SSY098 Project Descriptions 2025

- This document contains four project proposals. Each project has 40 available spots, and you may sign up for your preferred project via Canvas. Allocation is on a first-come, first-served basis, and you are free to choose any of the available options. Sign-up opens on **Monday, May 19 at 14:00**. Each project has its own supervisor (TA) who will be responsible for its supervision:
  - Project I: Analysis of a Classification Model - Sophia
  - Project II: Memory - Victor
  - Project III: Structure-from-Motion in 2D - Sofie
  - Project IV: Diffusion Models - Josef
- Each project consists of three parts: (i) A mandatory part, (ii) a theoretical part, and (iii) an advanced part.
  - In order to pass the project with a grade of 3, you need to complete the mandatory part.
  - Completing the mandatory part (i), and either the advanced part (ii) **or** the theoretical part (iii), will allow you to pass the project with a grade of 4.
  - Completing the mandatory part (i), and both the advanced part (ii) **and** the theoretical part (iii), will allow you to pass the project with a grade of 5.
  - The theoretical part (iii) consists of three questions, each of which is worth 1 point. You need 2 points to pass the theoretical part. Your answers will be graded on a continuous scale, i.e., we try to reward partially correct answers.
- Submission:
  - Use a similar structure as in the labs to organize your code. You should submit a Python notebook file `main.ipynb` and an HTML printout. You can choose between defining your functions directly in the notebook file or in a separate `functions.py` file. Submit all files through Canvas.
  - In addition to your Python code, you will need to write a short report describing your project and providing experimental results. If you submit answers to the theoretical questions, include your answers in the report as well. Please use the IEEE conference template found here for your report. Submit your report as a PDF file through Canvas.
- **Note:** Projects must be done individually and may not be carried out in groups. No collaboration is allowed, and you are only permitted to ask for help from the teaching assistants or the lecturer. Copying solutions, even partially, from any source is strictly prohibited. If you use any non-human source beyond the course material and literature, such as a research article or a book, you must include a proper reference. You may use LLMs, but your report must contain a clear, complete, and truthful description of how they were used. Any undocumented, misleading, or incorrect description of LLM use will be treated as cheating.

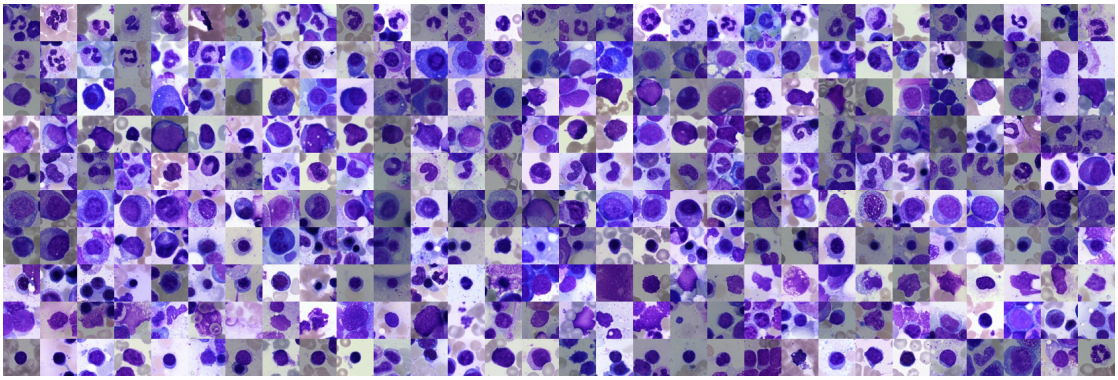

Figure 1: Some samples from the Bone Marrow Cell Classification dataset.

1 Project I: Analysis of a Classification Model

The goal of this project is to develop a model to classify trees based on images of their bark. This will be done with an at least partially convolutional neural network, which is then further improved.

1.1 Mandatory Part

Design a convolutional neural network, optionally with some fully connected layers, and train it on a dataset derived from the Bone Marrow Cell Classification dataset. The dataset was created by selecting a subset of the original images and resizing them, giving you a balanced set of classes. Training and test sets were separated randomly. Download here.

This file contains a train and a test directory, both then containing eight directories representing classes[1]. Elements of the dataset are $56 \times 56 \times 3$ colour images in the JPEG format with values from 0 to 255[2]. There are 32000 train and 24000 test images.

