatGPT

ChatGPT has attracted interest in its potential for simulating human-like characteristics and generating perception responses. While past research solely focusing on using ChatGPT for prototyping theories is limited, we present studies that revealed connections that may support its relevance to theory prototyping.

Simulation of sample profile: An important aspect of prototyping theories is the ability to generate sample profiles based on a specific population. Madelyn [29] reported LLM such as GPT-3 can generate diverse data points and maintain relationships between columns, making it a useful platform for quickly generating data for testing proof of concepts. It can provide statistical relationships when explicitly requested. However, its limitations include the uncertainty of accurately modeling the complexities of real-world data.

Theory of Mind (ToM) Proficiency: Kosinski [26] demonstrated a high success rate of using ChatGPT in Theory of Mind (ToM) tasks. ChatGPT's ability to comprehend and respond to human intentions, beliefs, and emotions signifies its potential in prototyping theories related to social cognition and understanding others. Utilizing ChatGPT's ToM proficiency, researchers can explore and prototype theories on empathy, social interaction, and psychological processes. The study by Brunet-Gouet et al. [4] highlights the ability of ChatGPT to infer intentions, track beliefs, and respond to questions about mental states. These capabilities can be leveraged in the context of theory prototyping, where researchers seek to simulate and examine theoretical constructs related to human cognition, psychology, and social interaction.

ChatGPT Personalities and Psychologies: Machine personalities and psychologies associated with language models have been studied by G. Jiang et al. [22]. Their study suggests that ChatGPT's responses and interaction patterns are influenced by its machine personality characteristics. Personality priming ChatGPT into various pseudo personalities and behaviour tendencies using a psychological prompt, such as the human-like moral judgments [11], could transform ChatGPT as participants in a survey and inform the prototyping of theories related to personality psychology, human-computer interaction, and user experiences. Current research focuses on utilizing ChatGPT for various individualized assessments, leaving a research gap in assessing its capability of understanding concepts and constructs, and prototyping theories.

Technology acceptance model in education

The integration of new technologies into learning and teaching has become an area of great interest in the field of education. Digital technologies are a vital tool in achieving the objective of ensuring inclusive and equitable access for all [19]. As it is crucial to understand why users adopt or reject specific technologies in educational settings, research on technology acceptance in teaching and learning contexts has gained popularity. The Technology Acceptance Model (TAM) has gained prominence as a scientific paradigm for examining the acceptance of learning technology. TAM originated from the Theory of Reasoned Action (TRA) [2] and has evolved into a key model for understanding the predictors of human behaviours regarding technology acceptance. Davis [9] proposed the TAM framework that emphasizes factors such as perceived ease of use (PEOU), perceived usefulness (PU), attitude (AT) and behavioural intention (BI) towards using technology, which influences use motivation. It has demonstrated its applicability across a wide range of technologies and user groups.

In the realm of technology acceptance literature in education [16,40], TAM has been widely utilized by numerous studies expanding upon or applying the original model. Researchers have delved into user intentions toward e-learning technology using TAM as well as additional constructs such as subjective norms, perceived enjoyment, perceived compatibility, perceived trust, flow, and perceived social influence [14,23,24]. Moreover, the applicability of TAM has been explored in various learning technologies including mobile learning, personal learning environments (PLEs), learning management systems (LMSs), and emerging technologies like virtual reality (VR) and artificial intelligence (AI) [17]. Furthermore, the adoption of TAM in educational research highlights its significance in comprehending the factors that influence technology acceptance among students, teachers, and other

stakeholders. These studies contribute to a growing body of knowledge on the acceptance of learning technology, offering insights for effective implementation and utilization in educational contexts. Due to word limitations, interested readers can refer to Sukackė [45] and Granić and Marangunić [17] for a historical review of the technology acceptance model.

Measure development and metrics for construct validity

This study focuses on the notion of construct measurement in scale development [28]. A construct is a purposefully designed term within a scientific realm, serving to effectively organize knowledge and guide research endeavours in describing and explaining a particular aspect or phenomenon [31]. Each construct is measured with multiple items and could exist at a higher level of abstraction than concepts. In this context, a concept refers to an abstract idea operationalised through a construct that represents a particular attribute or dimension being measured. For example, in behavioural research, concepts like perceived usefulness, attitude or intention are often assessed using scales comprising multiple items or questions [9,10].

Measure development processes are essential for ensuring the validity and reliability of measures used in research and assessment. These processes involve various steps, including defining the construct's domain, generating items, specifying dimensions, and investigating dimensionality [32]. According to Peter and Churchill [32], adhering to careful measure development processes leads to higher construct validity, minimizing chance and method variance for more reliable and valid measures. ChatGPT can facilitate the measure development process by leveraging its language generation and understanding abilities to assist in generating and refining construct prototypes, thereby enhancing the overall reliability of measures.

Internal consistency reliability is a measure of the consistency or homogeneity of items within a construct [18]. It assesses the extent to which the items within the measure are measuring the same underlying construct. In behavioural research, internal consistency reliability is commonly used when a scale or questionnaire consists of multiple items or questions that are intended to measure a particular construct. The aim is to determine if these items are consistently measuring the same construct or if they are capturing different aspects. There are several commonly used statistical techniques for assessing internal consistency reliability, such as Cronbach's alpha [8] and Composite reliability [48]. These measures range from 0.0 to 1.0, with higher values indicating greater internal consistency reliability. A value closer to 1 indicates that the items in the scale are highly correlated and are consistently measuring the same construct. Generally, the accepted standard for both of these indices is 0.70 or above [18].

Convergent validity is the extent to which all indicators are related to the constructs they are meant to measure and are not related directly to constructs they are not intended to measure [7]. The metric used for evaluating a construct's convergent validity is the average variance extracted (AVE) for all indicators on each construct. AVE values above 0.5 or 0.6 are often considered indicative of good convergence, although the speific threshold can vary depending on the research field or context.

Discriminant validity, on the other hand, checks for the uniqueness of a measure and its independence from other variables. It is indicated by predictably low correlations between the measure of interest and other measures that are not supposed to measure the same variable or concept. Low correlations with unrelated measures indicate discriminant validity. Unlike reliability, discriminant validity is not enhanced by high reliability.

When assessing discriminant validity, the common Fornell-Larcker (FL) approach [12] and the relatively new Heterotrait-Monotrait ratio of correlations (HTMT) approach [21] can complement each other [13]. The FL approach relies on comparing the square roots of the average variance extracted (AVE) with the correlations between constructs. If the AVE square roots are greater than the corresponding inter-construct correlations, discriminant validity is established. On the other hand, the HTMT approach offers two ways to assess discriminant validity: as a criterion and as a statistical test [21]. Using HTMT as a criterion involves comparing its values to certain threshold values, such as 0.85 or 0.90. If the HTMT value exceeds these thresholds, it indicates potential issues with discriminant validity. The statistical test entails examining the null hypothesis (H0: HTMT $\geq$ 1) against the alternative hypothesis (H1: HTMT $<$ 1). If the confidence interval encompasses the value of one, it suggests the presence of discriminant validity concerns.

