10907 Pattern Recognition

Lecturers
Prof. Dr. Ivan Dokmanić ⟨ivan.dokmanic@unibas.ch⟩

Tutors
Felicitas Haag ⟨felicitas.haag@unibas.ch⟩
Alexandra Spitzer ⟨alexandra.spitzer@unibas.ch⟩
Cheng Shi ⟨cheng.shi@unibas.ch⟩
Vinith Kishore ⟨vinith.kishore@unibas.ch⟩

Problem set 6

Math

Exercise 1 (Patchwise MLPs as convolutional nets ⋆⋆)**.**

Let $x \in \mathbb{R}^{H\times W\times 1}$ be a single-channel image and let k be a patch size (so we work with square patches of size $k \times k$). We will define an image-to-image architecture which uses an MLP instead of a convnet. It will simply apply the same MLP to all overlapping paches. For each *valid* location (i, j) (so $0 \le i \le H - k$, $0 \le j \le W - k$), define the patch

$$P_{ij} := x[i : i + k - 1,\ j : j + k - 1] \in \mathbb{R}^{k\times k}, \qquad p_{ij} := \mathrm{vec}(P_{ij}) \in \mathbb{R}^{k^2}.$$

Consider an MLP applied with *shared weights* to every patch:

$$\begin{aligned} z_{ij}^{(1)} &= W_1 p_{ij} + b_1, \qquad W_1 \in \mathbb{R}^{D\times k^2},\ \ b_1 \in \mathbb{R}^D, \\ a_{ij} &= \sigma\left(z_{ij}^{(1)}\right), \\ y_{ij} &= W_2 a_{ij} + b_2, \qquad W_2 \in \mathbb{R}^{C_{\mathrm{out}}\times D},\ \ b_2 \in \mathbb{R}^{C_{\mathrm{out}}}. \end{aligned}$$

where σ is a pointwise activation function.

Show that

1. *The first layer of the MLP can be implemented as a convolution:* Concretely: each row of the matrix W_1 is a (vectorized) filter impulse response, and the D rows constitute the D channels at the output of this layer.
2. *The second layer can be implemented as a* 1×1 *convolution.*

Now consider an MLP with L layers.

$$z_{ij}^{(1)} = W_1 p_{ij} + b_1, \qquad z_{ij}^{(\ell+1)} = W_{\ell+1}\sigma(z_{ij}^{(\ell)}) + b_{\ell+1},\ \ \ell = 1, \dots, L-1.$$

Argue that the network is equivalent to: one $k \times k$ convolution for W_1, followed by $L - 1$ many 1×1 convolutions for $W_2, \dots, W_L$, with σ between them. In other words, this MLP-based architecture is a convnet in disguise!

Exercise 2 (Closed-form forward noising process ⋆)**.**

Recall the forward diffusion process in $1D$,

$$x_t = \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, 1)$$

with $0 < \alpha_t < 1$ (NB: close to 1), and all ϵ_t independent of each other and of x_0.

1. Show that, conditional on x_{t-1}, x_t is normally distributed. Concretely,

$$x_t | x_{t-1} \sim \mathcal{N}(\sqrt{\alpha_t} x_{t-1}, 1 - \alpha_t).$$

2. Let $\overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Prove by induction that

$$x_t | x_0 \sim \mathcal{N}(\sqrt{\overline{\alpha}_t} x_0, 1 - \overline{\alpha}_T)$$

Exercise 3 (What is normal and what is not ⋆⋆)**.**

We again work with the 1D diffusion process from the last exercise, and we want to show that the transition probabilities for the reverse process are Gaussian *IF* we know and condition on x_0 (which in practice we don't—we want to sample it).

1. Using the result of the previous exercise for $t-1$ and t, write both x_{t-1} and x_t as affine functions of x_0 and independent standard normals. That is to say, show that there are some numbers a, b, c, d, e such that
$$x_{t-1} = a\,x_0 + b\,z_1, \qquad x_t = c\,x_0 + d\,z_1 + e\,z_2,$$
with independent $z_1, z_2 \sim \mathcal{N}(0,1)$.
2. Use this representation to compute the following quantities conditional on x_0:
$$\mathbb{E}[x_{t-1} \mid x_0], \quad \mathbb{E}[x_t \mid x_0], \quad \mathrm{Var}(x_{t-1} \mid x_0), \quad \mathrm{Var}(x_t \mid x_0), \quad \mathrm{Cov}(x_{t-1}, x_t \mid x_0).$$
3. Recall that if $\begin{pmatrix} u \\ v \end{pmatrix}$ is jointly Gaussian, then
$$U \mid V = v \sim \mathcal{N}\Big(\mathbb{E}[U] + \frac{\mathrm{Cov}(U,V)}{\mathrm{Var}(V)}\big(v - \mathbb{E}[V]\big),\ \mathrm{Var}(U) - \frac{\mathrm{Cov}(U,V)^2}{\mathrm{Var}(V)}\Big).$$
Use this with $U = x_{t-1}$, $V = x_t$ to show that
$$x_{t-1} \mid (x_t, x_0)$$
is Gaussian with the mean
$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{1-\overline{\alpha}_t}\left(x_t - \sqrt{\overline{\alpha}_t}\,x_0\right)\right).$$

