COMP 2211 Final Exam - Spring 2024 - HKUST
Question Paper

Date:	May 22, 2024 (Wednesday)
Time Allowed:	3 hours, 8:30–11:30 am

Instructions:

1. This is a closed-book, closed-notes examination.
2. There are **9** questions on **24** pages (including this cover page, and 5 blank pages at the end).
3. Write your answers in the **ANSWER BOOK** provided in black/blue ink. *NO pencil please, otherwise you are not allowed to appeal for any grading disagreements.*
4. All programming codes in your answers must be written in the Python version as taught in the class.
5. For programming questions, unless otherwise stated, you are **NOT** allowed to define additional classes, helper functions and use global variables, nor any library functions not mentioned in the questions.

Student Name	
Student ID	
Venue and Seat Number	

Problem 1 [10 points] True/False Questions

Indicate whether the following statements are true or false by putting **T** or **F** in the given table in the answer book. You get 1 point for each correct answer.

(a) The console output of the following Python code is `[1,2,3,4,5,6,7]`.

```
import numpy as np
# arange(start, stop): Values are generated within half-open interval [start,stop).
array = np.arange(1,10)
print(array[:-2])
```

(b) There is no way for the Naïve Bayes classifier to make a prediction if a categorical feature (e.g., color) has a new category (e.g., blue) not observed in the training data set.

(c) K-Nearest Neighbors classifier is a non-parametric machine learning algorithm with an assumption that the data are uniformly distributed.

(d) It is possible for K-Means Clustering to return empty clusters if certain initial centroid positions are unfortunate.

(e) If there are only two classes to predict, the following Multi-layer Perceptron (MLP) models will have the same output, given that they have the same initial weights and biases, and are trained in the same manner:

- MLP 1: Input layer, hidden dense layer with ReLU activation function, output layer with sigmoid activation function.
- MLP 2: Input layer, hidden dense layer with ReLU activation function, output layer with softmax activation function.

(f) An affine transformation may preserve distances and angles.

(g) The number of floating point multiplications involved when a 32×32 pixel RGB image is passed through a 2D convolutional layer with 8 3×3 kernels, padding of 3, and stride length of 1 is 8×3×3×36×36.

(h) Setting the dropout rate of a Convolutional Neural Network to 0.5 means that more than 50% of its layer's outputs are non-zero.

(i) Alpha-beta pruning can sometimes change the final decision made by the minimax algorithm, resulting in a different move being selected for the current player.

(j) Researches that involves human participants should require informed consent.

Problem 2 [10 points] Advanced Python: Image Processing with NumPy

You are working on an image processing project that involves manipulating arrays to perform various operations on images. You have been provided with a NumPy array `img` representing a grayscale image of size 512×512:

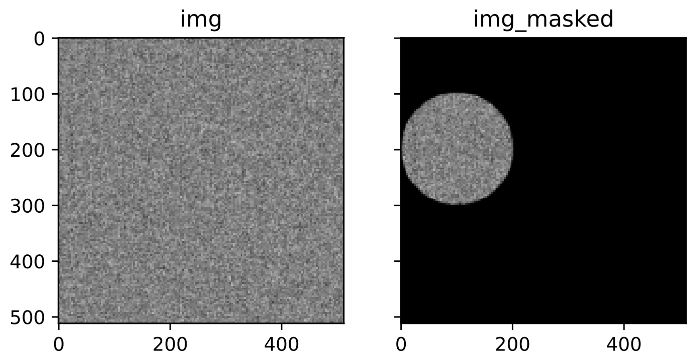
```python
import numpy as np
img = np.random.randint(0, 256, size=(512, 512), dtype=np.uint8)
# Note: random.randint(low, high=None, size=None, dtype=int)
# returns random integers from low (inclusive) to high (exclusive).
```

(a) [4 points] **Image masking:**

Implement the following function which creates a new array `img_masked` by applying a mask to `img`. The mask is defined as a 2D array mask of size 512×512, where `mask[i, j] = 1` if the pixel at `img[i, j]` is within a circle (boundary included) of radius 100 centered at a given position `[center_x, center_y]`, and `mask[i, j] = 0` otherwise.

```python
def apply_circle_mask(img, center_x, center_y):
    # --- YOUR CODE HERE ---
    return img_masked
```

Below is an example for `center_x = 100` and `center_y = 200`.



Do **NOT** use any explicit loop to implement the function. You may find the following functions useful for this question.

- `np.arange(start, stop, step)` returns spaced values within a given interval.
  - `start` (optional) is the start of interval. The default start value is 0.
  - `stop` is the end of interval. The returned interval does not include this value.
  - `step` (optional) is the spacing between values. The default spacing is 1.
- `np.square(a)` returns the element-wise square of an array.
  - `a` is the input array.

