NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

AI6126 Advanced Computer Vision

Homework Assignment 2

School of Computer Science and Engineering

Programme: MSAI

Date: 14 April 2024

Authored By: Tan Jie Heng Alfred (G2304193L)

Question 1

The receptive field of an element f in layer l is defined to be the size of the region in the input that produces the feature. Suppose we have a total of L layers, and r_l refers to the receptive field size of the output feature map with respect to feature maps f_l, we get can find r_{l-1} with the following recurrence relation[1],

$$r_{l-1} = s_l \times r_l + (k_l - s_l),$$

where k_l refers to the convolution kernel size at layer l and s_l refers to the stride of the kernel. Here, we are interested in finding r_0, and we have an equivalent closed form expression for it,

$$r_0 = \sum_{l=1}^{L} \left((k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1$$

To account for the dilation, d_l, of the kernel in layer l, we can replace k_l with $k'_l = d_l(k_l - 1) + 1$, giving

$$\begin{aligned} r_0 &= \sum_{l=1}^{L} \left((k'_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1 \\ &= \sum_{l=1}^{L} \left((d_l k_l - d_l) \prod_{i=1}^{l-1} s_i \right) + 1 \end{aligned}$$

Finally, to find the receptive for each of the hidden layer, H_i, $i \in \{1, 2, 3\}$, we only have to vary $L = 1, 2, 3$. We tabulate the results in the table below.

Hidden layer, H_i	**Receptive field, r_0**
H_1	$\sum_{l=1}^{1} \left((d_l k_l - d_l) \prod_{i=1}^{l-1} s_i \right) + 1$ $= (d_1 k_1 - d_1) + 1$ $= 5 - 1 + 1$ $= 5$
H_2	$\sum_{l=1}^{2} \left((d_l k_l - d_l) \prod_{i=1}^{l-1} s_i \right) + 1$ $= 5 + (d_2 k_2 - d_2) s_1$ $= 5 + (3 - 1) \times 2$ $= 9$
H_3	$\sum_{l=1}^{3} \left((d_l k_l - d_l) \prod_{i=1}^{l-1} s_i \right) + 1$ $= 9 + (d_3 k_3 - d_3) s_1 s_2$ $= 9 + (10 - 2)(2)$ $= 25$

[1] https://distill.pub/2019/computing-receptive-fields/

Question 2

(i) In the Fully Convolutional Network (FCN) architecture, there are several layers of feature downsampling and upsampling. The downsampling of features happens in the shallower part of the network, while the upsampling happens in the deeper part of the network. Downsampling reduces the spatial resolution of the features through operations like pooling and strided convolutions. This reduces the computational cost, as the network can process information more efficiently, while also capturing broader contextual information as the deeper features have a wider receptive field of the input image. However, since we want to perform segmentation, we require the output of FCN to be the same spatial resolution as the input image. As such, we have to perform upsampling to recover the spatial resolution, by using processes such as interpolation and transposed convolution.

(ii) We note that the formula in Question 1 does not take into account padding from Conv2d. An alternative approach is to use the equation that calculates the resolution of an output after convolution. Let n_l be the size of the output feature map from the l^{th} layer, and further suppose that the output feature maps of layer l will be the input feature maps of layer $l+1$. We further define n_0 to be the size of the input image. Then, after performing a convolution at layer l, using kernel of size k_l, stride s_l, padding p_l and dilation d_l, on the input feature of size n_{l-1}, we get the size of the output feature map as,

$$n_l = \frac{n_{l-1} + 2p_l - d_l(k_l - 1) - 1}{s_l} + 1.$$

Since we want to find the receptive field of the network (i.e., the last layer), we will apply this equation recursively, each time solving for n_{in}. Then, we first set $n_3 = 1$ and solve for n_2 as such,

$$n_3 = 1 = \frac{n_2 + 2(0) - 2(3-1) - 1}{1} + 1$$
$$\Rightarrow n_2 = 2(2) + 1 = 5$$

This means that, to obtain an output of 1×1 in the final layer, the second layer must output feature maps of size 5×5. Now, we find n_1, with $n_2 = 5$ as,

$$n_2 = 5 = \frac{n_1 + 2(2) - 1(5-1) - 1}{2} + 1$$
$$\Rightarrow n_1 = 4 \times 2 - 4 + 4 + 1 = 9$$

Finally, we find n_0 with n_1 as,

$$n_1 = 9 = \frac{n_0 + 2(0) - 1(3-1) - 1}{1} + 1$$
$$\Rightarrow n_0 = 8 + 2 + 1 = 11$$

This means that, to get an output size of 1×1 at the third layer, we require an input image size of 11×11. In other words, the receptive field of the third layer is 11×11.

