Mahir Pirmohammed

Vision Assignment

Assignments include the delivery of coding of a ML model and a short technical report describing a proposed ML application. The technical report consists of six items as follows.

1. Pitch:
   State the circumstances an organization or sets of users would be willing to fund the proposed ML application based on their perceived value. Value could be financial or any other type of outcome organizations or users need to obtain (e.g., institutional image, customer satisfaction, increased quality or efficiency).
2. Data source:
   Indicate with a link where you obtained the data. If you generated the data yourself, please provide a link to the code of the used approach.
3. Model and data justification:
   Justify why you chose a specific model to learn from the selected data. If you learned that a given more would be suitable for a type of data from a publication, please provide a link to the source. Note you still have to justify it with your own words (as always, limit to three sentences).
4. Commented examples:
   Indicate the input where trained model is applied, the output and whether it is as expected or any observations you may have.
5. Testing:
   Provide a confusion matrix with one or more metrics and comment the results.
6. Code and instructions to run it:
   Provide a link to the code and any required instructions to run it. Please include some testing examples so we can quickly experience what you experienced with the model.

Synthetic Card Generator with Surface Damage (Gum Stains)

1. Pitch
   - The Trading Card Grading Application Capstone Project is an organization who would be willing to fund a synthetic card generation model for the ability to create large amounts of data without having to do the tedious job of manual card scans.
   - Currently our stakeholder must manually scan thousands of cards of a specific type (e.g. gum stain, print hickeys, logo, no-logo, etc.) which can take many hours. Dirty damaged cards also leave dirt and other debris in the scanner which takes time to clean and disrupts the scanning process.
   - The synthetic card generation would give the project a way to generate vastly more data for training other AI models such as surface damage detection, improvements in centering scores, logo detection, and many other grading purposes.
2. Data Source
   - The data I used was provided by the Trading Card Grading Application Stakeholder. It is a collection of 1000+ Gum Stain cards he scanned manually. I am on this capstone project working on surface damage detection and I chose to specifically work on gum stains.
   - A sample of the dataset can be found in the *dataset* folder in the code. It is categorized into 4 categories that I manually did to the 1000+ gum stains to understand the distribution on what kinds of gum stains exist and if there are any imbalances. I also have a *ALL_GUM_STAIN_CARDS* folder that holds all the card images in one folder for the initial models I had trained.

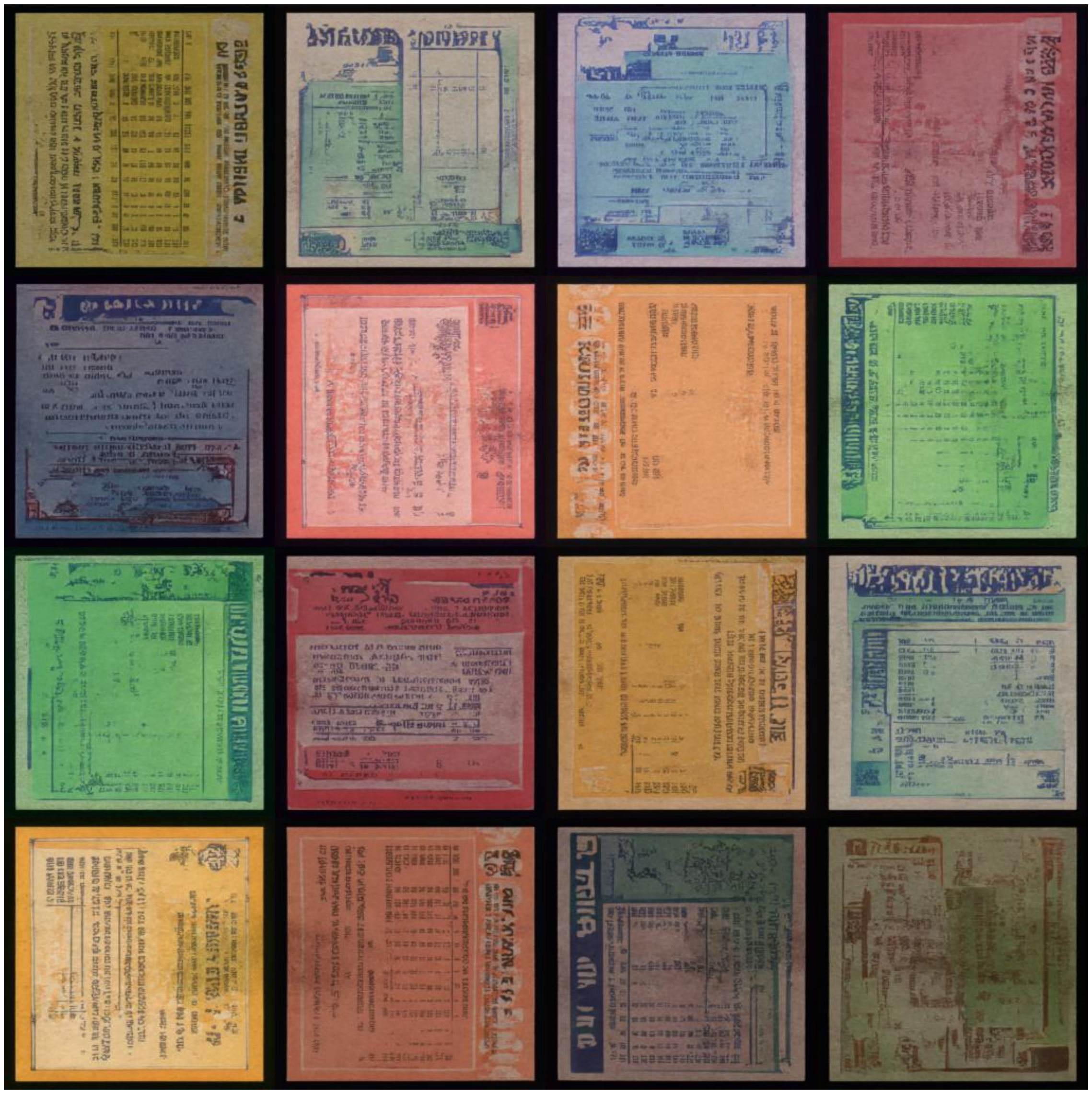
3. Model and Data Justification
    - I chose the U-net diffusion model because of its ability to extract key features from a set of images and predict the noise to be removed from the scheduler on the dataset of images.
    - Without changing many configuration settings from the Train a diffusion model tutorial on hugging face and giving it a dataset of gum stain cards, the diffusion was able to successfully generate synthetic cards with various irregular shaped gum stains so the model I used is justified through the results seen in section 4.
    - Things I changed were the resolution, number of epochs, training batch size, evaluation batch size, and the preprocessing of the images to be uniform rectangular shapes in the same vertical orientation.
    - The rectangular shape was slightly modified to comply with the UNet models down/up sampling process as seen in the huggingface UNet2DModel's sample_size parameter: *sample_size (int or Tuple[int, int], optional, defaults to None) — Height and width of input/output sample. Dimensions must be a multiple of 2 ** (len(block_out_channels) - 1).*