  This is equivalent to `a ** 2`.
- `np.sqrt(a)` returns the element-wise square root of an array.
  - `a` is the input array.

- `np.expand_dims(a, axis)` returns a new array with a new axis of size 1 inserted.
  - `a` is the input array.
  - `axis` is the position where the axis is to be inserted.

  If `axis` is 0, this is equivalent to `a[np.newaxis]` and `a[None]`. If `axis` is 1, this is equivalent to `a[:, np.newaxis]` and `a[:, None]`.

(b) [6 points] **Image blurring:**

Your task is to use NumPy to apply the following 3×3 blur filter to `img` with zero padding so that `img_blur` has the same shape as `img`.

```
blur_filter = np.array([[1/9, 1/9, 1/9],
                        [1/9, 1/9, 1/9],
                        [1/9, 1/9, 1/9]])
```

Implement the following function so that after the whole code snippet below is executed, `img_blur` stores the desired result.

```
def img_flatten_conv_1d(img, v):
    # --- YOUR CODE HERE ---
    return img_conv

v = blur_filter.sum(0) # 'sum(0)' returns the sum of the array elements over axis 0.
img_blur = img_flatten_conv_1d(img, v)
img_blur = img_flatten_conv_1d(img_blur.T, v).T
```

Do **NOT** use any explicit loop in your code. You may find the following functions useful for this question.

- `np.convolve(a, v, mode='full')` returns the discrete, linear convolution of two one-dimensional sequences.
  - `a` of shape $(N,)$: First one-dimensional input array (or array-like structure, e.g., list).
  - `v` of shape $(M,)$: Second one-dimensional input array (or array-like structure, e.g., list).
  - `mode` (optional): The convolution mode. It must be one of `'full'`, `'valid'`, `'same'`. **Note:** use `'valid'` for this question. When `'valid'` is used, `np.convolve` is equivalent to the following function:

```
def convolve_valid(a, v):
    if len(a) < len(v):
        a, v = v, a    # swap the array if v is longer than a
    c = np.zeros(len(a) - len(v) + 1)
    for i in range(len(c)):
        c[i] = np.sum(a[i:i+len(v)] * v[::-1])
    return c
```

Examples:

```
>>> np.convolve([1,2,3],[0,1,0.5], 'valid')
array([2.5])  # this is the output array
>>> np.convolve([1,2,3,4],[0,1,0.5], 'valid')
array([2.5, 3.5])  # this is the output array
```

- `np.zeros(shape, dtype=float)` returns a new array of given shape and type, filled with zeros.
  - `shape` is the shape of the new array, e.g., $(2, 3)$ or 2.
  - `dtype` (optional) is the desired data type for the array, e.g., `np.int8`. The default is `np.float64`.
- `np.concatenate((a1, a2, ...), axis=0)` joins a sequence of arrays along an existing axis.
  - `a1, a2, ...` is a sequence of arrays (or array-like structure, e.g., list). The arrays must have the same shape, except in the dimension corresponding to `axis` (the first, by default).
  - `axis` (optional) is the axis along which the arrays will be joined. If `axis` is None, arrays are flattened before use. The default is 0.
- `np.reshape(a, newshape)` gives a new shape to an array without changing its data.
  - `a` is the array to be reshaped.
  - `newshape` is the new shape. It should be compatible with the original shape. If an integer, then the result will be a 1D array of that length. One shape dimension can be -1. In this case, the value is inferred from the length of the array and the remaining dimensions (if any).

  This is equivalent to `a.reshape(newshape)`.
- `np.transpose(a)` returns the transpose of an array.
  - `a` is the input array.

  This is equivalent to `a.T`.

Hints:

(1) In this case, applying `blur_filter` to the image can also be done by consecutively applying two 1D filters, one vertically and the other horizontally, to the image.

(2) Since `np.convolve` only accepts 1D arrays, you may consider flattening the image array, applying `np.convolve` to the flattened array, and then reshaping it back to a 2D array.

(3) Be aware of the boundaries since `np.convolve` with `mode='valid'` does **not** pad the array and the output array does **not** always have the same shape as the input array. Also, remember to remove the padding (if any) after convolutions.

Problem 3 [12 points] Naïve Bayes, K-Nearest Neighbors and Perceptron

Below is some health data of patients with and without dengue fever.

Patient #	Diagnosis (B)	Body Temperature (Celsius) (e_1)	Pulse Rate (bpm) (e_2)
1	Dengue	40	60
2	Dengue	39	50
3	Dengue	38	70
4	No dengue	36	90
5	No dengue	36.5	75
6	No dengue	37	105

You can assume the data follows Gaussian distribution:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

where

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)^2}$$

(a) [6 points] Using the data above, calculate the following. Round off calculations to the fourth decimal place.

