An Imageless Internet

Joseph Hong
Computer Science, New York University Abu Dhabi
joseph.hong@nyu.edu

Advisors

Yasir Zaki
yasir.zaki@nyu.edu

Matteo Varvello
varvello@gmail.com

ABSTRACT

In recent years, there has been growing concern over the environmental impact of the internet, particularly the energy consumption and carbon emissions associated with data storage and transmission. One potential solution to reduce the carbon footprint of the internet is to replace images with generated images using machine learning algorithms. By generating images on demand instead of storing and transmitting images, the energy and carbon emissions associated with them could be reduced, alongside page load times as clients would no longer need to wait for an entire image to be downloaded. This paper is a feasibility study of how using image generation techniques can address these issues, and is also an exploration of the benefits that dynamically generated images can bring to the internet, both in terms of energy savings and speed.

1 INTRODUCTION

Images have always been a foundational and major form of communication and information exchange, and since their digitization they have been an essential part of web pages. With the advent of high-resolution displays and the growing demand for visually appealing content, images have undergone a transformation. They have evolved from small, compressed files to larger, high-quality images that showcase intricate details and vivid colors [1]. While this has undoubtedly enhanced the user experience and enabled more stimulating visual content, it has also given rise to several issues.

One of the key challenges lies in the energy consumption and load times associated with image-heavy web pages, particularly in low-bandwidth environments [2]. Traditional approaches rely on content delivery networks (CDNs) to store and transmit images from servers to end-users [3]. However, the larger file sizes of high-quality images have led to increased energy consumption in data centers and CDNs, which are responsible for storing and serving images to users across the internet.

CDNs have had to allocate more storage space and computational resources to accommodate the growing demand [4]. This translates to higher energy consumption by the servers that store and transmit these images, as well as the cooling systems required to maintain optimal operating conditions [5]. The transmission of these larger images also incurs costs in terms of network bandwidth and data transfer. Transmitting image packets from servers to end-users requires the packet to be passed through routers from one end to the other, which results in increased energy consumption— lower bandwidth and further distance from the CDN server can additionally lead to slower page load times [2]. As the demand for high-quality images continues to rise, these costs become even more substantial, both economically and environmentally [5].

Consequently, addressing the energy consumption and efficiency challenges associated with image generation and transmission has become a pressing concern. A promising solution is to shift the image generation process to the client-side, using the client's graphic hardware in order to generate images on the fly. This paper is a feasibility study of how using image generation techniques can address these issues, and is also an exploration of the benefits that dynamically generated images can bring to the internet, both in terms of energy savings and speed. In addition, this paper provides the framework for tools that could assist in this solution, such as a machine learning model that can provide optimal parameters for image generation given a prompt.

2 BACKGROUND

Over the past decade, the number of image requests per page has shown little to no change, with an approximate 9 percent increase between 2012 and 2023, currently at a median of 21 requests. However, the total size of images on webpages has increased dramatically, approximately 300 percent at 1,000 KB, in the same period. This means that, given today's average size of 2300 KB per webpage, images make up nearly 44 percent of a webpage's size. It is important to note that the number of image requests has *decreased* in the past decade, from an average of 42 to 21 requests per webpage [6]. This implies that, though the number of images per page may have decreased, their size— and in tandem their resolution —has increased.

It is not the first time that research into decreasing image sizes has been conducted, but to our knowledge the use of image generation techniques to replace images with a line of text has not yet been explored as an area of internet optimization.

2.1 The Generative Learning Trilemma

In the past, image generation techniques were not considered as viable candidates for reducing image sizes due to a fundamental challenge known as the generative learning trilemma. This trilemma arises from the trade-off between image quality, sampling speed, and the diversity of generated images [7].

Traditionally, the focus of image compression techniques was to reduce image sizes while preserving image quality as much as possible. However, the generative learning trilemma suggests that achieving high-quality and diverse image generation simultaneously while keeping the sampling time low is an inherently difficult task. In the context of the internet, where high-quality samples are preferred, image generation models either prioritized sample diversity, as with flow and diffusion models, or sampling speed, as with general adversarial networks (GANs), but never were able to achieve both [8-10]. Thus, this approach was not suitable for replacing images on the internet.

However, recent advancements in machine learning, particularly latent and stable diffusion [11], have shown promise in addressing the generative learning trilemma to some extent. These algorithms strike a balance between image quality, diversity, and sampling speed, making them viable options for generating images with reduced energy consumption and high-quality samples, all the while being computationally inexpensive and fast [11]. As a result, image generation has emerged as a potential solution to sustainable image storage and transmission, as well as faster page loading times.

