COMP4432 Machine Learning

Assignment #2

Instructions:
- Answer all 3 questions.
- Interpret the questions logically, show your steps and **write down your assumption(s) when necessary**.
- Please submit your answer to L@PU before the due date.
- Late Submission Policy
  - 3-hour “grace period” is given.
  - 10% off for every 3-hour late
- Plagiarism Policy
  - Both giver and receiver subject to the same penalty below
  - All the students involved not only will receive 0 marks for this assessment, but also will have an additional 50% penalty applied, e.g., 5 marks for a 10-mark assessment.

1. In our lecture notes, we have already seen the derivative of a sigmoid function wrt its weight parameters as follows.

$$f(x) = \frac{1}{1 + e^{-(wx+b)}}$$

we have

$$\frac{\partial}{\partial w}\left(\frac{1}{1+e^{-(wx+b)}}\right)$$

$$= \frac{-1}{(1+e^{-(wx+b)})^2}\frac{\partial}{\partial w}(e^{-(wx+b)}))$$

$$= \frac{-1}{(1+e^{-(wx+b)})^2} * (e^{-(wx+b)})\frac{\partial}{\partial w}(-(wx+b)))$$

$$= \frac{-1}{(1+e^{-(wx+b)})} * \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})} * (-x)$$

$$= \frac{1}{(1+e^{-(wx+b)})} * \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})} * (x)$$

$$= f(x) * (1 - f(x)) * x$$

Let us now consider the softmax function with the function form

$$f_{sm}(z_j) = \frac{e^{z_j}}{\sum_k e^{z_k}}.$$

Derive the derivative of softmax wrt z_j. You MUST use the symbols above, i.e., z_j, $f_{sm}(z_j)$, etc., to present your answer.

Hint:

Using the quotient rule and let $g(z_j) = e^{z_j}$ and $h(z_j) = \sum_k e^{z_k}$, we have

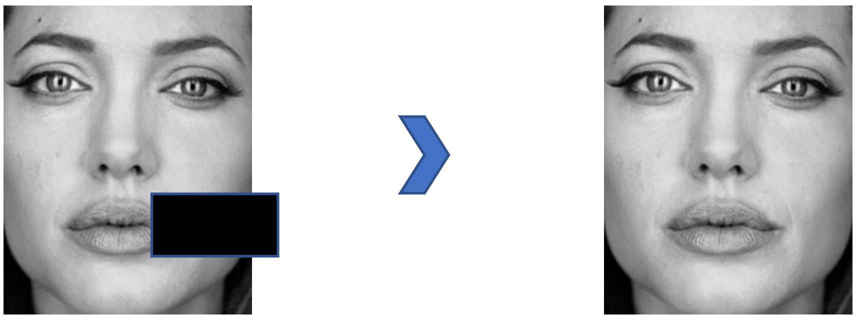
$$\frac{\partial f_{sm}(z_j)}{\partial z_l} = \frac{g\prime(z_j)h(z_j) - g(z_j)h\prime(z_j)}{[h(z_j)]^2}.$$

$$f_{sm}(z_j) = e^{z_j} / \sum_k e^{z_k}$$

$$\frac{\partial f}{\partial z_j} = \left[\left(\sum_k e^{z_k} \right) \cdot e^{z_j} - e^{z_j} \cdot \left(\sum_k e^{z_k} \right)' \right] / \left(\sum_k e^{z_k} \right)^2$$

$$= f_{sm}(z_j) - f_{sm}(z_j)^2$$

2. Image Inpainting refers to the task of rebuilding missing or damaged patches of an image. Given grayscale images of fixed size 300(h)x200(w) pixels with a rectangular patch of pixels of size 60(h)x100(w) pixels missing as exemplified on the left photo below, we want to rebuild the missing pixels like the right photo on the right. Recall that grayscale images consist of 1 pixel channel with pixel value from 0 to 255 (represented by 1 byte).



a) Suppose you are asked to use a multilayer perceptron (MLP) neural network as shown in Fig.1 to carry out the inpainting task, i.e., predicting the missing image patch.

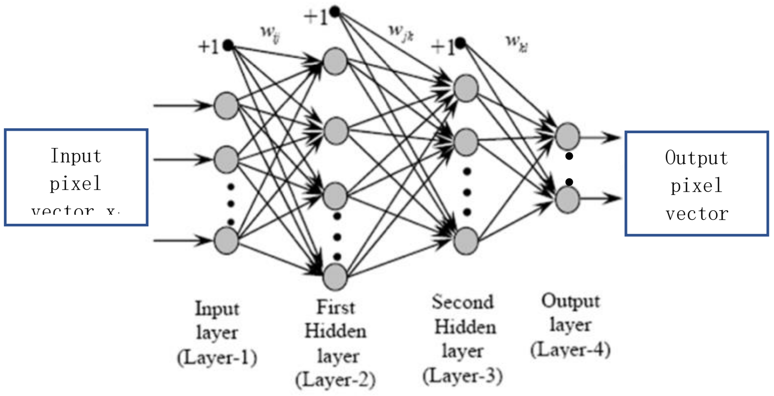


Fig. 1

(i) Describe how you formulate the problem by describing what should be the input (pixel vector x_i) and what should be the output (pixel vector sx_i).