(i) P(e_1 = 36 | B = No dengue)

(ii) P(e_2 = 85 | B = No dengue)

(iii) P(e_1 = 36 | B = No dengue) P(e_2 = 85 | B = No dengue) P(B = No dengue)

(iv) Calculate the posterior probability that the belief is 'No dengue' for the sample with evidence $E = \{e_1 = 36, e_2 = 85\}$ if
$P((e_1 = 36, e_2 = 85)|B = Dengue)P(B = Dengue) = 0.0000036$

(b) [4.5 points] Identify the reasons for inaccurate predictions when using the following number of folds to evaluate the performance of a 3-Nearest Neighbors classifier model on the given dataset (i.e., not including the sample introduced in part (a)) without shuffling.

(i) No. of folds = 6

(ii) No. of folds = 3

(iii) No. of folds = 2

(c) [1.5 points] If the true label of the sample $\{e_1 = 36, e_2 = 85\}$ is 'No dengue', will the perceptron model make a good prediction for the sample? Provide explanations with evidence for why or why not.

Problem 4 [11 points] Multi-layer Perceptron

(a) [5 points] Amanda wants to train a multi-layer perceptron to classify various renting options' popularity. Each housing option is represented with X, a three-dimensional vector (x_1, x_2, x_3).

x_1 is the noise level rated from 1 to 5. x_2 is the proximity to campus, measured in minutes it takes to arrive at the north gate. x_3 is the availability of food options rated on a scale from 1 (no food available) to 5 (a wide variety of food available). Amanda has collected the opinions of a group of students on their preferences, classifying the housing options into three types: low, medium, and high, represented as one-hot vector $Y = (y_1, y_2, y_3)$.

When designing the network architecture, Amanda has two ideas.

Model A

Layer (type)	Output Shape
dense_1 (Dense)	(None, 6)
dense_2 (Dense)	(None, 4)
dense_3 (Dense)	(None, 3)

Model B

Layer (type)	Output Shape
dense_1 (Dense)	(None, 3)
dense_2 (Dense)	(None, 5)
dense_3 (Dense)	(None, 2)
dense_4 (Dense)	(None, 3)

(i) In a multi-layer perceptron, suppose there are L hidden layers, each layer k (starting from 1) has l_k hidden nodes. The input data is a n-dimensional vector, and the output data is m-dimensional.

(I) Calculate the number of updated parameters in the MLP model. Represent your result using n, m, L, and l_k for $k = 1, \ldots, L$.

(II) Apply your result in part (a)(i)(I) to Model A and Model B separately.

(ii) Please help Amanda decide on which multi-layer perceptron (A or B) to choose for potentially better performance. Please briefly explain why.

(b) [1 point] Why do we use activation functions in multi-layer perceptron neural networks?

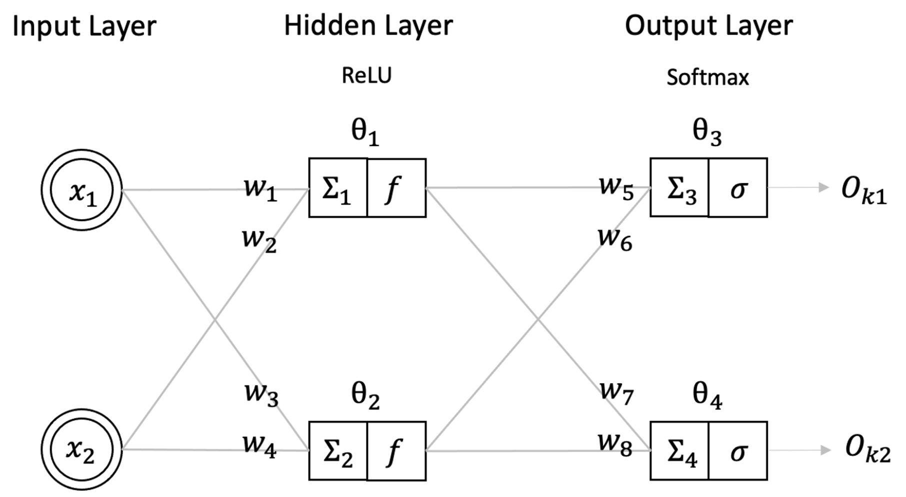
(c) [2 points] Consider the two activation functions we have learned in class:

- Sigmoid activation function: $\sigma(x) = \frac{1}{1+e^{-x}}$.
- Binary step activation function:

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{otherwise} \end{cases}$$

The binary step activation function gives a hard threshold at 0, and its gradients are 0 almost everywhere. What is the problem if we use the binary step activation function in a multi-layer perceptron network? If we want to avoid the problem while approximating the binary step activation function, how to make use of the sigmoid activation function to achieve that?

(d) [3 points] You are given an multi-layer perceptron model with the architecture shown below.