Both the FL approach and the HTMT approach provide researchers with valuable metrics to assess the distinctiveness of constructs in a measurement model. By utilizing multiple approaches, researchers can enhance the rigour and comprehensiveness of their evaluations of discriminant validity.

Nomological validity examines the degree to which a measure or construct behaves following established theoretical relationships and expectations. It involves assessing whether a measure demonstrates patterns of associations with other variables that are theoretically predicted or expected based on existing knowledge.

Measure development processes are crucial for establishing valid and reliable measures. Reliability, convergent validity, discriminant validity, and nomological validity are important metrics of measure evaluation: higher reliability generally leads to higher consistency, while convergent validity provides evidence of systematic variance; discriminant validity ensures the uniqueness of the measure, and nomological validity examines whether the measure behaves as expected. Appendix D provides a glossary of terms on theory and construct validation.

Research methodology

The methodology in this study involves a practical approach for leveraging ChatGPT in data generation, hypothesis evaluation, and metric assessment. We focus on designing prompts to guide ChatGPT in generating relevant data, supported by essential background information. The generated data undergoes a thorough evaluation of hypotheses to assess the model's fitness in addressing research inquiries. For quantifying and analysing performance metrics, we employ a combination of Partial Least Squares (PLS) and SPSS (Statistical Package for the Social Sciences). This framework ensures a robust evaluation of the generated responses, forming the basis for the study's findings and conclusions.

Prompt design

In ChatGPT, different prompt designs such as reframing can be utilized to achieve specific goals [30,51]. Instructional prompts offer explicit instructions, guiding ChatGPT's behaviours of generating responses aligned with a particular objective or style; contextual prompts provide background information, setting the conversation's context for better understanding; Socratic prompts use a questioning approach, encouraging critical thinking and exploring different perspectives [6]; seed prompts offer a starting point for ChatGPT to continue the conversation; evaluation prompts ask ChatGPT to assess given responses; creative prompts stimulate imaginative outputs like storytelling; and conditional prompts introduce specific constraints for controlled conversations. Choosing the appropriate prompt designs depends on the desired outcomes and the nature of the interaction with ChatGPT.

For the reported study, we used a mix of prompt designs to elicit responses from ChatGPT for generating participants' responses for theory prototyping, as shown in Fig. 1. The prompt design process, differs from other frameworks [27], as it necessitates GPT to act as human participants. Therefore, the prompt design framework needs to ensure that GPT comprehends the entire experiment requirements. The prompt aimed to create a structured and coherent conversation to elicit meaningful formatted responses from ChatGPT. An instructional prompt was used to provide the experiment scenario and clear instructions for answering survey questions. A combination of contextual prompts and in-

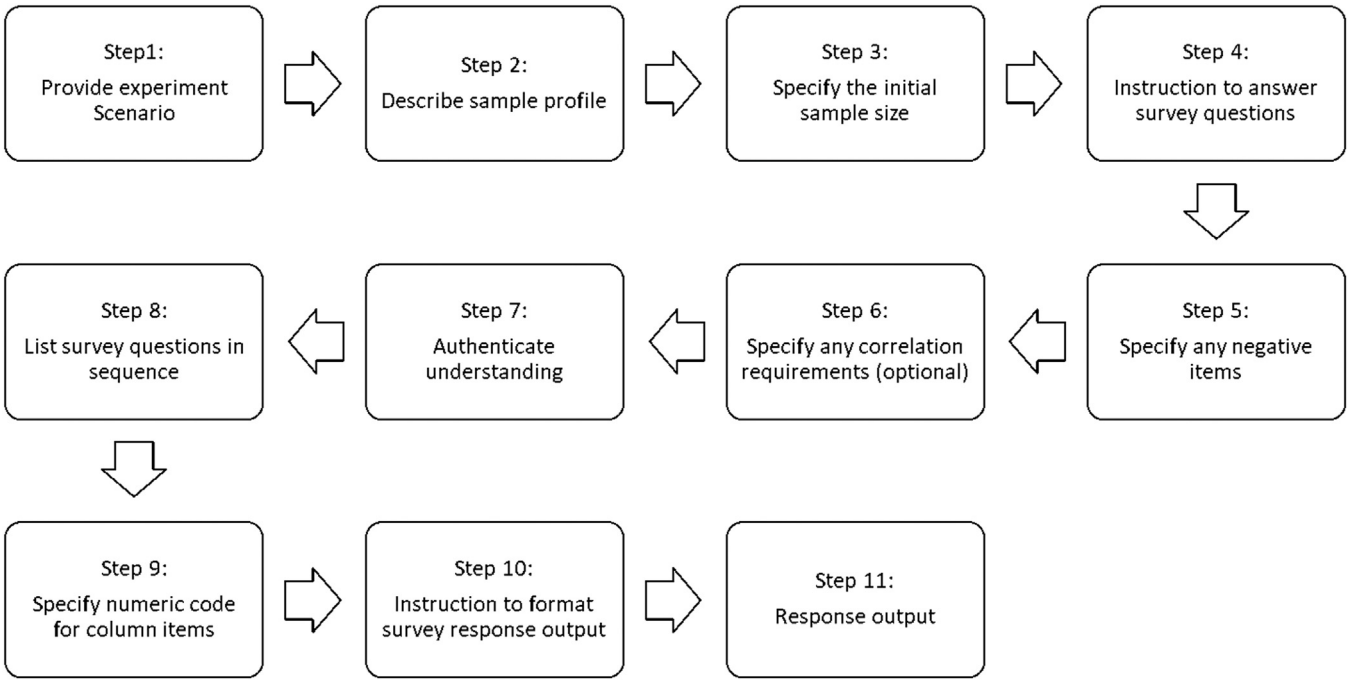


Fig. 1. The prompt design process.

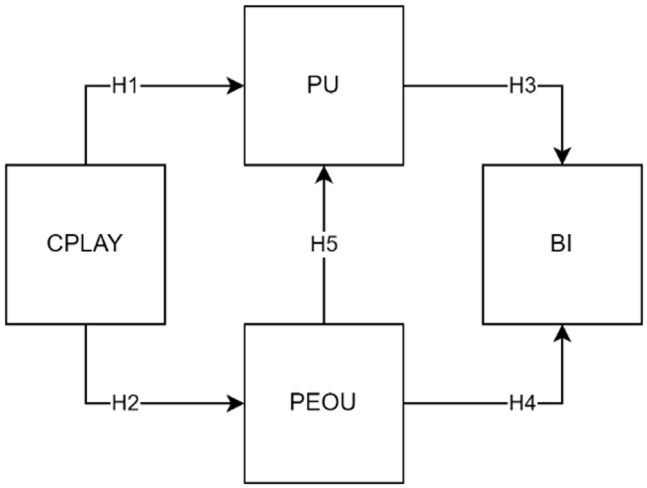


Fig. 2. Conceptual model in Study 1.

structional prompts was employed to describe the sample profile and authenticate understanding. The authentication prompt required ChatGPT to explain its reasoning before providing an answer. This process enhanced ChatGPT's reliability and overall performance by ensuring it understood the request [1]. The survey questions or conversation prompts were listed in sequence, and numeric codes were assigned to column items. Instructions were given for formatting the survey response output.