Even a fairly shallow model should be able to reach 70% accuracy on the test data with $15 - 20$ epochs of training. If you have a CUDA-enabled GPU (or a lot of time), you can also try for a deeper model.

Once you have this, evaluate with confusion matrices, training plots, and any further evaluation methods of your choice where your original model is lacking. Based on this choose **at least three** methods to improve your model in these aspects:

- **Pre-processing**: Modifying the data in some non-learned way can often improve training significantly. For example, you could use PCA to extract the dimensions carrying the most information.[3]
- **Early stopping**: Use a validation set split from the training data to decide when to stop training. The method cvpartition may come in handy for doing this in a stratified way.
- **Batch Normalisation**: Including batch normalisation layers before some or all of your activation functions often makes training markedly easier.

[1] The ImageFolder class might be helpful for loading this data.

[2] A different range might work better for your model. I suggest to already consider this for the base network, perhaps even apply standardisation (based on training data only!).

[3] Neural networks are theoretically able to implement the same transformations as PCA themselves, but applying them manually can help in some cases.

- **Optimisers and Activation Functions**: The choice of optimisers, activation functions, and their hyper-parameters is crucial for training. While Adam and ReLU are often solid options, it can pay to try others.
- **Data Augmentation**: Use rotations, flipping, cropping, or any other methods to create additional training images from the data you already have.
- **Other**: You are encourage to research other methods on your own.

Explain in detail why you chose these specific methods. With this you should reach a test accuracy of 81% or more[4].

Finally, do an ablation study with your improvements. I.e., compare your final model to additional models, each including all but one of your improvements, showing you what impact the absence of a given improvement has. This should be done based on test accuracy, as well as confusion matrices for all models (as well as any other metrics you consider useful).

What to include in the report. Your report should consist of the following:

- A short abstract describing the problem behind the project and outlining your solution.
- A "Method" section describing your approach: What network architecture are you using? What improvements did you choose and what problems are they supposed to solve? This section should be about one page long.
- A "Experimental Evaluation" section describing the experiments you perform with your approach as well as the results of these experiments:
  - Describe the parameters you used for training, i.e., your optimiser, the learning rate, whether you used momentum, etc.
  - Show the improvements you used. E.g., for early stopping plot the training and validation accuracy during training, for pre-processing visualise how you changed the images, etc.
  - Include the results of your ablation study. Do you see any pattern in the confusion matrices? Did your improvements achieve what you wanted them to? Compare what happened to the expected improvements you described in the "Method" section.
  - Choose a subset of test images and visualise (e.g., with a border) which ones were classified correctly and which one weren't. Explain why you think they were classified this way.

1.2 Advanced Part

To classify objects within a larger image, it is important to detect if a given frame contains an object at all. There are many methods to do so, most of which involve training the model with this capability to begin with. However, for the advanced part here, you will instead extend the previous classification network without any new training.

Specifically, your task is to separate the 2000 images provided here into in-distribution (ID) samples (from the earlier training and test set) and out-of-distribution (OOD) samples (in this case images of blood cells.).

[4]Note that normally you, of course, shouldn't attempt to improve a model based on test data results, as that may introduce a bias toward that dataset. Each phase would need each own new test data afterwards. However, it does commonly happen in practice, as there is often limited data available.

You should not modify the network, but rather use the information you can get from the activations of your network. Try to find a way to create a scalar score for each sample, usually from the output of one of the later layers but not necessarily the last. You are likely successful if you manage to get two distinct hills in a histogram of all the scores. Can you also figure out which hill corresponds to ID and which to OOD data[5]?

What to include in the report. Extend your report to include the following:

- Extend the "Method" section by a subsection describing how you searched for useful features, and what you did with them to detect OOD samples.
- Extend the "Experimental Evaluation" with a histogram showing the distribution of scores, colouring and labeling it according to which parts you believe to be ID/OOD data. Argue for your decision.

1.3 Theoretical Part

Answer the following questions in your report. Make sure to use your own words and justify your answers.

(a) Fully convolutional neural networks vs. neural networks with fully connected layers. Explain the purpose of fully connected layers in convolutional neural networks. In which scenarios would you apply a fully convolutional neural network instead of a neural network with a fully connected layer? Describe at least two, including at least one that cannot also be solved by adaptive pooling[6].