Parameter	Value
w_1	0.2
w_2	0
w_3	-0.1
w_4	0
w_5	0.1
w_6	0
w_7	0
w_8	0.2
θ_1	0
θ_2	0
θ_3	0
θ_4	0.1

- ReLU: $f(z) = max(0, z)$.
- Softmax: $f(z_i) = \frac{e^{z_i}}{\sum_{j=0}^{n-1} e^{z_j}}$, where $\mathbf{z} = [z_0, z_1, \ldots, z_{n-1}]$.

(i) For a sample with features x_1=1 and x_2=1, what are the outputs of the hidden layer and the output layer? If necessary, round off the values to two decimal places.

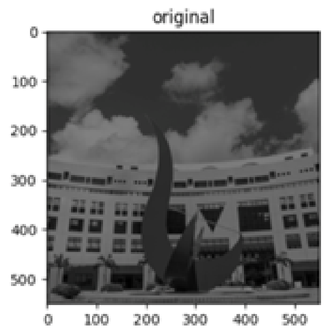
(ii) If the target labels have values T_{k1}=1,T_{k2}=0 for the sample in part d(i), calculate the new values of the weights: w_5, w_7, and w_1 after one round of backward propagation if the learning rate is 0.4. Round off the values to four decimal places.

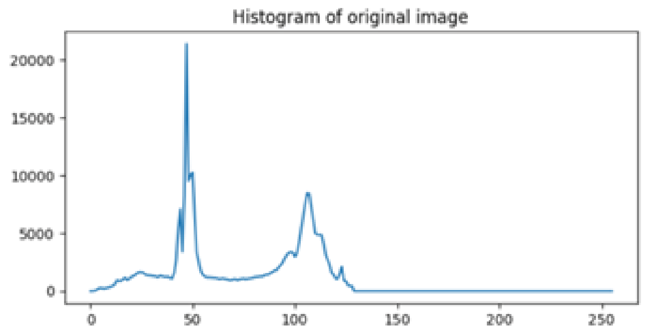
For reference, here are some of the equations used in the back propagation.

$$\delta_k = (O_k - T_k)O_k(1 - O_k)$$
$$\delta_j = O_j(1 - O_j)\sum_{k\in K}\delta_k w_{jk}$$
$$w_{jk} \leftarrow w_{jk} - \eta\delta_k O_j$$
$$w_{ij} \leftarrow w_{ij} - \eta\delta_j O_i$$

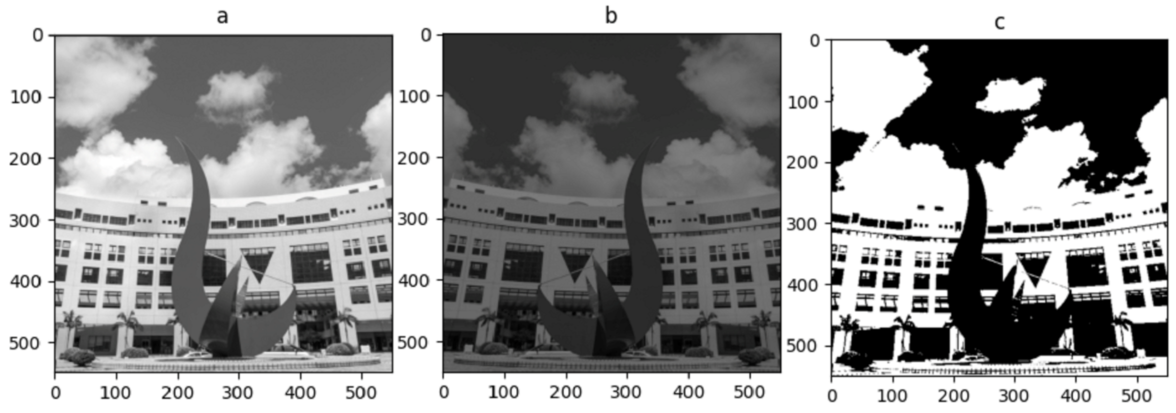
Problem 5 [13 points] Digital Image Processing

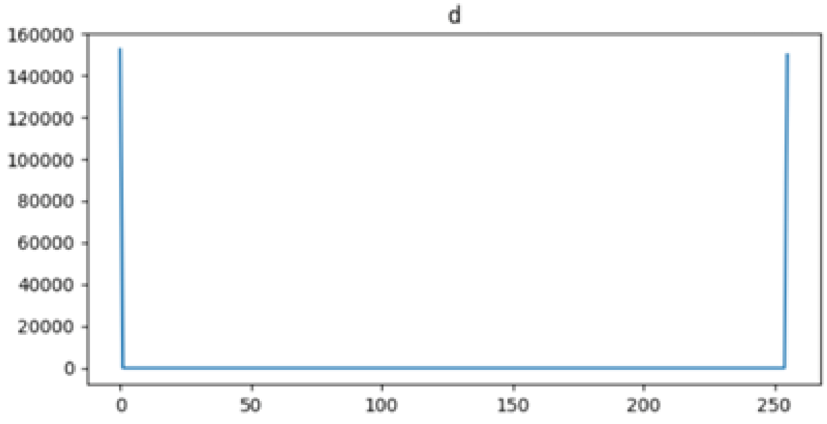
(a) [4.5 points] The following is a grayscale image of the HKUST redbird and its corresponding histogram. An grayscale image histogram is a distribution showing the frequency of occurrence of each gray-level value.