Background of study 1

Study 1 aims at creating a baseline model by leveraging the well-established and mature Technology Acceptance Model (TAM) [10] for subsequent comparison. The proposed conceptual model depicted in Fig. 2 includes the impact of computer playfulness (CPLAY) on students' perceived usefulness (PU) and perceived ease of use (PEOU) of ChatGPT, and how these factors collectively influence students' behavioural intentions (BI) towards adopting ChatGPT as a learning assistant. Computer playfulness (CPLAY) refers to the extent to which an individual's tendency to interact spontaneously, inventively, and imaginatively with computers [47]. It is an antecedent construct in the TAM3 model [46]. By incorporating the foundation Technology Acceptance Model (TAM), this study can replicate the original TAM and simultaneously validate a conceptual model over CPLAY, PU, PEOU, and BI and verify the hypotheses between these factors driving students' acceptance and adopting ChatGPT in a new learning context. The hypotheses in Study 1 were adapted from Davis [10] and Venkatesh and Bala [46] are as follows:

H1: Computer playfulness (CPLAY) positively influences students' perceived usefulness (PU) of ChatGPT as a learning assistant.
H2: Computer playfulness (CPLAY) positively influences students' perceived ease of use (PEOU) of ChatGPT as a learning assistant.
H3: Perceived usefulness (PU) positively influences students' behavioural intentions (BI) towards using ChatGPT as a learning assistant.
H4: Perceived ease of use (PEOU) positively influences students' behavioural intentions (BI) towards using ChatGPT as a learning assistant.
H5: Perceived ease of use (PEOU) positively influences students' perceived usefulness (PU) of ChatGPT as a learning assistant.

Data collection for study 1

In Study 1, the data collection involved utilizing ChatGPT to construct twenty student samples based on a student population profile. The profile included equal representation of gender with different majors, ages, years of study, and ChatGPT experience.

To gather responses, we prompted ChatGPT with the Technology Acceptance Model (TAM) questionnaires. By employing the TAM questionnaires, the intention is to gauge how ChatGPT's inner model comprehends the concept of perceived usefulness (PU), perceived ease of use (PEOU) and behavioural intention (BI) among the generated student samples. An additional construct of computer playfulness (CPLAY) [37] was included to assess its ability to discriminate against hedonic and utilitarian constructs. The scale used in this study ranged from 1

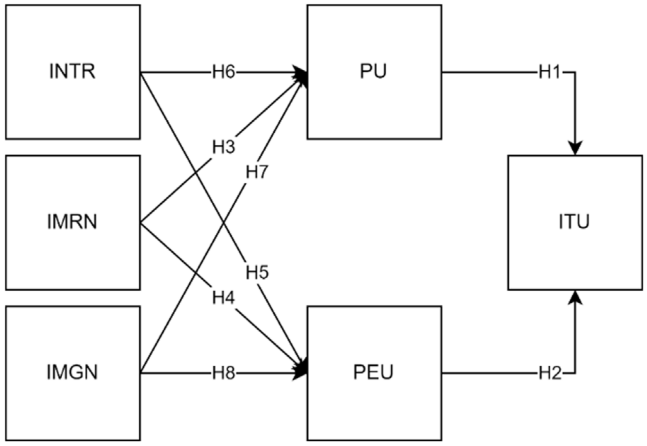


Fig. 3. Conceptual model in Study 1.

to 7, with 1 representing "Highly Unlikely," 2 representing "Unlikely," 3 representing "Somewhat Unlikely," 4 representing "Neutral," 5 representing "Somewhat Likely," 6 representing "Likely," and 7 representing "Highly Likely."

To ensure the robustness of the data collection, the process was repeated twenty times, generating a total of 400 samples. Each time ChatGPT generated twenty rows of responses based on a random set of student profiles. This methodology enabled ChatGPT to generate a sufficient sample size for conducting a structural equation analysis on how ChatGPT's inner model perceived the Technology Acceptance Model theory. The prompt is available in Appendix A. The wordings of the TAM constructs were modified from Davis [10] to align with the context of using ChatGPT as a learning assistant.

Background of study 2

The objective of Study 2 is to compare an existing TAM with different contexts and preferably to include a set of new constructs to assess ChatGPT's capabilities. To achieve this goal, we replicated the study by Barrett et al. [3]. Their study was based on user acceptance of a high-immersion virtual reality (VR) learning environment for English paragraph writing. A sample of 134 undergraduate students participated in their study, using a virtual reality system and a virtual reality learning program. A partial least squares structural equation modeling (PLS-SEM) analysis was employed to test the extended VR technology acceptance model, which was the same data analysis method used in our study. Their conceptual model as shown in Fig. 3 included exogenous variables such as Imagination (IMAG), Immersion (IMRN), and Interaction (INTR), with Perceived Ease of Use (PEU) and Perceived Usefulness (PU) mediating Intention to Use (ITU). The hypotheses in Study 2 follow exactly from Barrett et al. [3] are as follows:

H1: PU will have a strong, positive, and significant effect on learners' intention towards using the VR system.

H2: PEU will have a weak, positive, and nonsignificant effect on learners' intention towards using the high-immersion VR system.

H3: Immersion will be a strong, positive, and significant predictor for PU.

H4: Immersion will be a positive and significant predictor for PEU.

H5: Interaction will be a strong, positive, and significant determiner for PEU.

H6: Interaction will be a positive and non-significant predictor for PU.

H7: Imagination will exhibit a medium to large, positive, and significant effect on PU.

H8: Imagination will exhibit a medium, positive, and significant effect on PEU.

Table 1
Demographics summary of samples in Study 1.

Measure	Item	N (295)	%
Gender	Male	130	44.1 %
	Female	164	55.6 %
	No-binary	1	0.3 %
Age	18	7	2.4 %
	19	57	19.3 %
	20	76	25.8 %
	21	61	20.7 %
	22	63	21.4 %
	23	31	10.5 %
Year	1	74	25.1 %
	2	78	26.4 %
	3	76	25.8 %
	4	67	22.7 %
ChatGPT Experience	0	48	16.3 %
	1	74	25.1 %
	2	64	21.7 %
	3	65	22.0 %
	4	44	14.9 %

Data collection for study 2

In Study 2, the data collection method was identical to that in Study 1. It involved utilizing ChatGPT to create twenty student samples based on a defined student population profile. The profile included variables such as age, gender, majors, English ability, and AR experience.

We prompted ChatGPT with both AR and Technology Acceptance Model (TAM) questionnaires. The inclusion of the AR questionnaires aimed to assess how ChatGPT's inner model was able to comprehend the concepts of immersion (IMRM), imagination (IMGM), and interaction (INTR), which differed from the constructs of perceived usefulness (PU), perceived ease of use (PEU), and behavioural intention (BI) in the TAM questionnaire. By examining these responses, we can assess how ChatGPT's understanding of AR and its alignment with the theory of technology acceptance model.