4. Commented Examples

   Using the Hugging Face Train a diffusion model tutorial and noteboook, I was able to train a UNet model to generate cards.

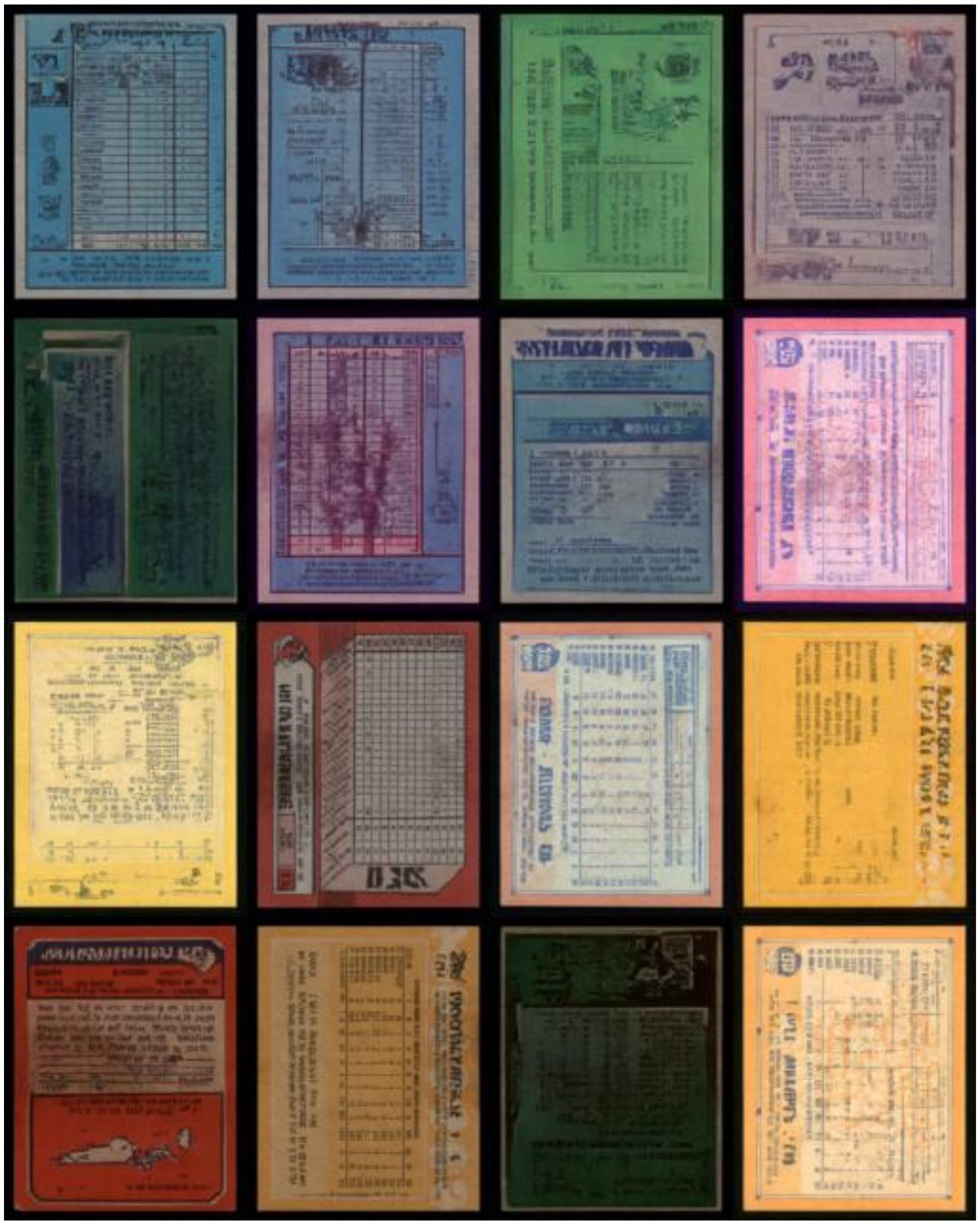
   I trained the following models using the same script but with different sample data and other hyperparameter changes:

    1. cards - This is the first attempt at using the tutorial notebook on the entire Gum Stain dataset. I only changed the number of epochs to 100 epochs.
        i. The 99th epoch was saved, and it created surprisingly good-looking cards although they were square in shape due to the script's preprocessing of the input images into 128 x 128 squares.

ii. 

iii. As can be seen in the above image, the model trained very well on the input datasets and captured very good characteristics of what the backs of baseball cards typically look like. You can also see that it did capture the gum stain characteristic itself given that all the cards have some gum stain on them.

2. all_gum_stain_cards - This was my second attempt where I changed the sample size to reflect the aspect ratio of the original cards. I used the entire gum stain dataset for this model. The new dimensions are 160 x 128. I also trained for 200 epochs.
    i. The 199th epoch below shows the great quality of the cards. You can also see that a few of the cards did have very realistic gum stains.

