CSCI 1430 Final Project Report:
Magic Eraser

Team Members: Andrew Li, Michelle Lu, Abhyudaya Sharma & Aaron Zhang.
TA Name: Siddarth Diwan. Brown University.

Abstract

Ever since the invention of photographs, people have captured unwanted objects and people in their images. Photobombs, crowds and solar glare are some things people try to remove from their images. Our project leverages a cutting-edge photo editing tool akin to Google's Magic Eraser: an artificial intelligence backed web application that not only seamlessly removes unwanted objects, but is also accessible to laypeople without knowledge of convoluted, difficult software like Photoshop and Gimp.

*The web application utilizes **YOLOv8** for image segmentation and **stable-diffusion-inpainting** for inpainting. The integration of these two models not only adheres to minimizing user effort, but also achieves high-fidelity image editing. The ongoing adaptation and improvement of these models further refine the tool's capabilities, balancing user-friendliness with sophisticated technological innovation.*

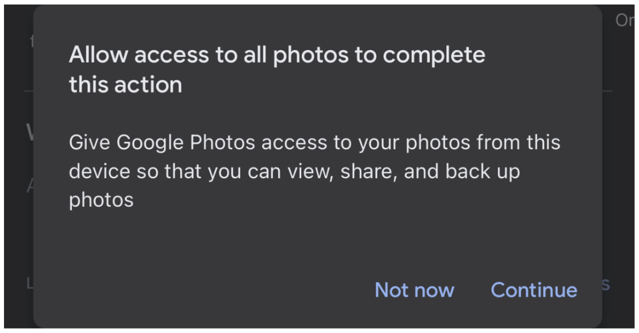
1. Introduction

With the advent of photography, there are several use cases where a user may want to remove unwanted elements from their photos, including photobombs, crowds, and solar glare. Though it is possible to use existing software like Photoshop and Gimp, these applications require high learning curves and are difficult to use.

Google introduced a photo editing tool in 2021 called Magic Eraser that easily eliminates unwanted people and things in a picture. Upon its release, this technology sparked much buzz about its ability to enhance the visual appeal and aesthetics of photographs.

However, there are limitations behind Google's Magic Eraser. The first is that Google's Magic Eraser automatically fills in details which may not be what the user wants. That is, there is no user prompt to the technology. The second is that Google's Magic Eraser requires access to an individual's complete photo library and does not work across all platforms (limited to iOS and Android).

Identifying these limitations, we implemented a photo editing tool akin to Google's Magic Eraser that grants users the flexibility to remove objects and inpaint the image in a less privacy-invasive way.



Allow access to all photos to complete this action

Give Google Photos access to your photos from this device so that you can view, share, and back up photos

Figure 1. Google prompts users for access to their complete photo library

The pipeline for the application consists of three Computer Vision tasks:

- Object Recognition
- Segmentation
- Inpainting

These three functionalities are combined into a web-application where users can upload a picture, get a list of recognized objects, choose which object to remove, and visually see the object segment to be removed in the image before the application removes it. The web application then displays an output image without the object in it that can be easily downloaded and exported.

2. Related Work

Google's Magic Eraser [1] is publicly available software tool that can remove unwanted objects from images. Users can use their phones to select objects for removal. Magic Eraser then automatically fills in the missing details after removing the selected objects.

In 2019, Xin Hong, Pengfei Xiong, Renhe Ji and Haoqiang Fan released DFNet [2], an inpainting model that took in as inputs the original image with objects removed as well as the mask of the object that was removed. They constructed several new neural network blocks that improved performance by forcing the model to train on the missing areas. The authors also developed new model losses that focused more on the model's performance on missing regions.

Rombach et al. debuted Stable Diffusion, a text-to-image generation model, in 2022 [3]. The authors also created an inpainting model by training on top of the existing Stable Diffusion v1 weights. The stable diffusion model was trained on 256 Nvidia A100 GPUs for a total training time of over 150,000 hours.

