Project 2

Moira's Dream Car 🚘

1 Description

You and your team just started working for the up-and-coming PushPop Inc. last month and you have anxiously been awaiting your first assignment. In a morning meeting, your manager Raymond states that a new project was commisioned by a client. The client wants to use generative AI to construct images of cars appealing to certain members of a focus group. Your task is to generate images of cars that maximizes the preferences of certain members of the focus group.

1.1 The Data

The data consists of 6000 RGB images of various cars with resolution 96px × 128px, alongside annotations from six members of a focus group. The focus group have rated each car on a scale of one to ten, encoded as $s_i \in \{0,\dots,9\}^6$ for each of the $i = 1,\dots,6$ responders. The responders are named **Raof, Moira, Louie, Ferdinando, Gragar, Esther**, and their scores are presented in this order. The dataset is conveniently packed in the `quix.QuixDataset` format.

1.2 The Advice

In the meeting, your manager and two of your coworkers chime in with some advice on how to proceed.

Raymond: Manager

"I've heard that GANs are out and that diffusion models are all the rage nowadays. Just a couple of days ago I made an image of cat wearing a cowboy hat using OpenAI, and it looked brilliant! Clearly, you should implement a diffusion model for your project and dazzle the client with beautiful cowboy-hat wearning cars and whatnot. I think you should definately go with a state-of-the-art option to really dazzle the client."

Cosette: Senior Developer

"There is not much data, and there is not a lot of time. A diffusion model from scratch is a lot of work, and generally requires a U-Net. What about training a simple variational autoencoder model instead, and condition on the scores for the respondents? This will be a lot less work and produce decent results, especially given the amount of data you have at hand. You could experiment with a VQ-VAE or try different priors for the latent dimension if you want."

Spike: Senior Developer

"Yeah, a VAE is a better option, but I think you should try a regularized autoencoder with a Normalizing Flow model over the latent dimension. Just train an autoencoder with an additional L^2 regularization on the latents, and then train an NF model with whatever prior you want on the latent dimension. I read somewhere that this improves the blurring artifacts which you often get with VAE models."

With these three options in mind, you start to wonder what approach would be the most appropriate and whether GANs would really be such a bad choice for the project? You will have to choose which—if any—advice to go with, and discuss your choices in your final report.

2 Detailed Project Description

This task is designed as an *experiental* learning experience. You are provided with a scenario, and relative freedom to pursue whichever option you find most appropriate for the task at hand. You are free to go with any of the three alternatives provided by your coworkers, or ignore them entirely to decide on a model you find most appropriate for the task. You are free to use any pretrained model and architecture you like for the project. All that is required is that your generative model needs to be conditioned on the responses from the focus group in some way of form, such that you can generate images according to the preferences of the selected respondents.

For Master Students: Your task is to generate 10 images which are optimal for pleasing **Moira**, the second respondent in the focus group.

For PhD Students: Your task is to generate 10 images which are jointly optimal for pleasing **Moira** and **Ferdinando**, the second and fourth respondents in the focus group, respectively.

Groups with one or more IN9310 students need to perform the task with joint optimization for **Moira** and **Ferdinando**.

2.1 Model Evaluation

At the end of the project, you will produce a set of 10 images from your model, which will be evaluted, and given a mean score by the respondent. The images should be stored in two files; one file called `images.pth` containing a PyTorch tensor with the images of shape $[10, 3, 96, 128]$. The second file `embeddings.pth` should contain the embeddings used to construct your images with your model, ideally of shape $[10, D]$ where D is the embedding dimension of your model. These embeddings should preferably include the conditioning vectors in some way or form.