(b) Generative models. (i) In your own words, explain why we optimize the Empirical Lower BOund (ELBO) rather than $\log(p(\mathbf{x}))$ when training a VAE. (ii) What are the major differences between VAEs and Diffusion probabilitic models? Answer in terms of the encoding/decoding processes in the two methods, and the latent variables.

(c) Triangulation. (i) Triangulation is used in order to create 3D points in Structure-from-Motion. Given a set of 2D pixel positions $\mathbf{x}_i$, $i = 1, ..., k$, in k images and the projection matrices $\mathtt{P}_i$ of these images, explain how to obtain the 3D point that best represents these 2D measurements.
(ii) Consider the setup with $k = 2$ images with known relative pose. Please identify in which case there are several 3D points that project exactly to $\mathbf{x}_1$ in the first image plane and to $\mathbf{x}_2$ in the second. Is it possible to derive the position of $\mathbf{x}_1$ and $\mathbf{x}_2$? Motivate your answer!

[5] Hint: You are allowed to use known data from the base part, but do not directly compare the images to find the inliers.

[6] The paper introducing the concept can be accessed via the Chalmers Library by selecting "Access Through Your Institution". Other resources may be used, e.g., this medium article on adaptive average pooling.

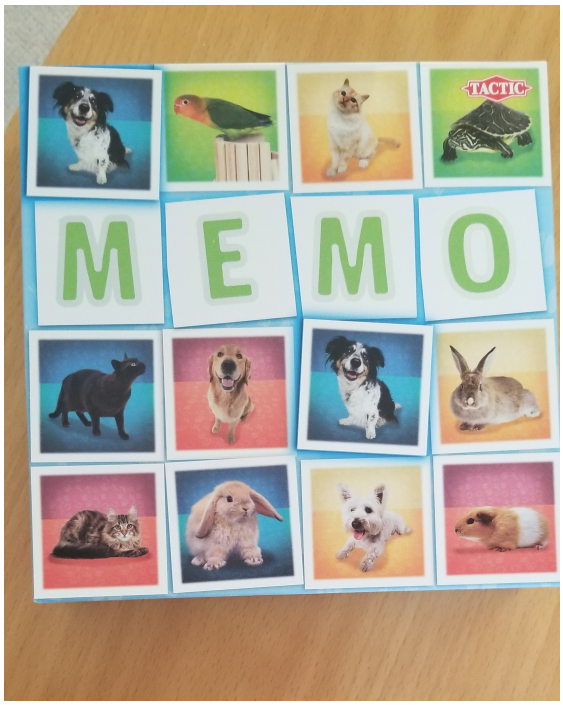

Figure 2: The card game “Memory”.

2 Project II: Memory

The goal of this project is to develop a vision system for playing “Memory”. The system will act as a guide and keep track of all matching pairs.

You can collect cards from the lecturer (or if you like, use your own cards). You will take your own images for the project, so you need for instance a mobile phone camera.

2.1 Mandatory Part

The vision system should have a number of capabilities in order to assist playing the game, from start to finish. The system shall keep track of how many correct card pairs each player has been able to match. Basic image analysis functionality includes:

- Detect the location of all cards in an image.
- Detect which cards are face up and which cards are face down.
- Determine if two cards depict the same picture or not.

The formal rules for the card game are as follows. *Collect as many matching card pairs as possible, that is, two cards with the same picture. Spread the cards face down on the table. One player starts and the turn passes clockwise. On each turn the player turns two cards, so that everyone can see the pictures on them. If the cards match, the player takes them. He may continue until he turns two cards that don't match. When this happens, he turns the cards back face down and the turn passes to the next player. The player who has the most pairs at the end of the game wins.*

For simplicity, we will assume only two players. Also, we will assume that the system is only fed the essential images of the card game. The first image will be an image with 16 cards face

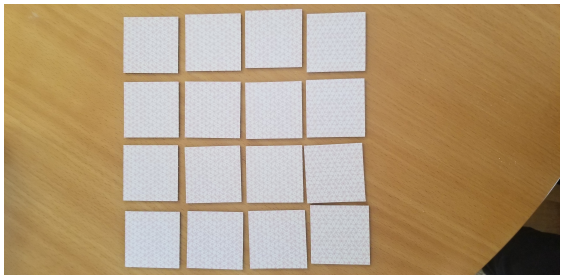

Figure 3: The card game "Memory", starting board. Your system should locate and draw bounding boxes as output.