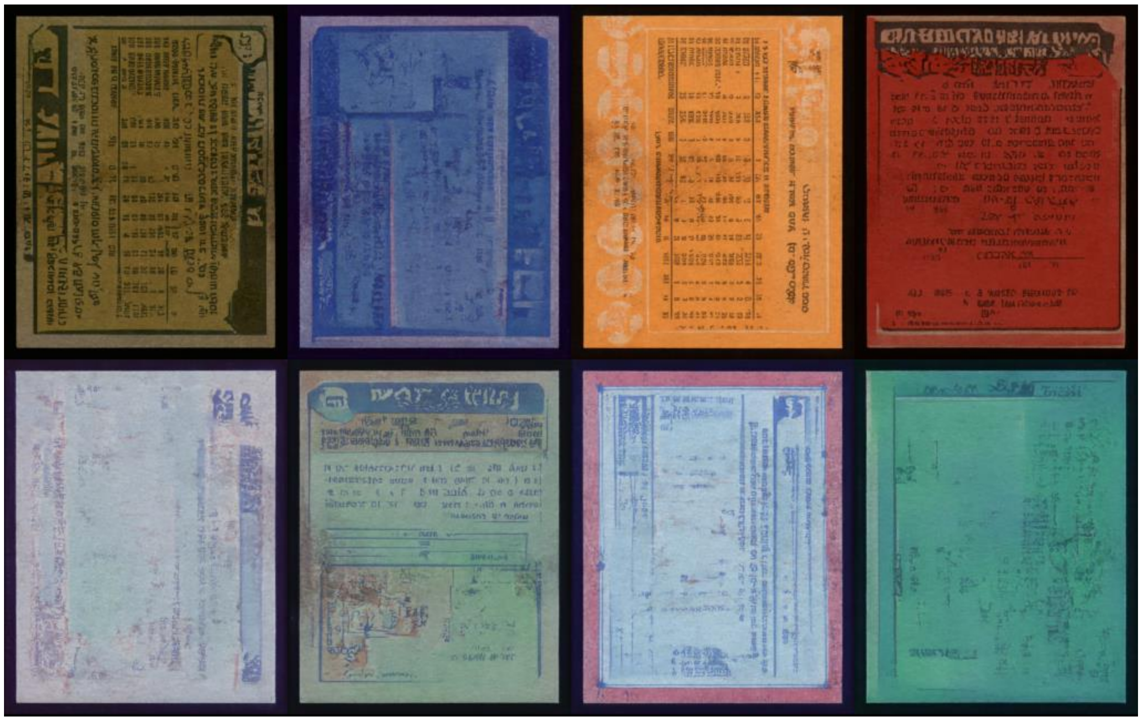
ii. 

iii. With over 100 more epochs and a change in the dimensions of the input data, the output finally appears like actual trading cards with more clarity in the features. The text is still very unreadable but because the purpose of these cards is to show gum stains this is not necessary for the model to learn. The model performed really well and we can see a variety of gums stains clearly.

3. higher_res_gum_stain_cards - This was my third attempt where I tried to increase the dimensions of the images to provide a higher quality synthetic card generator for use in our models. I used the entire gum stain dataset for this model.
    i. With the dimensions now at 320 x 256 the cards have much better resolution. The problem arises with hardware requirements where approximately doubling the resolution causes CUDA problems and almost 4-6x times longer to train. I only was able to do 100 epochs and that took almost 24 hours to run. I wanted to see what card dimensions I could run on my own physical hardware without using any cloud-based training.

ii. You are still able to see gum stains being created so if I was able to run this for 200 epochs or more, I do believe that we would see cards like in all_gum_stain_cards but at a higher image quality.

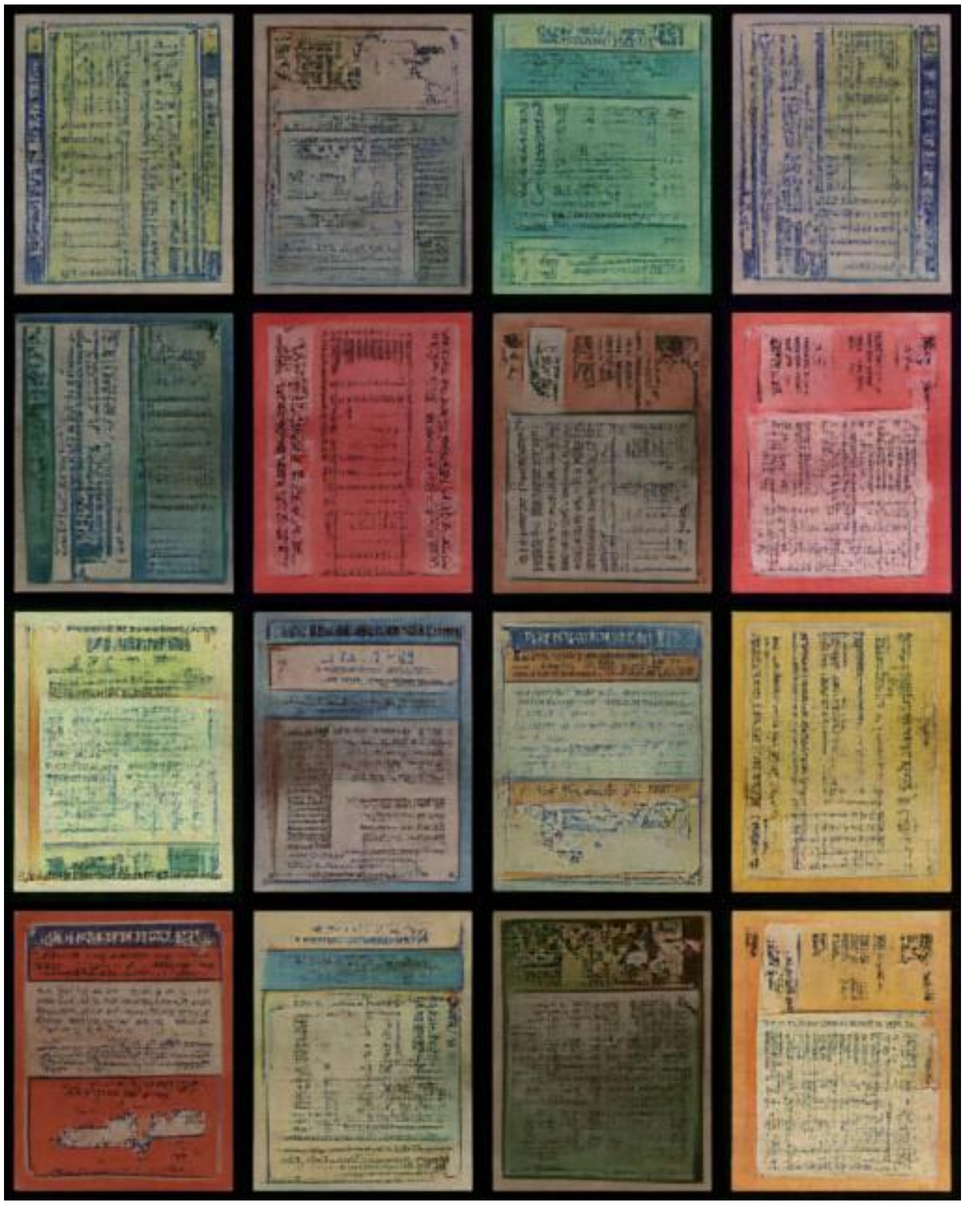
iii.



