Answer all problems following the instructions. Please submit a PDF report with pictures of the output for each step followed by relevant code snapshots/screenshots that executed the function.

1 Assignment Instructions

Assignments in EEE 598 are due by the date assigned. You must use Latex to typeset your reports. **We will not accept Jupyter notebooks as reports.** Your report should have a section for each problem, and present your approach to solving the problem, any relevant code and figures to explain your method, and your results. Think about this like a job, you need to explain to your supervisor/boss what you did without dumping a whole bunch of code, answering questions in comments (big NO), and using labeled figures with captions. All figures must have axes labeled and an appropriate title.

2 Problems

Problem 1. *Music Generation*

1. This problem will have you create AI generated music beats using a Recurrent Neural Network. The following references have been provided to aid you in this task and offer datasets to download as well, but you are free to use your own:
   a) `https://www.tensorflow.org/tutorials/audio/music_generation`
   b) `https://www.geeksforgeeks.org/music-generation-using-rnn/`

   While these tutorials use standard RNN layers or LSTMs, you will be expected to implement a GRU layer for your network. You should also have 1-2 additional parameters controlling the note generation beyond what is shown in these tutorials. (Examples of these parameters from the tutorials include 'temperature' and 'pitch')

2. In your report, talk about the parameters selected for note generation and sequencing. Also show learning graphs and how you implemented the GRU layer. While you cannot upload your generated music to a pdf file, you should upload a 10 second portion (preferably the most interesting sounding part in your song) to a drive, allow sharing, and include the link in your report

Problem 2. *Make your own Positional Encoding*

In this problem, we are going to explore positional encoding based on the popular SIREN paper, and then make our own encoding.

1. Read the SIREN paper: `https://arxiv.org/pdf/2006.09661`
2. Reproduce the SIREN architecture on a high resolution image (at least 1024×1024) to show that a MLP can fit this high resolution image. You can use the Google Colab notebook provided on the paper website as a guide: `https://www.vincentsitzmann.com/siren/`.
3. Design your own activation function to be used instead of the sinusoid positional encoding. Mathematically define and graph your activation function
4. Show how your SIREN network with your own custom activation function performs for high resolution image fitting. If it doesn't perform as well as SIREN itself, then hypothesize why this might be the case and run an experiment of your own design to lend evidence to your hypothesis.

Problem 3. *Vision Transformer (Segmentation Task)*
In this problem, we will explore the task of image segmentation using a pretrained model and build a small application around it. You will apply segmentation to a selfie image, simulate a blurred background effect, and use monocular depth estimation to create realistic lens blur.

1. Explore the Hugging Face Segmentation Models Library for image segmentation. Choose any model of your choice. Take a selfie with a complex background (avoid using a plain wall) and preprocess the image by resizing and cropping it to a final size of 512×512. Run the image through the chosen model and demonstrate that the model can successfully segment the human figure from the background. Display the input image and the output mask, where the background should be completely black, and only the human should appear in white.
2. Add a Gaussian blur with $\sigma = 15$ to the background of the segmented image, leaving the human figure sharp. This simulates the background blur effect often seen in video conferencing tools like Zoom. Display the input image and the output image with blurred background side by side.
3. Explore the Hugging Face Monocular Depth Estimation Models Library and choose any depth estimation model. Run the same input image (512×512) through the model and display the input image alongside the depth map generated by the model.
4. Normalize the depth map values to a range between 0 and 15 to represent the relative distance of objects from the camera. Using this normalized depth map, apply a variable Gaussian blur to the image, where the blur intensity is proportional to the depth of the objects. Specifically, objects that are farther away from the camera (having higher depth values) should appear more blurred, while closer objects (with lower depth values) should remain sharper. This simulates a realistic lens blur effect, similar to what is observed with a camera's depth of field. Display the original input image alongside the output image with the applied depth-based lens blur for comparison.
5. **Extra Credits:**
   (a) Create a Google Colab notebook that contains all the resources necessary to run those models and generate results. Ensure that the images are linked, and all dependencies are included, so the results can be reproduced by simply running the notebook. Share the link.

(b) Develop a Hugging Face Space app that showcases the Gaussian blur and lens blur effects based on user input. Share the app link and a screenshot of the final output.

Problem 4. *Generative AI*

1. Follow the PyTorch tutorial for DCGANs: `https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html`. Try to train your DCGAN to get as best performance as possible. Show the resulting output images of your trained GAN.
2. **New Dataset of Colored Squares:** Create your own dataset of 64x64 images, with black backgrounds and colored squares. The dataset should have squares of different colors, sizes, and orientations. Create a dataset of at least 1000 images. Save these images when you generate them.
3. Train DC-GAN on your dataset of colored squares. How well does it perform? Does it learn to create squares? If not, keep increasing the size of the dataset until it does.
4. **Collect a Dataset of Favorite Animal:** Now pick a favorite animal of yours. Collect a small dataset of 500 images of this animal, resize all the images to the same size of 64 x 64. To make collecting images easier, you can try the following extension: `https://chrome.google.com/webstore/detail/download-all-images/ifipmflagepipjokmbdecpmjbibjnakm?hl=en`. Or any other resources that you can find on the web to download images in bulk.
5. Train DC-GAN on your new dataset. Try various tricks to improve performance (either add more data beyond the 500 images, dataset augmentation, different learning rates, network architecture/sizes). There's no right answer here, just try to get some performance or document any mode collapse you get.
6. Implement a Diffusion Model of your choice for your new animal dataset. You may use existing codebases for your implementation, please just cite which resources you utilize. Explain its performance compared to DC-GAN, does it perform better or worse? Make sure to justify your claims with experiments/analysis. Again, we are not looking for perfect image generation, but want to see good faith effort to try and get a diffusion model to output interesting images. You must show: (1) a figure showing the encoding process and the series of images getting progressively noisier (2) a figure showing the decoding/inference procedure going from noise to image sample (quality will vary here).

3 Grading and Report

The assignment report is very important for the grading of this assignment. We will return reports that we are not satisfied with the presentation, this is to help you practice improving your written communication skills. Grading breakdown is as follows:

- Problem 1: 20 points
- Problem 2: 20 points

- Problem 3: 20 points
- Problem 4: 20 points
- Presentation (are answers clearly defined and easy to find, are code snippets and figures utilized well for the report, overall readability): 20 points
- **Total:** 100 points

Please only upload the PDF of the final report (we do not want any code). Also acknowledge any web sources, classmate help that you used in this assignment in an acknowledgements section.