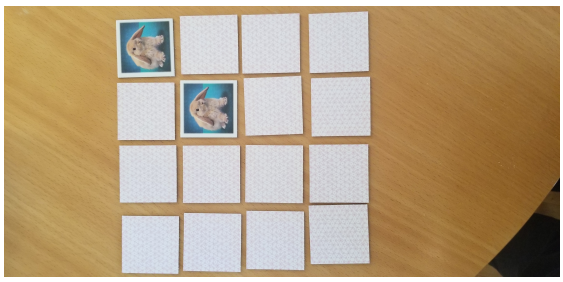

Figure 4: A matching pair! Your system should locate and draw **green** bounding boxes on the matching pair.

down, arranged in 4×4 grid, similar to Figure 3. The system will act as a guide and instruct the players. So, from the first image, the system shall localize the cards and the output should be bounding boxes of the cards and the instructions *Player 1, please choose two cards to turn.* Then, the system is fed the next image with two cards turned and the system shall locate which cards have been turned and also decide if they form a matching pair. The output should either be *Congratulations! A matching pair. Please pick up your matching card pair and then you may continue.* as in Figure 4 or *No match! Turn back the two cards face down again. Player 2 may then continue.* as in Figure 5. And so forth. In the end, when all pairs have been found, the system shall conclude how many pairs each player obtained and declare a winner (or a draw).

What to include in the report. Your report should consist of the following:

- A short abstract describing the problem behind the project and outlining your solution.
- A "Method" section describing your approach.

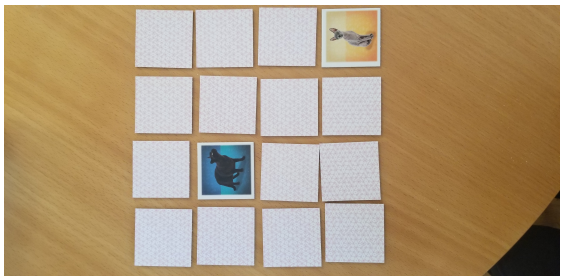

Figure 5: No matching pair! Your system should locate and draw **red** bounding boxes on the non-matching pair.

- A "Experimental Evaluation" section presenting a whole round of play, from start to finish with all intermediate output images and text captions. Note that the detected cards of interest should have bounding boxes as described in Figure 3, Figure 4 and Figure 5. You must also present appropriate quantitative metrics measuring how well face-up and face-down cards are detected, and how well cards are correctly identified.

.

2.2 Advanced Part

In the advanced part, you should extend the functionality of the system in order to handle arbitrary rotations of the cards. Hence, unlike Figure 3, where the cards are organized in a 4×4 grid, the cards can be spread out arbitrarily on the table. We will assume that no cards are overlapping, and as before, when a card is turned, it is put at (approximately) the same location. You should also demonstrate that your solution is robust to (small) camera pose changes, so that the camera is not viewing the cards from above (that is, the image plane is not parallel to the plane with the cards).

For the report, you should report the same things as in the mandatory part, but you should of course additionally include how you solved the problem of unknown orientation of the cards and you should show a whole round of play with cards spread out arbitrarily on the table with a camera that is not viewing the cards from above.

2.3 Theoretical Part

Answer the following questions in your report. Make sure to use your own words and justify your answers.

(a) Overfitting. Explain what overfitting is and what causes it. Name and briefly describe two methods to combat overfitting in the context of neural networks.

(b) Generative models. (i) In your own words, explain why we optimize the Empirical Lower BOund (ELBO) rather than $\log(p(\mathbf{x}))$ when training a VAE. (ii) What are the major differences between VAEs and Diffusion probabilitic models? Answer in terms of the encoding/decoding processes in the two methods, and the latent variables.