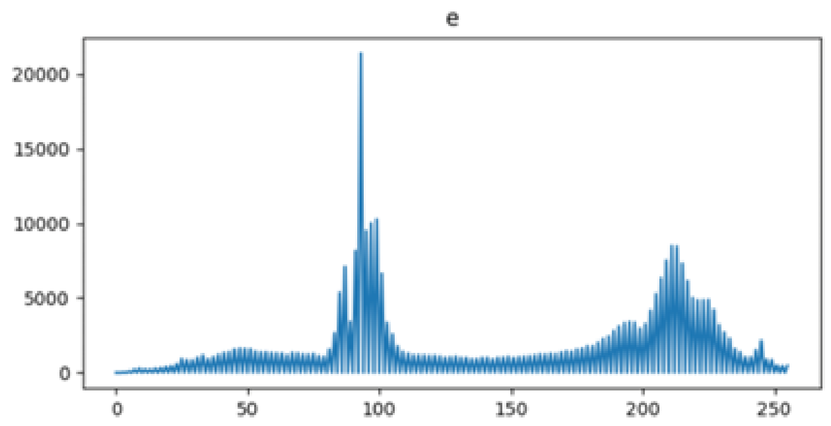


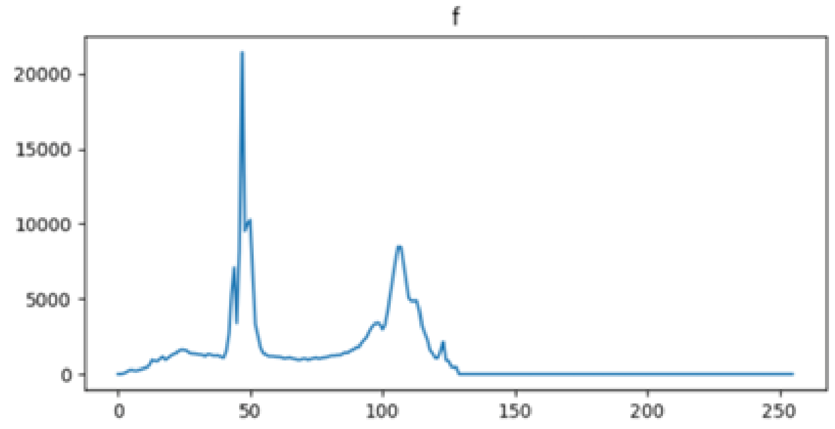


After some transformation on the original image, we get the transformed images shown below (first row). Their histograms are shuffled and shown in the rows below (second, third, and fourth row). Please state which transformations are applied to get the resulting images for (a),(b),(c), and what is the correct pairing for the images and histograms between {a,b,c} and {d,e,f}. Please also briefly state why.









(b) [3 points] Consider the following 3 × 3 image. Perform binary thresholding of the image using Otsu's method. The initial threshold is T=100, and we apply one iteration. What is the resulting threshold and the resulting image after thresholding? What is the advantage of using Otsu's Method for image thresholding compared to the regular image thresholding algorithm?

2	4	8
16	32	64
128	128	128

(c) [2 points] What is the resulting image of size 7 × 7 after adding reflection padding of size 2 on the original image in part (b)?

(d) [1.5 points] Please briefly explain the effect of convolving an image with the following kernels.

(i) Kernel 1:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

(ii) Kernel 2:

-1	0	1
-1	0	1
-1	0	1

(iii) Kernel 3:

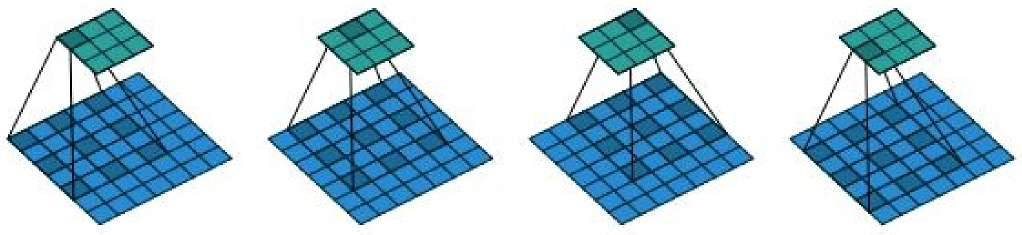
-1	-1	-1
-1	9	-1
-1	-1	-1

(e) [2 points] Is it possible to design a 3×3 kernel and apply convolution with the kernel to flip a 64×64 image horizontally or vertically? If yes, please give such a kernel. If not, please explain why.