3. Method

For optimal functionality of our Magic Eraser, we identify two underlying problems to solve: extracting objects from the background of an image and filling in missing regions left by selected objects in the image when prompted. Our approach, therefore, consists of two major components:

Segmentation Image segmentation is the process of identifying which pixels of an image belong to one object. For segmentation, we decided to use YOLOv8, a state-of-the-art pre-trained image segmentation model that accurately identifies thousands of different objects. We take the polygon surrounding the object and generate a mask image.

Inpainting Inpainting is the process of filling in missing details in an image. We implemented inpainting two ways:

- Reimplementing and training a custom DFNet model [2] from scratch in Tensorflow on 10 million images from the MIT Places2 dataset. [4]
- Using an existing pre-trained `stable-diffusion-inpainting` model from Hugging Face [3].

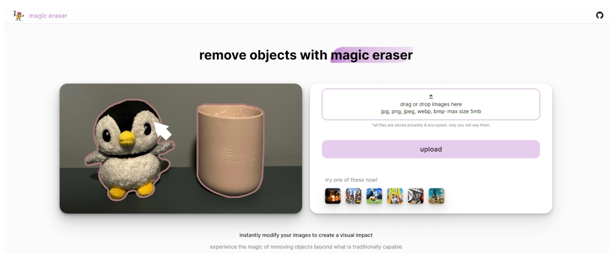
For training our DFNet model, we perused the original paper [2] and reimplemented their novel neural network blocks and custom losses in Tensorflow. We assembled an architecture identical to what was described in the paper and then trained it on a Windows 10 PC with an Nvidia 3060Ti GPU using WSL2 on Ubuntu 22.04. This model's architecture does not allow the user to provide a prompt and would try to automatically generate the missing contents.

By using the existing stable diffusion model, users can input a custom prompt for inpainting, giving them more flexibility about the contents that will be inpainted. Since this model is trained on over 2 billion images, the quality of the outputs are far beyond anything we could achieve with our limited budget. This is the model we use for our web application.

4. Results

4.1. Web Application

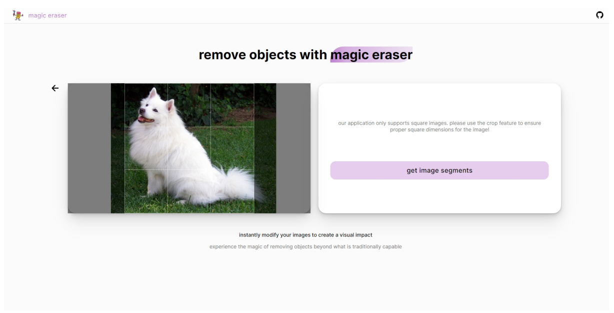
We created a web application for users to uitilize the models we used in a seamless manner. The user is first prompted to with a screen to upload their own image or choose an existing photo:


magic eraser
remove objects with magic eraser
drag or drop images here
jpg, png, jpeg, webp, bmp· max size 5mb
upload
try one of these now!
*all files are stored privately & encrypted. only you will see them.

instantly modify your images to create a visual impact

experience the magic of removing objects beyond what is traditionally capable

Upon uploading an image, the user must crop the image so that it is a square, as the inpainting model requires images to be 512x512. Users have the ability to zoom in on the region they would like to use, in addition to moving the highlighted region:

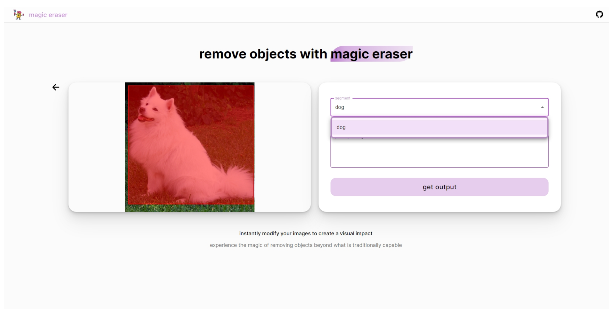


our application only supports square images. please use the crop feature to ensure proper square dimensions for the image!

