# CSE555 – Deep Learning (Spring 2025)

# Homework #1

**Hand-in Policy**: Via Teams. Late submission policy as announced in the course syllabus.
**Collaboration Policy**: You are expected to do your own work. No collaboration is permitted.
**Grading**: This homework will be graded on the scale 100.

**Description**: Experiments with Deep Neural Networks.

You are expected to use the Python language and Keras library unless otherwise noted. You will prepare a report including your code and results (in a Jupyter Notebook). The expected report format is shown at the end of this document. You are encouraged to use Google Colab for computational needs to do this work.

**Part 1: Training a deep feed forward network for multidimensional regression.**

Assume that you are given a polynomial with multiple (8) inputs and multiple outputs (6) of the form:

$$(y_1, y_2, \dots, y_6) = \mathrm{P}(x_1, x_2, \dots, x_8),$$

where the exact polynomials are:

$$\begin{gathered}
y_1 = 2x_1x_3 - x_1x_5 + x_3x_8 + 2x_1^2x_8 + x_5 \\
y_2 = x_1x_5x_6 - x_3x_4 - 3x_2x_3 + 2x_2^2x_4 - 2x_7x_8 - 1 \\
y_3 = 2x_3^2 - x_5x_7 - 3x_1x_4x_6 + x_1^2x_2x_4 - 1 \\
y_4 = -x_6^3 + 2.\,1x_1x_3x_8 - x_1x_4x_7 - 3.2x_5^2x_2x_4 - x_8 \\
y_5 = x_1^2x_5 - 3x_3x_4x_8 + x_1x_2x_4 - 3x_6 + x_1^2x_7 + 2
\end{gathered}$$

Model a feed forward neural network (FFNN) such that:

$$(y_1, y_2, \dots, y_5) = \mathrm{FFNN}(x_1, x_2, \dots, x_8).$$

Training and validation data:

- Given the polynomials, generate $N_t$ instances of data pairs of the form: $\{(x_1^t, x_2^t, \dots, x_8^t), (y_1^t, y_2^t, \dots, y_5^t)\}$.
- When needed, add some noise to the training data from a normal distribution with mean $\mu$ and standard deviation $\sigma$ such that the training data becomes: $\{(x_1^t, x_2^t, \dots, x_8^t), (y_1^t + N(\mu,\sigma), y_2^t + N(\mu,\sigma), \dots, y_5^t + N(\mu,\sigma))\}$.
- Similarly, create an additional $N_v$ instances $\{(x_1^v, x_2^v, \dots, x_8^v), (y_1^v, y_2^v, \dots, y_5^v)\}$ for validation of the trained model.

Model training:

1. Choose $N_t$ and $N_v$ to be 1000 and 500 respectively.
2. In your training data add some noise to $y_i$'s from a normal distribution with $\mu = 0.0$ and $\sigma = 0.001$.
3. Build a feed forward network with exactly 3 hidden layers:
   - Each layer should include exactly 6 nodes in the beginning.

    - Use a combination of activation functions in these layers (use the same activation for each node at a given layer).
4. Define your loss function:
    - Use MSE for loss function.
5. Train your algorithm with SGD.
    - Use appropriate learning rates and the number of epochs.
    - Report on the training and validation errors.
6. Repeat Steps 2-4 with another set of activation functions (3 different combinations), learning rates (3 different schemes) and number of epochs (after finding a reasonable number of epochs in the first trial, increase by 50% for 2 times).
7. Choose your best parameters after Step 5.
8. Add new nodes at a time to each hidden layer:
    - Start from the first hidden layer, add two nodes, train, and record results.
    - Move to the second hidden layer, add two nodes, train, and record results.
    - Move to the third hidden layer, add two nodes, train, and record results.
    - Repeat Step 8 until bias and variance curve is drawn (remember the bias-variance curve discussed during lecture on March 11, 2025).
9. Increase $N_t$ by 10% and repeat Step 8.
10. Report all your results.

**Part 2: 2D Object Recognition using CNNs**

You are provided with a Python library that can generate objects of various shapes and sizes. This code is given in: [https://github.com/TimoFlesch/2D-Shape-Generator](https://github.com/TimoFlesch/2D-Shape-Generator)

You are asked to use the images generated by this function to train a CNN that recognizes the shape of the object in each image. You are asked to generate various images of the following shape classes:

**Oval**: Oval shaped (of varying shape, size, orientation, and location in the image).

**Rectangle**: Rectangular shaped (of varying shape, size, orientation, and location in the image).

**Triangle**: Triangular shaped (of varying shape, size, orientation, and location in the image).

**Poly 5**: Polygonal shapes with 5 sides (of varying shape, size, orientation, and location ..).

**Poly 6**: Polygonal shapes with 6 sides (of varying shape, size, orientation, and location …).

**Poly 7**: Polygonal shaped with 7 sides (of varying shape, size, orientation, and location …).

**Star 5**: A five-vertex star (of varying shape, size, orientation, and location …).

**Star 8**: An eight-vertex star (of varying shape, size, orientation, and location…).

You are expected to do the following:

1. Generate your data (online or offline) with 128x128 pixel images. Your data should have salt and pepper noise added to the images. Use only black-and-white (binary) colors.
2. Start with AlexNet. Change the input layer to handle binary images. Change the number of outputs to the right number of object classes.
3. Train the network with the data and report results.
    a. Use at least two different learning rate adjustment schemes.
    b. Use at least three different activation functions.

4. Change the network architecture.
    a. Change the number of nodes in the fully connected layer by 10% three times and repeat Step 3.
    b. Continuing with 15% of the nodes in the fully connected layer, remove the third layer from the output and repeat Step 3.
    c. Continuing with 20% of the nodes in the fully connected layer, remove the third and fourth layers from the output and repeat Step 3.

**What to hand in:** You are expected to hand in the following

- **Homework1_StudentNumber_LastName_FirstName.ipynb:** your Python Jupyter Notebook used to obtain the result provided in the report – along with any additional files needed to run your notebook).
- **Homework1_StudentNumber_LastName_FirstName.pdf:** your report in PDF format as explained below.
- You **MUST** follow this convention of submission file naming as some of your identities cannot be determined from your Teams profile.

The report format should be something like the following:

**Part 1: Model a deep feed forward network for regression.**

Code:

| |
|---|
| |

Results:

| |
|---|
| |

Comments and Discussions:

| |
|---|
| |

**Part 2: Object recognition using CNNs.**

Code:

| |
|---|
| |

Results:

| |
|---|
| |

Comments and Discussions:

| |
|---|
| |