(c) Triangulation. (i) Triangulation is used in order to create 3D points in Structure-from-Motion. Given a set of 2D pixel positions $\mathbf{x}_i$, $i = 1, ..., k$, in k images and the projection matrices $\mathtt{P}_i$ of these images, explain how to obtain the 3D point that best represents these 2D measurements.
(ii) Consider the setup with $k = 2$ images with known relative pose. Please identify in which case there are several 3D points that project exactly to $\mathbf{x}_1$ in the first image plane and to $\mathbf{x}_2$ in the second. Is it possible to derive the position of $\mathbf{x}_1$ and $\mathbf{x}_2$? Motivate your answer!

3 Project III: Structure-from-Motion in 2D

In this project, you will implement a Structure-from-Motion pipeline in 2D, which is equivalent to stitching together multiple 2D images of a planar surface.

3.1 Mandatory Part

You can start by modifying your RANSAC implementation from Lab 3. Rather than estimating an affine transformation, you should implement a function that estimates the homography between four correspondences (see the DLT algorithm presented in the lectures). When implementing the DLT algorithm, you should apply *normalization* to the input points. For each set of points you should:

- Translate the points so that their centroid is at the origin.
- Scale the points so that the average distance from the origin is $\sqrt{2}$.

This normalization is done using a similarity transform matrix T, which is applied to all input points before running DLT. After estimating the homography H' from the normalized points, you obtain the final homography as:

$$H = T_{\text{target}}^{-1} \cdot H' \cdot T_{\text{source}}.$$

Given your RANSAC-based method for estimating the homography between two images, implement the following pipeline:

- For a given set of images, extract SIFT features in each image. Next, match SIFT features between pairs of images and estimate a homography for each pair. Store each estimated homography, together with the inlier matches to it. Use code from Lab 3 if possible (e.g., for feature extraction and matching).
- Select the image pair with the largest number of inliers as the starting pair. The first image thereby provides a global coordinate system and the homography `H` provides the mapping from the second image into the global coordinate system.[7]
- Iteratively add the other images: Among all images not yet included, select the image I that has the largest number of homography inliers with any of the images already included in the reconstruction. Let J be the corresponding image in the reconstruction and $\texttt{H}_{J,I}$ the homography from I to J. Use $\texttt{H}_{J,I}$ and the known mapping $\texttt{H}_J$ from the coordinate system of J into the global coordinate system to compute the global homography $\texttt{H}_I$ for image I. This step is the 2D equivalent to the "Extend motion" step from the Structure-from-Motion pipeline presented in the lecture.
- Finally, create a sparse map of the scene: For each image I in the reconstruction, use $\texttt{H}_I$ to map all SIFT features found in I into the global coordinate system. Visualize the resulting 2D point cloud.

You can use the "graf" and "wall" (both showing changes in viewpoint), as well as the "bark" and "boat" (both showing zoom and rotation changes) datasets from here to evaluate your approach.

[7] Assuming that your homography maps pixels in the second into the first image. If this is not the case, you need to invert the homography.

What to include in the report. Your report should consist of the following:

- A short abstract describing the problem behind the project and outlining your solution.
- A "Method" section describing your approach: Provide pseudo code for your 2D Structure-from-Motion system and explain each step in your own words. This section should be about one page long.
- A "Experimental Evaluation" section describing the experiments you perform with your approach as well as the results of these experiments. Specifically,
  1. List the thresholds that ared used in your RANSAC-based homography estimation for each dataset in a table. The thresholds include the inlier threshold that you use for each dataset and the corresponding maximum number of RANSAC iterations you run.
  2. Visualize the 2D sparse maps in the global coordinate for each dataset and indicate the number of the image regarded as global coordinate in the caption.[8]
  3. The dataset also provides a ground truth homography from the first image in the sequence to each other image. Use these ground truth homographies to create sparse maps and visualize them next to your maps. Make sure to visualize the sparse map in the same global coordinate as 2) for each dataset.
  4. For each dataset, measure the average residual between each sparse point in your map and the corresponding point position in the ground truth map. Report the results in a table.

3.2 Advanced Part

As a post-processing step, improve your 2D Structure-from-Motion pipeline by using the 2D equivalent of bundle adjustment.