2.2 Stable Diffusion

Stable Diffusion (SD), the image generation that this paper uses to test the effectiveness of the solution, is a modified latent diffusion model (LDM) [11]. LDMs, also known as denoising diffusion probabilistic models, are a variant of diffusion models that combine the concepts of diffusion processes and latent variable models. These models offer a powerful framework for image generation by explicitly modeling a latent space and utilizing diffusion steps to gradually refine the latent variables.

In LDMs, the generative process involves iteratively updating the latent variables rather than directly modifying the pixel values. The latent variables capture high-level representations of the images, such as semantic features or abstract concepts. Through the diffusion steps, the latent variables are refined to generate coherent and realistic images. In addition, working in the latent space means that LDMs can compute data faster than a diffusion model while maintaining image quality and diversity— addressing all three areas of the trilemma [7].

2.2.1 Limitations

In general, a larger number of inference steps, which denoise the image and give it more detail, result in a more convincing output image, directly related to the image quality [11]. However, there are some areas where this does not always apply: human hands and faces. The majority of images depicting humans that showed appendages, hands, or faces tended to be awkward or unconvincing. In this case, a larger number of inference steps proves to be helpful, but it does not solve the issue. However, SD is able to depict landscapes, animals, and items even with a low number of inference steps.

This is likely due to the dataset that SD was trained on, the LAION Aesthetic dataset [12], composed of a large number of paintings and artistic photography. Perhaps as a result of this, SD is capable of generating convincing images of landscapes, animals, or items even with a low number of iterations, while it struggles to generate realistic portrayals of humans, particularly in context of their hands, appendages, and faces. This meant that the training set would probably have to be diversified in order to produce more realistic, artefact-free results. In addition to this, other image modification models such as GFPGAN, which helps restore faces in images, were examples of possible fixes to the issue [13].

2.2.1 Negative Prompts

Negative prompts are also a tool that can help address the artefact issue with human limbs. These allow the user to tell the SD model what not to display in the result, and is crucial to the sampling process in SD 2.0 because of how it was trained. This allows a user to change specific details in images, such as specifying for an absence of snow or removing a specific color, or something more general, by adding disfigurement or blurriness to the negative prompt. The negative prompt does not come in the code by default and requires the addition of a `neg_prompt` as an argument in the unconditional model, run alongside the generative one to guide the process [14].

2.3 Page Load Time

The page load time refers to the amount of time taken to load an entire webpage with its contents. This usually comes from a CDN server, which provides data packets from a cache.

2.4 CDN Servers

CDN servers are crucial components of the internet infrastructure that store and distribute content, including images, to end-users efficiently. Here, there are two main forms of energy consumption that occur regarding images: transmission and storage.

In image transmission, it is not only the CDN server's transmission that must be considered, but also the routers along the path from the server to the client. Depending on the distance of the client from the closest CDN server, the energy consumption will differ. To analyze the efficiency in energy consumption of transmission, there is a need to compare the energy consumption of an image transmission from a CDN server to that of generating an image from the client-end *after* receiving the textual prompt from the CDN server.

In terms of storage, CDN servers require significant amounts of electricity to power and maintain the server hardware, including the storage devices and associated cooling systems. The energy consumption of these servers is continuous, as they need to be operational 24/7 to serve content to users globally. The scale of CDN operations, with numerous servers distributed across different locations, further amplifies the energy requirements [15].

When considering image generation as a solution to image storage, this is synonymous to analyzing the amount of storage space that can be saved when converting images into a *combined prompt*, the concatenation of the `alt-text` of an image file and a specialized prompt generated by an `img2prompt` model, meant to not only give context but also the specifications of a given image. In addition, the number of times the image will be accessed is also important as it would mean the image would be dynamically generated each time, resulting in further energy consumption.

3 METHODOLOGY

3.1 Image Collection

The image-collecting script, or web crawler, was written on Python using the Selenium library [16]. It would take a list of URLs, then from each page collect all elements within the website that contained the `<img>` tag. From these elements, the web crawler then extracts and records the alt-text and dimension attributes into a CSV file. While testing the initial script, there were three main observations and resulting modifications that needed to be made.

The first was that there were many images that had no alternate text available. However, the larger elements that were within oftentimes had contextual text that could replace the alt-text attribute's role in the image generation process. In addition, these images could be placed deep within `<a>` elements while having their contextual alternate text in different `<p>` elements. Thus, the script was modified to search within elements, both within and outside of the image in question, to find contextual alternate text in the case that there was no alternate text as a replacement to its `alt-text` attribute. Some testing led to the conclusion that dynamically programming this would be difficult, so some websites were hardcoded to match their image placement format.

