Deep Learning With Tensor Flow 1 (CSE 3793)

ASSIGNMENT-2: NEURAL NETWORK FOR REGRESSION AND CLASSIFICATION

Instruction:

- **Submit one well-structured notebook per task.**
- **Provide brief discussion (2-5 lines) for each experiment and final instruction.**

1. Write a Python code to build a TensorFlow-Keras model (Sequential API) to implement a single-layered perceptron (one Dense unit) that predicts house price from area.

- Follow the given instructions:

   a. Import the given packages: numpy, pandas, matplotlib, and from Keras import via tensorflow and tensorflow.keras.

   b. Generate a synthetic data set for 200 samples for area values from a Normal distribution with mean = 0 and std = 1, scale the std to 2.5, and shift the mean by 25.

   – area = 2.5 * random(200) + 25 (use appropriate NumPy function).

   – Create the target value price = 25 * area + noise, where noise is integer noise in [20, 49].

   c. Normalize both features (area) and target (price) using Min–Max normalization.

   d. Build and train the model using Sequential with a single Dense layer (units=1). For training, the loss is set to mean squared error, the optimizer is SGD, the batch size is 32, the validation split is 0.2, and the model is run for 100 epochs.

   e. Report the training and validation MSE for all 100 epochs. Predict the price for a given area (e.g., 28.0) and plot Actual vs Predicted prices.

2. Write a Python code to build a regression model in TensorFlow–Keras to predict MPG (fuel efficiency) from the UCI Auto MPG dataset using: Keras Normalization layer for input standardization (mean 0, std 1), a feed-forward network with two hidden layers (ReLU), and Adam optimizer and MSE loss.

- Follow the given instructions:

   a. Import the given packages: numpy, pandas, matplotlib, seaborn and Keras modules (Dense, Normalization).

   b. Load the Auto MPG dataset from the UCI repository: URL: https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data

   – Column names as ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', '$model\_year$', 'origin']

   – Treat '?' as NaN; drop rows with NaNs.

   – Drop the origin column; keep 7 input features.

   c. Split the data into 80

   d. Visualize relationships with Seaborn pairplot (features + mpg).

   e. Separate inputs (features) and label (mpg).

f. Normalize inputs using Keras Normalization (fit only on training features).
g. Build a Keras model with Normalization with an input layer dense layer and an output layer
h. Compile and train and train the model by setting the following parameters:
– optimizer='adam', loss='mean_squared_error'
– epochs=100, validation_split=0.2
i. Plot the training vs validation loss curves.
j. Evaluate on test set and Predict mpg for test features, also Plot True vs Predicted (diagonal reference line)

3. Write a Python code to build a logistic regression classifier for MNIST digits using TensorFlow–Keras.

- Use Flatten to convert 28×28 images to 784-D vectors.
- Normalize the images to [0,1][0,1][0,1].
- Train a single dense layer model (multiclass logistic regression).
- Use the sparse Categorical Cross-Entropy Loss and Adam Optimizer.
- Train for 50 epochs with validation_split=0.2.
- Plot training vs. validation loss.
- Visualize a test image and the predicted class probabilities.
- Follow the given instructions:

a. Import the given packages: numpy, pandas, matplotlib, and from Keras import via tensorflow and tensorflow.keras as K and from Keras Dense, Flatten.
b. Load the MNIST dataset as tf.keras.datasets.mnist.load_data().
c. Preprocess the data where scale pixels by 255 for floats in [0,1][0,1][0,1] and labels as int32.
d. Build model the sequential model as Sequential([Flatten(input_shape=(28,28)), Dense(10, activation='softmax')]).
e. Compile the model by providing the optimizer as'adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'].
f. Train the model for 50 epochs and set validation_split to 0.2.
g. Plot the loss curves.
h. Predict on test set; for an index i, show (a) the image with predicted vs true label and confidence, and (b) a bar chart of predicted probabilities.