Similarly, this study utilized a 7-point Likert scale, where a rating of 1 indicated 'strongly disagree', 2 represented 'disagree', 3 denoted 'somewhat disagree', 4 indicated 'neither agree nor disagree', 5 represented 'somewhat agree', 6 denoted 'agree', and a rating of 7 indicated 'strongly agree'.

Similar to study 1, the process was repeated twenty times, resulting in a total of 400 samples. Each iteration involved ChatGPT generating twenty rows of responses based on a randomly generated student profile. Appendix B depicts the specific prompt used in this study. The constructs and background information were adopted directly from Barrett et al. [3] to make meaningful comparisons.

Results

Study 1

Data analysis was conducted using SmartPLS4 [36] and SPSS 26 [44]. Table 1 illustrates the demographic distribution of samples in Study 1. During the sample generation process, the likelihood of ChatGPT's generating samples that deviate from the prescribed criteria was low. Note that an instance occurred in Study 1 where ChatGPT generated a sample with non-binary gender. After eliminating duplicate samples, Study 1 comprised 295 distinct data points, exhibiting an uneven gender distribution with 55.6 % females. Most of the generated participants were in the age range of 20 to 22, and the distribution of their years in university was fairly uniform. Participants' experiences with ChatGPT varied, ranging from a minimum rating of 0 (16.3 %) to a maximum rating of 4 (14.9 %), simulating a diverse range of users.

Table 2
Items loadings and constructs reliability in Study 1.

Construct	Loading	Cronbach's alpha	Composite reliability (rho_a)	Composite reliability (rho_c)	Average variance extracted (AVE)
BI1	0.982	0.960	0.965	0.98	0.961
BI2	0.979				
CPLAY1	0.947	0.959	0.96	0.97	0.891
CPLAY2	0.946				
CPLAY3	0.927				
CPLAY4	0.955				
PEOU1	0.906	0.968	0.971	0.974	0.862
PEOU2	0.941				
PEOU3	0.929				
PEOU4	0.933				
PEOU5	0.926				
PEOU6	0.937				
PU1	0.958	0.973	0.973	0.978	0.88
PU2	0.930				
PU3	0.933				
PU4	0.941				
PU5	0.918				
8PU6	0.947				

Table 3
The Heterotrait–Monotrait ratio of correlations (HTMT) in Study 1.

Construct	BI	CPLAY	PEOU
CPLAY	0.846		
PEOU	0.769	0.86	
PU	*0.93*	0.889	0.861

Table 2 illustrates the loading of items and construct reliability in Study 1. The loading and reliability values consistently exceeded the acceptable threshold of 0.7, indicating strong convergence validity [18]. Factor loading values in Study 1 were high, suggesting that the selected items effectively represented and measured the underlying constructs. Cronbach's alpha values for constructs in Study 1 showed strong internal consistency: Behavioral Intention (BI) at 0.960, Computer Playfulness (CPLAY) at 0.959, Perceived Ease of Use (PEOU) at 0.968, and Perceived Usefulness (PU) at 0.973. These scores demonstrate the reliability of the measurement scales for each construct.

In this study, we utilized the Heterotrait–Monotrait ratio of correlations (HTMT) to evaluate discriminant validity among the primary constructs. Table 3 displays the HTMT values for the relationships between Behavioral Intention (BI), Computer Playfulness (CPLAY), Perceived Ease of Use (PEOU), and Perceived Usefulness (PU). While most HTMT values hover around the threshold of 0.85, the correlation between Perceived Usefulness (PU) and Behavioral Intention (BI) produced an HTMT value of 0.93, slightly exceeding the recommended threshold. This finding prompts a potential need for further refinement of the measurement model to ensure robust discriminant validity.

Table 4 and Fig. 4 display the structural path coefficients and overall structural model from the SmartPLS analysis in Study 1. The coefficients are presented for the original sample (O), sample mean (M), and standard deviation (STDEV), along with T statistics (|O/STDEV|) and P values. The results indicated positive relationships between Computer Playfulness (CPLAY) to both Perceived Ease of Use (PEOU) and Perceived Usefulness (PU). Moreover, the path from Perceived Usefulness (PU) to Behavioral Intention (BI) was strong. However, the relationship between Perceived Ease of Use (PEOU) and Behavioral Intention (BI) was negative and lacked statistical significance. These findings offer valuable quantitative insights into the relationships among the studied constructs. When comparing the correlation values and path coefficients with the meta-analysis conducted by Yousafzai et al. [52], Study 1 exhibited correlation coefficients that fell towards the higher end of the range, while the path coefficients exceeded the upper limit.

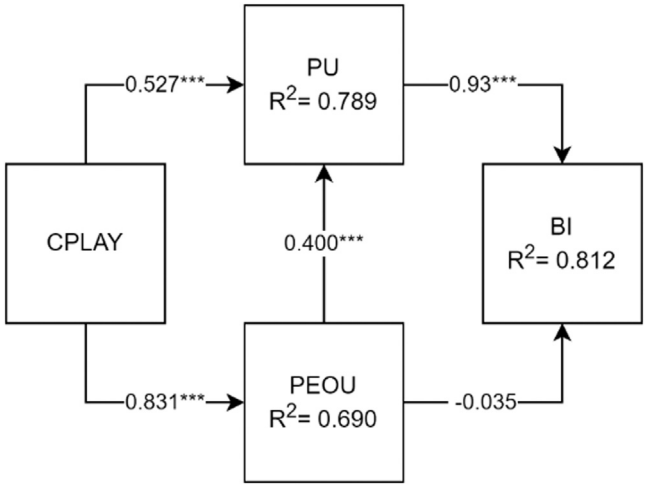


Fig. 4. Structure model in Study 1.
$^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$

Study 2

In Study 2, the demographic profile of the generated sample, outlined in Table 5 after eliminating duplicates, encompasses 240 unique participants. The gender distribution is 37.5 % male and 62.5 % female, representing a ratio of 1 to 2. The age distribution displays a balanced spread, with 37.1 % at 18, 32.1 % at 19, and 30.8 % at 20. In terms of English proficiency, 6.3 % rated themselves at level 1, while 34.6 % assessed themselves at levels 2 and 3 each, and 24.6 % at level 4. Here, level 1 indicates low English ability, and level 4 signifies strong English ability. Concerning familiarity with virtual reality (VR), 13.8 % positioned themselves at level 1, 25.4 % at level 2, 33.8 % at level 3, and 27.1 % at level 4.

Table 6 presents the loading and construct reliability in Study 2, where the SmartPLS method was utilized to assess various constructs, including Imagination (IMGM), Immersion (IMRN), Interaction (INTR), Perceived Usefulness (PU), Perceived Ease of Use (PEU), and Intention to Use (ITU). The assessment involved factor loadings, Cronbach's alpha, composite reliability (rho_a), composite reliability (rho_c), and average variance extracted (AVE).

Table 4
Structure paths coefficients in Study 1.