Exercise 4 (Predicting x_{t-1} vs predicting noise ε)**.**

Consider a single step of the 1D forward process
$$x_t = \sqrt{\alpha_t}\,x_{t-1} + \sqrt{\beta_t}\,\varepsilon_t, \qquad \beta_t = 1 - \alpha_t, \quad \varepsilon_t \sim \mathcal{N}(0,1).$$
We compare two possible training objectives:

1. directly regress x_{t-1} from x_t,
2. predict the noise ε_t from x_t and estimate x_{t-1} as (x_t − estimated noise)

Assume we parameterize the reverse mean via a noise-predictor network $\varepsilon_\theta(x_t, t)$ as
$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \sqrt{\beta_t}\,\varepsilon_\theta(x_t, t)\right).$$

1. Express the true x_{t-1} in the same form:
$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \sqrt{\beta_t}\,\varepsilon_t\right).$$
2. Show that
$$x_{t-1} - \mu_\theta(x_t, t) = \frac{\sqrt{\beta_t}}{\sqrt{\alpha_t}}\big(\varepsilon_\theta(x_t, t) - \varepsilon_t\big).$$
3. Deduce that the two MSE losses
$$L_x(\theta) = \mathbb{E}\big[(x_{t-1} - \mu_\theta(x_t, t))^2\big], \qquad L_\varepsilon(\theta) = \mathbb{E}\big[(\varepsilon_t - \varepsilon_\theta(x_t, t))^2\big]$$
are proportional:
$$L_x(\theta) = \frac{\beta_t}{\alpha_t}\,L_\varepsilon(\theta),$$
where the factor β_t/α_t does not depend on θ. Conclude that minimizing MSE on x_{t-1} is equivalent to minimizing MSE on ε_t in the sense that they will give the same θ and ultimately the same estimate of x_{t-1}.

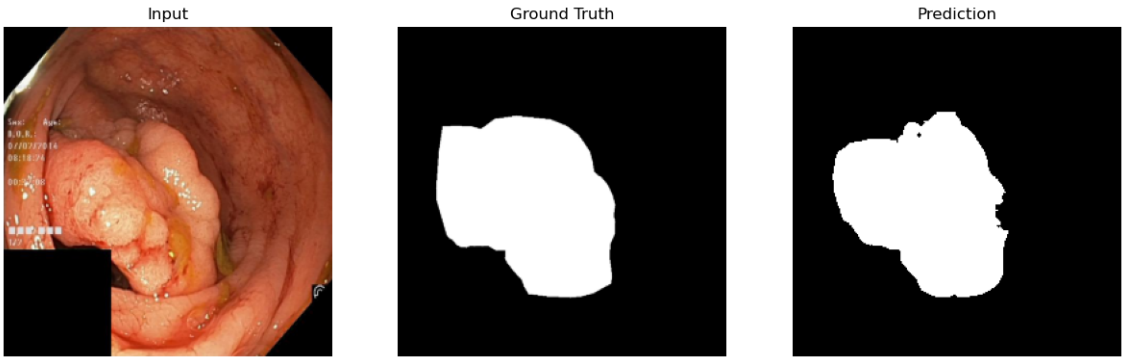

Figure 1: Segmentation and prediction for a sample image in the dataset.

Coding

For this problem set, we do not provide Gradescope autograding, because the expected outputs are plots and other visualizations, which are difficult to assess automatically. Instead, we will include reference plots in the model solution.

