Computer Vision 1: Assignment 5
(Due date: 24.01.2022)

Submission instructions:

- For each programming task, submit a `.py` source code file or a Jupyter/Colab notebook file. Do not include any images or data files you used.
- For each pen & paper task, submit a `.pdf` file. Type your solution in LaTeX, Word, or another text editor of your choice and convert it to PDF. Do not submit photographs or scans of handwritten solutions!
- In all submissions, include at the top names of all students in the group.
- Choose one person in your group that submits the solution for your entire group.

Important note! Task 3 relates to the last lecture, published only some days before the deadline. You may therefore treat this task as **optional**, and solve it if you wish. The requirement of solving assignments is therefore also reduced from 12 tasks to 11 tasks. Solving Task 3 of this assignment sheet still gives points towards the requirement for solving assignments.

Task 1: Clustering for image retrieval (pen & paper)

In 2017, Flickr launched a feature called "Similarity Search" which allows users to find similar images given a query image. A demo and technical description can be found at: `http://code.flickr.net/2017/03/07/introducing-similarity-search-at-flickr/`.

Based on this article, briefly answer the following questions about how clustering algorithms for large-scale nearest neighbor image search can be implemented in practice.

1. What is the "feature vector" for clustering in this application, and how is it obtained?
2. Consider the direct k-nearest neighbours method to return k images most similar to a query image: extract the feature vector of the query image, and compare it to the feature vectors in the database to find the k feature vectors corresponding to the most similar images. Why is a direct implementation of this method not computationally feasible at a large scale (in the order of billions, 10^9, of images to search among)? Mention at least two reasons.
3. The idea of product quantization is to split every large feature vector $\vec{x}$ into subvectors $\vec{x}_i$, $i = 1, \ldots, n$, so that $\vec{x} = \begin{bmatrix}\vec{x}_1 & \vec{x}_2 & \ldots & \vec{x}_n\end{bmatrix}$ and then use k-means clustering on the subvectors separately. Then given a query vector $\vec{q}$, its nearest cluster center can be found by clustering each subvector $\vec{q}_i$ independently. Why does this help speed up nearest neighbour search?

Task 2: Cross-entropy and label smoothing regularization (pen & paper)

In this task, we review the motivation for and interpretation of *label smoothing regularization*, a technique used in training neural networks for large scale image classification.

Consider an image classification problem with m mutually exclusive classes. The typical loss function used to train classification networks is the cross-entropy loss, defined as

$$\ell(\vec{q}, \vec{p}) = -\sum_{i=1}^{m} q_i \log p_i, \tag{1}$$

where $\vec{p} = \begin{bmatrix} p_1 & p_2 & \dots & p_m \end{bmatrix}^T$ is a vector of class probabilities output by the neural network after a softmax activation function, and $\vec{q} = \begin{bmatrix} q_1 & q_2 & \dots & q_m \end{bmatrix}^T$ is a vector of probabilities expressing the desired or ground truth output.

1. Each p_i is formed from the activation values $\vec{z} = \begin{bmatrix} z_1 & z_2 & \dots & z_m \end{bmatrix}^T$, also called the *logits*, from the output layer before the softmax activation function:

   $$p_i = \frac{\exp(z_i)}{\sum\limits_{j=1}^{m} \exp(z_j)}. \tag{2}$$

   Show that $\frac{\partial \ell(\vec{q},\vec{p})}{\partial z_i} = p_i - q_i$. What are the minimum and maximum values of this partial derivative? Hint: recall that p_i and q_i are probabilities.

2. An image with true class h is typically assigned a vector $\vec{q}$ such that

   $$q_i = \begin{cases} 1 & \text{iff } i = h \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

   i.e., exactly the element corresponding to the true class equals 1 while others equal 0. Show that in such a case, $\ell(\vec{q},\vec{p}) = -\log p_h$. That is, when we minimize cross-entropy, we are (equivalently) maximizing the log likelihood $\log p_h$ of the true class.

3. The theoretical maximum of $\log p_h$ is $\log 1 = 0$ (Why?). This maximum is approached when the logit z_h of the ground truth class is much larger than all other logits z_i, $i \neq h$. Szegedy et al.[1] argue that this causes two problems:

   - Over-fitting. If the model learns to assign full probability to the ground-truth label for each training example, it is not guaranteed to generalize.
   - Encourages the differences between the largest logit and all other logits to become large. Combined with the bounded gradient $\frac{\partial \ell(\vec{q},\vec{p})}{\partial z_i}$, this reduces the ability of the model to adapt to new data. Intuitively, the model becomes too confident about its predictions.

   To address these problems, they propose *label smoothing regularization* (LSR), where instead of the one-hot ground truth vector $\vec{q}$, a new vector $\vec{q'} = \begin{bmatrix} q'_1 & q'_2 & \dots & q'_m \end{bmatrix}^T$ is used that is a weighted mixture of $\vec{q}$ and a uniform distribution $\vec{u} = \begin{bmatrix} \frac{1}{m} & \frac{1}{m} & \dots & \frac{1}{m} \end{bmatrix}^T$. Specifically, $\vec{q'} = (1-\epsilon)\vec{q} + \epsilon\vec{u}$, where $\epsilon \in (0,1)$ is a weight term typically chosen small, e.g., 0.1. When $\ell(\vec{q'},\vec{p})$ is used as the loss function, the network is no longer encouraged to make the largest logit much larger than all others. It was empirically shown that LSR consistently improved top-1 and top-5 error on ImageNet classification by about 0.2%.

   Show that:

   $$\ell(\vec{q'},\vec{p}) = (1-\epsilon)\ell(\vec{q},\vec{p}) + \epsilon\ell(\vec{u},\vec{p}). \tag{4}$$

   This allows an alternative interpretation. The second term $\ell(\vec{u},\vec{p})$ penalizes the network for deviations from the prior u, while the first term encourages the network to match the one-hot vector $\vec{q}$.

[1] Szegedy et al., "Rethinking the Inception Architecture for Computer Vision", CVPR 2016.

Task 3: Convolutional layers (pen & paper)

See note at start of assignment sheet concerning this task!

Recall from the lecture that it is typical to think of the inputs and outputs of convolutional layers as 3-dimensional volumes. It is in fact so common, that this fact is implicitly assumed in many sources and must be used by the reader to interpret the sizes of the layer inputs/outputs, and their number of parameters, if required. Spatial dimensions are often given, but the third dimension must be inferred from other information.

The table below shows a segment of two convolutional layers in a CNN, showing the typical information available to a reader of a design document. Based on the table and the interpretation in the lecture of inputs/outputs as 3D volumes, answer the questions below. You can ignore batch size. **Justify all your answers.**

Layer index	Number of filters	Filter size	Bias?	Activation	Spatial dimension of output
$l-1$	32	3 x 3	yes	ReLU	112 x 112
l	64	3 x 3	yes	ReLU	112 x 112

1. Given the information in the table only, can you infer the full size of the 3D volume that is the input to layer $l-1$? If yes, what is it? If not, why not? Note: layer $l-1$ is not necessarily the first layer in the network.

2. Can you infer the size of the 3D volume that is the input to layer l (equivalently, the output of layer $l-1$)? If yes, what is it? If not, why not?

3. Does layer l apply any kind of padding on its input? Why can you say this is or is not the case?

4. How would the spatial dimensions of the output of layer l change if bias is removed? Why?

5. How many parameters does layer l have? Besides the final answer, also give an equation and explain where the values come from.