- **Form feature tracks:** If feature i in image I and feature j in image J form an inlier correspondence, and if feature k in image K and feature j in image J form an inlier correspondence, then i, j, and k form a feature track. Extend this definition to include all images where consistent matches exist.
- **Estimate initial 2D point positions:** For each track, transform all observed feature positions into the global coordinate system using the corresponding homographies. Compute the global point position as the average of these transformed coordinates.

Once you have obtained a set of global 2D point estimates, perform a non-linear optimization to **jointly** refine the 2D coordinates of all global points, **and** the homographies (assume the bottom-right element is fixed to remove scale ambiguity).

The goal is to minimize the reprojection error for all observed features. You should solve this problem using the Gauss-Newton method, i.e. at each iteration n, compute a parameter update

$$\theta^{(n+1)} = \theta^n - (J^\top J)^{-1} J^\top \mathbf{r}$$

where θ includes all 2D point coordinates and homography parameters stacked into a single vector, and J is the Jacobian of the residual vector $\mathbf{r}$ with respect to θ.

[8] The sparse map simply means all estimated 2D points, shown in the coordinate system of the reference image.

Hint 1: You do not need to compute the Jacobian analytically. Instead, you can use **finite difference approximation**. For each parameter θ_k, approximate:

$$\frac{\partial \mathbf{r}}{\partial \theta_k} \approx \frac{\mathbf{r}(\theta_k + \epsilon) - \mathbf{r}(\theta_k)}{\epsilon}$$

You should only compute residuals for actual feature observations. Each residual depends only on one 2D point and one homography, so the Jacobian is sparse.

Hint 2: To reduce computational cost, use a random subset of 2D points and their observations when building the residuals and Jacobian.

After optimization, reconstruct the sparse 2D map by transforming all observed features from each image into the global coordinate system using the optimized homographies.

What to include in the report. Extend your report to include the following:

- Extend the "Method" section by a subsection by adding a description of the method that you implemented in your own words.
- Extend the "Experimental Evaluation" section by adding visualizations of the 2D maps generated with your more advanced pipeline. Also measure the errors in your 2D point positions with respect to the ground truth map you generated before.

3.3 Theoretical Part

Answer the following questions in your report. Make sure to use your own words and justify your answers.

(a) Bundle Adjustment in 2D. Given a set of 2D points and the homographies mapping points from a global coordinate system to the coordinate system of each image, derive the equation that should be minimized during Bundle Adjustment. As for standard Bundle Adjustment, the goal is to minimize a sum of squared reprojection errors between 2D points and the corresponding measurements in the images. Explain your equation and make sure to explain all variables that you use.

(b) Generative models. (i) In your own words, explain why we optimize the Empirical Lower BOund (ELBO) rather than $\log(p(\mathbf{x}))$ when training a VAE. (ii) What are the major differences between VAEs and Diffusion probabilitic models? Answer in terms of the encoding/decoding processes in the two methods, and the latent variables.

(c) Triangulation. (i) Triangulation is used in order to create 3D points in Structure-from-Motion. Given a set of 2D pixel positions $\mathbf{x}_i$, $i = 1, ..., k$, in k images and the projection matrices $\mathtt{P}_i$ of these images, explain how to obtain the 3D point that best represents these 2D measurements.
(ii) Consider the setup with $k = 2$ images with known relative pose. Please identify in which case there are several 3D points that project exactly to $\mathbf{x}_1$ in the first image plane and to $\mathbf{x}_2$ in the second. Is it possible to derive the position of $\mathbf{x}_1$ and $\mathbf{x}_2$? Motivate your answer!

4 Project IV: Diffusion Models

Diffusion probabilistic models have recently become quite popular in Computer Vision due to their ability to generate high-resolution realistic images. In this project, you will be diving deep into the actual working of such models. You will work your way through the math involved and the tricks used to train neural network models to accomplish the generative task using this method. Due to limited data and computation available, you will be working with simple toy data.

Notes

1. It is recommended to use PyTorch for mandatory part-2 and the advanced part.
2. blog 1 and blog 2 are excellent sources that can serve as tutorials for the intrinsics of diffusion models.

4.1 Introduction

As discussed in the lectures, a (denoising) diffusion model consists of 2 parts:

1. A pre-defined forward diffusion process q that gradually adds small amounts of Gaussian noise to the original data, until you end up with pure noise.
2. A learned reverse (denoising) model p_θ that removes the noise to eventually produce something that hopefully looks like your original data.

