# 上 海 交 通 大 学 试 卷（A 卷）

（ 2021 至 2022 学年 第 二 (春季) 学期 ）

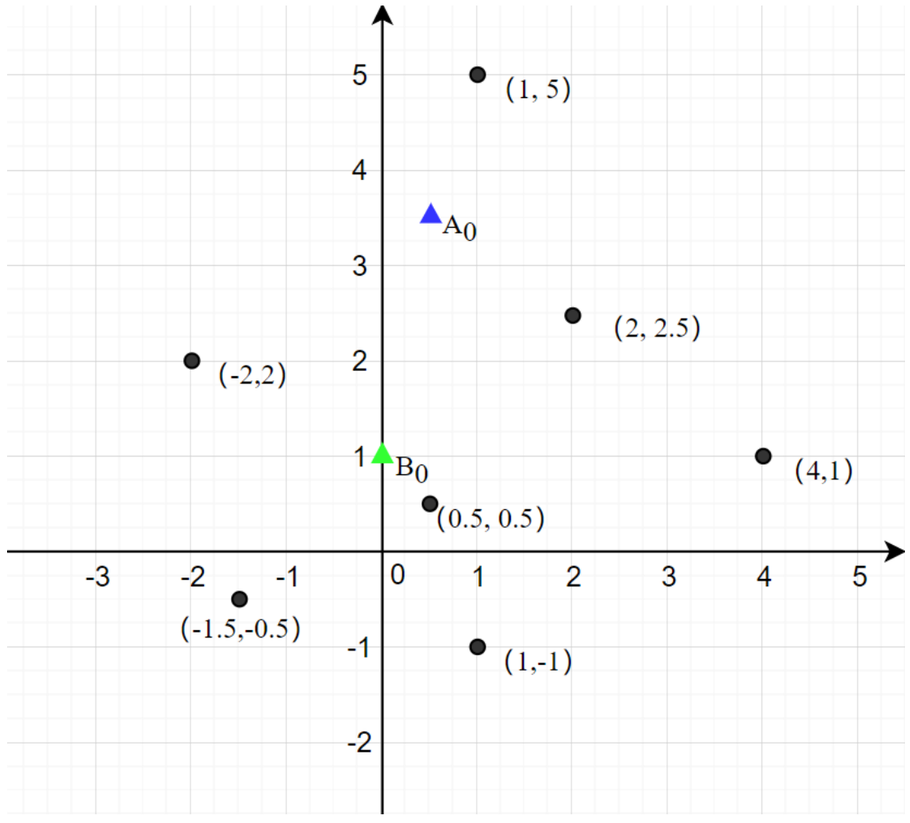
班级号_____ 学号_____ 姓名 _____

课程名称 _____ 成绩 _____

(一) 、有一组未分类的二维数据如图所示。要求将其分成 2 类，命名为：类 A，类 B。给定 A，B 两类的点集中心初始坐标 $A_0$ (0.5, 3.5) , $B_0$ (0, 1)。

(1) 请简述 K-Means 聚类算法的流程（7 分）。

（2）请计算 K-Means 迭代 2 次后 $A_2$ , $B_2$ 的位置，并给出计算过程 （8 分）。






我承诺，我将严格遵守考试纪律。

承诺人：_____

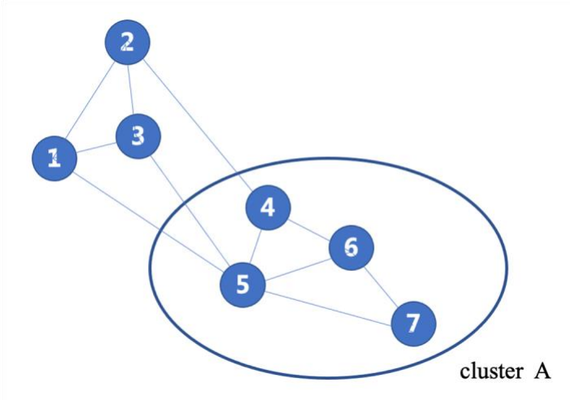
| 题号 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 得分 | | | | | | | | | | |
| 批阅人(流水阅卷教师签名处) | | | | | | | | | | |

