Instructor: Alina Vereshchaka

Assignment 2

Autoencoder and Transformer Architectures

Checkpoint: April 2, Tue, 11:59 pm

Due date: April 11, Thu 11:59pm

Welcome to Assignment 2! This assignment focuses on developing both theoretical and practical skills related to autoencoders and transformer networks. It consists of four theoretical and hands-on exercises where you will derive and analyze network setups, work with various datasets, and implement, train, and adjust different neural network architectures. Libraries usage:

- You can use inbuilt functions or define some functions from scratch.
- All libraries are allowed, except those with pre-trained models or predefined architectures. Submissions with pre-trained models or predefined architectures will not be considered.

Part I: Theoretical Part – Autoencoders for Anomaly Detection in Manufacturing [10 points]

In manufacturing industries, ensuring the quality of products is important. Detecting anomalies or defects in the production process can be challenging, especially in large-scale operations. Traditional methods often rely on manual inspection or rule-based systems, which can be time-consuming and prone to human error.

Scenario: A manufacturing company “ElectroGuard Innovations” produces electronic components, and they want to improve their quality control process by implementing an automated anomaly detection system. The company collects sensor data from the production line, which includes various parameters such as temperature, pressure, and voltage readings. Anomalies in these readings could indicate potential defects in the components.

Objective: To develop an automated anomaly detection system using an autoencoder neural network to identify defects in the manufacturing process.

Autoencoder Architecture:

- Input layer: 1000-dimensional vectors
- Hidden layer 1: Fully connected layer with 512 units and ReLU activation
- Bottleneck layer: Fully connected layer with 32 units
- Hidden layer 2: Fully connected layer with 512 units and ReLU activation

- Output layer: Fully connected layer with 1000 units and sigmoid activation
- Autoencoder is trained using MSE loss

TASKS:

1. Calculate the total number of parameters in the autoencoder, including weights and biases.
2. Generate a computational graph for the autoencoder.
3. Discuss potential challenges and limitations (at least 4) of using autoencoders for anomaly detection in manufacturing.
4. Propose potential improvements or extensions (at least 4) to the system to enhance its effectiveness in detecting anomalies.

In your report for Part I:

- Provide all the answers to questions in Tasks.
- You may provide the answer using handwritten notes, typed in the MS Word or Latex.
- The submitted version has to be converted to a single pdf and named as

  UBIT TEAMMATE1_TEAMMATE2_assignment2_part1.pdf
  e.g., avereshc_atharvap_assignment2_part1.pdf

  Or it can be combined with your report for this assignment.

Part II: Autoencoders for Anomaly Detection [30 points]

In this part of our assignment, we will implement autoencoder and explore it's application for anomaly detection. **NOTE: The min accuracy to be achieved for this task is 80%.**

DATASETS

Below is a list of datasets. Select ONE dataset based on your preference.

- Yahoo S5 Dataset: real server logs from a popular web service, containing various anomalies such as spikes, dips, and shifts in user traffic
- Hard Drive Test Data: Daily Snapshot of Each Operational Hard Drive
- Numenta Anomaly Benchmark: dataset for evaluating anomaly detection algorithms, featuring a variety of time-series data from different domains, including financial market data.

STEPS:

1. Select one dataset from the list above. Briefly justify your choice in the report (e.g., relevance to your field of study)
2. Data exploration and preprocessing:
    - Read, preprocess, and print the main statistics about the dataset
    - Clean and prepare the data for modeling (e.g. handling missing values, normalization, feature engineering).
    - Use libraries like matplotlib, seaborn, or plotly to create at least 3 informative visualizations that reveal insights about the data and potential anomalies (e.g., histograms, time-series plots, scatter plots).
    - Prepare the dataset for training. Divide the preprocessed data into training, validation, and testing sets.
3. Autoencoder model building:
    - Implement a standard Autoencoder or a Variational Autoencoder (VAE) for anomaly detection.
    - Experiment with architectures. Build and train at least 3 different autoencoder architectures for anomaly detection. Consider experimenting with:
        - Different layer types (Dense, LSTM for time series, Conv1D for sequential data)
        - Number of hidden layers and units
        - Activation functions (ReLU, sigmoid)
4. Model evaluation and training:
    - Train each autoencoder architecture using the training data and validation data for monitoring performance.
    - Choose appropriate evaluation metrics (e.g., reconstruction error, precision-recall for anomaly classification.
    - Compare the performance of the different architectures on the validation set. Discuss which architecture performs best and why (consider factors like reconstruction error and anomaly detection accuracy).
5. Save the weights of the trained network that provides the best results. Saving and loading models (PyTorch)
6. Discuss the results and provide relevant graphs:
    a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.
    b. Plot the training and validation accuracy over time (epochs).
    c. Plot the training and validation loss over time (epochs).
    d. Generate a confusion matrix using the model's predictions on the test set.
    e. Calculate and report other evaluation metrics such as Precision, recall and