The second observation had to do with the format of the images on websites. A presumption that had been made was that image elements on websites would use the `<img>` tag in an isolated format. This was not the case, as different websites would not only use different tags such as `<pic>` and `<figure>`, but also would sometimes forgo using HTML elements altogether, loading the image from CSS using the `background-img` property without an alternate text. This required that the script look for different image element formats.

Finally, the third observation was that the script was collecting images that were either unnecessary or did not need to be part of the converting process. These included advertisements or icons of the websites, of which the former would be constantly changing and the latter requested to be kept constant by the owner of the website. To address this, the script was modified to look for keywords such as *analytic*, *advertising*, *marketing* and checked the dimensions of an image to ignore any images that would be icons, usually with dimensions smaller than 100 by 100 pixels.

3.2 Additional Image Statistics

Google BigQuery was the main source for understanding how much of the top webpages use images and how much of the page's size they take up [17]. To do this, the script would perform a SQL query to the HTTP archive database labeled `202209\_01\_desktop` and receive a result. However, this took a few seconds for each query and the plan was to check the top 100 thousand, then top 1 million webpages. Thus, a faster method needed to be implemented.

In response to this, the script went through two more iterations of lookups. The first was a script that created subprocesses to search different sections of the database to find matching queries. This method, though faster than a single process lookup, still took over 6 hours to go through 20 thousand queries. The second iteration utilized the Python dictionary, mapping the entire database before performing lookups. Though this took a slightly longer startup time, the lookups were almost instantaneous, completing the lookup of all 1 million webpages in less than five seconds.

Not all websites were found, however, with only 20 thousand of the top 100 thousand and 300 thousand of the top 1 million webpages having entries in the BigQuery database. Regardless, the results of how images play a large role in not only content delivery but also webpage size became increasingly clearer. Images accounted for an average of 42 percent of total composition and 50 percent of total size in Tranco's top 1 million webpage [18].

Table 1. Statistics of top 1 million pages

Aggregate Image Requests / Total Requests	
Percent	Requests
41.8	15141692 /36258945
Aggregate Image Size / Total Size	
Percent	Mbytes
50.3	806545 /1355126

3.3 Image Generation

3.3.1 Stable Diffusion 2.0

SD 2.0 addressed many of the computational inefficiencies that existed in its earlier versions, as well as adding new features. It included the xformers library by default, which implemented the memory-efficient attention modification to the SD model. In addition, the precision of the model was set to autocast, meaning that it allows the model to use half precision in the U-Net component of the model, thus increasing speeds even further [19].

The new model also uses a new text encoder, one of the core components of SD, an open-source version of CLIP. This did not change computations, but rather affected the contents of the outputs themselves. OpenCLIP does not have the same number of celebrities and artists in its dataset as the previous models' text encoders. Thus, it is harder to represent specific art styles or celebrities with SD 2.0 [20].

As a result of this, SD 2.0 falls behind the previous versions when it comes to representing art styles or celebrities. On the other hand, it is much more capable of producing realistic images, particularly with the help of negative prompts, which allow a user to specify what should not be generated in the output image. Negative prompts do, however, affect the performance of the model, portions of samples to taking up to 4 seconds longer during inference (see Figure 1, 2).

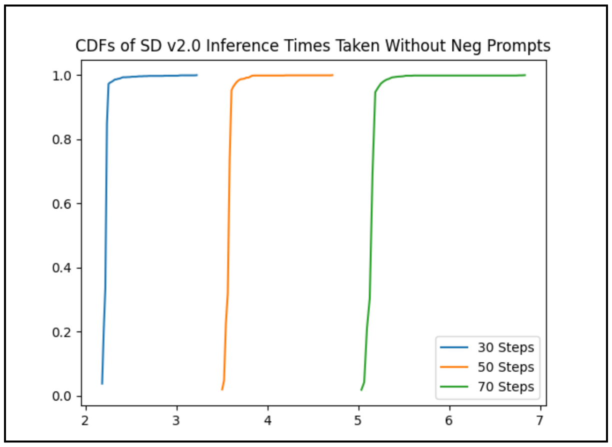

Figure 1. Ratio of samples (Y) and their inference times (X) without negative prompts