Path	Original sample (O)	Sample mean (M)	Standard deviation (STDEV)	T statistics (|O/STDEV|)	P values
CPLAY->PEOU	0.831	0.831	0.018	44.916	0.00
CPLAY -> PU	0.527	0.527	0.051	10.295	0.00
PEOU -> BI	−0.035	−0.036	0.049	0.71	0.477
PEOU -> PU	0.4	0.401	0.056	7.178	0.00
PU -> BI	0.93	0.931	0.042	21.902	0.00

Table 5
Demographics summary of samples in Study 2.

Measure	Item	N (240)	%
Gender	Male	90	37.5 %
	Female	150	62.5 %
Age	18	89	37.1 %
	19	77	32.1 %
	20	74	30.8 %
English Ability	1	15	6.3 %
	2	83	34.6 %
	3	83	34.6 %
	4	59	24.6 %
Familiar with VR	1	33	13.8 %
	2	61	25.4 %
	3	81	33.8 %
	4	65	27.1 %

In Table 6, factor loadings convey the strength of the relationship between items and their respective constructs, ranging from 0.759 to 0.962. Reliability measures, including Cronbach's alpha and composite reliability, provide insights into the internal consistency of constructs. For example, IMGM exhibits a Cronbach's alpha of 0.85, rho_a of 0.863, rho_c of 0.909, and AVE of 0.769. Similarly, ITU displays a Cronbach's alpha of 0.783, rho_a of 0.783, rho_c of 0.873, and AVE of 0.697, indicating high reliability for both constructs.

The acceptable criterion for factor loading is values above 0.7, signifying a strong relationship between items and constructs. Additionally, a Cronbach's alpha above 0.7 is deemed acceptable for reliability, ensuring internal consistency. Composite reliability values above 0.7 further indicate good reliability. All loading and reliability values were within the acceptable range, suggesting good convergence validity [18]. Loading values of TAM constructs in Study 2 appeared to be slightly lower than in Study 1.

Table 7 provides a detailed examination of the Heterotrait–Monotrait ratio of correlations (HTMT) in Study 2. The HTMT is a metric for assessing discriminant validity among constructs in the Technology Acceptance Model (TAM). However, an observation in Table 7 raised concerns, as two values among the TAM constructs exceeded 1, and one value exceeded 0.9. This suggests a potential issue with discriminant validity, indicating that certain constructs may not be sufficiently distinct from one another. A close examination revealed that the TAM

Table 6
Items loadings and constructs reliability in Study 2.

Construct	Loading	Cronbach's alpha	Composite reliability (rho_a)	Composite reliability (rho_c)	Average variance extracted (AVE)
IMGM1	0.848	0.85	0.863	0.909	0.769
	(0.831)	(0.837)			(0.754)
IMGM2	0.923				
	(0.896)				
IMGM3	0.857				
	(0.883)				
IMRN1	0.962	0.922	0.925	0.951	0.865
	(0.852)	(0.835)			(0.752)
IMRN2	0.898				
	(0.879)				
IMRN3	0.93				
	(0.871)				
INTR1	0.856	0.83	0.839	0.898	0.745
	(0.779)	(0.715)			(0.634)
INTR2	0.861				
	(0.755)				
INTR3	0.872				
	(0.851)				
ITU1	0.837	0.783	0.783	0.873	0.697
	(0.912)	(0.858)			(0.778)
ITU2	0.823				
	(0.874)				
ITU3	0.845				
	(0.860)				
PEU1	0.846	0.761	0.78	0.862	0.675
	(0.886)	(0.872)			(0.723)
PEU2	0.858				
	(0.856)				
PEU3	0.759				
	(0.847)				
PU1	0.874	0.719	0.722	0.877	0.78
	(0.911)	(0.801)			(0.716)
PU2	0.893				
	(0.873)				

Note: The bottom values were taken from Barrett et al. [3] for comparison.

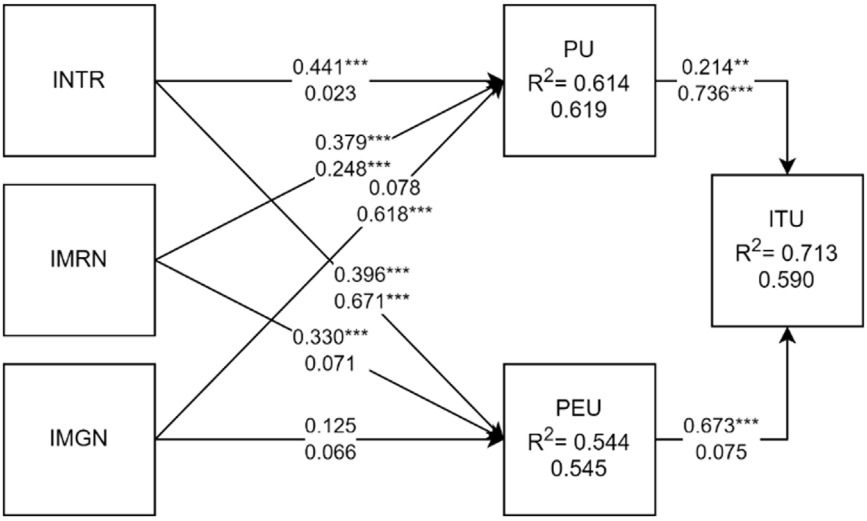


Fig. 5. Structural Model in Study 2.
Note: The bottom values were taken from Barrett et al. [3] for comparison. * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Table 7
The Heterotrait–Monotrait ratio of correlations (HTMT) in Study 2.

Construct	IMGM	IMRN	INTR	ITU	PEU
IMRN	0.841				
INTR	0.627	0.633			
ITU	0.822	0.823	0.749		
PEU	0.703	0.757	0.802	1.065	
PU	0.76	0.841	0.89	0.952	1.006

constructs' wordings in Study 1 maintained the standard phrasing from Davis [10], whereas, in Study 2, the wordings differed from the standard TAM, which could contribute to the observed variations in the HTMT ratios.

Table 8 and Fig. 5 depict the structural model and path coefficients in Study 2. Most of the path coefficients were significant except IMGM -> PEU and IMGM -> PU. R-squared values indicated a higher degree of variance explained in Study 2, suggesting a good overall fit.

For the TAM construct, Study 2's correlation coefficients were within the meta-analysis range, but the PEU->ITU path coefficient exceeded the upper range.

Discussion

This study centres on evaluating ChatGPT's processing of diverse constructs within the provided TAM theories and appraising the validity of relationships between different constructs using GPT-generated survey responses. This assessment is conducted through structural equation models in two studies. During the data gathering, we noted a tendency for ChatGPT to generate duplicated responses in both Study 1 and Study 2. The duplicate response rate was found to be 26.25 % in Study 1 and 40 % in Study 2. To ensure data quality and avoid redundancy, these duplicate responses were removed, resulting in a usable sample size of 295 for Study 1 and 240 for Study 2.

For the experiments, we selected a response length of 20 rows to overcome limitations encountered while manually working with ChatGPT. The slow response and frequent interruptions of the ChatGPT platform when reaching response limits necessitated the selection of a smaller response size. Despite these challenges, the selected response length enabled the collection of data for analysis and efficient data generation.