iv.

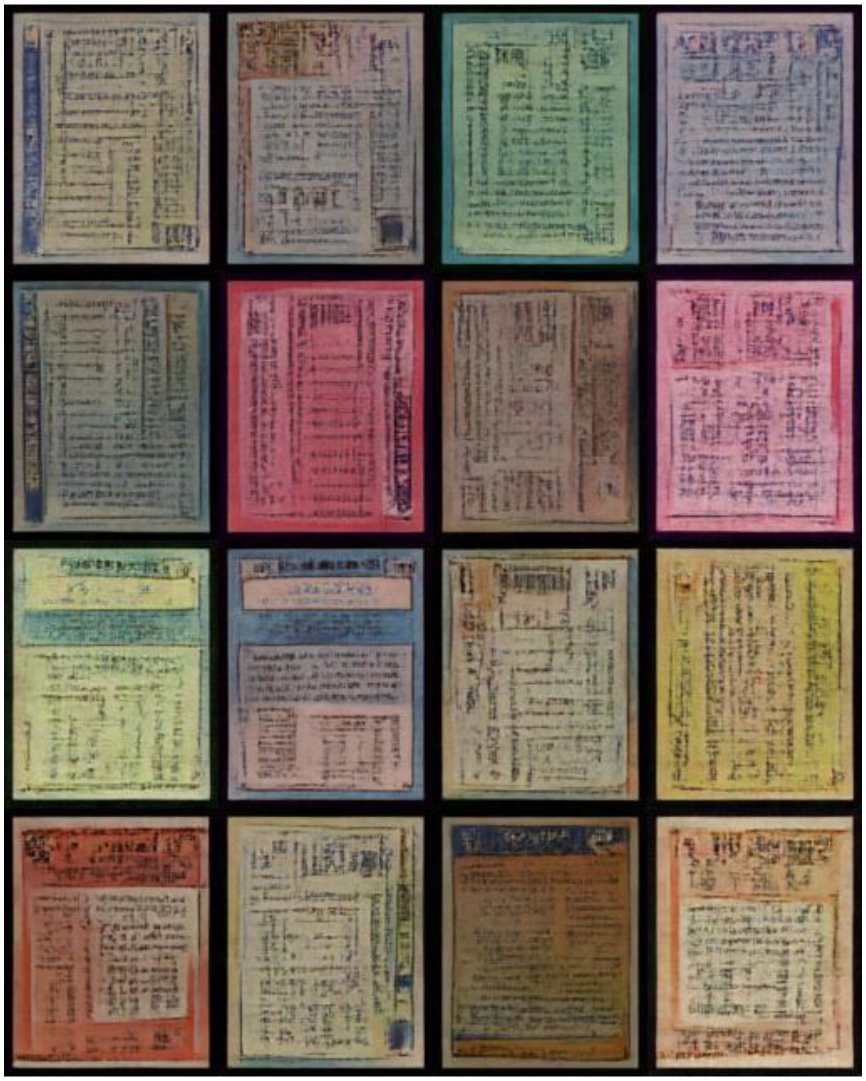
v. The models output for this was good for some cards and very terrible for others. I believe if this was given another 100 epochs similar to the all_gum_stain_cards model then the cards would come out very similar but at a higher resolution. This would be the ideal model to use if hardware requirements were not restricting me.

4. I then tested two different specific categories I had found when annotating these gum stain cards by hand: Full and White/Marks. These were both trained on the lower 160 x 128 resolution for 150 epochs due to hardware and time constraints. I was having issues with the GPU cache not emptying.
    i. FULL_GUM_STAIN_CARDS - Full means the gum stain was across most of the card, and I would have to use one big bounding box. I used the dataset/FULL_CARD dataset to train this model.
        1. In the 149th epoch samples generated by the model I was able to get more cards with clearer gum stains.

2. 
3. Due to the more category specific cards with large gum stains, the model definitely learned what a gum stain is along with the characteristics of the cards. You can clearly see large gray stains on some of the cards. If I had given it 50+ epochs I believe the gum stains would be clearer in all the cards.