Assuming a data point to have been sampled from an original distribution $\mathbf{x}_0 \sim q(\mathbf{x})$. We sequentially add Gaussian noise to it at timesteps $1 \dots T$ as per the conditional distribution:

$$q(\mathbf{x_t}|\mathbf{x_{t-1}}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta\mathbf{I}) \tag{1}$$

where the step sizes are controlled by the variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$.

4.1.1 Mandatory task 1

If somehow the reverse process $q(\mathbf{x_{t-1}}|\mathbf{x_t})$ was available, one would be able to produce meaningful data samples by repeatedly applying this denoising process starting from pure noise. Here we will consider a simple 1D case: $\mathbf{x_0} \sim \mathcal{N}(\mathbf{x_0}; m, p)$ and the forward diffusion process as mentioned above in (1). Assuming that the distribution of $\mathbf{x_0}$ is known, the reverse conditional distribution $q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0})$ can be computed in closed form, using the Bayes rule and the product of Gaussian distributions.

- Choose your own parameters (m, p) for the distribution of $\mathbf{x_0}$ and sample from this.
- Run the forward diffusion process for a number of timesteps, $T = 50$. You can experiment with different variance schedules (eg. linear, quadratic, etc.), but for simplicity, you can start with a constant schedule such that $\beta_t = 0.25\ \forall t \in \{1 \dots T\}$. By repeating this process for various inputs, verify that the values at time T approximately follow a standard normal distribution. Visualize a few forward trajectories and also the distribution of x_T computed over at least 100 trajectories starting from different realizations of $\mathbf{x}_0$ by plotting x_t against t for $t \in (1 \dots T)$.
- Compute the reverse process as a function of $\mathbf{x_t}$, $\mathbf{x_0}$ and t. You can take help from the derivations mentioned here.

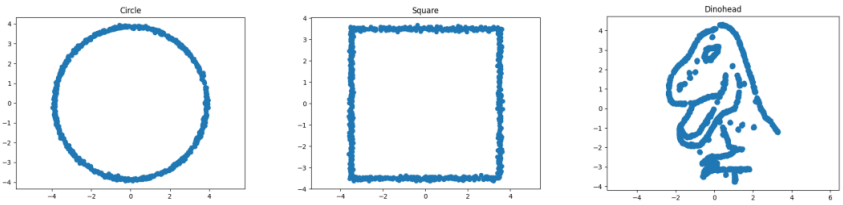

Figure 6: 2D point cloud datasets for mandatory task 2 - *Circle*, *Square* and *Dinohead*

- Sample $\mathbf{x_T}$ from a standard normal distribution and repeatedly apply the reverse process computed above to obtain $\mathbf{x_0}$. Visualize a few of the reverse trajectories.

4.1.2 Mandatory task 2

In this task you will actually predict the $q(\mathbf{x_{t-1}}|\mathbf{x_t})$ using a neural network with parameters θ. Hence this prediction/appoximation will be termed as $p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t})$. To learn this, you will closely follow the approach presented in [1]. You are provided sets of 2D points in the following shapes - a *Circle*, a *Square*, and *Dinohead* - an outline of a (rather simplistic) dinosaur head (Fig. 6). The task is to learn to produce something close to these starting from standard Gaussian noise. You are supposed to train a different model for each shape. This is not a great example to showcase the generative ability of diffusion methods, however, it is interesting in the sense that it allows understanding and implementing the intrinsic details of the method.

Use the following steps as guidelines to implement a training regime that produces the required output. While training the model for a particular shape example, do the following within each training loop:

- Sample a batch (randomly selected small subset) of points and for each point - a timestep, and noise - this is the true noise.
- Predict the 2D noise $\epsilon_\theta(\mathbf{x_t}, t)$ using $\mathbf{x_t}$ and t as inputs.
- Using squared error between the predicted and true noise as the loss, back-propagate and update the model parameters using gradient descent.

Other general tips:

- Instead of feeding the 2D points and the timesteps as inputs to the neural network directly, it is advisable to feed in *positional embeddings* of these inputs. High-frequency sinusoidal embeddings perform well. A python class extending torch.nn.Module for sinusoidal embeddings is provided in the *Stuff_for_DiffusionModelsProject* folder on Canvas. You may use it as reference as it might or might not match your required format exactly.
- An MLP (multi-layer perceptron) of 3-4 layers should be able to accomplish this task, trained for enough steps. Training such an MLP with Adam optimizer and learning rate in the range of 10^{-3} to 10^{-4} has been found to be effective.

