In the name of God

University of Tehran
School of Electrical and Computer Engineering

Neural Networks and Deep Learning

Homework 4

Question 1	
Designer TA Name	Kiana Hooshanfar
Email	k.hooshanfar02@gmail.com
Question 2	
Designer TA Name	Ehsan Forootan
Email	ehsan.forootan@ut.ac.ir
Submission Deadline	2025/05/25

Contents

1 Rules **3**

2 Question 1: Image Captioning with a Hybrid ResNet50 + LSTM-GRU Network **4**
2.1 Introduction . . . 4
2.2 Dataset and Pre-processing (30 points) . . . 4
2.2.1 Dataset Selection . . . 4
2.2.2 Image Pre-processing . . . 4
2.2.3 Text (Caption) Pre-processing . . . 4
2.2.4 Data Splitting . . . 5
2.3 Model Implementation (30 points) . . . 5
2.3.1 Encoder Implementation: . . . 5
2.3.2 Decoder Implementation: . . . 5
2.3.3 Encoder and Decoder Integration (End-to-End Model) . . . 5
2.4 Training and Evaluation Model . . . 5
2.4.1 Training (15 points) . . . 5
2.4.2 Model Evaluation (25 points) . . . 5
2.4.3 Bonus (5 points) . . . 6

3 Question 2: Time Series Prediction for Clinical Event **7**
3.1 Introduction . . . 7
3.2 Methodology . . . 7
3.3 Data Preparation and Statistical Analysis . . . 7
3.4 Training Deep Learning Models . . . 8
3.5 Plotting Results and Analyzing Answers . . . 8
3.6 Maximum Log-Likelihood Estimation Method . . . 8

1 Rules

Before answering the questions, please read the following points carefully:

- Prepare a report of your answers in the format provided on the course page on the Elearn system, named `REPORTS_TEMPLATE.docx`.
- It is recommended to do the assignments in groups of two. (More than two people are not allowed, and individual submissions do not receive extra credit). Note that there is no requirement for group members to remain the same throughout the semester. (i.e., you can do the first assignment with person A and the second assignment with person B, etc.).
- The quality of your report is of great importance in the grading process; therefore, please mention all the points and assumptions you considered in your implementations and calculations in the report.
- In your report, according to what is provided in the sample template, include captions for figures and tables.
- It is not necessary to provide detailed explanations of the code in the report, but the results obtained from it must be reported and analyzed.
- Analysis of the results is mandatory, even if not explicitly asked for in the question.
- The teaching assistants are not obliged to run your codes; therefore, any results or analysis requested in the question must be clearly and completely presented in the report. Failure to comply with this will result in a deduction from your grade.
- Codes must be prepared in a notebook with the `.ipynb` extension. At the end of the work, all the code must be run, and the output of each cell must be saved in this submitted file. For example, if the output of a cell is a plot that you have included in your report, this plot must also be present in the code notebook.
- In case of plagiarism, the score of all involved individuals will be set to -100.
- The only authorized programming language is **Python**.
- Using pre-written code for the assignments is strictly forbidden. If two groups use a common source and submit similar codes, it will be considered cheating.
- The late submission policy is as follows: After the submission deadline, you have up to one week to submit with a penalty. After this one week, your grade for that assignment will be zero.
  - First three days: No penalty.
  - Day 4: 5% penalty.
  - Day 5: 10% penalty.
  - Day 6: 15% penalty.
  - Day 7: 20% penalty.
- The maximum score that can be obtained for each question is 100. If the total points for a question exceed 100, and you score more than 100, it will not be applied.
  - For example, if your score on question 1 is 105 and on question 2 is 95, your final assignment score will be 97.5, not 100.
- Please place the report, codes, and other attachments in a folder with the following name, compress it, and then upload it to the Elearn system:
  `HW[Number]_[Lastname]_[StudentNumber]_[Lastname]_[StudentNumber].zip`
  (Example: `HW1_Ahmadi_810199101_Bagheri_810199102.zip`)
- For groups of two, submission by one member is sufficient, but it is recommended that both members upload the assignment.

2 Question 1: Image Captioning with a Hybrid ResNet50 + LSTM-GRU Network

2.1 Introduction

Image captioning is a branch of computer vision and machine learning that gives systems the ability to automatically generate textual descriptions for images. In this approach, deep learning models—usually Convolutional Neural Networks (CNNs)—first extract the visual features of an image. Then, with the help of Recurrent Neural Networks (RNNs) or transformer-based architectures, these features are converted into fluent and understandable sentences. Applications of this technology include improving accessibility for visually impaired individuals to visual content, intelligent image search, and content analysis in social networks.

In this question, we intend to generate textual captions for images from the Flickr8k dataset. For this purpose, we will use different Encoder-Decoder structures, where in the first stage, visual features of the image are extracted using pre-trained models, and in the second stage, these feature vectors are converted into meaningful textual descriptions.