First, results of the two studies provided an evidence that ChatGPT can generate sample responses that align with the Technology Acceptance Model (TAM), as reflected in the loading and reliability analysis presented in Tables 2 and Table 6. Both Study 1 and Study 2 demonstrate valid models with high R-squared values of 82 % and 71 %, respectively. All the hypotheses in Study 1 were supported except H4. In Study 2, hypotheses that were not supported are H2, H6, H7 and H8.

It is interesting to note that in Study 2, the items PU3 and PEU4 contained negative wording, resulting in low loadings of 0.29 and 0.57, respectively, and they were subsequently removed from the measurement model. Further investigation is needed to understand how negative wording items impact ChatGPT's responses [38].

Second, despite the presence of high correlations among the constructs in Study 1 (see Table C3), the square roots of Average Variance Extracted (AVE) still exceeded the corresponding items in the correlation matrix, satisfying the Fornell-Larcker criterion [12]. This finding suggests that the indicators within each construct exhibit stronger internal consistency and stronger relationships with their respective constructs compared to other constructs. In Study 2, the intercorrelation coefficients among the items mostly fall within the moderate range, as shown in Table C4. Additionally, Study 2 also met the Fornell-Larcker criterion, indicating satisfactory discriminant validity. However, to fully assess the discriminant validity of the measurement model, the HTMT analysis should also be considered. The HTMT analysis revealed there was a tendency for high intercorrelation coefficients among the TAM constructs, which may impact ChatGPT's ability to discriminate between the constructs. In Study 1 the collinearity statistics (VIF) was between 1 and 4.7 while in Study 2 it was between 1.5 to 2.4. Although a VIF value of not larger than five suggests that low collinearity exists among the TAM constructs [18], additional investigation into the TAM constructs is still warranted.

Third, Study 1 exhibited higher reliability values compared to Study 2, as shown in Table 2 and Table 6. The wording style for the perceived usefulness (PU) statements in the two studies was not the same. In Study 1, the statements focused on the use of ChatGPT in a learning context, emphasizing benefits such as efficiency, performance improvement, productivity, and usefulness. These statements demonstrated a higher level of cohesion and a specific thematic focus, and the wording followed the standard TAM constructs. Conversely, the PU statements in Study 2 centred around the impact of a Virtual Reality (VR) system on understanding paragraph structure and its potential to enhance academic writing. While the statements in Study 2 were more specific, they also covered a narrower aspect compared to the broader scope in Study 1. The wordings were different from standard TAM constructs.

Fourth, when asking ChatGPT to assess the similarity between the items in the two sets of PU constructs, ChatGPT assigned a score of 8 in Study 1 and a score of 6 in Study 2, based on the construct items provided. This suggests that ChatGPT perceived greater similarity among the items in Study 1 which may contribute to the higher reliability val-

Table 8
Structure paths coefficients in Study 2.

Path	Original sample (O)	Sample mean (M)	Standard deviation (STDEV)	T statistics (|O/STDEV|)	P values
IMGM -> PEU	0.125	0.123	0.068	1.845	0.065
IMGM -> PU	0.078	0.072	0.091	0.858	0.391
IMRN -> PEU	0.33	0.335	0.073	4.543	0.000
IMRN -> PU	0.379	0.383	0.09	4.239	0.000
INTR -> PEU	0.396	0.397	0.058	6.819	0.000
INTR -> PU	0.441	0.443	0.05	8.798	0.000
PEU -> ITU	0.673	0.661	0.078	8.672	0.000
PU -> ITU	0.214	0.22	0.068	3.129	0.002

ues. In summary, the differences in wording style and thematic focus between the PU statements in Study 1 and Study 2, along with ChatGPT's scoring render further investigation.

Similarly, IMRN has a higher reliability value than INTR. We asked ChatGPT to determine the level of similarity between the items in the IMRN and INTR constructs. Here is ChatGPT's response: "Considering that all three statements in IMRN revolve around the subjective experience of immersion and engagement in the VR learning environment, I would assign a score of 8 out of 10 for the level of similarity. The statements share a common theme and address similar aspects of the VR experience.

INTR, on the other hand, consists of statements that focus on different aspects of interaction and ease of use within the VR system, such as identifying paragraph features, selecting colours, and browsing help menus. As a result, the level of similarity between the statements in INTR is lower. I would assign a score of 5 out of 10 for the level of similarity within INTR." In the context of prototyping theory, ChatGPT can assist in evaluating constructs that are not clearly defined or unambiguous [28].

Fifth, when we compared the structural models of Study 2 and the reference structural model as shown in Fig. 5, there were both similarities and differences between the structural models. The R-squared values indicate how well the models fit the data. In this case, the R-squared values were similar for both models, except for the "ITU" variable. ChatGPT had a higher R-squared value for "ITU" (0.71) compared to the reference model (0.59). This suggests that ChatGPT's model may better explain the variance in the "ITU" variable. R-squared values for PU and PEU were similar for the two models. However, path coefficients were not entirely similar. PEU→ITU was statistically significant for the simulated model but not significant in the reference model. In terms of the AR constructs, there were differences between the two models. Specifically, the relationships INTR→PU, IMRN→PEU, and IMGM→PU diverged between the models. These disparities may indicate that ChatGPT perceives the concepts related to the flow of augmented reality (AR) differently from the reference model. Another factor that may contribute to the difference was the presentation of the questionnaire. In the reference paper, all items were presented randomly, whereas, in this study, a sequential presentation of construct items was used.

Sixth, there were biases in the generated samples in terms of gender and ChatGPT experiences in Study 1. ChatGPT tended to generate more female students with higher ChatGPT experiences. As a result, the constructs' responses of PU and BI from these female students were significantly higher compared to the male samples. In Study 2, ChatGPT again generated more female students with higher English ability and AR experiences. Consequently, the constructs' responses from male students were lower compared to female students. Refer to Appendix C for the descriptive statistics in Study 1 and Study 2. These sample biases in both studies highlight the potential influence of gender and prior experiences on the constructs' responses generated by ChatGPT. It is important to consider and control for these biases when interpreting the conceptual capability of ChatGPT.

Overall, the experiments demonstrated that ChatGPT can generate relevant responses aligned with the TAM constructs which demonstrated the nomological validity of the constructs within the TAM framework based on the empirical evidence gathered from the study. The measurement models exhibited good validity, and while there were some challenges in discriminant validity due to high intercorrelations, the findings provided valuable insights into the abilities of ChatGPT to comprehend, discriminate and associate the relationship between theoretical constructs.

Practical implications

The research holds important implications for AI software developers interested in leveraging ChatGPT for theory prototyping.

First, there is an opportunity to integrate ChatGPT into software applications specifically designed for theory prototyping. By incorporating ChatGPT as a tool for generating responses aligned with theoretical constructs, developers can contribute to the development of interactive and intelligent systems for theory development.

Second, there is a need for dedicated platforms or software tools that harness the capabilities of ChatGPT for theory prototyping. Developers can design user-friendly interfaces and workflows that streamline the process, allowing researchers to create theoretical constructs, generate responses, and analyse results efficiently. Such platforms have the potential to accelerate the pace of knowledge advancement by providing researchers with a valuable tool for theory exploration and refinement.