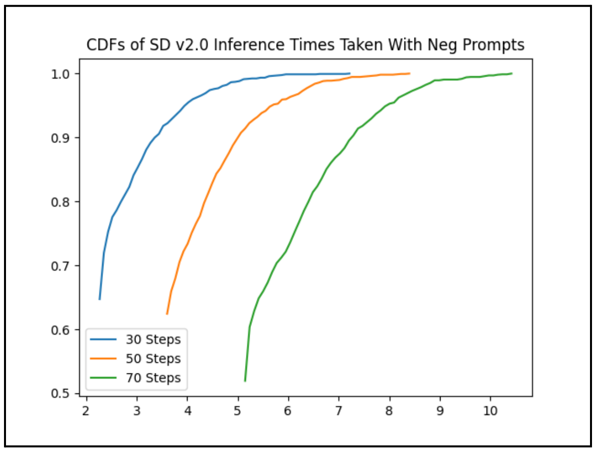

Figure 2. Ratio of samples (Y) and their inference times (X) with negative prompts

3.2.2 High Performance Cluster

We used the institution's GPU high performance cluster (HPC) to access Nvidia A100 PCIe graphics cards for testing purposes [21]. In order to do this, we needed to update the driver of the server to support CUDA 11.2 and higher.

3.2.3 Parameters of Interest

There are several parameters used in SD, but of these there are two that have a large influence on the performance of the model. The first of these is the *guidance scale*, which tells the model how closely to stick to the prompt that it is given. In many cases, a default value of 7.5 is given for SD models [19], as it allows the model to create high-quality images at lower iterations— the second parameter of importance. A higher number of iterations— called *inference steps* —affects the model in two ways. On one hand, it improves the quality of an image, particularly fine-grain details [11]. On the other hand, it increases the sampling time, something undesirable in the context of using image generation as a substitute for non-generated images. Thus, it is important to find the balance between guidance scale and inference steps to get the result with the highest quality *given a certain level of inference time*.

We label a small pool of prompts in the eight categories of `["food", "building", "landscape", "object", "animal", "public figure", "person/people", "sports"]` and find the parameters at which each category is considered acceptable, which we call the *optimal parameters*. Acceptable is defined as the *minimum inference steps* at which an image's contents are recognizable. With this, we train a convolutional neural network (CNN) to classify a prompt given its context and assign the given optimal parameters.

3.3 Page Load Times

To imitate the page loading, we created the tool `WebDiffusion`, meant to act as a proxy of the server, delivering content to a device on demand. When a request for an image is made, the request is redirected to a local server, which provides the textual prompt corresponding to the image. This prompt is used to generate an image using SD, which is then sent to the client-end device.

3.4 Energy Consumption

As aforementioned, there are two areas of interest when it comes to the energy consumption of CDN servers and images: transmission and storage.

3.4.1 Transmission

Obtaining a clear formula for the energy consumption of transmitting a data packet can be challenging due to various factors and complexities involved in the transmission process. The energy consumption depends on multiple variables, including the network infrastructure, transmission medium, equipment efficiency, packet size, and data transfer protocols [5].

The energy consumption of transmitting a data packet can vary depending on the specific network infrastructure and equipment used. Different network technologies, such as wired, wireless, or cellular networks, have varying energy efficiency characteristics and protocols. Additionally, the efficiency of network equipment, such as routers, switches, and network cards, can vary across different manufacturers and models [22].

The size of the data packet being transmitted plays a significant role in energy consumption. Larger packet sizes generally require more energy to transmit due to the increased data payload. However, modern data transfer protocols, such as TCP/IP, often break data into smaller packets for efficient transmission, and the overhead associated with packet headers and acknowledgments can also impact energy consumption [23].

Data transfer protocols involve additional overhead for error correction, flow control, congestion control, and other functionalities. These protocol overheads contribute to the total energy consumption during packet transmission. The specific implementation and efficiency of these protocols can vary, making it challenging to derive a universal formula for energy consumption.

Even so, in order to understand the benefits that image generation may bring, there needs to be some kind of standard to compare its functionality to [23].

$$P = P_{idle} + \mathrm{E_P} R_{pkt} + \mathrm{E_{S\&F}} R_{byte}.$$

Here, P is the total power consumption, P_{idle} represents the power consumption of a router when idle, $\mathrm{E_P}$ is the energy required for per-packet processing and $\mathrm{E_{S\&F}}$ is the energy required to store and forward a packet. R_{byte} is the input byte rate, while R_{pkt} is the input packet rate.