instantly modify your images to create a visual impact

experience the magic of removing objects beyond what is traditionally capable

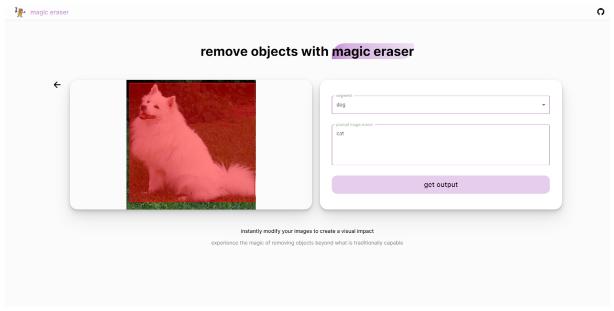
Once the user is happy with the selected region, they are then prompted with a dropdown box to select the object they would like to mask and inpaint. In the event that there are multiple objects with the same label (ie, multiple dogs), users may refer to the image on the left, as a red box will appear to denote the object currently selected:



instantly modify your images to create a visual impact

experience the magic of removing objects beyond what is traditionally capable

Once the desired object has been selected, the user may enter a prompt for our model to use for inpainting purposes. In this example, the prompt was "cat":

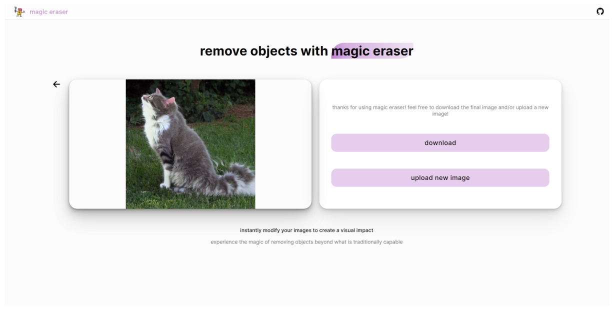


remove objects with magic eraser

instantly modify your images to create a visual impact

experience the magic of removing objects beyond what is traditionally capable

The user is then prompted with the resulting image and are given the option to either download the image or upload a new one:



remove objects with magic eraser

thanks for using magic eraser! feel free to download the final image and/or upload a new image!

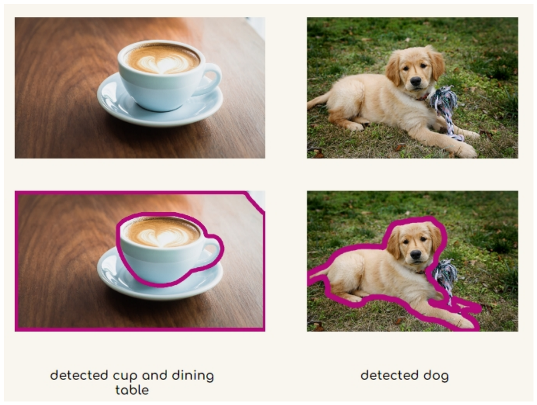
instantly modify your images to create a visual impact

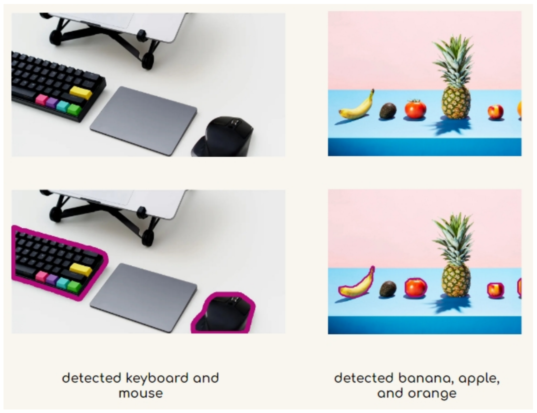
experience the magic of removing objects beyond what is traditionally capable

As previously mentioned, there are two main components to our web application: segmentation and inpainting. The results of this will be discussed in the following two sections.