Furthermore, the identification of biases in ChatGPT-generated samples highlights the responsibility of software developers to address and mitigate biases in their applications. Collaboration with researchers is crucial to implement strategies that enhance diversity and reduce biases in the responses generated by ChatGPT. By curating data, making algorithmic adjustments, and ensuring ongoing monitoring, developers can ensure that the outputs of the software are fair, unbiased, and reflective of diverse perspectives.

Lastly, prompt design considerations play a significant role in the outcomes of theory prototyping. Software developers, in collaboration with researchers, can establish guidelines and best practices for prompt design to improve the reliability and validity of generated responses. Clear instructions, contextual clarity, and minimization of ambiguity are essential factors that developers can focus on to enhance the quality of the data and support researchers in deriving meaningful insights.

Conclusion and future research

The study presented in this paper is not intended to be exhaustive but rather aims to open conversation for future research and underscore the potential of utilizing ChatGPT or LLMs as a benchmark tool for theory prototyping. The results indicate that ChatGPT can generate responses that align with the theoretical constructs of TAM, demonstrating its ability to process complex concepts. The high R-squared values obtained in the two experimental studies demonstrated the substantial explanatory power of the proposed models, which indicates that ChatGPT can capture and represent the underlying relationships among constructs. This paves the way for exploring and testing theories with ChatGPT as human participants in a simulated environment, which can save time and resources compared to traditional methods in behavioural research.

This research has identified several limitations that warrant further investigation. First, the presence of data duplications [41] poses a concern, potentially impacting the accurate evaluation of ChatGPT as a theory prototyping benchmark tool. The issues related to duplicated responses can be addressed through methodological refinements. Researchers can develop techniques to identify and filter out duplicated responses, ensuring data quality and integrity. This will enhance the reliability of findings and strengthen the validity of the conclusions drawn from the analysis. Second, variations in prompts such as correlation specification, negative wordings and priming used on ChatGPT may influence the generated survey responses, potentially introducing bias or inconsistencies [33] which needs further investigations. Third, it is important to note that the main theory examined in this research is the technology acceptance model specifically in the context of augmented reality (AR) and learning with ChatGPT. Further research is needed to validate the applicability and generalizability of the findings in other contexts beyond the scope of this research.

In addition to research issues triggered by these limitations, there are three interesting perspectives for future research. First, the validation of constructs, including factors loading, internal consistency reliability, and convergence reliability, provides a foundation for further investigation in this direction. In other words, researchers can build upon validated constructs to explore various theoretical frameworks and test hypotheses in a controlled and interactive manner. This has the potential to enhance the efficiency and effectiveness of theory development and refinement. Moreover, of most importance, the identification of potential biases in ChatGPT-generated samples presents an opportunity for further exploration. Understanding and mitigating these biases can contribute to the development of more robust and reliable models. Researchers can investigate feasible ways to enhance the diversity and representativeness of the generated responses, ensuring that the findings are applicable and generalizable across various demographic and contextual factors. Finally, future research can delve into more complex and nuanced theoretical models, expanding the scope of theory prototyping using LLMs.

Availability of data and material

Data is available on request from the author.

Funding

Not Applicable.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Tiong-Thye Goh: Writing – original draft, Methodology, Formal analysis, Data curation, Conceptualization. **Xin Dai:** Writing – original draft, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Yanwu Yang:** Writing – original draft, Project administration, Investigation, Formal analysis, Conceptualization.

Acknowledgements

The initial version of this manuscript was pre-printed on arXiv on June 4, 2023 (https://arxiv.org/abs/2307.05488).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.tbench.2024.100153.

References

[1] Ajitesh, K. (2023). *ChatGPT prompts design tips & examples*. Retrieved 25-06 from https://vitalflux.com/chatgpt-prompts-design-tips-examples/.
[2] I. Ajzen, Understanding attitudes and predicting social behavior, Englewood cliffs (1980).
[3] A.J. Barrett, A. Pack, E.D. Quaid, Understanding learners' acceptance of high-immersion virtual reality systems: insights from confirmatory and exploratory PLS-SEM analyses, Comput. Educ. 169 (2021) 104214.
[4] Brunet-Gouet, E., Vidal, N., & Roux, P. (2023). *Do conversational agents have a theory of mind? a single case study of chatgpt with the hinting, false beliefs and false photographs, and strange stories paradigms*. https://hal.science/hal-03991530.
[5] A. Chan, GPT-3 and InstructGPT: technological dystopianism, utopianism, and "contextual" perspectives in AI ethics and industry, AI. Ethics 3 (1) (2023) 53–64.
[6] E.Y. Chang, Prompting large language models with the socratic method, 2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC), 2023.
[7] G.W. Cheung, H.D. Cooper-Thomas, R.S. Lau, L.C. Wang, Reporting reliability, convergent and discriminant validity with structural equation modeling: a review and best-practice recommendations, Asia Pacific J. Manage. (2023), doi:10.1007/s10490-023-09871-y.
[8] L.J. Cronbach, Coefficient alpha and the internal structure of tests, Psychometrika 16 (3) (1951) 297–334, doi:10.1007/BF02310555.
[9] F.D. Davis, A Technology Acceptance Model for Empirically Testing New End-User Information systems: Theory and Results, Massachusetts Institute of Technology, 1985].
[10] F.D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS quarterly (1989) 319–340.
[11] D. Dillion, N. Tandon, Y. Gu, K. Gray, Can AI language models replace human participants? Trends Cogn. Sci. (Regul. Ed.) 27 (7) (2023) 597–600, doi:10.1016/j.tics.2023.04.008.
[12] C. Fornell, D.F. Larcker, Evaluating structural equation models with unobservable variables and measurement error, J. Market. Res. 18 (1) (1981) 39–50.
[13] G. Franke, M. Sarstedt, Heuristics versus statistics in discriminant validity testing: a comparison of four procedures, Internet Res. 29 (3) (2019) 430–447, doi:10.1108/IntR-12-2017-0515.
[14] T.T. Goh, B. Yang, The role of e-engagement and flow on the continuance with a learning management system in a blended learning environment, Int. J. Educ. Technol. High. Educ. 18 (1) (2021) 49, doi:10.1186/s41239-021-00285-8.
[15] Gozalo-Brizuela, R., & Garrido-Merchan, E.C. (2023). ChatGPT is not all you need. a state of the art review of large generative AI models. *arXiv preprint arXiv:*2301.04655.
[16] A. Granić, Educational technology adoption: a systematic review, Educ. Inf. Technol. (Dordr) 27 (7) (2022) 9725–9744.
[17] A. Granić, N. Marangunić, Technology acceptance model in educational context: a systematic literature review, Br. J. Educ. Technol. 50 (5) (2019) 2572–2593, doi:10.1111/bjet.12864.
[18] J.F. Hair Jr, G.T.M. Hult, C.M. Ringle, M. Sarstedt, N.P. Danks, S Ray, Partial Least Squares Structural Equation Modeling (PLS-SEM) Using R: A workbook, Springer Nature, 2021.
[19] A. Haleem, M. Javaid, M.A. Qadri, R. Suman, Understanding the role of digital technologies in education: a review, Sustain. Oper. Comput. 3 (2022) 275–285, doi:10.1016/j.susoc.2022.05.004.
[20] Hartmann, J., Schwenzow, J., & Witte, M. (2023). The political ideology of conversational AI: converging evidence on ChatGPT's pro-environmental, left-libertarian orientation. *arXiv preprint arXiv:*2301.01768.
[21] J. Henseler, C.M. Ringle, M. Sarstedt, A new criterion for assessing discriminant validity in variance-based structural equation modeling, J. Acad.Market. Sci. 43 (2015) 115–135.
[22] Jiang, G., Xu, M., Zhu, S.C., Han, W., Zhang, C., & Zhu, Y. (2022). MPI: evaluating and Inducing personality in pre-trained language models. *arXiv preprint arXiv:*2206.07550.
[23] X. Jiang, T.T. Goh, X. Chen, M. Liu, B. Yang, Investigating university students' online proctoring acceptance during COVID-19: an extension of the technology acceptance model, Aust. J. Educ. Technol. 39 (2023) 47–64, doi:10.14742/ajet.8121.
[24] X. Jiang, T.T. Goh, M. Liu, On students' willingness to use online learning: a privacy calculus theory approach [original research], Front. Psychol. 13 (2022), doi:10.3389/fpsyg.2022.880261.
[25] D. Kaur, S. Uslu, K.J. Rittichier, A. Durresi, Trustworthy artificial intelligence: a review, ACM Comput. Surv. 55 (2) (2022) Article 39, doi:10.1145/3491209.
[26] Kosinski, M. (2023). Theory of mind may have spontaneously emerged in large language models. *arXiv preprint arXiv:*2302.02083.
[27] L.S. Lo, The CLEAR path: a framework for enhancing information literacy through prompt engineering, J. Acad. Librariansh. 49 (4) (2023) 102720, doi:10.1016/j.acalib.2023.102720.
[28] S.B. MacKenzie, P.M. Podsakoff, N.P. Podsakoff, Construct Measurement and Validation Procedures In MIS and behavioral research: integrating new and existing techniques, MIS Q. 35 (2) (2011) 293–334, doi:10.2307/23044045.
[29] Madelyn, G. (2022). *Can You generate realistic data with GPT-3? We explore fake dating with fake data*. Retrieved 20-06 from https://www.tonic.ai/blog/can-you-generate-realistic-data-with-gpt-3.
[30] Mishra, S., Khashabi, D., Baral, C., Choi, Y., & Hajishirzi, H. (2021). Reframing instructional prompts to GPTk's language. *arXiv preprint arXiv:*2109.07830.
[31] J.P. Peter, Construct Validity: a review of basic issues and marketing practices, J. Market. Res. 18 (2) (1981) 133–145, doi:10.1177/002224378101800201.