We adopted the same theoretical system that the authors used in the paper, assigning values to the various equipment used in the study. This included the installed bandwidth capacity and the energy parameters as follows:

Table 2. Energy consumptions for different devices [23]

Device	P_{idle} (W)	$\mathrm{E_P}$ nJ/pkt	$\mathrm{E_{S\&F}}$ nJ/byte
Enterprise Ethernet Switch	36.2	40	0.28
Edge Ethernet Switch	631	1571	9.4
Metro IP Router	352	1375	14.4
Edge IP Router	576	1707	10.2

In addition to this, there are also costs at the CDN server level. This includes the server energy consumption, as well as the transmission energy consumption. For the sake of simplicity, we assume that a surrogate server on the same Tier ISP where the request was generated *has* the image (is a hit). The server energy consumption can be defined as

$$E_{server} = \sum_{m} Br_m E_{sr}$$

Here, B is the content size, r_m is the number of requests for the *m*, the content, and E_{sr} is the server energy consumption per bit [22].

On the other hand, the transmission energy can be defined as

$$E_{transmission} = \sum_{m} Br_m [E_{sr}(H^A_{sd} + 1) + E_l H^A_{sd}$$

Here, H^A_{sd} refers to the number of hops to fetch content from a surrogate on the same Tier ISP, E_l refers to the link energy consumption per bit, and E_r refers to the router energy consumption per bit [22].

3.4.2 Storage

As CDN servers typically store multiple copies or replicas of content to ensure high availability and reliability, storage becomes one of the most energy-consuming aspects from a long-term perspective. The maintenance of redundant copies requires additional storage space, which increases energy consumption. Redundancy is crucial to prevent data loss and ensure uninterrupted content delivery, but it comes at the cost of increased energy requirements.

The storage devices used in CDN servers, such as hard disk drives (HDDs) or solid-state drives (SSDs), consume energy during data read and write operations. These devices require power for spinning disks, moving read/write heads, and managing data access. The longer data, particularly an image, is stored in a CDN server, the less likely it is to be accessed in the long-term future. However, as the amount of stored content increases, the number and capacity of storage devices also increase, leading to higher energy consumption [5].

A possible formula for energy consumption is the following:

$$E_{storage} = \sum_{m} Bn_{m} P_{st} t$$

Here, t is the time period in which the energy consumption is computed, n_m is the replicas of m, and P_{st} is the energy consumption for storage per bit [22].

3.4.3. Generation
Due to the nature of an electronic device, there is a lot of fluctuation in the amount of power that is drawn at a given point in time, so we determined the method of calculating the energy consumption as the amount of power the GPU draws when generating an image. Thus we define the energy consumption for an image as:

$$P = (M_{pc}U)T_i$$

Here, P is once again the total power consumed, T_i is the total time of inference in seconds, U is the utilization factor of the GPU, and M_{pc} is the maximum power consumption of the specified GPU. Though it is very unlikely that a GPU will draw at maximum power, it still provides a good reference as it is the ceiling the GPU will reach before throttling speed to lower temperatures [24], and also considers CPU utilization, which tends to be constant.

3.4.4 Other Costs
There are other costs that are associated with server storage, transmission, and image generation, such as the constant power consumption for computers and routers. However, as this is a comparison between the two methods, the additional costs that are common between them are not considered in this paper.

4 SIGNIFICANCE

This method involves leveraging the computational capabilities of users' devices, which have been improving manifold over the past years. By utilizing machine learning algorithms and frameworks directly on the user's device, images can be generated dynamically, reducing the reliance on server-side transmission, and decreasing load times. This approach not only alleviates the strain on CDNs and reduces energy consumption but also enhances the user experience, particularly in low-bandwidth environments where network limitations are prevalent.

By adopting client-side image generation, the internet ecosystem can achieve multiple benefits. It minimizes the energy consumption associated with server-side image transmission, thereby reducing carbon emissions. Additionally, it empowers users by leveraging their devices' processing power and decreases the reliance on bandwidth-intensive operations. As a result, image-heavy web pages can be loaded faster, ensuring a smoother browsing experience for users, regardless of their network limitations.

5 RESULTS

5.1 Collected Images and Prompts

The script was used to collect information from 500 of the top 1 million websites on Tranco's list. However, some of the webpages failed to load and, even with modifications, the script was able to collect 1,870 images and their contexts from 200 different webpages.

Here, the `img2prompt` model was used to generate additional annotation for the images that were collected to give more information for the SD model to use when generating an image, creating *combined prompts*. On average, each of these prompts consisted of 194 bytes.

5.2 Image Generation

The combined prompts were used to generate images once using the Nvidia A40 and another time the Nvidia A100 40GB PCIe [21,25].