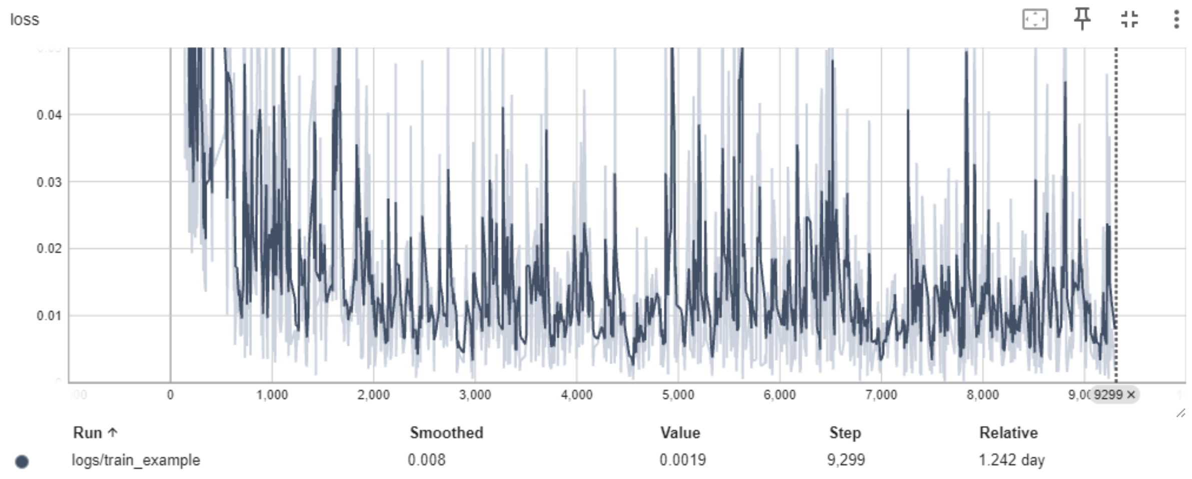
ii. WHITE_OR_MARKS_GUM_STAIN_CARDS - White/Marks are gum stain cards that had any sort of gum stain white or other color residue remaining. I used the dataset/WHITE_MARKS dataset to train this model.

1. In the 149th epoch samples generated by this model, I was not able to see the white marks. I believe this is due to the nature of the cards having bigger dark gum stains along with the smaller white marks so due to the nature of the data it could not extract those features in the 150 epochs.

2. 

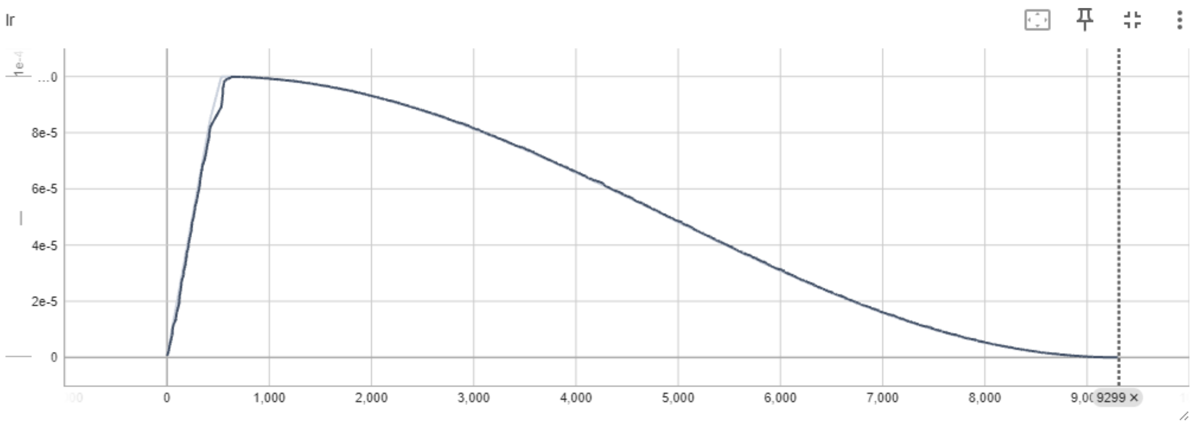
iii. Due to the results between the two categories above, I did not train models for the LIGHT and SMALL categories.

5. Testing
   - I could not make a confusion matrix for this type of model, but I was able to obtain statistics on the diffusion model's loss and learning rate as a function of each time step through the training process for each of the different models I trained:
   - cards

i. 

Run ↑	Smoothed	Value	Step	Relative
● logs/train_example	0.008	0.0019	9,299	1.242 day

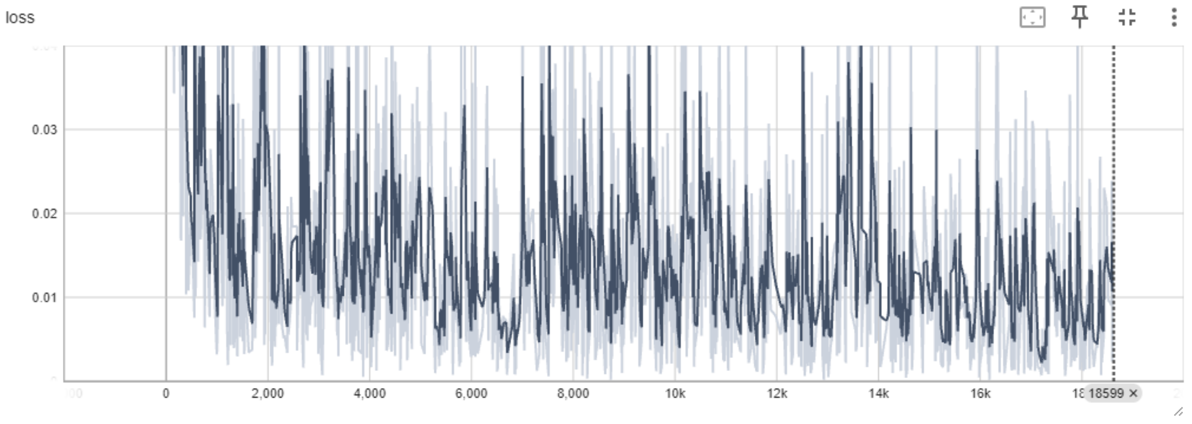
The loss of the diffusion model for 9,299 steps for the cards model