2.2 Dataset and Pre-processing (30 points)

2.2.1 Dataset Selection

First, download the Flickr8k dataset from its respective website. Display a few sample images along with their corresponding captions to familiarize yourself with the data structure and file format.

2.2.2 Image Pre-processing

- **Resizing:** Resize all images to a fixed dimension (e.g., 224x224 pixels) to be compatible with the input of your CNN model.
- **Normalization:** Scale the pixel values so that their distribution has a mean close to zero and a standard deviation of one. This helps in making the learning of visual features by the neural network more stable and faster.

2.2.3 Text (Caption) Pre-processing

- **Lowercase:** Convert all letters to lowercase to resolve case-sensitivity issues.
- **Remove Punctuation:** Remove punctuation marks, unnecessary symbols, and irrelevant numbers.
- **Tokenization:**
  - Map each word to a unique numerical ID.
  - Create a dictionary that assigns the entire vocabulary of your dataset to their corresponding numbers.
  - Add special tokens like `<sos>` (start of sentence), `<eos>` (end of sentence), `<pad>` (padding), and `<unk>` (unknown words) to this dictionary and explain the purpose of each.
  - Save this dictionary as a JSON file to be used as a tokenizer in subsequent steps.
- **Uniform Caption Length:**
  - To facilitate computations in the decoder, set the length of all captions to a fixed value (e.g., 20 words).
  - If a caption is shorter, pad it with the `<pad>` token to the desired length.
  - Explain why padding is necessary for model training (all inputs must have the same dimensions).

2.2.4 Data Splitting

Divide the dataset into three parts with the following ratios: 80% for training (Train), 10% for validation (Validation), and 10% for testing (Test). Ensure that the splitting is done based on images, meaning no image should appear in more than one set.

2.3 Model Implementation (30 points)

CNN-RNN models are among the most common deep learning methods for multimedia tasks like generating textual descriptions for images. In this architecture, a Convolutional Neural Network (CNN) is first used as an encoder to extract visual features from the image. Then, a Recurrent Neural Network (RNN) or more advanced variants like LSTM or GRU act as a decoder, taking the feature vectors and generating descriptive sentences. In this section, we aim to implement a model for the image captioning problem, inspired by the designs and parameters presented in this article.

2.3.1 Encoder Implementation:

Use a pre-trained CNN model (ResNet50). Remove the final Fully Connected layer so that instead of classifying, you can extract the image feature vector. Examine and note the dimensions of the output feature vector.

2.3.2 Decoder Implementation:

- Use an Embedding layer to map words to continuous vectors. (Embeddings create a more compact vector space compared to One-hot encoding and better preserve the semantic relationships between words).
- Implement the decoder structure according to the paper (LSTM-GRU) to generate a sequence of words.
- (Pass the image feature vector as the initial hidden state to the LSTM. At the end, use a Linear layer along with a Softmax function to predict the next token (word)).

2.3.3 Encoder and Decoder Integration (End-to-End Model)

Define a custom class named `HybridModel` that incorporates both the Encoder and Decoder. The implementation should be such that it first passes the image through the Encoder and then uses its output to generate a text sequence with the Decoder. How do we turn the encoder and decoder into an end-to-end model that can be trained?

2.4 Training and Evaluation Model

2.4.1 Training (15 points)

Use an appropriate loss function to calculate the error and train the model according to the training parameters (learning rate, batch size, number of epochs, and other settings) suggested in the paper.

2.4.2 Model Evaluation (25 points)

- Plot the training and validation loss over each epoch.
- Compare the Greedy Search and Beam Search algorithms. Explain which one provides better quality.
- Calculate and compare the results generated by Greedy Search and Beam Search algorithms. Then, select five images and display the captions generated by both methods for each image.
- Identify and analyze a few examples of model errors (e.g., cases where the model incorrectly identifies objects or fails to understand the relationships between them).

2.4.3 Bonus (5 points)

Research common evaluation metrics for caption generation models. Then, briefly explain the functionality of BLEU Score (BLEU-1 to BLEU-4) based on n-grams. In the next step, calculate the BLEU-1 to BLEU-4 values on the test set and report the results in a table with different model configurations (Greedy vs. Beam Search).

3 Question 2: Time Series Prediction for Clinical Event

3.1 Introduction

Successfully modeling complex multivariate time series and their ability to predict future events is crucial for various applications in science, engineering, and business. In clinical settings, the ability to predict future events for a patient based on past clinical observations, such as previous prescriptions, lab results, or past physiological signals, can help predict a wide range of future events. This allows healthcare professionals to intervene before an event occurs or to prepare the necessary resources to deal with it. In this question, using various methods, we will try to create a good model for predicting blood markers. For this purpose, the MIMIC-III dataset and a specific part of it, related to blood tests of patients, have been selected from here. Also, for the general methodology of time series prediction, the following article has been used: Lee_Hauskrecht_AIME_2019.pdf.