2.2 Loading the Data

The data for the project is located in the folder `/projects/ec517/data/CarRecs`, and consists of six `.tar` files which contains the `.jpg` images and `.scores.npy` files containing the respondent scores for each image. In addition, the folder contains a list of seek indices for the `.tar` files, named `train.idx`. Due to the nature of the task, there is no validation set required, hence the `.tar` file and index for the validation set is empty. To load the data, we recommend using the `quix` module, which is included in the `in5310` venv for the EduCloud resources for the course. You can use the following code for loading the data:

```
# Import necessary modules
import quix # For quixdata public repo, use import quixdata as quix
import torchvision.transforms as T

# Specify data folder
datapath = '/projects/ec517/data/'

# Define mean and std from ImageNet data
in_mean = [0.485, 0.456, 0.406]
in_std = [0.229, 0.224, 0.225]

# Define postprocessing / transform of data modalities
postprocess = (                         # Create tuple for image and class...
    T.Compose([                         # Handles processing of the .jpg image
        T.ToTensor(),                   # Convert from PIL image to torch.Tensor
        T.Normalize(in_mean, in_std),   # Normalize image to correct mean/std.
    ]),
    T.ToTensor(),                       # Convert .scores.npy file to tensor.
)

# Load training data
data = quix.QuixDataset(
    'CarRecs',
    datapath,
    override_extensions=[               # Sets the order of the modalities:
        'jpg',                          # ...load image first,
        'scores.npy'                    # ...load scores second.
    ],
).map_tuple(*postprocess)
```

From here, you can create a dataloader directly using `data`. For better shuffling behaviour, we also recommend you use

```
with traindata.shufflecontext():
    # do training loop...
```

instead of enabling shuffling in the dataloader. This ensures better IO performance on the server and speeds up training. If you are using a custom virtual environment, you can install `quix` using the terminal commands

```
# For ssh
pip install git+ssh://git@github.uio.no/xpt/quix.git
# For https
pip install git+https://github.uio.no/xpt/quix.git
```

3 Project Report

Your final report should be written like a workshop/conference paper. You can take a look at different papers from major conferences (CVPR, ICCV, ECCV, NeurIPS, ICML, or ICLR) to get a general sense of how the papers look and are organized. You can use the LaTeX template provided in the example repo (see the `report` branch) as a starting point.

Your report should follow the same structural format as the first project. In this project, we place more emphasis on your ability to find relevant research to discuss your findings, contrast your work, and motivate or argue convincingly for your findings and any claims you make in the process. In particular we want you to focus on:

- Why did you decide (a priori) on your selected approach?
- In hindsight (a posteriori), would you have selected a different approach?
- What worked well, or what failed in your approach, and why?

As before, we restrict the report to have a maximum of 8 pages, excluding references, but we recommend the report to be around 6–7 pages this time. We also encourage you to use figures and tables to present your methods and results. Remember to properly cite any figures you use that you haven't made yourself. If you have more specific questions of the report requirements and style, either refer to the instructions for ICLR 2023 or ask someone from the teaching team.

You must include a "Group Contributions" section (before the references) where you detail what each member of your group did. You could have a paragraph detailing what each member was responsible for and what tasks did they perform, or you could draw a contribution matrix detailing the main steps and responsibilities and report the percentage of involvement of each team member. This section *does not count towards your page limit.*

4 Evaluation

Your submission consists of the project report and the code in your group's GitHub at the deadline.
The deadline is Friday 26/09. The grade is pass/fail, and will be defined by the following aspects:

1. **Project report** **85%**
   (a) Introduction, motivation, and background 10%
   (b) Related work
      - Exposition of relevant literature to the task 5%
      - Relating relevant results to your experimental findings 10%

   (c) Explanation of your methodology 15%
   (d) Experiments, discussion, and results
      i. Arguments, inference, and evidence for optimization of preferences 15%
      ii. Motivating your a priori choice of model 10%
      iii. Posterior discussion on choice of model 10%
      iv. Discussion on image quality and what worked/failed 10%
2. **Code** **15%**

Your language usage won't be graded, but your ability to present your results, ideas, and how they are supported will be. Each other point will be evaluated according to the completeness and correctness of the requested items.

5 Submission

Your submission must be through your assigned group on Github and Devilry. Use the repository named `g#-p2` for your group number and commit all the code and report there. You can start out by cloning this example repo, which has one `main` branch for pushing the code and a `report` branch for pushing the report files. For instance,

```
1 $ git clone --bare git@github.uio.no:2025-s2-in5310-in9310/example-p2.git
2 $ cd example-p2
3 $ git remote rm origin
4 $ git remote add origin git@github.uio.no:2025-s2-in5310-in9310/g#-p2.git
5 $ git push --force --all origin
```

will clone the `example-p2` repository into your own group. Note that in line 4 you need to replace `#` with your group number.