（二）、给定一个图结构（Graph structure）的如图一所示：请：

**1**）根据图一写出它的 Adjacency matrix，  Degree Matrix，和  Laplacian Matrix；（8 分）

**2**）给定一个 cluster (簇) A  如图二所示，请计算图割 cut (A)  和 A 的 Conductance Score；（9  分）

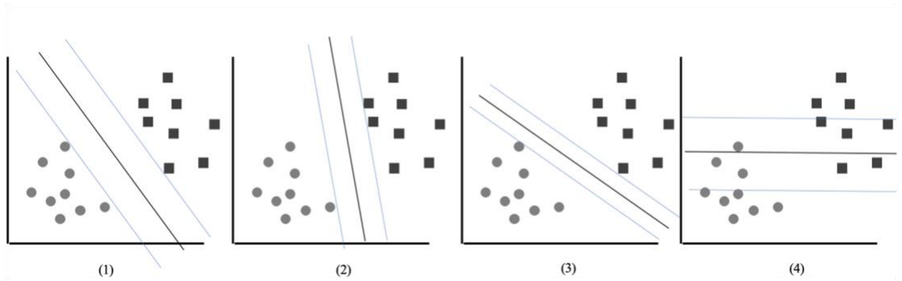
**3)**  (附加题) 在计算大型图分割的过程中，我们需要寻求 Laplacian 矩阵的第二小的特征值$\lambda_2$和它对应的特征向量。请阐述寻找第二小特征值的原因，**需列出关键公式**。（5 分）




（图一）

（三）、支持向量机 SVM

**1）** 请在下图给出的四个间隔（Margin）中选出最优的间隔。（2 分）




2） 请给出 SVM 中最大化 Margin (间隔) 的目标函数， 并解释该目标函数与限制条件的含义（4 分）

3） 结合第 2）问中的间隔 $\gamma$ 的形式，请推导：$\gamma = \frac{1}{\|w\|}$ (5 分)

**4）** 请简述 SVM 中限制条件（如下方公式）的意义 （6 分）

$$s.t. \forall i, y_i(w \cdot x_i + b) \geq 1$$

（四）、一位果商现有一小批榴莲样本，其中样本属性分为：刺的软硬，刺的密度。其中刺的软硬属性拥有三个可能的取值{扎手，渐软，不扎手}； 刺的密度属性有两个取值{密集，稀疏}。果商根据统计顾客的反馈，得到了如下属性与榴莲甜度的统计数据：

| 刺的软硬 | 刺的密度 | 甜度 |
|---|---|---|
| 扎手 | 稀疏 | 不甜 |
| 渐软 | 密集 | 甜 |
| 不扎手 | 稀疏 | 甜 |
| 扎手 | 密集 | 不甜 |
| 不扎手 | 密集 | 不甜 |
| 渐软 | 稀疏 | 不甜 |
| 渐软 | 密集 | 甜 |
| 扎手 | 密集 | 甜 |
| 不扎手 | 密集 | 不甜 |
| 扎手 | 稀疏 | 甜 |
| 不扎手 | 稀疏 | 甜 |
| 扎手 | 密集 | 不甜 |
| 不扎手 | 稀疏 | 甜 |
| 渐软 | 密集 | 不甜 |