Run ↑	Smoothed	Value	Step	Relative
● logs/train_example	0	0	9,299	1.242 day

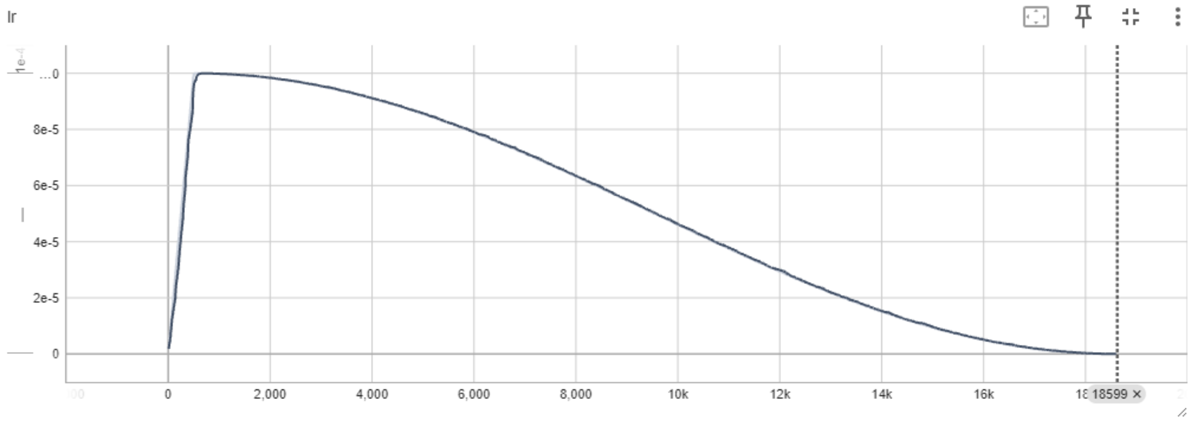
ii. The learning rate of the diffusion model 9,299 for the cards model. The learning rate starts to decrease after 500 steps of warmup.

- all_gum_stain_cards



Run ↑	Smoothed	Value	Step	Relative
● logs/train_example	0.0107	0.0104	18,599	4.636 hr

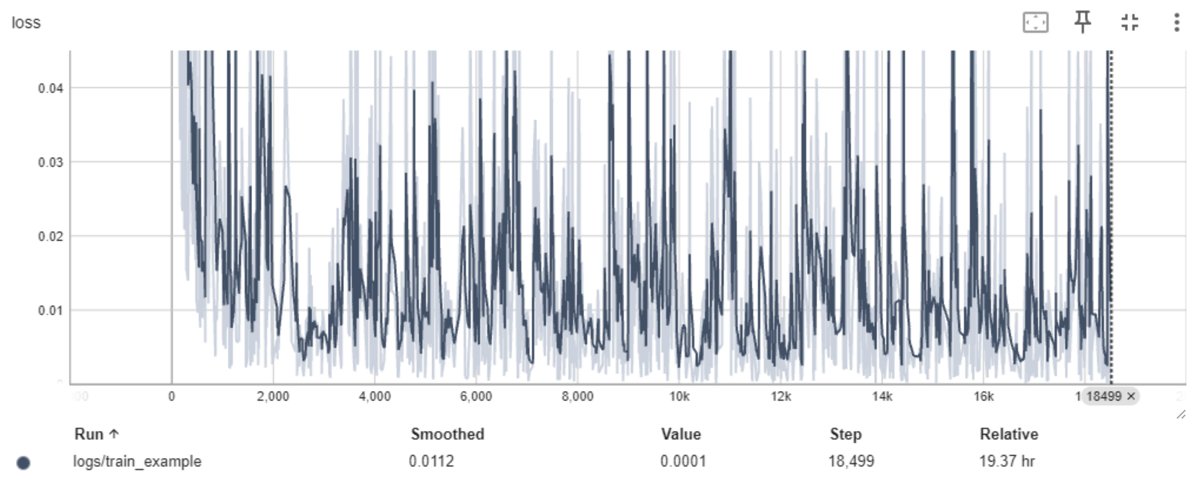
i. The loss of the diffusion model for 18,599 steps for the all_gum_stains_cards model. Looks very similar to the cards model. This took 4.646 hours to train.



Run ↑	Smoothed	Value	Step	Relative
● logs/train_example	0	0	18,599	4.636 hr

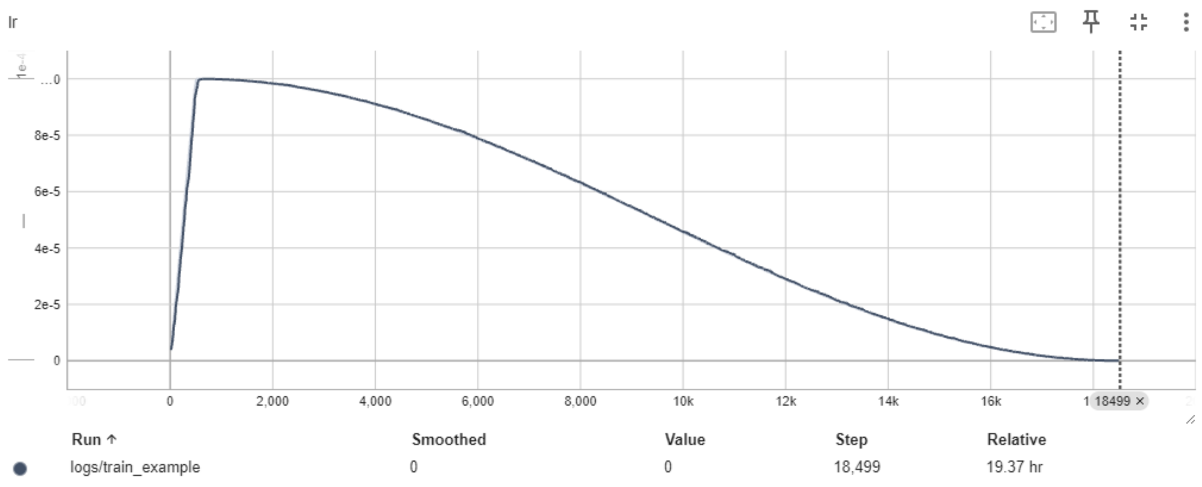
ii. The learning rate of the diffusion model for 18,599 steps for all_gum_stain_cards model. Again, this looks very similar to the card model. The learning rate starts to decrease after 500 steps of warmup.