5.1 Code

Your submission must have the following subfolders in the `main` branch of your repository:

- `output`: a directory where your application should produce all the generated files, including a state dict with model weights (e.g., `final_model.pth`), the generated images (`images.pth`), and the embeddings used to generate the images (`embeddings.pth`). 🚀 This folder and all its contents must be added to the artifacts in case you setup a workflow for your code.
- `src`: a directory containing all your source code. You only need to submit files that are not derived from other files or through compilation. In case some processing is needed, prefer to submit a script that does that instead of submitting the files.
- `Makefile`: a makefile that executes your code through a docker image or that is setup in a way that construct its own environment for reproduction. You are encouraged to use the standard pytorch image, `pytorch/pytorch`, but other solutions are possible. ⚠️ Docker is not installed on the Educloud servers. If you're using the Educloud servers to run your code, you can use the `in5310` virtual environment instead of docker to ensure that we can execute your code. More detailed instructions can be found here. Discuss with the teaching team early on your setup to ensure its reproduction. The code will be executed through a standard call to `make`, so other dependencies must be provided by you under that constraints. 🚀 Moreover, note that your code **must** run without additional prompt or changes from the user.

Additionally, you need to write a script `validate_project2.py` with a function `load_my_models()`. This should take no arguments and return your final model with all weights loaded. The teaching team will import this script for testing your models when we evaluate your projects. Use the `validate_project2.py` example available in the example repo as a starting point. You only have to complete the `load_my_models()` function. The reason you have to write this script is to ensure that we load your model correctly!

5.2 Evaluation and Expectations

Your model will be evaluated in three steps:

1. Passing your provided embeddings / latents through your model and comparing with your submitted images.
   ⚠️ It is therefore essential to make sure the ordering of the embeddings and images are correctly aligned!
2. Checking that the images are reasonably distinct from all the images in the original dataset.
   ⚠️ Make sure that the images produces are not simply copies of images from the dataset.
3. Evaluating the preference scores of your images, which is simulated by a seperate model.

We do not expect your model to produce high fidelity images which are indistinguishable from the original data, but we do want you to discuss the quality of the generated images in your report 📝.

Furthermore, we do not place significant weight on how well your model aligns with the scores of the respondents. Instead we want to hear how well you expect the generated images to align with the scores in your report, and importantly how you make this argument based on relevant research and experimental evidence 📝.

5.3 Report

All files pertaining to the report should be uploaded in a separate `report` branch. You need to include the following:

- All the source files (e.g., `report.tex`) that produce your report must be here.
- 🚀 Your report **must** be written using LaTeX (and friends) and compiled within the workflow in this branch. ⚠️ Consequently, **the PDF must not be committed**. You can use the images from `adnrv/texlive` to build your PDFs. **Only the PDF of the report** must be added to the `artifacts` path on the workflow setup, and not other intermediary files of your report (e.g., images, log files, auxiliary files). You can follow the `example` repository (or directly clone it into your own repository) to reuse the existing workflow.
- Your report must show all your work for the given project, including images (labeled appropriately, that is, following the convention given) and other outputs needed to explain and convey your work. **The constraints of this report are explained in § 3.**
- Your report in the last commit before the deadline **must** be the same as the one submitted in Devilry.

🚀 The last commit before the submission deadline will be used to review your code and the report. Do not worry about executing times for building the report. However, **you must ensure that your code works as no attempt will be made to patch or run broken code**.

6 Notes

- ℹ️ Note that there are several implementations that you can find on the internet. This project is for **you to implement** the algorithms. Thus, **do not submit code from others**. And if you re-use code from someone for a non-restricted part, disclose it in your report and code.
- ℹ️ All the submissions must be self-contained and must be executable in a Linux environment. Your code could execute in the docker image `pytorch/pytorch`, available at docker hub (`https://hub.docker.com/r/pytorch/pytorch/`). If not, then you must ensure that your entry point sets up the environment needed to reproduce your work without further inputs from the user.
- ℹ️ It is your responsibility to make sure your code compiles and executes correctly. No effort will be made to run your code besides executing `make`.
- ℹ️ You must program in Python. If you need to install other packages you must do so within your `Makefile` as automatic prerequisites.

Please contact someone from the teaching team at a reasonable time before the deadline if you are having problems.

6.1 Final Note

⚠️ *Training generative models from scratch is tricky*, especially under the time constraints of the project. If you feel like the results are not as you initially expected, focus on detailing your modeling decisions, reporting on what works and what does **not** work in the report. We advise you to be **thorough in your literature search**; read related papers, reference them, and contextualize your modeling choices.