请给出**计算过程**与**结果**：

a) *Info* (甜度)　（3 分）

b) *Info* $_{\text{刺的密度}}$ (甜度)　（3 分）

c) *Gain* (刺的密度)　（3 分）

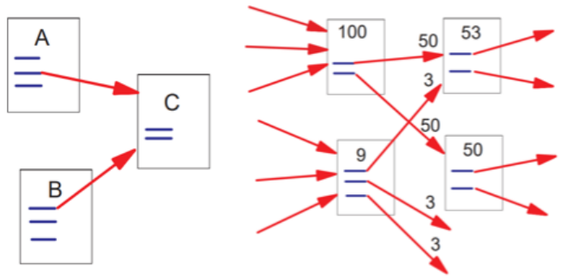
d) *SplitInfo* (刺的密度)　（4 分）

e) *GainRatio*(刺的密度)　（4 分）

f) 附加题：请画出完整决策树，并给出计算依据（4 分）

（五）、PageRank

**1） 结合图 1 简述 PageRank 算法的基本思想；**（5 分）




$$PR(C) = \frac{PR(A)}{L(A)} + \frac{PR(B)}{L(B)}$$

图 1

**2） 写出图 2 代表的状态转移矩阵 S；**（6 分）

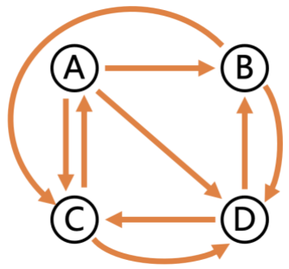
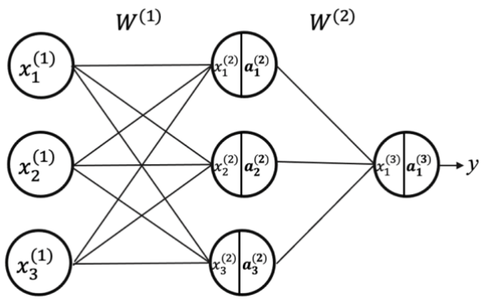



图 2

3）已知$P_0$，$P_i$，与 2）中的状态转移矩阵，写出 PageRank 的递推公式 $P_{i+1} = f(P_0, P_i, S, \alpha)$（6 分）

4）PageRank 的递推公式还有另外一种齐次形式，$P_{i+1} = AP_i$，请写出 $A$ 矩阵的表达式，并且证明 3）中的递推公式与 4）中的 $A$ 矩阵等价，默认$P_i$ 元素和为 1（提示：数学归纳法）(5 分)

（六）、一个 Fully-Connect（全连接）网络示意图如下

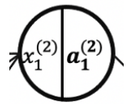



**图注解释:**

上标 $^{(l)}$ 代表第 $l$ 层，下标 $_k$ 代表当前层内的第 $k$ 个元素。例如：$x_1^{(2)}$代表第 2 层中的第 1 个元素；$W^{(1)}$ 为第一层到第二层之间的权重矩阵。$W^{(1)}$中的第 2 行第 1 列元素$w_{21}^{(1)}$代表第一层中的第 2 个元素$x_2^{(1)}$与第二层中第 1 个元素$x_1^{(2)}$连线上的权重。$a_1^{(2)} = f(x_1^{(2)})$，$f$为激活函数：

$$f(x) = \begin{cases} x, & if\ x > 0 \\ 0.1\,x, & if\ x \le 0 \end{cases}$$

下图所示的神经元代表：




**将$x_1^{(2)}$经过$f(x)$函数后得到$a_1^{(2)}$，并将$a_1^{(2)}$传递给后续神经元或输出为结果**

**给定初值：**

$$x_1^{(1)} = 1,\ \ x_2^{(1)} = 2,\ \ x_3^{(1)} = -1$$

$$W^{(1)} = \begin{bmatrix} -2 & 1 & 3 \\ 1 & 5 & -1 \\ -2 & 2 & 1 \end{bmatrix},\ \ W^{(2)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

**请给出计算过程与结果：**

**1）** 网络的输出值$y = a_1^{(3)}$= **？** （9 分）

**2）** 网络输出值$y$对$w_{21}^{(2)}$的偏导数$\frac{\partial y}{\partial w_{21}^{(2)}}$= **？**（8 分）

（七）、（附加题）论文阅读与推导：请阅读以下论文（局部），并回答相应问题（10 分）

# Spatial Transformer Networks

**Max Jaderberg** **Karen Simonyan** **Andrew Zisserman** **Koray Kavukcuoglu**

Google DeepMind, London, UK
{jaderberg,simonyan,zisserman,korayk}@google.com

## Abstract

Convolutional Neural Networks define an exceptionally powerful class of models, but are still limited by the lack of ability to be spatially invariant to the input data in a computationally and parameter efficient manner. In this work we introduce a new learnable module, the *Spatial Transformer*, which explicitly allows the spatial manipulation of data within the network. This differentiable module can be inserted into existing convolutional architectures, giving neural networks the ability to actively spatially transform feature maps, conditional on the feature map itself, without any extra training supervision or modification to the optimisation process. We show that the use of spatial transformers results in models which learn invariance to translation, scale, rotation and more generic warping, resulting in state-of-the-art performance on several benchmarks, and for a number of classes of transformations.

## 1 Introduction

Over recent years, the landscape of computer vision has been drastically altered and pushed forward through the adoption of a fast, scalable, end-to-end learning framework, the Convolutional Neural Network (CNN) [21]. Though not a recent invention, we now see a cornucopia of CNN-based models achieving state-of-the-art results in classification [19, 28, 35], localisation [31, 37], semantic segmentation [24], and action recognition [12, 32] tasks, amongst others.