3.2 Methodology

a) How is time series prediction generally performed? Name different models in time series prediction and, in one line, explain the general method for each with the help of a formula or a diagram. (5 points)

b) How is the State-Space Markov Event prediction method generally performed? Design a Dense network for it and provide a summary diagram of the model. Is the use of layers like Normalization allowed? Which Activation Function should be used? For the number of hidden neurons, use the settings from the paper (W equal to 128). Also, select the Learning Rate from the paper. For the Batch-size, use values of 32, 16, or 1. (To obtain the input and output sizes, refer to the statistical analysis section). (5 points)

c) How do LSTM-based event prediction methods work? Describe the method and its Loss-function and explain a Forward pass of the prediction. (5 points)

d) What is the general structure of a Bi-Directional LSTM? What advantage does this model layer offer over a regular LSTM layer? Create 4 networks with the architectures GRU, Bi-Directional GRU, LSTM, and Bi-Directional LSTM. Note that all models for predicting the next moments require at least one Fully-Connected layer in their Head. Use hyperparameter settings similar to part b. (5 points)

3.3 Data Preparation and Statistical Analysis

a) The Train, Validation, and Test data have been provided to you in CSV format. Read them and show the first 5 rows. The 'Hour' column indicates the hour and represents the progress of the time series. Obtain the Correlation matrix for the Validation data. Then select the columns with the lowest correlations (e.g., below 5 percent) and show them in a plot. Finally, select the 20 columns with the lowest correlation. Why should the lowest correlation be chosen? In the end, group the data based on the patient number. This means that each patient will have a time series. (5 points)

b) Show the time series of one patient in the 'HR' column throughout Train, Validation, and Test. Take the discrete derivative of this time series once. Run the ADF test from the Statmodels library on it. Is the probability less than 5 percent? Continue the process of testing and differentiating until the probability is less than 5 percent. This number of times will be your d. (5 points)

c) Then plot two diagrams, Autocorrelation and Partial AutoCorrelation. Consider the first point of sharp decline in the two diagrams as AR and MA. Then, fit a SARIMAX model with an external input as the 'O2Sat' column. Fit the model, make a time prediction, and plot it on the main 'HR' diagram. (5 points)

d) Report the R2-score values for the test set. Also, compare and analyze the prediction and actual plots from the previous part. At the end, normalize all the data with MinMax. Be careful that the hour column should be normalized separately. Also, be careful to use the normalizer of the Validation and Test columns from the normalizer used for training. Why are normalizers based on mean and variance not used here? (5 points)

3.4 Training Deep Learning Models

a) The data from the previous section should be windowed. One Window represents the length of the time series that the model sees, and the other Window represents the length of time that must be predicted. How can these two lengths be approximately determined from the previous section? Report the two window lengths. Note that for training in this section, the 20 columns with the least Correlation should be used. Also, explain the Explained Variance Score metric. Why might this metric be suitable for time series? (5 points)

b) Train the Dense, LSTM, and GRU models on the training data. In a plot, report the loss function value for training and validation for each Epoch. Note that the time series of each patient is considered a data group, and you have time series for the number of patients. (10 points)

c) Train the Bi-Directional models and repeat the plot from the previous section. (10 points)

d) In a table, report the values of Loss, MSE, MAE, R2-score, and Cosine-Distance for all models. Which is the best model in terms of R2-Score? Is the best model simply the one that reports the best average as a better prediction? (5 points)

3.5 Plotting Results and Analyzing Answers

For each of the 5 deep learning models, plot the predictions over time for the 'HR' column. Which models performed better than SARIMAX? Analyze the results on both the metrics and the trends for each of the 6 models. (15 points)

3.6 Maximum Log-Likelihood Estimation Method

a) Suppose a hospital's financial management system needs patient statistical information and prediction of these statistics. This system only requires a 2nd-degree approximation of the distribution function at each moment. Therefore, the hypothesized distribution function is designed as follows:

$$\begin{aligned} \mu(t) &= E\{X(t)\} \\ \sigma(t,s) &= E\{X(t)X(s)\} \\ P(X(t)=X) &= f(\mu(t),\sigma(t)) \end{aligned}$$

The goal of this method is to minimize the difference between the predicted value P by the network and the actual P of the output window. Design an LSTM-based network that receives an input Window, the mean and variance values at each moment for each column, and outputs the mean and variance values at each moment for each column. Why is it correct to separate the variances and means for each column and assume them to be independent? Can the previous structures be used? Can we still benefit from the Loss-Function of the previous sections? Do we need to change the Activation Function in the last layer? (5 points)

b) With the help of the network, train the data. Display the Loss-Function results during training for the training and Validation data together in a plot. Report the R2-score value for the test data. (10 points)

c) Add noise with a Laplace distribution to the data. Using the Bi-LSTM network from section 4, predict the time series for one patient. Then plot the mean and variance for this prediction. (Also use the model from section 2-6-b to predict the mean and variance over time and place it next to the Bi-LSTM results and also the mean and variance of the original data). Which method was more robust to noise? Why? (10 points)