- higher_res_gum_stain_cards



	Run ↑	Smoothed	Value	Step	Relative
●	logs/train_example	0.0112	0.0001	18,499	19.37 hr

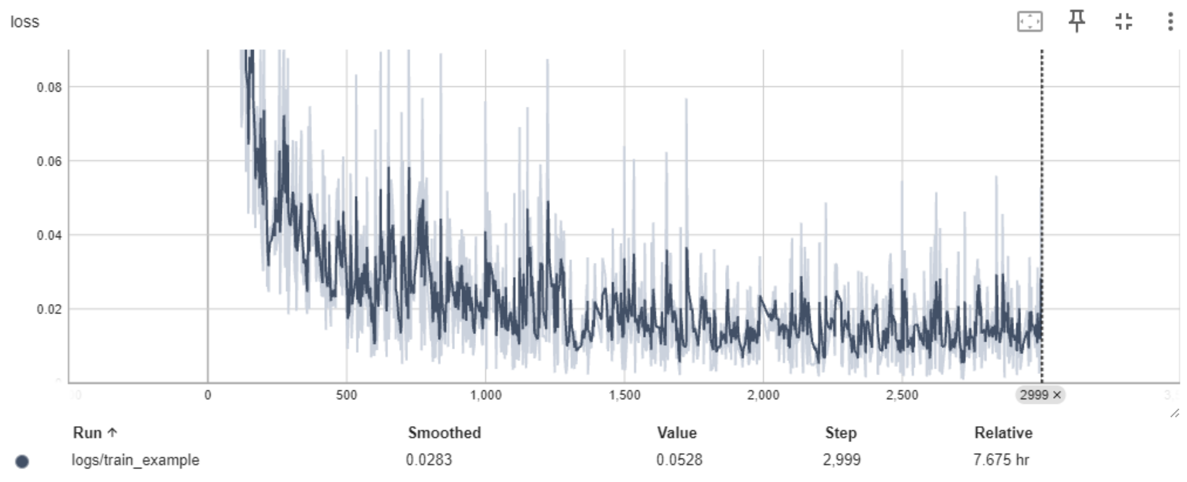
i. The loss of the diffusion model for 18,499 steps for the higher_res_gum_stain_cards model. Looks very similar to the previous models except that the last stretch of the step has a very high spike again. This took 19.37 hours to train



	Run ↑	Smoothed	Value	Step	Relative
●	logs/train_example	0	0	18,499	19.37 hr

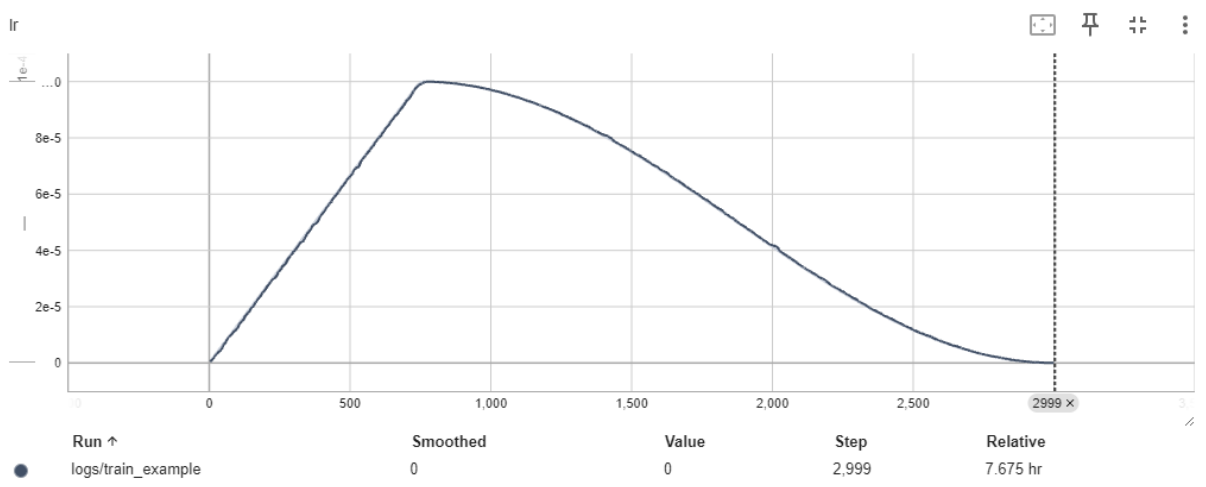
The learning rate of the diffusion model for 18,499 steps for all_gum_stain_cards model. This looks very similar to the card model as well. The learning rate starts to decrease after 500 steps of warmup.

- FULL_GUM_STAIN_CARDS



	Run ↑	Smoothed	Value	Step	Relative
●	logs/train_example	0.0283	0.0528	2,999	7.675 hr

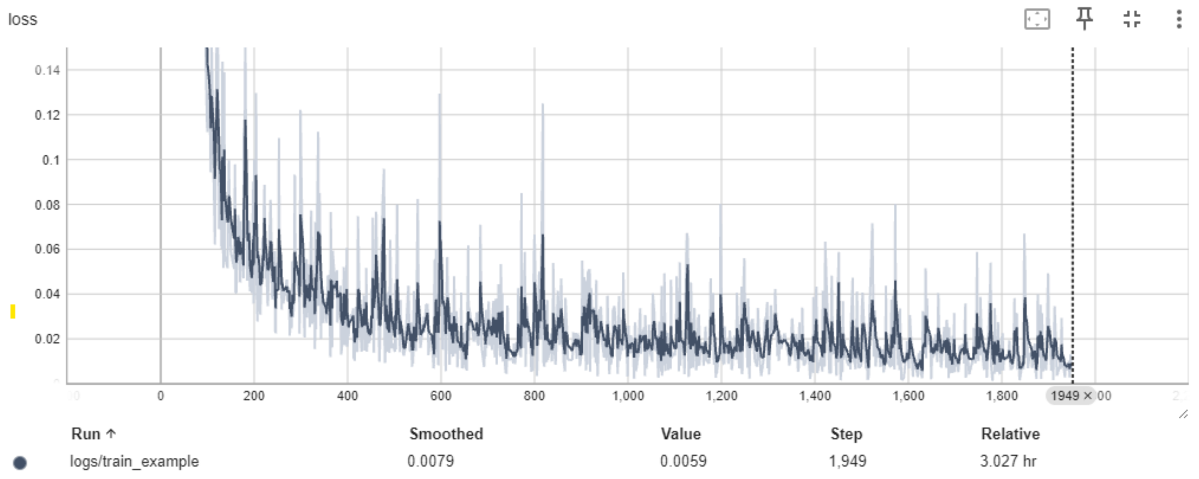
i. The loss of the diffusion model for 2,999 steps for the FULL_GUM_STAIN_CARDS model. The loss is much more stable with a gradual decrease in loss.