Problem 6 [13 points] Dilated Convolution and Dropout

(a) [10 points] Dilated convolution is a variation of the standard convolution operation that involves skipping input elements by a certain dilation rate. By doing this, the convolutional kernel can be made to "see" a larger area of the input data without actually increasing the number of parameters it has. The dilation rate determines the spacing between the values in the kernel.

The figure below demonstrates how a 3×3 kernel is applied to a 7×7 image using a dilation factor of 2.



Given the incomplete implementation of the Python function `dilated_convolution` that takes the following inputs:

- `input_array`: a 2D NumPy array representing the input data
- `kernel`: a 2D NumPy array representing the convolutional kernel
- `dilation_rate`: an integer representing the dilation rate (default value is 1)
- `stride`: an integer representing the stride (default value is 1)
- `padding`: a string representing the padding type, either 'valid' (no padding) or 'same' (padding to preserve input dimensions) (default value is 'valid')

and returns a 2D NumPy array representing the output of the dilated convolution operation.

Complete the missing parts of the function using NumPy, without using any specialized deep learning libraries so that the execution of the test script produces the required output. Make sure that your implementation supports stride and padding options.

You may find the following formula for determining the size of output image of **regular image convolution** useful for this question.
(Size of image dimension - Size of kernel dimension + 2 × Padding) / Stride + 1

```
import numpy as np

# Zero pad the input_array. For example
# a = [[1,2,3]]
# np.pad(a, ((1,2),(3,4)), 'constant')
# >> array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
# >>        [0, 0, 0, 1, 2, 3, 0, 0, 0, 0],
# >>        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
# >>        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])

def dilated_convolution(input_array, kernel, dilation_rate=1,
                        stride=1, padding='valid'):
    # Apply padding to the input array if specified
    if padding == 'same':
        pad = #______ TODO 1 ______
        padded_input = np.pad(input_array, ((pad, pad), (pad, pad)), mode='constant')
    else:
        padded_input = input_array

    # Calculate the output shape
    output_rows = #______ TODO 2 ______
    output_cols = #______ TODO 3 ______

    kernel_rows, kernel_cols = kernel.shape

    # Initialize the output array with zeros
    output_array = np.zeros((output_rows, output_cols))

    # Iterate through the kernel and perform the convolution
    for i in range(kernel_rows):
        for j in range(kernel_cols):
            # Calculate the input indices for the current kernel position
            input_row_indices = #______ TODO 4 ______
            input_col_indices = #______ TODO 5 ______

            # Perform the convolution and accumulate the results in the output array
            output_array += #______ TODO 6 ______

    return output_array
```

```
# Test script
input_array = np.array(np.arange(100).reshape(10,10))

kernel = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])

dilation_rate = 2; stride = 2; padding = 'same'

output_array = dilated_convolution(input_array, kernel,
                                   dilation_rate, stride, padding)
print(output_array)

# Output:
# [[ 22.  26.  30.  34.   8.]
#  [ 62.  66.  72.  78.  34.]
#  [102. 126. 132. 138.  74.]
#  [142. 186. 192. 198. 114.]
#  [ 80. 142. 146. 150. 154.]]
```

You may find the following functions useful for this question.

- `np.arange(start, stop, step)` returns spaced values with a given interval.
  - `start` (optional) is the start of interval. The default start value is 0.
  - `stop` is the end of interval. The returned interval does not include this value.
  - `step` (optional) is the spacing between values. The default spacing is 1.
- `np.array(object)` creates an array.
  - `object` is an array or any sequence.
- `np.reshape(a, newshape)` gives a new shape to an array without changing its data.
  - `a` is the array to be reshaped.
  - `newshape` is the new shape. It should be compatible with the original shape. If an integer, then the result will be a 1D array of that length. One shape dimension can be -1. In this case, the value is inferred from the length of the array and the remaining dimensions (if any).

  This is equivalent to `a.reshape(newshape)`.
- `ndarray.shape` returns a tuple of array dimensions.
- `ndarray.zeros(shape, dtype=float)` returns a new array of given shape and type, filled with zeros.
  - `shape` is the shape of the new array, e.g., (2,3) or 2.
  - `dtype`(optional) is the desired data type for the array, e.g., `np.int8`. The default is `np.float64`.
- `range(n)` returns a numeric series starting with `0` and extending up to but not including `n`.