We can adopt the similar configuration proposed in [1] and [2].

- The input pixel vector is of size 300 * 200 * 2. Here it has two layers. The first layer is the original picture with missing parts. The second layer is a binary mask layer which indicates which pixel in the original picture is missing. The necessarily of the second layer is that we should explicitly tell the model which part it needs to complete.
- The output vector is of size 300 * 200.

(ii) How training data should be **prepared** to train the MLP based image inpainting model?

Before training, we need to first collect human face images. After that, we need to preprocess it by randomly erase some part of the image. Meanwhile, we need to prepare a binary mask vector to store where the missing part is. We then pair each incomplete image with its binary mask vector as a data point. If needed, we can also do some feature extraction (using autoencoder etc.) to process the data.

b) Suppose now the MLP in part (a) is enhanced with convolutional layers and pooling layers so that a CNN is resulted to carry out greyscale image inpainting. For the following table of architecture, how many learnable parameters are there in each of the specified layers? Show the formula or calculations in your answers. Also, there exist TWO unknown parameters in the specification column, namely, U_1 and U_2, write down their correct values.

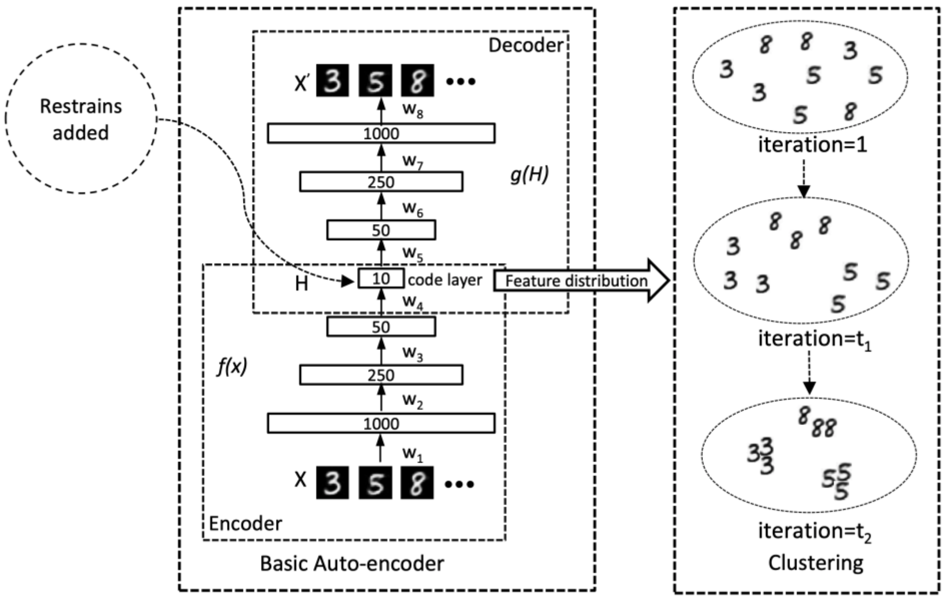
Layer in CNN	Specification	Number of learnable parameters (**formula answer is acceptable**)	Memory size
Input Layer	Your solution in part (a) 300 * 200 * 2	0	300 * 200 * 2
1st Convolutional Layer	64 3x3xU_1(2) filters; stride=1; no zero padding	(3 * 3 * 2) * 64	298 * 198 * 64
1st Max Pooling Layer	2x2 window; stride=2	0	149 * 99 * 64
2nd Convolutional Layer	256 3x3xU_2(64) filters; stride=1; no zero padding	(3 * 3 * 64) * 256	147 * 97 * 256
2nd Max Pooling Layer	2x2 window; stride=2	0	73 * 48 * 256
Input layer of fully connected (fc) feedforward network	Just the flattened output from previous layer	0	73 * 48 * 256
1st hidden layer of fc feedforward network	N_{L2} hidden neurons	(73 * 48 * 256 + 1) * N_{L2}	N_{L2}

2nd hidden layer of fc feedforward network	N_{L3} hidden neurons	$(N_{L2}+1) * N_{L3}$	N_{L3}
Output layer	Your solution in part (a)	$(N_{L3}+1) * 300 * 200$	

3. Autoencoder is a widely used unsupervised deep learning model. Describe how it can be used in the following applications.

   a) k-means clustering of face images
   Given a collection of human face images, it is aimed to cluster them into some natural groups of similar faces.