A desirable property of a system which is able to reason about images is to disentangle object pose and part deformation from texture and shape. The introduction of local max-pooling layers in CNNs has helped to satisfy this property by allowing a network to be somewhat spatially invariant to the position of features. However, due to the typically small spatial support for max-pooling (*e.g.* $2 \times 2$ pixels) this spatial invariance is only realised over a deep hierarchy of max-pooling and convolutions, and the intermediate feature maps (convolutional layer activations) in a CNN are not actually invariant to large transformations of the input data [6, 22]. This limitation of CNNs is due to having only a limited, pre-defined pooling mechanism for dealing with variations in the spatial arrangement of data.

In this work we introduce a *Spatial Transformer* module, that can be included into a standard neural network architecture to provide spatial transformation capabilities. The action of the spatial transformer is conditioned on individual data samples, with the appropriate behaviour learnt during training for the task in question (without extra supervision). Unlike pooling layers, where the receptive fields are fixed and local, the spatial transformer module is a dynamic mechanism that can actively spatially transform an image (or a feature map) by producing an appropriate transformation for each input sample. The transformation is then performed on the entire feature map (non-locally) and can include scaling, cropping, rotations, as well as non-rigid deformations. This allows networks which include spatial transformers to not only select regions of an image that are most relevant (attention), but also to transform those regions to a canonical, expected pose to simplify recognition in the following layers. Notably, spatial transformers can be trained with standard back-propagation, allowing for end-to-end training of the models they are injected in.



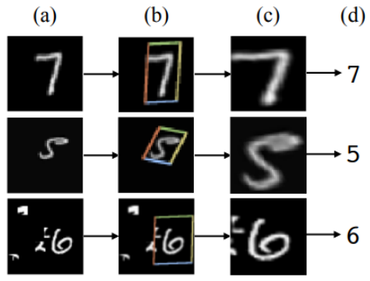



Figure 1: The result of using a spatial transformer as the first layer of a fully-connected network trained for distorted MNIST digit classification. (a) The input to the spatial transformer network is an image of an MNIST digit that is distorted with random translation, scale, rotation, and clutter. (b) The localisation network of the spatial transformer predicts a transformation to apply to the input image. (c) The output of the spatial transformer, after applying the transformation. (d) The classification prediction produced by the subsequent fully-connected network on the output of the spatial transformer. The spatial transformer network (a CNN including a spatial transformer module) is trained end-to-end with only class labels – no knowledge of the groundtruth transformations is given to the system.

Spatial transformers can be incorporated into CNNs to benefit multifarious tasks, for example: (i) *image classification:* suppose a CNN is trained to perform multi-way classification of images according to whether they contain a particular digit – where the position and size of the digit may vary significantly with each sample (and are uncorrelated with the class); a spatial transformer that crops out and scale-normalizes the appropriate region can simplify the subsequent classification task, and lead to superior classification performance, see Fig. 1; (ii) *co-localisation:* given a set of images containing different instances of the same (but unknown) class, a spatial transformer can be used to localise them in each image; (iii) *spatial attention:* a spatial transformer can be used for tasks requiring an attention mechanism, such as in [14, 39], but is more flexible and can be trained purely with backpropagation without reinforcement learning. A key benefit of using attention is that transformed (and so attended), lower resolution inputs can be used in favour of higher resolution raw inputs, resulting in increased computational efficiency.

The rest of the paper is organised as follows: Sect. 2 discusses some work related to our own, we introduce the formulation and implementation of the spatial transformer in Sect. 3, and finally give the results of experiments in Sect. 4. Additional experiments and implementation details are given in Appendix A.

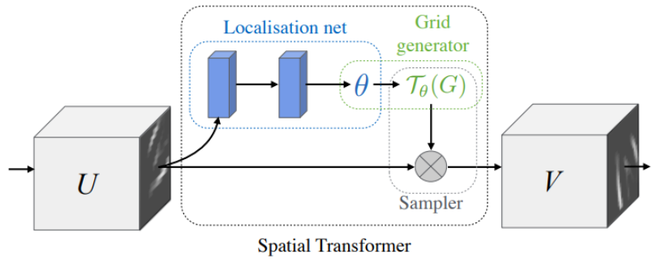



Figure 2: The architecture of a spatial transformer module. The input feature map $U$ is passed to a localisation network which regresses the transformation parameters $\theta$. The regular spatial grid $G$ over $V$ is transformed to the sampling grid $\mathcal{T}_\theta(G)$, which is applied to $U$ as described in Sect. 3.3, producing the warped output feature map $V$. The combination of the localisation network and sampling mechanism defines a spatial transformer.