4.2. Segmentation

Using YOLO, we were able to not only detect the presence of objects, but also identify what these objects were. Some examples of this are shown below:




detected keyboard and mouse
detected banana, apple, and orange

4.3. Inpainting

As previously mentioned, we employed two separate tactics to handle the task of inpainting:

DFNet Our outputs of tests conducted on our trained-from-scratch DFNet model were somewhat poor. Please see Figure 2 for an example. We attribute the inaccuracy to tight time constraints and a large dataset. More specifically, even on a powerful Nvidia 3060Ti GPU, each epoch took an average of approximately 12 hours, leaving us only able to train the model for around 10 epochs. We believe that with more compute and time availabililty, our DFNet model would be able to perform as well as, or even exceed, what Hong et al. were able to achieve.

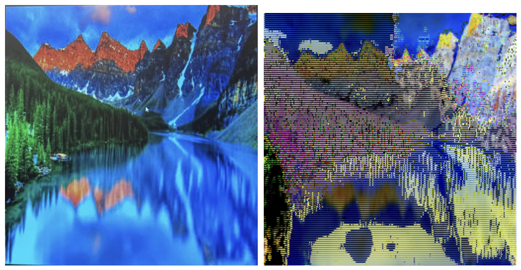


Figure 2. DFNet Output after 10 epochs

Stable Diffusion Inpainting We demonstrated our WebApp, which employs the Stable Diffusion Inpainting model, and its functionality to several qualified acquaintances in the Brown Computer Science department before asking them to try the Magic Eraser out for themselves. The most common feedback our WebApp received was that, while not perfect, the Stable Diffusion model did a far better job of achieving the user's expected functionality following segmentation. Many also appreciated the incorporation of natural language processing embedded within the Stable Diffusion interface.

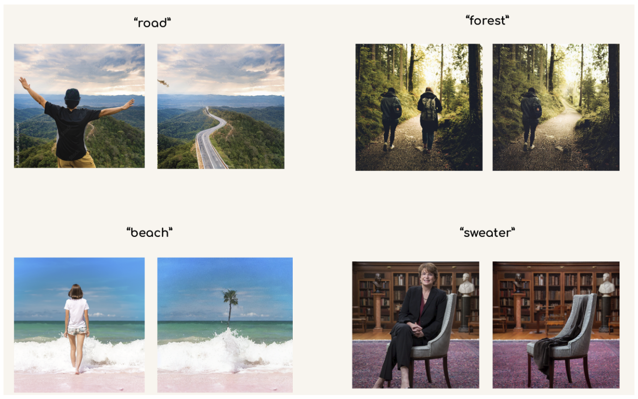


Figure 3. Stable Diffusion Inpainting Outputs

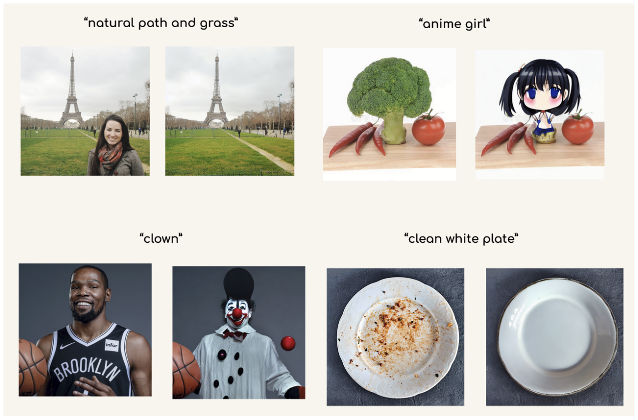


Figure 4. More Stable Diffusion Inpainting Outputs

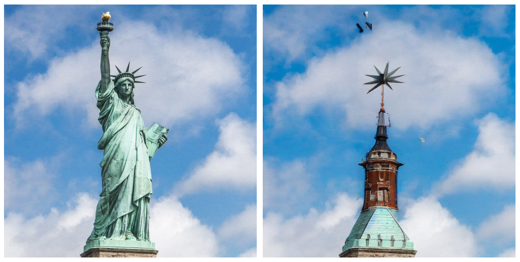


Figure 5. Bad Stable Diffusion Outputs (prompt = "sky and clouds")

In figures 4 and 5, we include several examples of pairs of inputs and outputs to our WebApp along with a user query. For most images, the model performs well. However, there are some cases when the model's outputs are not what we would have expected.