Exercise 5 (Segmentation — ⋆⋆)**.**

Pixel-wise image segmentation is an important computer vision task that involves dividing an image into different segments representing various objects or regions of interest. Such a method can aid in medical diagnostic scenarios where experts can sift through large amounts of data quickly or assist in the automatic diagnosis of several conditions. In this exercise, we will focus on segmenting polyps used in gastrointestinal disease detection. We will use a popular dataset, Kvasir SEG (`https://datasets.simula.no/kvasir-seg/`) [1] to train and test segmentation networks. The data is already contained in the `python_scripts.zip file`, so you do not need to manually download it. The segmentation network $f_\theta(\cdot)$ simply takes an image x as input and predicts the segmentation map y. To train such types of networks, we need to ensure that the predicted output $f_\theta(x)$ is as close to the true output y as possible. Closeness can be defined in several ways, and one such method is Binary Cross Entropy (BCE) loss, commonly used in binary classification problems. Binary cross-entropy loss is defined as

$$\text{BCE}(y,p) = -\frac{1}{N^2}\sum_{i,j}\Big(y[i,j]\log(p[i,j]) + (1-y[i,j])\log(1-p[i,j])\Big).$$

where $p = \sigma(f_\theta(x))$ and σ denotes the sigmoid function. In the notebook, apply a sigmoid to the network output before computing BCE, Dice loss, and IoU. Additionally, several metrics have been developed that are effective for segmentation problems, with one of the recent popular ones being the Dice loss [2], which is defined as

$$DiceLoss(y,p) = 1 - \frac{2\sum_{i,j} y[i,j]p[i,j]}{\sum_{i,j}(y[i,j]+p[i,j])+\epsilon},\ \epsilon > 0$$

To train the network in the assignment the two losses are combined as follows:

$$\min_\theta \mathbb{E}_{y,x\sim p_{x,y}}\Big(BCE(y,\sigma(f_\theta(x))) + \lambda_{DiceLoss} DiceLoss(y,\sigma(f_\theta(x))).\Big)$$

where $\lambda_{DiceLoss} \geq 0$ is a user-defined parameter. A sample image and its segmentation are given in Figure 1. The quality of segmentation is typically evaluated using the Intersection over Union (IoU) metric. We have provided the function to compute this metric.

In this exercise, you will test two types of networks: U-Net [3], a popular network that has been successfully used for several biomedical applications, and a simple Vision Transformer (ViT)

[4], which employs a transformer-like architecture for computer vision tasks. We have provided you with a simple ViT architecture in the file `./models/vit.py` and the U-Net architecture in the file `./models/unet.py`. We have also provided you with a notebook `segment.ipynb` where the necessary libraries and data are loaded. Your task in this assignment is:

1. Split the data into train, validation, and test sets, using 20% for validation and 20% for testing. The validation set is used to tune the hyperparameters of the network, and the test set is used to report the results.

2. Choose an appropriate batch size and initialize the data loader for each set. Hint: use `TensorDataset`.

3. Define the U-Net model (`UNet(3, 1).to(device)`), the loss, and the optimizer. We have provided you with the following parameters; you can choose to vary them:

   (a) `EPOCHS`: number of epochs trained ,

   (b) `LR`: learning rate,

   (c) `LAMBDA_DICE`: the weight $\lambda_{DiceLoss}$ in the loss function .

   Use `torch.nn.BCELoss()` for BCE loss and `smp.losses.DiceLoss('binary')` for dice loss. You can use the Adam optimizer or experiment with other optimizers. Note that the loss function is defined for a single image; however, you can use it for multiple images at once.

4. Complete the training loop of the code. Evaluate the network's performance using the validation set in terms of the IoU metric using the provided function. You can choose how often you want to evaluate your network on the validation set. Plot the loss curves.

5. Evaluate the model on the test set and report the IoU metric. Plot a few examples from your test set and the corresponding predictions. Note that the plots should show results from several cases, including examples where it worked well, cases where it missed the segmentation completely, and cases where the network partially recovered the segmentations.

6. Report on the hyperparameters used and observations on the test set in the cell `Observations and Results`.

7. Report on the possible reasons for using binary cross-entropy loss for training the model.

8. Repeat the above steps using the vision transformer. We have provided you with a simple vision transformer model in the class `SegmentViT`. Complete the steps again and note down your observations and results in the cell `ViT Observations and Results` and compare the results with those of U-Net.

References

[1] Debesh Jha, Pia H. Smedsrud, Michael A. Riegler, Pål Halvorsen, Thomas de Lange, Dag Johansen, and Håvard D. Johansen, *Kvasir-seg: A segmented polyp dataset*, in *MultiMedia Modeling: 26th International Conference, MMM 2020, Daejeon, South Korea, January 5–8, 2020, Proceedings, Part II 26*, pages=451–462, year=2020, organization=Springer.

[2] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, *Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations*, in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*, pp. 240–248, 2017, Springer.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, *U-net: Convolutional networks for biomedical image segmentation*, in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages=234–241, year=2015, organization=Springer

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and others, *An image is worth 16x16 words: Transformers for image recognition at scale*, *arXiv preprint arXiv:2010.11929*, year=2020.