What to include in the report

- An introduction section describing the basics of diffusion probabilistic models, the underlying distributions, and the applications of such methods. This section should also explain the details of how the denoising task can be accomplished by a noise-predicting neural network. (About 1-2 pages)

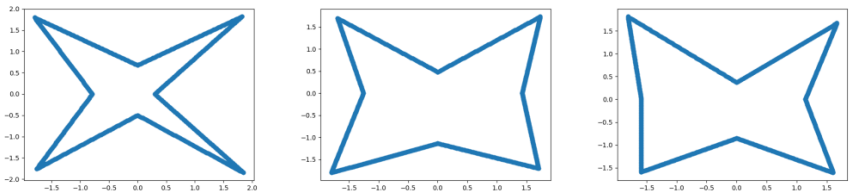

Figure 7: Examples of data samples from the *stars* dataset.

- A method section with 2 subsections for the 2 mandatory tasks.
    1. In the first subsection, explain in your own words how the reverse process can be computed in closed form in the given setting.
    2. In the second subsection, explain in detail the steps involved in training your neural network and list the hyper-parameters used. Also, explain the steps involved in generating data at inference time.
- An experimental results section, again containing two subsections.
    1. For the first task, include the plots for forward, and backward trajectories and also the resulting distributions (of $\mathbf{x_T}$ in the forward process and $\mathbf{x_0}$ in the reverse process).
    2. For the second task, include scatter plots of the resulting 2D points from your inference process (reverse sampling from 2D Gaussian noise). Show results obtained at different stages of the training process (eg. after every 1k training steps). Also, include analysis over any other hyper-parameter whose impact you found interesting.

4.2 Advanced part

In the advanced part, you attempt to generate (somewhat) new data. Like 4.1.2, the data is still in the form of 2D points, however, instead of learning to generate points in a single pattern, you will try to generate different realizations of a shape template (examples shown in figure 7). You are provided 1024 data samples with 1024 points each. You can devise your own strategy for this or you can have roughly the same approach as before except the following major change.

1. Instead of your network producing noise for each point independently, the network would take in information about all points in the batch to estimate the noise for the batch.
   Tip : Since the output noise should permute similarly upon the input points being permuted in a certain way, the additional information from other points should be something that is permutation invariant - like the average position of the points.

Upon successful implementation, you should be able to generate samples which resemble the input data (perhaps noisy versions of them).

4.3 Theoretical Part

Answer the following questions in your report. Make sure to use your own words and justify your answers.

(a) Evaluating a generative model. Generative models should produce something meaningful - which resembles the training data, however the generated samples should also be diverse in nature. Consider a specific application of 3D shape generation, where you'd like to generate 3D points in shapes of chairs. Suppose you train a Diffusion model with a training data of 3D points sampled on 500 different chairs. You have 100 chairs kept as test data. Suggest ways of evaluating for your model (evaluation metrics along with how you would use them) that capture both the aspects - the reconstruction quality and the diversity.

(b) Generative models. (i) In your own words, explain why we optimize the Empirical Lower BOund (ELBO) rather than $\log(p(\mathbf{x}))$ when training a VAE. (ii) What are the major differences between VAEs and Diffusion probabilitic models? Answer in terms of the encoding/decoding processes in the two methods, and the latent variables.

(c) Triangulation. (i) Triangulation is used in order to create 3D points in Structure-from-Motion. Given a set of 2D pixel positions $\mathbf{x}_i$, $i = 1, ..., k$, in k images and the projection matrices $\mathtt{P}_i$ of these images, explain how to obtain the 3D point that best represents these 2D measurements.
(ii) Consider the setup with $k = 2$ images with known relative pose. Please identify in which case there are several 3D points that project exactly to $\mathbf{x}_1$ in the first image plane and to $\mathbf{x}_2$ in the second. Is it possible to derive the position of $\mathbf{x}_1$ and $\mathbf{x}_2$? Motivate your answer!

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020. 12