4.4. Technical Discussion

From training the DFNet model, we learnt that training complex deep learning models requires a significant amount of compute and that it quickly gets expensive when we need to rent GPUs from cloud providers. Hyperparameter tuning, another important puzzle that needs to be solved during training, also becomes significantly harder when one training epoch takes several hours (which in our case took approximately 12 hours). For most students and organizations, to get good results while not burning money, we feel like using and/or building on top of existing pre-trained models is the only way to achieve good results.

Another aspect worth considering is the storage of images that are being uploaded. Currently, we utilize **UploadThing** to store the images uploaded by users. Because we are currently using the free version, there is a limitation of 2GB. In the event that we max out on this limit, any calls to our API to upload images will error. To mitigate this, we can manually go in and delete existing images stored, but this would be an undesirable solution since this compromises the integrity of the storage application. Furthermore, this also raises questions about user privacy, as we do not explicitly state how we store the images they upload and the duration in which we do so. Because we delegate the storage of images to **UploadThing**, this means that our app's reliability and confidentiality depends on **UploadThing**. If it were to be compromised, this would have an incredibly large impact on our application.

Moreover, the input file size for our specific stable diffusion model is restricted to 512 x 512 pixels. To comply with this limitation, all input images are cropped to this square format, and we provide tools for users to adjust zoom and shift scales to fine-tune the focus of the image before uploading. However, this approach presents challenges for editing non-square photos since information in the photo will inevitably be lost. While users may include a border around their image to ensure that all of the objects in the image are included, this may compromise the resolution of their images, depending on the original dimensions. In essence, this restriction can affect the final quality and authenticity of the edited photo.

Finally, our web application is developed using Next.js and deployed on Vercel. Utilizing third-party services like Vercel means we must adhere to their operational conditions and limitations. That is, any server outages or periods of excessive traffic on Vercel's end can directly impact the availability and performance of our application. Additionally, we are operating on Vercel's free tier, which while does grant us a majority of functionality, there are a few missing features such as password protection and access to web analytics and speed insights. Without these features, it becomes challenging to refine and improve the application effectively.

5. Conclusion

We present a user-friendly Magic Eraser tool that allows users to remove unwanted objects from images. From our results, we can conclude that AI models are impressive and generally work well for the majority of input images. However, it is still possible for them to create non-realistic outputs that do not resemble the real world.

6. Social Impact

We discuss several potential social impacts that our Magic Eraser may have. One concern that our Magic Eraser mitigates is user privacy. Google's Magic Eraser requires access to a user's full photo library, which some may feel uncomfortable with. In contrast, our model only requires the input image a user chooses without asking for permission to access any other photos in the user's library. Although we do not have a privacy policy at this moment, it is possible for us to delete uploaded images after a short duration of time (say 24 hours). This will ensure that users' sensitive images are not exposed even if there is a data breach or vulnerability in our software.

Another potential issue with images generated by our Magic Eraser is the creation of fake information. In the past few years, fake news have become a real problem. It is not impossible to imagine that fake news can lead to a different election outcomes. There is also a real danger of fake news disrupting peace and causing communal violence. The problem is not limited to just large-scale global or regional disinformation campaigns; fake information could be an occupational or legal hazard. For instance, a fake Twitter post saying that a company will refuse to provide health insurance can sow the seeds of distrust in the eyes of its employees.

Our Magic Eraser isn't immune from these issues. Although most of our outputs are easily distinguishable from real-life images, a malicious actor could still use our tools to swindle unknowing individuals. Given the right input photos and prompts, it's possible for certain output images to look more realistic and convey/elicit specific responses. This issue becomes further compounded when the malicious actor knows the other individual personally – an added level of trust and personal connection makes it easier to scam unsuspecting individuals. Furthermore, these images may not be carefully analyzed to determine their authenticity. Many times, we look at photos in an effort to gain additional information, without questioning its source or message. This instinct, while efficient for our day-to-day activities, unfortunately amplifies the risks that our Magic Eraser poses.