(b) [3 points] You are given an incomplete implementation of the Python function `dropout`, which implements the Dropout technique for regularization in neural networks. The function takes the following input:

- `input_array`: a 2D NumPy array representing the input data
- `p`: a dropout probability

The function generates a binary mask with the same shape as the input, where each element is 1 with probability `1-p` and 0 with probability `p`. The input is then multiplied element-wise by the mask, effectively dropping out random elements. The function returns a NumPy array representing the output of the dropout operation.

Complete the missing part of the function using NumPy, without using any specialized deep learning libraries so that the execution of the test script produces the required output.

You may find the following function useful for this question.

- `np.random.rand(d0, d1, ..., dn)` returns an array of the given shape and populate it with random samples from a uniform distribution over $[0, 1)$.
  - `d0, d1,..., dn` (optional) represent the dimensions of the returned array, must be non-negative. If no argument is given, a single Python float is returned.

```
import numpy as np

def dropout(input_array, p):
    mask = #______ TODO ______
    return input_array * mask
```

Problem 7 [18 points] Convolutional Neural Network

(a) [8 points] Consider the following Keras implementation of a Convolutional Neural Network:

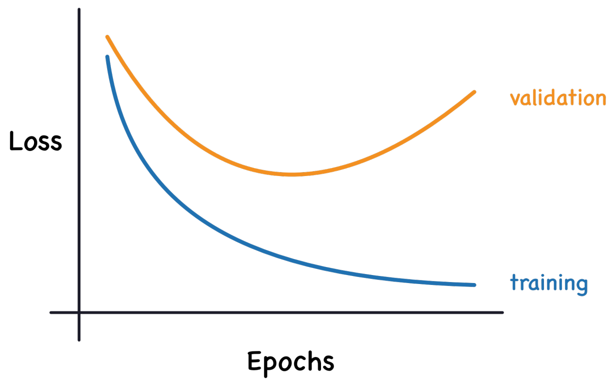
```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Dense, Flatten

model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5, 5), padding='same',
                 activation='relu', input_shape=(32, 32, 3)))
model.add(Conv2D(filters=64, kernel_size=(3, 3), padding='same',
                 activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(filters=64, kernel_size=(3, 3), padding='same',
                 activation='relu', strides=(2, 2)))
model.add(Conv2D(filters=64, kernel_size=(3, 3), padding='same',
                 activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(units=128, activation='relu'))
model.add(Dense(units=10, activation='softmax'))
```

(i) What is the padding size (in terms of border width) of the first convolutional layer?

(ii) Fill in the blanks in the answer book in the `model.summary()` of the network. Assume all convolutional layers have biases. Numbers shown as `?` are kept secret and you may ignore them.
Hint: When padding = "same" and strides = 2, the output has the half size of the input.

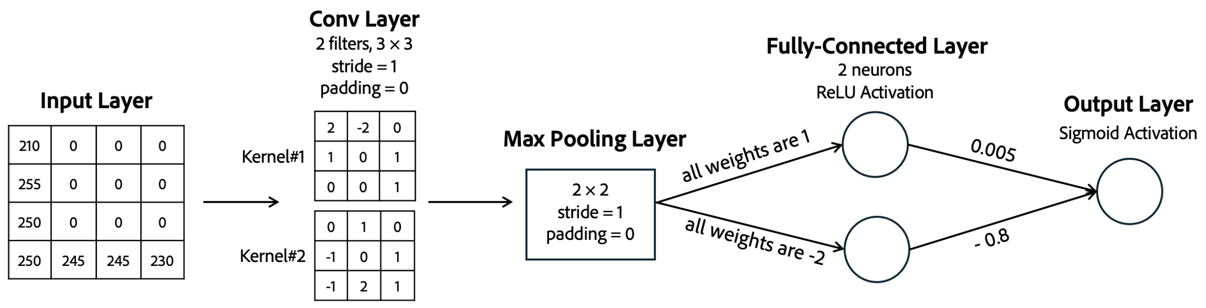
(iii) Suppose you observe the following loss curves as you train the Convolutional Neural Network. What is the most likely problem with the model? When does the problem usually occur? Describe a common way to mitigate the problem and explain why it works (in a few sentences).



(b) [5 points] What is the output of the following parts of this very tiny convolutional neural network? This network was trained to classify whether the input 4×4 image depicts the letter “L” or not. Stride is 1, padding size is 0 for all layers, and the pool size for the max pooling layer is (2, 2). Show your calculation step by step. You only need to keep 1 decimal number.

- ReLU activation function $f(z) = max(0, x)$.
- Sigmoid activation function $f(z) = \frac{1}{1+e^{-z}}$, where $e = 2.718$.

Note: Assume the given kernels have already been flipped.