## 3 Spatial Transformers

In this section we describe the formulation of a *spatial transformer*. This is a differentiable module which applies a spatial transformation to a feature map during a single forward pass, where the transformation is conditioned on the particular input, producing a single output feature map. For multi-channel inputs, the same warping is applied to each channel. For simplicity, in this section we consider single transforms and single outputs per transformer, however we can generalise to multiple transformations, as shown in experiments.

The spatial transformer mechanism is split into three parts, shown in Fig. 2. In order of computation, first a *localisation network* (Sect. 3.1) takes the input feature map, and through a number of hidden layers outputs the parameters of the spatial transformation that should be applied to the feature map – this gives a transformation conditional on the input. Then, the predicted transformation parameters are used to create a sampling grid, which is a set of points where the input map should be sampled to produce the transformed output. This is done by the *grid generator*, described in Sect. 3.2. Finally, the feature map and the sampling grid are taken as inputs to the *sampler*, producing the output map sampled from the input at the grid points (Sect. 3.3).

The combination of these three components forms a spatial transformer and will now be described in more detail in the following sections.

### 3.1 Localisation Network

The localisation network takes the input feature map $U \in \mathbb{R}^{H \times W \times C}$ with width $W$, height $H$ and $C$ channels and outputs $\theta$, the parameters of the transformation $\mathcal{T}_\theta$ to be applied to the feature map: $\theta = f_{\text{loc}}(U)$. The size of $\theta$ can vary depending on the transformation type that is parameterised, *e.g.* for an affine transformation $\theta$ is 6-dimensional as in (10).

The localisation network function $f_{\text{loc}}()$ can take any form, such as a fully-connected network or a convolutional network, but should include a final regression layer to produce the transformation parameters $\theta$.

### 3.2 Parameterised Sampling Grid

To perform a warping of the input feature map, each output pixel is computed by applying a sampling kernel centered at a particular location in the input feature map (this is described fully in the next section). By *pixel* we refer to an element of a generic feature map, not necessarily an image. In general, the output pixels are defined to lie on a regular grid $G = \{G_i\}$ of pixels $G_i = (x_i^t, y_i^t)$, forming an output feature map $V \in \mathbb{R}^{H' \times W' \times C}$, where $H'$ and $W'$ are the height and width of the grid, and $C$ is the number of channels, which is the same in the input and output.

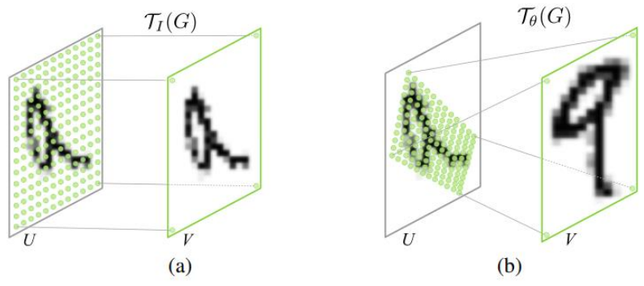



Figure 3: Two examples of applying the parameterised sampling grid to an image $U$ producing the output $V$. (a) The sampling grid is the regular grid $G = \mathcal{T}_I(G)$, where $I$ is the identity transformation parameters. (b) The sampling grid is the result of warping the regular grid with an affine transformation $\mathcal{T}_\theta(G)$.

For clarity of exposition, assume for the moment that $\mathcal{T}_\theta$ is a 2D affine transformation $\mathtt{A}_\theta$. We will discuss other transformations below. In this affine case, the pointwise transformation is

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \mathtt{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{1}$$

where $(x_i^t, y_i^t)$ are the target coordinates of the regular grid in the output feature map, $(x_i^s, y_i^s)$ are the source coordinates in the input feature map that define the sample points, and $\mathtt{A}_\theta$ is the affine transformation matrix. We use height and width normalised coordinates, such that $-1 \leq x_i^t, y_i^t \leq 1$ when within the spatial bounds of the output, and $-1 \leq x_i^s, y_i^s \leq 1$ when within the spatial bounds of the input (and similarly for the $y$ coordinates). The source/target transformation and sampling is equivalent to the standard texture mapping and coordinates used in graphics [8].