5.2.1 Performance
The A100 sampled images of 512x512 in 3.8 seconds at 50 steps. In comparison, with the A40 on SD 1.4, a 512x512 image at 20 steps took 3 seconds.

An image at 20 steps using SD v2 and the A100 took 1.5 seconds to complete (see Table 2).
These results were in line with the benchmarking of the SD model done by Lambda Labs [26], which included not only the A100 but also other GPUs and their sampling times for images generated with 50 inference steps (see Table 3).

Table 3. Average Speed of A100 in HPC server using with SD 2.0

Steps	Time Taken (sec)
10	0.960
20	1.681
30	2.474
40	3.118
50	3.828
60	4.752
70	5.386
80	6.249
90	6.751

Table 4. Speed of various GPUs on SD [26]

GPU	Precision	Time (sec)
Nvidia A100 80GB PCIe	half	3.74
Nvidia GeForce RTX 3090	half	4.83
Nvidia RTX A6000	half	5.03
Nvidia RTX A5500	half	5.05
Nvidia GeForce RTX 3080	half	5.59
Quadro RTX 8000	half	5.93

5.2.2 Oneflow

SD, when coupled with the features and optimizations offered by the OneFlow deep learning framework, brings notable speed improvements to image generation tasks.

When combined with OneFlow, it benefits from the framework's efficient GPU acceleration, memory management techniques, model parallelism, optimization algorithms, and hardware-awareness, resulting in faster image generation, up to 49.65 iterations per second [27].

5.2.3 Qualitative Analysis

We performed a qualitative analysis on the generated images through user feedback on Prolific, having them rate the score of the image on a 5-point scale from "Very Poor" to "Very Good." The images for the qualitative analysis were generated at 20 inference steps to keep neutrality between the different parameters that may have been required for each subject. The categories of food and landscape scored between "Good" and "Ok" but the other categories were either ranging between the minimum threshold of "Good" or between "Ok" and "Poor."

5.3 Page Load Times

We used three Web performance metrics: SpeedIndex (SI), Page Load Time (PLT), and the time it took for the page to become visually complete (VC). However, of these VC is the best metric as most of the page loading time is spent in the generation of the images. This is due to how the SI would be triggered prematurely as it measures how quickly visual content appears, and the PLT triggered too late due to the time it takes for all images to be fully generated and displayed, as we did not implement lazy loading.

The use of the intermediary image generator caused a delay in the VC of about 5 seconds in a simulated fast connection of 100 Mbps 20 ms RTT, as opposed to the 2.3 second delay in a 20 Mbps 100 ms RTT connection. This difference is due to how the image generator is not affected by the network, as it is meant to be client-end. As a result, 40 percent of webpages benefited from the intermediary image generator under the simulated slow connection.

5.4 Transmission Costs

For this study, we use the case of `edition.cnn.com`. We observed that `traceroute` to `edition.cnn.com` resulted in the packet traversing through 9 routers. 4 of these we identified as campus/metro routers, and 5 of them as high-capacity edge routers [23].

As high-capacity edge switches store data before feeding them into edge routers, we identified 2 edge switches along the path and 1 enterprise switch from the university network.

5.4.1 Transmitting Full Images

When considering the average size that images take in a website, approximately 1,000 KB, and assuming a 1,500-byte packet length [23], the number of packets would be approximately 667 packets. This incurs about 22.967 J in energy consumption. On the other hand, if the packet length was 100 bytes, this would incur 172.298 J. The smaller the packet size, the larger the energy consumption on transmission.

As for the server, assuming a hop length of 3 hops to reach the server to fetch content, a request of 1 user, a link energy consumption of $1.48*10^{-9}$ J/bit, a router energy consumption of $1.2*10^{-8}$ J/bit, and a server energy consumption of $2.81*10^{-7}$ J/bit [22], the server energy consumption is 2.28 J and the transmission energy consumption is 0.42 J. With this, the total energy consumption in transmission would be 25.635 J with 1,500-byte packets and 174.966 J with 100-byte packets.

5.4.2 Transmitting Textual Prompts

On the other hand, if we consider the average textual prompt of 194 bytes, 200 if including a few more characters for input parameters, then with the same network the energy consumption in transmission would be $4.27*10^{-5}$ J with 1,500-byte packets, or $5.99 * 10^{-5}$ J with 100-byte packets.

5.5 Storage Costs

Using `web.archive.org` to access archived CNN pages showed that most images were still being stored on their servers. Scraping 10 years' worth of news resulted in 73,881 unique images, approximately 5.91 GB of storage.