We also need to consider how our Magic Eraser can be used in the context of bias and fairness. The use of tools such as our Magic Eraser to remove certain objects or people from photos could potentially perpetuate biases or reinforce stereotypes. For instance, selectively editing out individuals from diverse backgrounds could contribute to underrepresentation in media and perpetuate harmful stereotypes. Similarly, indiscriminate applications of our Magic Eraser could be made towards inadvertently erasing culturally significant elements from photos, leading to cultural appropriation or erasure. Users should be mindful of the cultural context of the images that are inputted and consider the implications of removing or altering certain elements of the image. Finally, given that the Stable Diffusion Inpainting model was trained on subsets of LAION-2B(en), which consists of images that are primarily limited to English descriptions, this means texts and images from communities that communicate through other languages may be insufficiently accounted for.

The environmental impact of training and using models is also a concern. It is estimated that training just the stable diffusion inpainting model produced over 11,250 kg of carbon dioxide. According to the EPA, equivalent to driving a petrol car for over 42,000 km. Until we can ensure that the electricity for training deep learning models is from renewable sources, deep learning models can have a significant contribution towards global warming and climate change. Furthermore, each time a user tries to use our Magic Eraser, we are wasting precious energy that could have been used for filtering water or growing food in developing countries.

We also reflect upon the potential impact of our Magic Eraser on user's self-perception. Tools like our Magic Eraser may increase the prevalence of heavily edited photos on social media and thus, may contribute to unrealistic beauty standards and adversely impact individuals' self-image and self-esteem. This could have ramifications for mental health and well-being, particularly among young people.

Legal and copyright issues can arise with our Magic Eraser. The use of Magic Eraser-like tools to edit or remove copyrighted elements from photos may raise legal and copyright concerns. Users need to be aware of copyright laws and licensing agreements when editing and sharing photos to avoid infringing on the rights of content creators.

Regarding copyright, it is important to know what the copyright status of the images used to train the model was. If the training dataset consisted of copyrighted images, and the model learns to reproduce said images verbatim, the user may be in violation of copyright laws. In such situation, the onus may also lie on us, the developers of the model. The dataset curators might also share some responsibility. As far as we know, the existing copyright law was written decades ago when generative AI models were not a thing. Since the law has not been amended to keep up with recent developments, it is unclear who the blame should fall onto.

Our solution makes it easy to remove objects; however, it also removes the legitimacy that humans have generally attributed to images. Images hold immense value – an image can literally capture history, an image can be the deciding factor in a judicial trial, an image can evoke passionate emotions and bring change in our world. By dismembering the authenticity that is inherent of images, generative AI models have the potential to destroy the fabric of trust that underlies human lives. Software developers, including us, have the responsibility to correctly place an indelible mark on AI-produced content that cannot be removed. Although there are several such efforts in the works, until every developer adopts one such standard, we have, unfortunately, eroded all trust in our digital communications.

References

[1] Google Magic Eraser. https://blog.google/products/photos/magic-eraser/. Accessed: 2024-05-14. 1

[2] Xin Hong, Pengfei Xiong, Renhe Ji, and Haoqiang Fan. Deep fusion network for image completion. In *Proceedings of the 27th ACM international conference on multimedia*, pages 2033–2042, 2019. 1, 2

[3] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022. 2

[4] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2

Appendix

Team contributions

All team members contributed to the success of this project.

Abhyudaya Sharma & Aaron Zhang Set up the dev environment on a Windows PC with Nvidia 3060Ti GPU. Worked on implementing and training DFNet from scratch in Tensorflow. Implemented the web application backend using FastAPI and Hugging Face. Created the production app environment and deployed the backend on GCP using an Nvidia T4 equipped cloud virtual machine. Also worked on the project poster and this report.

Andrew Li & Michelle Lu Worked on the YOLO segmentation model for object detection and masking. Built the frontend of the web application using Next.js and TypeScript, starting with high-fidelity Figma prototypes. When building the web application, thoroughly considered user experience and workflow to provide a seamless experience. Integrated web application with UploadThing as the database and deployed web application on Vercel. Also worked on the project poster and this report.