   There are several ways to use autoencoder to enhance k-means. One of the classical methods is proposed in [3]. This method achieved 90 percent accuracy in YaleB dataset, which is for face images clustering. It first uses autoencoder to learn the representation of the images. The learned representation is normally with lower dimensionality. Then, the extracted features will be used for k-means clustering. The overall framework is shown below. As shown in this Figure, the hidden layer is used as the feature distribution for clustering. Although the figure uses MNIST as an example, it is the same idea with using face images.



   Notice that here it only applies a basic autoencoder. More specifically, the encoder maps an input x_i to its hidden representation h_i. The mapping function is usually non-linear and the following is a common form:

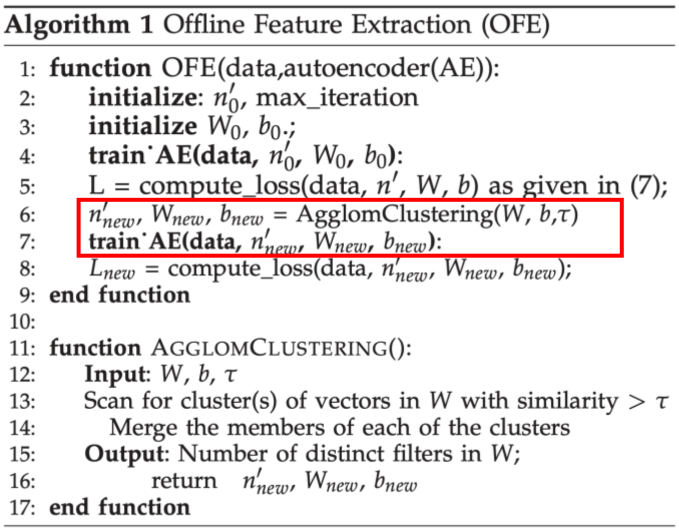
$$h_i = f(x_i) = \frac{1}{1 + \exp(-(W_1 x_i + b_1))},$$

   where W_1 is the encoding weight, b1 is the corresponding bias vector.

b) Hierarchical clustering of animal images
Given a collection of animal images, it is aimed to build a hierarchy of clusters of similar animal types.

Autoencoder can also be applied with hierarchical clustering methods to reduce redundance of the filters. Many works show that a good number of encoding and decoding filters of layered autoencoders are duplicative thereby enforcing two or more processing filters to extract the same features due to filtering redundancy. [4] proposed a new way to solve this problem, ***named aggo-SAE.*** The concept is illustrated with **Sparse Autoenconders (SAE)** using MNIST and NORB datasets. With similar idea, it can also be applied to animal images.

The following algorithm is the core methodology of this paper. It employs one of the most widely used hierarchical clustering method: **Agglomerative Clustering**. Notice in the red box, the clustering results are sent back to the autoencoder. Using this method, the SAE can represent the data with fewer filters and the learned representation will be used for the clustering job.



Algorithm 1 Offline Feature Extraction (OFE)

```
function OFE(data,autoencoder(AE)):
    initialize: $n'_0$, max_iteration
    initialize $W_0$, $b_0$.;
    train˙AE(data, $n'_0$, $W_0$, $b_0$):
    L = compute_loss(data, $n'$, $W$, $b$) as given in (7);
    $n'_{new}$, $W_{new}$, $b_{new}$ = AgglomClustering($W$, $b$,$\tau$)
    train˙AE(data, $n'_{new}$, $W_{new}$, $b_{new}$):
    $L_{new}$ = compute_loss(data, $n'_{new}$, $W_{new}$, $b_{new}$);
end function

function AGGLOMCLUSTERING():
    Input: $W$, $b$, $\tau$
    Scan for cluster(s) of vectors in $W$ with similarity $> \tau$
        Merge the members of each of the clusters
    Output: Number of distinct filters in $W$;
            return  $n'_{new}$, $W_{new}$, $b_{new}$
end function
```

For these two applications, state clearly the input-to-hidden and hidden-to-output arrangements of the autoencoder model used (e.g. convolutional autoencoder, denoising autoencoder, etc.), and how the representation learnt by autoencoder is used to carry out the clustering tasks.

References:

[1] Yu J, Lin Z, Yang J, et al. Generative image inpainting with contextual attention[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 5505-5514.

[2] Iizuka S, Simo-Serra E, Ishikawa H. Globally and locally consistent image completion[J]. ACM Transactions on Graphics (ToG), 2017, 36(4): 1-14.

[3] Song C, Liu F, Huang Y, et al. Auto-encoder based data clustering[C]//Iberoamerican congress on pattern recognition. Springer, Berlin, Heidelberg, 2013: 117-124.

[4] Ayinde B O, Zurada J M. Clustering of receptive fields in autoencoders[C]//2016 International Joint Conference on Neural Networks (IJCNN). IEEE, 2016: 1310-1317.