[32] J.P. Peter, G.A. Churchill, Relationships among research design choices and psychometric properties of rating scales: a meta-analysis, J. Market. Res. 23 (1) (1986) 1–10, doi:10.2307/3151771.
[33] Ramlochan, S. (2023). *Unlocking AI with priming: enhancing context and conversation in LLMs like ChatGPT*. Retrieved 05-06-2023 from https://www.promptengineering.org/unlocking-ai-with-priming-enhancing-context-and-conversation-in-llms-like-chatgpt/.
[34] Rao, H., Leung, C., & Miao, C. (2023). Can chatgpt assess human personalities? a general evaluation framework. *arXiv preprint arXiv:*2303.01248.
[35] P.P. Ray, ChatGPT: a comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope, Internet Things Cyber-Phys. Syst. (2023).
[36] Ringle, C.M., Wende, S., & Becker, J.M. (2022). *SmartPLS 4. oststeinbek: smartPLS*. https://www.smartpls.com.
[37] F. Rondan-Cataluña, J. Arenas-Gaitán, P. Ramírez-Correa, A comparison of the different versions of popular technology acceptance models: a non-linear perspective, Kybernetes (2015) 44, doi:10.1108/K-09-2014-0184.
[38] M.J. Roszkowski, M. Soven, Shifting gears: consequences of including two negatively worded items in the middle of a positively worded questionnaire, Assess. Eval. High. Educ. 35 (1) (2010) 113–130.
[39] D. Rozado, The political biases of chatgpt, Soc. Sci. 12 (3) (2023) 148.
[40] R. Scherer, F. Siddiq, J. Tondeur, The technology acceptance model (TAM): a meta-analytic structural equation modeling approach to explaining teachers' adoption of digital technology in education, Comput. Educ. 128 (2019) 13–35, doi:10.1016/j.compedu.2018.09.009.
[41] Schwab, P.N. (2023). *ChatGPT: 1000 texts analyzed and up to 75,3% similarity*. Retrieved 05-06-2023 from https://www.intotheminds.com/blog/en/chatgpt-similarity-with-plan/.
[42] A. Shafeeg, I. Shazhaev, D. Mihaylov, A. Tularov, I. Shazhaev, Voice assistant integrated with chat GPT, Ind. J. Comput. Sci. 12 (1) (2023).
[43] M. Shidiq, The use of artificial intelligence-based chat-gpt and its challenges for the world of education; from the viewpoint of the development of creative writing skills, Proceeding of International Conference on Education, Society and Humanity, 2023.
[44] I. Statistics, IBM Corp. Released 2019. IBM SPSS Statistics for Windows, Version 26.0, IBM Corp. Google Search, Armonk, NY, 2019.
[45] V. Sukackė, Towards extending the original technology acceptance model (tam) for a better understanding of educational technology adoption. society. integration. education, in: Proceedings of the International Scientific Conference, 2019.
[46] V. Venkatesh, H. Bala, Technology acceptance model 3 and a research agenda on interventions, Decis. Sci. 39 (2) (2008) 273–315.
[47] J. Webster, J.J. Martocchio, Microcomputer playfulness: development of a measure with workplace implications, MIS Q. 16 (2) (1992) 201–226, doi:10.2307/249576.
[48] C.E. Werts, D.R. Rock, R.L. Linn, K.G. Jöreskog, A general method of estimating the reliability of a composite, Educ. Psychol. Meas. 38 (4) (1978) 933–938, doi:10.1177/001316447803800412.
[49] Wong, J., & Kim, J. (2023). ChatGPT is more likely to be perceived as male than female. *arXiv preprint arXiv:*2305.12564.
[50] Y. Yang, C. Zhang, K. Zhao, Q. Wang, The shifting role of information processing and management in interdiscipline development: from a collection of tools to a crutch? Inf. Process. Manage. 60 (4) (2023) 103388.
[51] Yaroslav, S. (2023). *The power of prompting: unleashing the full potential of ChatGPT*. Retrieved 25-06 from https://yarspirin.hashnode.dev/the-power-of-prompting-unleashing-the-full-potential-of-chatgpt.
[52] S.Y. Yousafzai, G.R. Foxall, J.G. Pallister, Technology acceptance: a meta-analysis of the TAM: part 2, J. Model. Manage. 2 (3) (2007) 281–304.