University of Michigan
EECS 504: Foundations of Computer Vision
Fall 2022. Instructor: Andrew Owens.

Problem Set 10: Diffusion Model and Epipolar Geometry

Posted: Wednesday, Nov 30, 2022	**Due:** Thursday, Dec 8, 2022

Please convert your Colab notebook to a PDF file and submit the PDF file to Gradescope, *make sure to label your answers*. We have included the PDF conversion script at the end of the notebook. Nothing needs to be submitted to Canvas.

The starter code can be found at:
https://drive.google.com/file/d/1ANact7KpC2BJg_034Uw2gaiLV2Fbzl4v/view?usp=sharing

We recommend editing and running your code in Google Colab, although you are welcome to use your local machine instead.

This problem set is only required for EECS 504 students.

Problem 10.1 *Diffusion model (EECS 504 only)*

In this problem, you will be implementing some components of a denoising diffusion probabilistic model [1].

(a) **(3 points)** Implement the attention layer in the Attention U-Net backbone[2]. You will first implement single head self attention that takes a sequence x as input and outputs three matrices Q, K, V, standing for *query*, *key* and *values*. Each of the matrices is a linear transformation of x, mapping the input channel to d_k, d_k, and d_v, respectively. Attention is computed:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where d_k is a constant, corresponding to the last dimension (channel dimension) of K and Q. Suppose we have h heads, then the multi-head attention is the concatenation of h single head attention layers

$$\text{MultiHead}(Q, K, V) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right) W^O$$
$$\text{where head}_\text{i} = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

where $W_i^Q \in \mathbb{R}^{d_{in} \times d_k}, W_i^K \in \mathbb{R}^{d_{in} \times d_k}, W_i^V \in \mathbb{R}^{d_{in} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{in}}$. In this problem, we set both d_k and d_v to $\frac{d_{out}}{\text{h}}$. A final linear layer with weight W^O maps back to the input dimension.

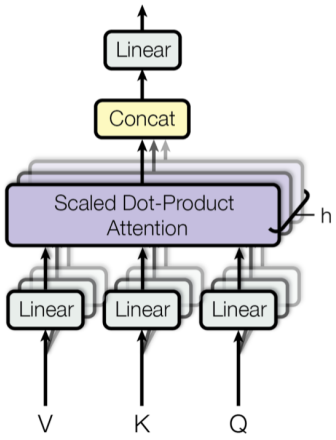

Figure 1: Overall multi-head attention layer.

(b) **(2 points)** Implement classifier-free guidance in the `sample` function [3]. Classifier-free guidance allows the diffusion model to generate class-conditioned samples without training an extra classifier. For each time step, the model generates both class-conditioned and and unconditioned velocity and compute with the following function to obtain the classifier-guided velocity.

$$\tilde{\boldsymbol{\epsilon}}_t = w\boldsymbol{\epsilon}_\theta\left(\mathbf{z}_t, \mathbf{c}\right) - (w-1)\boldsymbol{\epsilon}_\theta\left(\mathbf{z}_t\right)$$

In practice, you can first make a copy of z and t, and use -1 to indicate the class of unconditioned sample. Then stack the three things separately on the batch dimension to compute two velocity in parallel.

You can refer to Classifier-Free Diffusion Guidance for detailed math foundation of this part.

(c) **(1 point)** Run the training function and train for 20 epochs. It should produce image grids. The images in each row should be samples from the same class.

Problem 10.2 *Epipolar geometry (EECS 504 only)*

In this problem, you will plot *epipolar lines*. Given an input pixel in a *reference view*, we will plot the line containing all of its possible correspondences in a second view.

The reference view will be located at the origin, with an identity rotation matrix:

$$P_1 = K[I \mid 0], \tag{1}$$

while the second view's projection matrix is given by:

$$P_2 = K[R \mid t]. \tag{2}$$

Both cameras have the same intrinsics matrix K. For a review of epipolar geometry, please see this book chapter from Hartley and Zisserman.

(a) **(3 points)** Fill in the `coordinate_transform` function. First, obtain the ray $\mathbf{v} = K^{-1}p$ that connects the reference camera's center of projection to pixel p on the image plane (using homogeneous coordinates for p). We will walk along the ray at various distances $d_1, d_2, ...d_N$. For each one, create a 3D point $\mathbf{X}_i = d_i\mathbf{v}$. Project it into the second view. The provided code will then draw a dot at this position. After processing all N depth values, you should see a line. The correspondence to p should lie somewhere on this line.

(b) **(0 points)** Fill in the `compute_fundamental_matrix` function. We will estimate the fundamental matrix $F \in \mathbb{R}^{3\times3}$ from camera pose. Note that it is also possible to directly estimate F from correspondences, without the full camera pose. First compute the epipole:

$$\mathbf{e} = KR^T\mathbf{t}. \tag{3}$$

We will then compute F between another image frame with respect to the reference frame.

$$F = K^{-T}RK^T[e]_\times, \tag{4}$$

where $[e]_\times$ is the cross product of the epipole.

(c) **(0 point)** Fill in the `visualize_epipolar_line` function. For any pixel $\mathbf{p}$ in the reference frame, we can compute the line $\mathbf{u} = F\mathbf{p}$. The pixel in the other image that corresponds to $\mathbf{p}$ falls along $\mathbf{u}$. The vector $\mathbf{u} = [a, b, c]^T$ represents a line in the form $ax + by + c = 0$.

Acknowledgements. The diffusion problem was based on a problem set by L335 in Cambridge University, which is based on code from Katherine Crowson https://github.com/crowsonkb/v-diffusion-pytorch.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[3] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.