F1 score (more details). You can use sklearn.metrics.precision_recall_fscore_support

In your report for Part II:

1. Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? How many entries and variables does the dataset comprise?
2. Describe the details of your autoencoder models, including the layers, activation functions, and any specific configurations employed.
3. Discuss the results and provide relevant graphs:
   a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.
   b. Plot the training and validation accuracy over time (epochs).
   c. Plot the training and validation loss over time (epochs).
   d. Generate a confusion matrix using the model's predictions on the test set.
   e. Report any other evaluation metrics used to analyze the model's performance on the test set.
4. Discuss the strengths and limitations of using autoencoders for anomaly detection.

Part III: Theoretical Part – Transformers [20 points]

In this part of our assignment, we explore the theoretical foundation behind Transformers.

TASKS:

1. Break down the mathematical operations involved in self-attention. Explain how the input sequence $x = (x_1, x_2, \ldots, x_N)$ is processed through these stages:

   - **Linear transformations.** Describe how three linear transformations project the input sequence into:

     - **Query (q):** this vector plays the role of "asking questions" about the sequence elements.
     - **Key (k):** this vector helps determine which elements in the sequence are relevant for answering the queries.

     Value (v): this vector contains the actual information from each element in the sequence. Briefly explain how these transformations are typically performed using weight matrices (denoted by W_q ,W_k,and W_v) and bias vectors (optional).

- **Scaled dot-product attention.** Explain how the model calculates attention scores using the query (q_i) and key (k_j) vectors. Include the formula for this step and discuss the role of the square root of the key vector dimension (d_k) in the normalization process.
- **Weighted sum.** Describe how the calculated attention scores are used to weight the value vectors (v_j). Essentially, this step focuses on the relevant elements in the sequence based on the attention scores. Explain the mathematical formula for this weighted sum.
- **Optional output transformation:** Explain the purpose of the optional final linear transformation (W_o) and bias term (b_o) in generating the latent representation z. How does this step potentially impact the final representation?

2. Draw the computational graph that depicts the flow of data through the self-attention mechanism. Include all the transformations mentioned in Question 1.

In your report for Part III:

- Provide all the answers to questions in Tasks.
- You may provide the answer using handwritten notes, typed in MS Word or Latex.
- The submitted version has to be converted to a single pdf and named as

  UBIT TEAMMATE1_TEAMMATE2_assignment2_part3.pdf
  e.g., avereshc_atharvap_assignment2_part3.pdf

  Or it can be combined with your report for this assignment.

Part IV: Build Transformer with PyTorch [40 points]

In this part, we will implement a Transformer model from scratch using the PyTorch framework. We will train the model on a provided dataset, exploring various optimization techniques and hyperparameter tuning to achieve optimal performance. **NOTE: The min accuracy to be achieved for this task is 80%.**

DATASETS

List of Datasets. Select ONE dataset based on your preference.

STEPS:

Step 1: Data Exploration and Preprocessing

1. Select one dataset from the list above. Briefly justify your choice in the report
2. Data exploration:
   - Read, preprocess, and print the main statistics about the dataset
   - Use libraries like matplotlib, seaborn, or plotly to create at least 3 informative visualizations that reveal insights about the data and potential anomalies (e.g., polarity distribution, word count distribution, vocabulary size etc).
3. Text preprocessing:
   - Text cleaning. Remove punctuation, stop words, and unnecessary characters.
   - Text lowercasing. Ensure all text is lowercase for consistent representation.
   - Tokenization. Break down the text into individual words (or tokens). Explore libraries like `nltk` or spaCy for tokenization functionalities. You can also use keras tokenizer or Pytorch tokenizer.
   - Vocabulary building. Create a vocabulary containing all unique tokens encountered in the dataset.
   - Numerical representation. Convert tokens into numerical representations using techniques like word embedding (e.g., Word2Vec, GloVe).

Step 2: Model Construction

1. Embeddings and positional encoding. Define an embedding layer to map tokens into numerical vectors. If using pre-trained embeddings, ensure they are compatible with your model's input dimension.
2. Implement the core Transformer architecture:
   - Encoder: Utilize *nn.TransformerEncoder* with multiple *nn.TransformerEncoderLayer* instances. Each layer typically comprises a multi-head self-attention mechanism, a feed-forward layer, and layer normalization.
   - Decoder: Employ *nn.TransformerDecoder* with multiple *nn.TransformerDecoderLayer* instances. These layers incorporate masked self-attention, multi-head attention over the encoder outputs, and a feed-forward layer with layer normalization.
3. Depending on your task (e.g., classification, sequence generation), define an appropriate output layer. For classification tasks, you might use a linear layer with a softmax activation function.