(iii) For a given number of parameters, dilated convolution offers a larger effective receptive field than a standard convolution. This in turns allow the model to capture more spatial context, which is important in semantic segmentation tasks as they often require the capturing of relationships between distant parts of the image.

(iv)
$$\begin{bmatrix} 1 & 2+2 & 3+4 & 6 \\ 2 & 3+4+1 & 4+6+2 & 8+3 \\ 3 & 4+6+2 & 5+8+3 & 10+4 \\ 0 & 0+3 & 0+4 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 7 & 6 \\ 2 & 8 & 12 & 11 \\ 3 & 12 & 16 & 14 \\ 0 & 3 & 4 & 5 \end{bmatrix}$$

Question 3

The KL divergence between two distributions P and Q measures how different the two distributions are. Mathematically, the

$$D_{KL}(P,Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$
$$= \mathbb{E}_P\left[\log\frac{p(x)}{q(x)}\right] (1)$$

where $p(x)$ and $q(x)$ are the probability densities of P and Q respectively.
In a Variational Autoencoder, we have the objective of maximizing $p_\theta(x)$ (equivalently, $\log p_\theta(x)$), as

$$\mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x)] = \log p_\theta(x)$$
$$= \underbrace{D_{KL}\left(q_\phi(z|x), p_\theta(z|x)\right)}_{\geq 0} - D_{KL}\left(q_\phi(z|x), p(z)\right) + \mathbb{E}_{z\sim q_\phi(z|x)}(\log p_\theta(z|x))$$
$$\geq \mathbb{E}_{z\sim q_\phi(z|x)}(\log p_\theta(z|x)) - D_{KL}\left(q_\phi(z|x), p(z)\right),$$

where $q(z|x) = q_\phi(z|x) \approx p_\theta(z|x)$ is parametrized by ϕ, and $p_\theta(x)$ reconstructs x, parametrized by θ.
Therefore, the KL divergence of the univariate $q(z|x) = \mathcal{N}(\mu, \sigma^2)$ and $p(z) = \mathcal{N}(0,1)$ is,

$$-D_{KL}\big(q(z|x), p(z)\big) = -D_{KL}\left(q_\phi(z|x), p(z)\right)$$
$$= -\int q(z|x) \log\frac{q(z|x)}{p(z)} dz$$
$$= -\int q(z|x)\log q(z|x) - q(z|x)\log p(z)\, dz$$

Now, for a univariate Gaussian distribution,$Y\sim\mathcal{N}\left(\mu_y, \sigma_y^2\right)$, with probability density function $f(x)$, we have

$$\int f(x)\log f(x)\, dx = -\frac{1}{2}\big(1 + \log 2\pi\sigma_y^2\big),$$

since,

$$\int f(x)\log f(x)\, dx = \int f(x)\log\left(\frac{1}{\sigma_y\sqrt{2\pi}}\exp -\frac{1}{2}\left(\frac{x-\mu_y}{\sigma_y}\right)^2\right) dx$$
$$= \int f(x)\left[-\log\sigma_y\sqrt{2\pi} - \frac{1}{2}\times\frac{\big(x-\mu_y\big)^2}{\sigma_y^2}\right] dx$$
$$= -\log\sigma_y\sqrt{2\pi} - \frac{1}{2\sigma_y^2}\mathbb{E}_Y\big(x-\mu_y\big)^2$$
$$= -\frac{1}{2}\log 2\pi\sigma_y^2 - \frac{1}{2} = \frac{1}{2}\big(1+\log 2\pi\sigma_y^2\big)$$

Hence, we get,

$$-D_{KL}\big(q(z|x), p(z)\big) = -\int q(z|x)\log q(z|x) - q(z|x)\log p(z)\, dz$$
$$= \frac{1}{2}(1+\log 2\pi\sigma^2) + \int q(z|x)\log p(z)\, dz$$
$$= \frac{1}{2}(1+\log 2\pi\sigma^2) + \int \frac{1}{\sigma\sqrt{2\pi}}\exp -\frac{1}{2}\left(\frac{z-\mu}{\sigma}\right)^2\left[\log\left(\frac{1}{\sqrt{2\pi}}\exp -\frac{z^2}{2}\right)\right] dz$$
$$= \frac{1}{2}(1+\log 2\pi\sigma^2) + \int \frac{1}{\sigma\sqrt{2\pi}}\exp -\frac{1}{2}\left(\frac{z-\mu}{\sigma}\right)^2\left[-\frac{1}{2}\log 2\pi - \frac{z^2}{2}\right] dz$$