(i) Feature map after Max-Pooling

(ii) Feature in the fully-connected layer

(iii) Output

(c) [5 points] In video processing, 3D convolution is particularly valuable due to its ability to capture temporal dynamics as well as spatial features, which is essential for understanding content across frames. Unlike 2D convolution, which only analyzes spatial patterns within a single frame, 3D convolution extends the analysis across multiple consecutive frames, treating time as the third dimension alongside height and width. This approach allows a Convolutional Neural Network (CNN) to not only perceive static features in individual frames but also to understand movements and changes over time, which are crucial for tasks such as action recognition, video classification, and anomaly detection in video streams.

Suppose we have video data of shape (32, 400, 300, 3), corresponding to (#keyframes, width, height, #channels). In the first layer, we want to apply a 3D convolution with 100 kernels in the shape of (3, 3, 3), corresponding to (keyframe, width, height). The padding is (0, 2, 2), and the stride is (1, 2, 2). Please answer the following question with calculation steps or rationales.

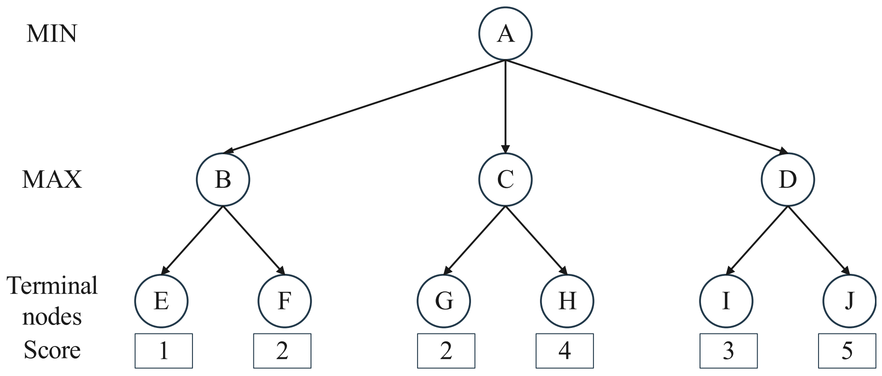
(i) What is the shape of the output after this layer?

(ii) How many weight parameters are there?

(iii) How many bias parameters are there?

Problem 8 [10 points] Minimax and Alpha-Beta Pruning

(a) [5 points] The figure below shows a tree in a minimax game with two players.

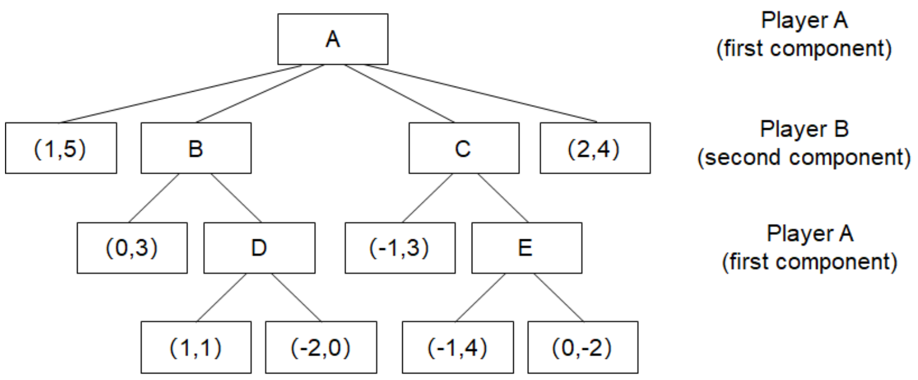


(i) Calculate the best score for the non-terminal nodes with the minimax algorithm.

(ii) Suppose we are using an alpha-beta pruning algorithm. Indicate which edge will be pruned. Gives the alpha-beta values when running. We denote the edge between nodes A and B as AB.

(iii) How do you make a minimax algorithm to find the shortest path to victory? Explain in one sentence.

(b) [5 points] Consider the non zero-sum generalization in which the sum of the two players' utilities are not necessarily zero. Because player A's utility no longer determines player B's utility exactly, the leaf utilities are written as pairs (U_A, U_B), with the first and second component indicating the utility of that leaf to player A and player B respectively. In this generalized setting, player A seeks to maximize U_A, the first component, while player B seeks to maximize U_B, the second component.



(i) Complete the table in the answer book by estimating the value (as pairs) at each of the internal node. Assume that each player maximizes their own utility.

(ii) Is alpha-beta pruning still applicable in this case? Briefly explain why and provide an example. (Hint: you can think about the case where $U_A(s) = U_B(s)$ for all nodes.)

Problem 9 [3 points] Ethics of Artificial Intelligence

From what areas we should ensure the AI models in production in their organizations function ethically?

-------------------- END OF PAPER --------------------

/* Rough work */

/* Rough work */

/* Rough work */

/* Rough work */

/* Rough work */