Step 3: Training the Transformer

1. Preparing for training:
   - Divide the preprocessed data into training, validation, and testing sets using a common split ratio (e.g., 70:15:15 or 80:10:10).

- Choose an appropriate loss function (e.g., cross-entropy loss for classification) and an optimizer (e.g., Adam) to update model parameters during training.

2. Training loop:
    - Forward pass. Pass the input data through the Transformer model to generate predictions.
    - Calculate loss. Compute the loss between predictions and true labels using the chosen loss function.
    - Backward pass. Backpropagate the loss to calculate gradients for each model parameter.
    - Update parameters. Utilize the optimizer to update model parameters based on the calculated gradients.

Step 4: Evaluation and Optimization

1. After each training epoch, assess the model's performance on the validation set. Monitor metrics like accuracy or loss to track progress.
2. Explore at least 3 optimization techniques to improve the performance of your Transformer model. E.g.:
    - Regularization (L1/L2): Penalize large model weights to prevent overfitting.
    - Dropout: Randomly drop out neurons during training to introduce noise and reduce overfitting.
    - Early stopping: Halt training when validation performance plateaus to prevent overtraining.
    - Learning rate tuning: Experiment with different learning rates to find the optimal value for convergence and performance.
3. Save the weights of the model that provides the best results. Check the saving and loading of models (Pytorch).
4. Discuss the results and provide the following graphs:
    - Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.
    - Plot the training and validation accuracy over time (epochs).
    - Plot the training and validation loss over time (epochs).
    - Generate a confusion matrix using the model's predictions on the test set.
    - Calculate and report other evaluation metrics such as Precision, recall and F1 score (more details). You can use sklearn.metrics.precision_recall_fscore_support
    - Plot the ROC curve.

In your report for Part IV:

1. Describe the Transformer architecture you have defined.
2. Describe how the techniques (regularization, dropout, early stopping) have impacted the performance of the model.
3. Discuss the results and provide the relevant graphs:
    - Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.
    - Plot the training and validation accuracy over time (epochs).
    - Plot the training and validation loss over time (epochs).
    - Generate a confusion matrix using the model's predictions on the test set.
    - Provide the Precision, recall and F1 score.
    - Plot the ROC curve

Bonus points [max 10 points]

Vision Transformer (ViT) for Image Classification [7 points]

Implement a vision transformer paper. to solve a problem defined in Assignment 1, Part I. Compare the results with VGG and ResNet.

EfficientNet [3 points]

Use EfficientNet to solve a problem defined in Assignment 1, Part I. Compare the results with VGG and ResNet. You can use pre-defined EfficientNet models.

Bonus part submission:

- Create a separate zip folder named as UBIT TEAMMATE1_TEAMMATE2_assignment2_bonus.zip e.g., avereshc_ankithba _ assignment2_bonus.
- Include all the files needed in the folder.
- Report is not required, you can include all the comments as part of your Jupyter Notebook/Script.

ASSIGNMENT STEPS

1. Submit checkpoint (April 2)

- Complete Part I, II and III of the assignment
- For the checkpoint, it is ok if your report is not fully completed. You can finalize it for the final submission.
- Include all the references at the end of the report. There is no minimum report size requirement, just make sure it includes all the information required.
- Do not include a dataset as part of your submission
- Add all your assignment files in a zip folder including .ipynb files, draft report as pdf, part V.2 as a pdf and saved weights as a zip folder. Name zip folder with all the files as: UBIT TEAMMATE1_TEAMMATE2_assignment2_checkpoint.zip (e.g., avereshc_ atharvap_assignment2_checkpoint.zip).
- Submit at UBLearns > Assignments

2. Submit final results (April 11)

- Fully complete all parts of the assignment and submit to UBLearns > Assignments
- In your report or Jupyter notebook include all the references that were used to complete the assignment or the task. You can include that at the end of the report.
- Your Jupyter notebook should be saved with the results. If you are submitting python scripts, after extracting the ZIP file and executing command python main.py in the first level directory, all the generated results and plots you used in your report should appear printed out in a clear manner.
- **Add .txt file “dataset_assignment2_part4.txt” and include a link that points to the dataset you have selected. Do not submit the dataset itself.**
- Report (as a pdf file): combine the reports for all parts into one pdf file named as