5.5.1 Storing Full Images

Assuming that these images will remain in the server for the next year and the power consumption per bit stored is $7.84*10^{-12}$ [22], with at least 5 replicas across all CDN servers, the total energy consumption of storage is approximately 138,248.79 kJ.

5.5.2 Storing Textual Prompts

On the other hand, using textual prompts would mean that 73,881*200 = 14.78 MB would be stored. The total energy consumption of storage in this case is 345.65 kJ.

5.6 Generation Costs

Our observations during the image generation process showed that there was an average of 70 percent GPU utilization with 4 cores. Given that the Nvidia A100 40GB PICe graphics card has a maximum power consumption of 250 W [21], we performed calculations for SD image generation with and without the OneFlow implementation (See Table 4).

Though this may be an overestimation of the actual consumption that occurs in the image generation process, it provides insight on the relationship between inference time, steps, and energy consumption.

Table 5. Power consumed in image generation

	Without OneFlow		**OneFlow**	
Steps	30	50	30	50
Power Consumed (J)	432.95	669.9	105.74	176.234

7 CONCLUDING REMARKS

While image generation has made significant advancements and can produce high-quality images at relatively low cost and speed, there are several reasons why it is not feasible as a solution to replacing images on the internet.

The most intuitive reason is that these models do not yet outperform the current systems that are set in place, in terms of speed and energy consumption. Even though client-end hardware has improved and continues to improve at an impressive rate, only high-end GPUs are able to run these models at fast speeds. The A100 is by no means a graphics card for the average desktop user. Though other GPUs *can* be used to generate images, they still do not have the ability to generate samples at high speed— and thus, the trilemma appears once more.

Despite this, it is important to note that there continue to be improvements on both the hardware and software ends: inference speeds have doubled between the first release of SD and SD 2.0, and have doubled again with the OneFlow implementation. As an open-source model, SD has seen many improvements as developers have taken the model apart and found ways to make it faster and more efficient [28]. This will likely continue with the updates that come, as well as if the upcoming SD XL is open-source [29].

Another reason why image generation will likely be unable to replace images entirely is due to specific image requirements. Models such as

Midjourney v5 have been shown to sample very realistic images, but at the same time certain applications of images require specific attributes or characteristics that are challenging to recreate. This might be medical imaging, scientific research, or historical documentation.

It is interesting to note, however, that— given a model can produce highly realistic outputs —image generation could find an application in "compressing" archived images, as this already shows savings in space and energy consumption, particularly if the archived images are seldom accessed. This should be taken with a grain of salt, however, as not to conflict with the need to maintain historical documentation. In any case, there are a lot of images on news sites or blogs that might not need to be entirely specific or factually accurate. Stock photos, images of food, generic images of landscapes or buildings; these are some of the categories that do not necessarily need to be kept as a full image on a server. Using a textual prompt to preserve or even amplify the photo could be a more cost-efficient way to store "images" on the internet.

In addition, image generation models may also have an application in *reconstructing* lost images. If an archived article or post has a broken link, the proposed intermediary could be used to generate an image using either the alternate text or context of the page.

As a feasibility study, the paper's aim was not to attempt to prove that image generation techniques outperform images on the internet today, but rather to see if there are areas in which image generation could help improve the internet.