$$= \frac{1}{2}(1 + \log 2\pi\sigma^2) + \mathbb{E}_{q(z|x)}\left(-\frac{1}{2}\log 2\pi - \frac{z^2}{2}\right)$$
$$= \frac{1}{2}(1 + \log 2\pi\sigma^2) - \frac{1}{2}\log 2\pi - \frac{1}{2}\mathbb{E}_{q(z|x)}(z^2)$$
$$= \frac{1}{2}(1 + \log \sigma^2) - \frac{1}{2}\mathbb{E}_{q(z|x)}((z-\mu)^2 + 2\mu z - \mu^2)$$
$$= \frac{1}{2}(1 + \log \sigma^2) - \frac{1}{2}\left[\underbrace{\mathbb{E}_{q(z|x)}(z-\mu)^2}_{\text{Variance of } q(z|x)} + 2\mu \underbrace{\mathbb{E}_{q(z|x)}(z)}_{\text{Mean of } q(z|x)} - \mu^2\right]$$
$$= \frac{1}{2}(1 + \log \sigma^2) - \frac{1}{2}(\sigma^2 + 2\mu^2 - \mu^2) = \frac{1}{2}(1 - \sigma^2 - \mu^2 + \log \sigma^2)$$

Equivalently, we have $D_{KL}\big(q(z|x), p(x)\big) = \frac{1}{2}(\sigma^2 + \mu^2 - 1 - \log \sigma^2)$.

Question 4

Variational Autoencoder (VAE) and Vector Quantized Autoencoder (VQVAE) are models used for data generation. The key difference between the two models is that VAE uses a continuous latent representation, whereas VQVAE uses a discrete latent representation instead. VQVAE achieved this by mapping the continuous latent space into a set of predefined codebooks (i.e., discrete values). This helps avoid the posterior collapse issue, where the model's variational distribution (i.e., $q_\phi(z|x)$), which approximates $p_\theta(z|x)$, collapses towards the prior distribution $p(z)$.

Question 5

In the training of Generative Adversarial Network (GAN), one very common obstacle is the issue of mode collapse. This occurs when the generator starts to produce a limited set of outputs (i.e., hence becoming the modes), even though the distribution of the underlying data is much richer. For instance, if the GAN is trained on cat images, then a mode collapse could manifest in the generator producing only one species of cat, when the training data contains much more species of cats. This can occur when the discriminator becomes much better, relative to the generator's generation, at discriminating the 'real' and 'fake' images. Mathematically, the objective function of the minimax game of GAN is as follows:

$$\min_G \max_D \left(\mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{z\sim p(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right]\right)$$

If the discriminator D is too good, then the second summand $\mathbb{E}_{z\sim p(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right]$ will be large, as $D\big(G(z)\big) \approx 0$ for many $z\sim p(z)$. Since the objective of the generator G is to minimize the objective, it would then learn to generate only a small subset of $G(z)$ such that $D\big(G(z)\big) \not\approx 0$ is much larger. This results in the mode collapse.

To address this, we can adopt unrolled GANs, where we take k steps with the discriminator in each iteration, and backpropagate through all of them to update the generator. In other words, the generator "unrolls" a few steps of the discriminator's training process to predict its future behaviour with respect to its current trajectory. This discourages the generator from overfitting to a specific type of output, hence addressing the mode collapse issue.

Question 6

The use of autoencoders aims to compress the data, such that the reconstructed data has minimal information loss, which is implicitly defined (e.g., by $L2$ loss). Suppose we have a flattened data, $x \in \mathbb{R}^n$, and the autoencoder maps it to a latent variable $z_x \in \mathbb{R}^m$. If $n \leq m$, then the autoencoder could learn an (almost) 'identity' mapping, such that,

$$z_x = \begin{bmatrix} x \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where the first n dimensions are the entries of x and the remaining $(m - n)$ dimensions are 0. Here, there is no compression of data and, instead, have additional redundancy.
On the flipside, by enforcing that the latent representations have a lower dimension, we ensure that the model learns to remove redundancies and irrelevant information while compressing the data. Furthermore, it can learn a better, more efficient and abstract representation of our data, which could be used for better performance on other downstream tasks.