  UBIT TEAMMATE1_TEAMMATE2_assignment2_report.pdf
  e.g., avereshc_atharvap_assignment2_report.pdf
- Theoretical Part (Part I and III) can be combined with a final report or submitted as a separate file named as
  UBIT TEAMMATE1_TEAMMATE2_assignment2_part_1_and_3.pdf
  e.g., avereshc_atharvap_assignment2_ part_1_and_3.pdf
- Code (as ipynb file with saved outputs): separate file for Part II and Part IV named as
  UBIT TEAMMATE1_TEAMMATE2_assignment2_part_2.ipynb
  UBIT TEAMMATE1_TEAMMATE2_assignment2_part_4.ipynb
- Pickle or .h5 files with saved models that generate the best results for your model for each parts named as

  UBIT TEAMMATE1_TEAMMATE2_assignment2_part_#.h5
  e.g., avereshc_atharvap_assignment2_ part_2.h5
- For Bonus part, create a separate folder named as UBIT TEAMMATE1_TEAMMATE2_assignment2_bonus.zip

- All assignment files and bonus folder should be packed in one ZIP file named: UBIT TEAMMATE1_TEAMMATE2_assignment2.zip

  e.g. avereshc_ atharvap_ assignment2.zip
- Submit your ZIP file at UBlearns > Assignments
- Verify that the submission was successful, e.g., download the submitted files and verify they are all correct. Only files submitted on UBlearns are considered for evaluation.
- You can make unlimited number of submissions and only the latest will be evaluated

Notes:

- Ensure that your code follows a clear structure and contains comments for the main functions and specific attributes related to your solution. You can submit multiple files, but they all need to be labeled with a clear name.
- Recheck the submitted files, e.g. download and open them, once submitted and verify that they open correctly

Assignments Grading

Checkpoint submission:

- Graded based on the 0/1 grade.
  - “1” is assigned, if more than 80% of the work has been fully completed for the checkpoint-related parts. “0” is assigned for all other cases
- Obtaining “0” for a checkpoint submission results in a fixed 30 points loss from the final assignment evaluation.

Final submission:

- Graded based on the X out of 100 points + bonus [if applicable]
- During the final evaluation, all the parts are evaluated, so please include a final version of all the parts of the assignment in your final submission.

Notes:

- Only files submitted on UBlearns are considered for evaluation.
- Files from local device/GitHub/Google colab/Google docs/other places are not considered for evaluation

- We strongly recommend submitting your work in-progress on UBlearns and always recheck the submitted files, e.g. download and open them, once submitted

Academic Integrity

The standing policy of the Department is that all students involved in any academic integrity violation (e.g., plagiarism in any way, shape, or form) will receive an F grade for the course. The catalog describes plagiarism as “Copying or receiving material from any source and submitting that material as one’s own, without acknowledging and citing the particular debts to the source, or in any other manner representing the work of another as one’s own.”. Refer to the Office of Academic Integrity for more details.

Important Information

This assignment can be done individually or in a team of up to two students. The grading rubrics are the same for a team of any size.

- No collaboration, cheating, and plagiarism is allowed in assignments, quizzes, the final exam or final project.
- All the submissions will be checked using SafeAssign as well as other tools. SafeAssign is based on the submitted works for the past semesters as well the current submissions. We can see all the sources, so you don't need to worry if there is a high similarity with your Checkpoint submission.
- The submissions should include all the references. Kindly note that referencing the source does not mean you can copy/paste it fully and submit as your original work. Updating the hyperparameters or modifying the existing code is a subject to plagiarism. Your work has to be original. If you have any doubts, send a private post on piazza to confirm.
- All group members and parties involved in any suspicious cases will be officially reported using the Academic Dishonesty Report form. What does that mean?
    - In most cases, the grade for the assignment/quiz/final project/midterm will be 0 and all bonus points will be subject to removal from the final evaluation for all students involved.
    - Final grade reduction (e.g. if you obtain B, that will result in B-)
    - Those found violating academic integrity more than once throughout their program will receive an immediate F in the course.

  Please refer to the Academic Integrity Policy for more details.
- The report should be delivered as a separate pdf file. You can combine report for Part I, Part II, Part III and Part IV into the same pdf file. You can follow the NIPS template as a report structure. You may include comments in the Jupyter

Notebook; however, you will need to duplicate the results in a separate pdf file.

- All the references can be listed at the end of the report. There is no minimum requirement for the report size, just make sure it includes all the information required.
- For the Bonus part, no report is needed.

Late Days Policy

You can use up to 5 late days throughout the course that can be applied to any assignment-related due dates. You do not have to inform the instructor, as the late submission will be tracked in UBlearns.

Important Dates

April 2, Tue, 11:59 pm - Checkpoint Submission is Due

April 11, Thu, 11:59 pm - Final Submission is Due