The transform defined in (10) allows cropping, translation, rotation, scale, and skew to be applied to the input feature map, and requires only 6 parameters (the 6 elements of $\mathtt{A}_\theta$) to be produced by the localisation network. It allows cropping because if the transformation is a contraction (i.e. the determinant of the left $2 \times 2$ sub-matrix has magnitude less than unity) then the mapped regular grid will lie in a parallelogram of area less than the range of $x_i^s, y_i^s$. The effect of this transformation on the grid compared to the identity transform is shown in Fig. 3.

The class of transformations $\mathcal{T}_\theta$ may be more constrained, such as that used for attention

$$\mathtt{A}_\theta = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \end{bmatrix} \tag{2}$$

allowing cropping, translation, and isotropic scaling by varying $s$, $t_x$, and $t_y$. The transformation $\mathcal{T}_\theta$ can also be more general, such as a plane projective transformation with 8 parameters, piecewise affine, or a thin plate spline. Indeed, the transformation can have any parameterised form, provided that it is differentiable with respect to the parameters – this crucially allows gradients to be backpropagated through from the sample points $\mathcal{T}_\theta(G_i)$ to the localisation network output $\theta$. If the transformation is parameterised in a structured, low-dimensional way, this reduces the complexity of the task assigned to the localisation network. For instance, a generic class of structured and differentiable transformations, which is a superset of attention, affine, projective, and thin plate spline transformations, is $\mathcal{T}_\theta = M_\theta B$, where $B$ is a target grid representation (*e.g.* in (10), $B$ is the regular grid $G$ in homogeneous coordinates), and $M_\theta$ is a matrix parameterised by $\theta$. In this case it is possible to not only learn how to predict $\theta$ for a sample, but also to learn $B$ for the task at hand.

### 3.3 Differentiable Image Sampling

To perform a spatial transformation of the input feature map, a sampler must take the set of sampling points $\mathcal{T}_\theta(G)$, along with the input feature map $U$ and produce the sampled output feature map $V$.

Each $(x_i^s, y_i^s)$ coordinate in $\mathcal{T}_\theta(G)$ defines the spatial location in the input where a sampling kernel is applied to get the value at a particular pixel in the output $V$. This can be written as

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c k(x_i^s - m; \Phi_x) k(y_i^s - n; \Phi_y) \quad \forall i \in [1 \ldots H'W'] \quad \forall c \in [1 \ldots C] \tag{3}$$

where $\Phi_x$ and $\Phi_y$ are the parameters of a generic sampling kernel $k()$ which defines the image interpolation (*e.g.* bilinear), $U_{nm}^c$ is the value at location $(n, m)$ in channel $c$ of the input, and $V_i^c$ is the output value for pixel $i$ at location $(x_i^t, y_i^t)$ in channel $c$. Note that the sampling is done identically for each channel of the input, so every channel is transformed in an identical way (this preserves spatial consistency between channels).

In theory, any sampling kernel can be used, as long as (sub-)gradients can be defined with respect to $x_i^s$ and $y_i^s$. For example, using the integer sampling kernel reduces (3) to

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \delta(\lfloor x_i^s + 0.5 \rfloor - m) \delta(\lfloor y_i^s + 0.5 \rfloor - n) \tag{4}$$

where $\lfloor x + 0.5 \rfloor$ rounds $x$ to the nearest integer and $\delta()$ is the Kronecker delta function. This sampling kernel equates to just copying the value at the nearest pixel to $(x_i^s, y_i^s)$ to the output location $(x_i^t, y_i^t)$. Alternatively, a bilinear sampling kernel can be used, giving

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|) \tag{5}$$

问题：

（1）请按你的理解设计 Figure.2 中 localization net 应有的形式，假定输入的 feature map 为 H=50，W=50，channel 为 10。以伪代码的形式设计神经网络，并简述每一层的作用即可。（5 分）

（2）请简述公式（3）中 kernel 函数 k()的作用。（5 分）