REFERENCES

1. Li, Yiyi, and Ying Xie. “Is a Picture Worth a Thousand Words? An Empirical Study of Image Content and Social Media Engagement.” *Journal of Marketing Research*, vol. 57, no. 1, American Marketing Association, Feb. 2020, pp. 1–19. https://doi.org/10.1177/0022243719881113.
2. Rajiullah, Mohammad. “Towards a Low Latency Internet: Understanding and Solutions.” *ResearchGate*, Nov. 2015, https://doi.org/10.13140/RG.2.1.4328.3601.
3. “Faster Web Through Client-Assisted CDN Server Selection.” *IEEE Conference Publication | IEEE Xplore*, 1 Aug. 2015, ieeexplore.ieee.org/document/7288411.
4. “Power Consumption and Energy Efficiency in the Internet.” *IEEE Journals & Magazine | IEEE Xplore*, 1 Apr. 2011, ieeexplore.ieee.org/document/5730522.
5. Jin, Chaoqiang, et al. “A Review of Power Consumption Models of Servers in Data Centers.” *Applied Energy*, vol. 265, Elsevier BV, May 2020, p. 114806. https://doi.org/10.1016/j.apenergy.2020.114806.
6. “HTTP Archive: Page Weight.” *httparchive.org*, May 2023, httparchive.org/reports/page-weight#reqImg.
7. Xiao, Zhisheng, et al. “Tackling the Generative Learning Trilemma With Denoising Diffusion GANs.” *arXiv (Cornell University)*, Cornell University, Dec. 2021, https://doi.org/10.48550/arxiv.2112.07804.
8. A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. “Zero-shot text-to-image generation.” CoRR, abs/2102.12092, 2021.
9. Nichol, Alex, et al. “GLIDE: Towards Photorealistic Image Generation and Editing With Text-Guided Diffusion Models.” *arXiv (Cornell University)*, Cornell University, Dec. 2021, https://doi.org/10.48550/arxiv.2112.10741.
10. Tao, Ming. “DF-GAN: A Simple and Effective Baseline for Text-to-Image Synthesis.” *arXiv.org*, 13 Aug. 2020, arxiv.org/abs/2008.05865.
11. Rombach, Robin. “High-Resolution Image Synthesis With Latent Diffusion Models.” *arXiv.org*, 20 Dec. 2021, arxiv.org/abs/2112.10752.
12. *LAION-Aesthetics | LAION*. laion.ai/blog/laion-aesthetics.
13. TencentARC. “GitHub - TencentARC/GFPGAN: GFPGAN Aims at Developing Practical Algorithms for Real-world Face Restoration.” *GitHub*, github.com/TencentARC/GFPGAN.
14. Woolf, Max. “Stable Diffusion 2.0 and the Importance of Negative Prompts for Good Results.” *Max Woolf’s Blog*, 28 Nov. 2022, minimaxir.com/2022/11/stable-diffusion-negativ e-prompt.
15. Islam, Saiful, and Jean-Marc Pierson. “Evaluating Energy Consumption in CDN Servers.” *Lecture Notes in Computer Science*, Springer Science+Business Media, 2012, pp. 64–78. https://doi.org/10.1007/978-3-642-32606-6_6.
16. “Selenium.” *Selenium*, www.selenium.dev.
17. “BigQuery Enterprise Data Warehouse | Google Cloud.” *Google Cloud*, cloud.google.com/bigquery.
18. *A Research-oriented Top Sites Ranking Hardened Against Manipulation - Tranco*. tranco-list.eu.
19. Stability-Ai. “GitHub - Stability-AI/Stablediffusion: High-Resolution Image Synthesis With Latent Diffusion Models.” *GitHub*, github.com/Stability-AI/StableDiffusion.
20. Cusick, Bill. “Stable Diffusion 2.0 Release — Stability AI.” *Stability AI*, Feb. 2023, stability.ai/blog/stable-diffusion-v2-release.
21. “A100 GPU’s Offer Power, Performance, and Efficient Scalability.” *NVIDIA*, www.nvidia.com/en-us/data-center/a100.
22. “Energy Consumption for Data Distribution in Content Delivery Networks.” *IEEE Conference Publication | IEEE Xplore*, 1 May 2016, ieeexplore.ieee.org/document/7511356.
23. Vishwanath, Arun, et. al. “Modeling Energy Consumption in High-Capacity Routers and Switches.” *IEEE Journals & Magazine | IEEE Xplore*, 1 Aug. 2014, ieeexplore.ieee.org/document/6848762.
24. Spetko, Matej, et al. “DGX-A100 Face to Face DGX-2—Performance, Power and Thermal Behavior Evaluation.” *Energies*, vol. 14, no. 2,

MDPI, Jan. 2021, p. 376. https://doi.org/10.3390/en14020376.

25. “NVIDIA A40 for Visual Computing.” *NVIDIA*, www.nvidia.com/en-us/data-center/a40.
26. LambdaLabsML. “GitHub - LambdaLabsML/Lambda-diffusers.” *GitHub*, github.com/LambdaLabsML/lambda-diffusers.
27. Oneflow Inc. “GitHub - Oneflow-Inc/Oneflow: OneFlow Is a Deep Learning Framework Designed to Be User-friendly, Scalable and Efficient.” *GitHub*, github.com/Oneflow-Inc/oneflow.
28. Automatic. “GitHub - AUTOMATIC1111/Stable-diffusion-webui: Stable Diffusion Web UI.” *GitHub*, github.com/AUTOMATIC1111/stable-diffusion-webui.
29. Stable Diffusion XL. “Stable Diffusion XL Model - SDXL Beta - Stable Diffusion XL.” *Stable Diffusion XL*, May 2023, stablediffusionxl.com.