**University of Central Missouri**

**Department of Computer Science & Cybersecurity**

**CS5720 Neural Networks and Deep Learning**

**Summer 2025**

**Home Assignment 1.**

**Student name: M. Siddartha Reddy**

**700774070**

**Submission Requirements:**

* 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 BrightSpace.
* 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.

1. **Tensor Manipulations & Reshaping**

**Task: Tensor Reshaping & Operations**

1. Create a random tensor of shape (4, 6).
2. Find its rank and shape using TensorFlow functions.
3. Reshape it into (2, 3, 4) and transpose it to (3, 2, 4).
4. Broadcast a smaller tensor (1, 4) to match the larger tensor and add them.
5. Explain how broadcasting works in TensorFlow.

**Expected Output:**

* Print rank and shape of the tensor before and after reshaping/transposing.

1. **Loss Functions & Hyperparameter Tuning**

**Task: Implement and Compare Loss Functions**

1. Define true values (y\_true) and model predictions (y\_pred).
2. Compute Mean Squared Error (MSE) and Categorical Cross-Entropy (CCE) losses.
3. Modify predictions slightly and check how loss values change.
4. Plot loss function values using Matplotlib.

**Expected Output:**

* Loss values printed for different predictions.
* Bar chart comparing MSE and Cross-Entropy Loss.

1. **Train a Neural Network and Log to TensorBoard**

**Task Description**

1. Load the MNIST dataset and preprocess it.
2. Train a simple neural network model and enable TensorBoard logging.
3. Launch TensorBoard and analyze loss and accuracy trends.

**Expected Outcome:**

* The model should train **for 5 epochs** and store logs in the "logs/fit/" directory.
* You should be able to visualize **training vs. validation accuracy and loss** in TensorBoard.

**4.1 Questions to Answer:**

1. **What patterns do you observe in the training and validation accuracy curves?**

**Training Accuracy Curve:**

Usually shows a steady upward trend as the model continues to learn and minimize the training loss.

Reflects how well the model is fitting to the training data.

**Validation Accuracy Curve:**

Initially follows a similar upward trend as training accuracy.

May plateau or even decline if the model starts overfitting—i.e., it memorizes training data but performs poorly on unseen data.

**Typical Pattern:**

A gap between training and validation accuracy suggests the model is learning training data well but not generalizing to validation data

1. **How can you use TensorBoard to detect overfitting?**

TensorBoard provides visual cues that can help detect overfitting:

**Compare training vs. validation loss:**

If training loss keeps decreasing while validation loss starts increasing, the model is likely overfitting.

**Accuracy curves:**

A large gap between training and validation accuracy where training accuracy is high but validation accuracy is stagnant or dropping is a strong indicator of overfitting.

**Use of TensorBoard features:**

Zoom in on specific epochs to observe the divergence.

Monitor learning curves in real time during training.

1. **What happens when you increase the number of epochs?**

**Early Epochs:**

The model learns useful patterns from the data.

Both training and validation accuracy usually improve.

**More Epochs:**

Training accuracy may continue to improve.

Validation accuracy may plateau or degrade due to overfitting.

**Too Many Epochs:**

The model may memorize the training data.

Validation performance decreases, indicating poor generalization.

**Solution:**

Use early stopping or regularization (like dropout) to prevent overfitting when training for many epochs.