**University of Central Missouri**

**Department of Computer Science & Cybersecurity**

**CS5720 Neural Networks and Deep Learning**

**Summer 2025**

**Home Assignment 2. (Cover Ch 5, 6)**

**Student name: ALCHURI NIKHITHA**

**Submission Requirements:**

* Total Points: 100
* Once finished your assignment push your source code to your repo (GitHub) and explain the work through the ReadMe file properly. Make sure you add your student info in the ReadMe file.
* Submit your GitHub link and video on Bright Space.
* Comment your code appropriately ***IMPORTANT.***
* Make a simple video about 2 to 3 minutes which includes demonstration of your home assignment and explanation of code snippets.
* Any submission after provided deadline is considered as a late submission.

**Question 1: Convolution Operations with Different Parameters**

**Task: Implement Convolution with Different Stride and Padding (10 points)**

Write a Python script using **NumPy and TensorFlow/Keras** to perform **convolution** on a **5×5 input matrix** using a **3×3 kernel** with varying parameters.

1. Define the following **5×5 input matrix**:

A number with black text

AI-generated content may be incorrect.

1. Define the following **3×3 kernel**:

A number lines with numbers

AI-generated content may be incorrect.

1. Perform **convolution operations** with:
   * **Stride = 1, Padding = ‘VALID’**
   * **Stride = 1, Padding = ‘SAME’**
   * **Stride = 2, Padding = ‘VALID’**
   * **Stride = 2, Padding = ‘SAME’**
2. Print the **output feature maps** for each case.

**Expected Output**

Print the output feature maps for

* + **Stride = 1, Padding = ‘VALID’**
  + **Stride = 1, Padding = ‘SAME’**
  + **Stride = 2, Padding = ‘VALID’**
  + **Stride = 2, Padding = ‘SAME’**

**Question 2: CNN Feature Extraction with Filters and Pooling**

**Task 1: Implement Edge Detection Using Convolution (15 points)**

Write a Python script using **NumPy and OpenCV (cv2)** to apply **edge detection** on an image using a **Sobel filter**.

* Load a grayscale image (you can use any sample image).
* Apply the **Sobel filter** for **edge detection** in the **x-direction** and **y-direction**.
* Display the original image and the filtered images.

Use the following Sobel filters:

A diagram of a mathematical equation

AI-generated content may be incorrect.

**Task 2: Implement Max Pooling and Average Pooling (15 points)**

Write a Python script using **TensorFlow/Keras** to demonstrate **Max Pooling** and **Average Pooling**.

* Create a **random 4x4 matrix** as an input image.
* Apply a **2x2 Max Pooling** operation.
* Apply a **2x2 Average Pooling** operation.
* Print the original matrix, max-pooled matrix, and average-pooled matrix.

**Expected Output**

**Task1: Edge Detection using Sobel Filter**

* Display **three images**:
  1. **Original Image**
  2. **Edge detection using Sobel-X**
  3. **Edge detection using Sobel-Y**

**Task2: Pooling Operations on Random 4×4 Matrix**

* Printed **original matrix, max pooled matrix, and average pooled matrix**.

**Q3: Data Preprocessing - Standardization vs. Normalization**

**Task:** Data preprocessing ensures that input data is in the right format for neural networks. You will explore **standardization and normalization** techniques on a dataset and analyze their impact.

1. Load the **Iris dataset** from sklearn.datasets.
2. Implement **Min-Max Normalization** on all numeric features and print the transformed dataset.
3. Implement **Z-score Standardization** and compare it with normalization by visualizing distributions using histograms.
4. Train a simple **Logistic Regression** model before and after applying these transformations and compare the accuracy.
5. Explain in which scenarios **normalization** vs. **standardization** is preferable for deep learning.

## ANS: **Normalization vs. Standardization in Deep Learning**

### **When to Use Normalization (Min–Max Scaling):**

* **Unknown or non-Gaussian data distributions**: If feature distributions aren’t Gaussian, or you have no prior knowledge, min–max scaling ensures features are in a consistent range, typically [0, 1].
* **Distance-based models & neural nets**: Algorithms like k‑NN, k‑means, and neural networks greatly benefit from normalization because it ensures no single feature dominates in distance computation.
* **Bounded inputs (e.g., images)**: When feature values are already bounded (e.g., pixel intensities 0–255), normalization preserves their structure and improves convergence.

### **When to Use Standardization (Z‑Score Scaling):**

* **Gaussian-like or unknown distributions**: When data approximates a normal distribution, standardizing to mean 0 and variance 1 helps models that assume such distributions.
* **Algorithms sensitive to variance and outliers**: Techniques like linear/logistic regression, SVM, PCA, and algorithms using gradient descent converge faster and behave more stably when inputs are standardized.
* **Handling outliers**: Standardization is less sensitive to extreme values; outliers influence the mean and variance but don't squash the entire feature range

***Hint:*** *Use MinMaxScaler, StandardScaler, and train\_test\_split from sklearn.preprocessing and sklearn.model\_selection.*