Run ↑	Smoothed	Value	Step	Relative
● logs/train_example	0	0	2,999	7.675 hr

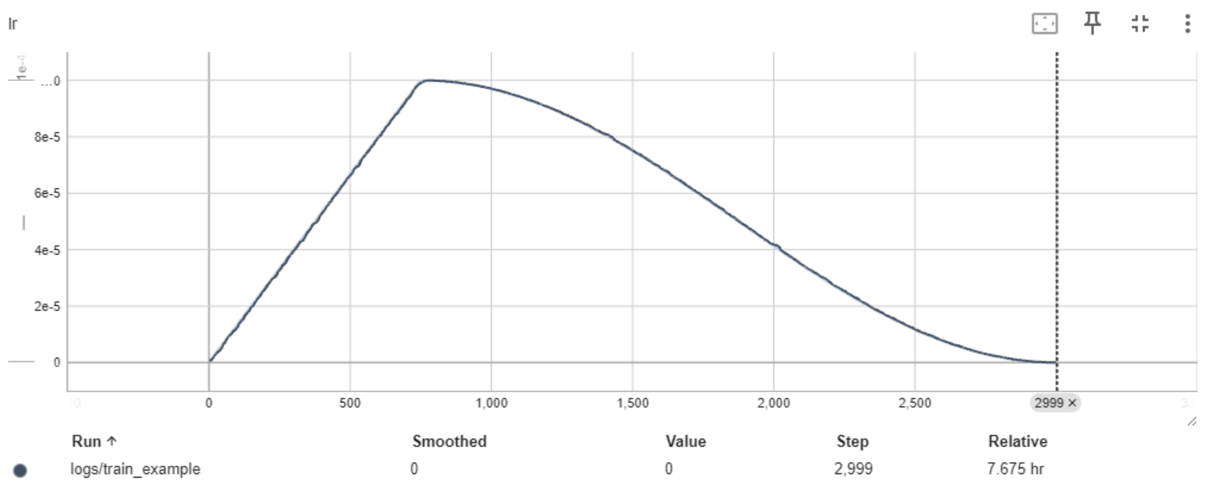
ii. The learning rate of the diffusion model for 2,999 steps for FULL_GUM_STAIN_CARDS model. The learning rate starts to decrease after 750 steps of warmup.

- WHITE_OR_MARKS_GUM_STAIN_CARDS



Run ↑	Smoothed	Value	Step	Relative
● logs/train_example	0.0079	0.0059	1,949	3.027 hr

i. The loss of the diffusion model for 1,949 steps for the WHITE_OR_MARKS_GUM_STAIN_CARDS model. This looks similar to the FULL_GUM_STAIN_CARDS model. I believe due to the more smaller datasets and tweaking of hyperparameters I was able to get a much stabler model to train on. This was again using the 160 x 128 sizes but with a higher lr warmup steps and a larger training batch size.



Run ↑	Smoothed	Value	Step	Relative
● logs/train_example	0	0	2,999	7.675 hr

ii. The learning rate of the diffusion model for 2,999 steps for WHITE_OR_MARKS_GUM_STAIN_CARDS model. The learning rate starts to decrease after 750 steps of warmup.

6. Code and Instruction to Run it
   - The code is at: https://github.com/megamp15/stable_diffusion_training
   - I have two notebooks
     i. The training notebook is to train the diffusion model and upload it to hugging face
     ii. The generator notebook is to pull the diffusion model from hugging face and generate cards using it.
   - The instructions are on the repository README but repeated here:
     i. The training notebook: *diffusion_model_training.ipynb*
        1. Install Python. I used version 11.9
        2. Create a virtual env: `python3 -m venv .venv`
        3. In the terminal activate the virtual env: `source .venv/Scripts/activate` or follow how to do this on your operating system. I am on windows.
        4. In the same terminal run the following command: `pip3 install ipykernel ipywidgets matplotlib accelerate`
        5. Follow the steps on the pytorch website to install additional libraries like torchvision and to use CUDA if available on your device.
        6. Open the *diffusion_model_training.ipynb* jupyter notebook
        7. If you are using vscode, click select kernel on the upper right corner, select python environments and then the name of the virtual environment you created which should be .venv
        8. Run the first cell to install libraries required for the notebook.
        9. Run the second cell to login to hugging face to upload the model once it is done training

10. Configure the fourth cell with your desired hyperparameters. You will also configure your hugging face repository name here. (Make sure the repository is created and it is public.)
11. Continue to run the rest of the cells to follow the tutorial on how to train a diffusion model but now using data from the *dataset/* directory.
12. Note: The dataset directory only has a sample of the type of gum stain cards I have been working with in my capstone. I cannot provide all the data.

ii. The generator notebook: *card_image_gen.ipynb*

1. If you are using VS Code, select the kernel as seen in step 7 in the earlier How to run the code section above.
2. Run the first cell to install any packages that may be missing
3. Run the rest of the cells one by one:
    a. Choose one of the following models you want to run by changing the `model` variable in the fourth cell:
        i. cards
        ii. all_gum_stain_cards
        iii. higher_res_gum_stain_cards
        iv. FULL_GUM_STAIN_CARDS
        v. WHITE_OR_MARKS_GUM_STAIN_CARDS
4. Run the fifth cell. This will pull down the model from hugging face and use the trained diffusion model to generate a card.
5. The images are generated and saved to the *generated_cards/*directory.