TeIAS | Tehran Institute for Advanced Studies

Applied Data Analysis — Assignment 3

Instructors: Dr.Salavati TA: Peyman Naseri

In Homework 1, you practiced EDA, Data Cleaning, and Feature Engineering.
In Homework 2, you applied classical Machine Learning models for regression and classification.

In this homework, you will transition into **Deep Learning** and work with modern neural network architectures.
The goal of this assignment is **not** to train an excessively large number of models,
but to understand how neural networks work, experiment with key components, and explain your observations clearly.

> You are free to choose **any deep learning framework** (PyTorch, TensorFlow, or Keras).
> We strongly encourage you to briefly research these frameworks and mention in your notebook *why you selected the one you used*.

Submission Guidelines

Your results must be presented in a **well-documented Jupyter Notebook (.ipynb)**.
Keep your notebook clean, modular, and reproducible. The focus is on clarity, not quantity.

How to Submit

1. Use the **same GitHub repository** you used for Homework 1 & 2.
2. You may also share your notebook via **Google Colab** (ensure link access is open).
3. Your final submission must include:
   - GitHub link
   - Colab link (if applicable)
   - The final `.ipynb` notebook file
4. You may write your code modularly (e.g., `.py` files imported into the notebook).

After submission, a short **in-person session** will be scheduled for you to explain and review your work.

Collaboration Policy

All homeworks must be done **individually**.

Evaluation Criteria

You will be graded **qualitatively**, based on:

- The models are implemented correctly
- The notebook is clear, readable, and well-commented
- You provide meaningful explanations for the experiments
- You analyze *why* performance changes when you adjust architecture or hyperparameters

- You avoid unnecessary model training or excessive computation
- Figures, tables, and comments help understanding

Bonus Points

You can earn extra credit by adding meaningful and insightful elements, such as:

- A clear and informative `README.md` explaining your experiments
- Interactive visualizations (e.g., loss curves, comparison dashboards)
- Conducting small but thoughtful **error analysis** (show misclassified samples, etc.)
- Visualizing feature maps for CNNs
- Explaining trade-offs between architectures

Avoid unnecessary complexity, heavy training, or flashy additions — they do **not** earn extra points.
Focus on clarity, insight, and good explanations.

Late Submission Policy:
A **10% penalty** will be applied for each late day.

Generative AI Policy

The use of tools such as **ChatGPT, Claude, Gemini**, or other similar AI assistants is **allowed and encouraged** but **use them wisely**.

- Try to solve each problem yourself first.
- Then, you may use AI tools to check, improve, or compare your results.
- Remember: what matters most is **understanding** the code you submit, not who wrote it.
- Be cautious, large models often "hallucinate" or produce inaccurate results.

Important: It is recommended that you use the course's Ai teaching assistant before the deadline and upload your answers, approximate scores, and suggestions for improving your implementations.

Ai ADS Assistant(چاکر شما)

The goal is to help you become confident in solving real data science problems independently.

Homework Components

1. Multilayer Perceptron (MLP) — Foundations of Deep Learning

Using the dataset of your choice (preferably the same one used in previous homeworks), implement **fully-connected neural networks** with your chosen framework.

A. Core Tasks

Implement MLPs for:

- Binary classification
- Regression

Show:

- Training & validation performance
- Loss curves
- Final evaluation metrics

B. Network Tuning & Experiments

Organize your experiments into the following categories and discuss the effect of each change (with short comments, not just code):

1. Training & Optimization

- Try different **optimizers**: SGD, SGD+momentum, Adam

- **Learning rate** variations (too small / good / too large)
- Learning rate scheduling
- **Batch** size effects
- **Early stopping**
- Number of **epochs**

2. Architecture & Representation

- Depth: number of **hidden** layers
- Width: **neurons** per layer
- **Activation functions**: ReLU, LeakyReLU, Tanh, Sigmoid
- **Weight initialization**: Xavier, He, random
- **Batch Normalization**

3. Regularization & Stability

- L1 / L2 weight regularization
- Activity regularization
- **Dropout**
- Gradient clipping (optional)

Your goal is to show **how changes affect performance**, not to search for a perfect model.You do **not** need an exhaustive grid search. The goal is to **see and explain trends**: what helps, what hurts, and why.

C. Discussion Questions

1. Why are neural networks so powerful**?**
2. Why does training become more difficult as we go deeper?
3. (Optional) If MLPs can approximate any function with a single hidden layer (Universal Approximation Theorem), what unique benefits does *depth* provide that cannot be achieved simply by increasing width?

2. Convolutional Neural Networks (CNNs) — Image Modeling

Choose an image-based dataset or your own.

A. Build and Train a CNN

- Create a convolutional neural network with **at least a few convolutional layers**, followed by pooling and fully-connected layers
- Train it and show training curves and performance metrics (accuracy, loss, etc.).

B. Use Key CNN Components and Explain Their Effect

Use and **briefly analyze the effect** of:

- **kernel** sizes (receptive field)
- **strides**
- numbers of **filters**
- **pooling types and pooling window sizes** (e.g., max-pooling vs. average pooling)
- Depth of the network

For each experiment, briefly **comment on how the change affects**:

- Model capacity
- Overfitting vs. underfitting
- Training time and performance

C. Data Augmentation

Apply and briefly analyze the effect of **data augmentation** techniques such as:

- Random flips, rotations, crops
- Normalization and/or color jitter (if applicable)

Explain:

- How augmentation impacts overfitting and generalization

D. Transfer Learning

Choose **one pretrained model** (e.g., VGG19, ResNet, EfficientNet, MobileNet, etc.) and use it for your task via:

- Feature extraction, or
- Fine-tuning (partial or full)

Clearly state:

- Which pretrained model you chose and why
- Which layers you froze (if any)
- How performance compares to your own CNN from part A

E. Discussion Question

- Why are CNNs fundamentally more parameter-efficient than MLPs when dealing with high-dimensional inputs such as images? Under what conditions could an MLP theoretically match CNN performance, and why is this unrealistic in practical scenarios?

3. Recurrent Neural Networks (RNNs) — Sequence Modeling

If your original dataset is not **sequential**, you may pick a public dataset (text, time series, or any sequential data).

A. Implement Three Models

Using your chosen framework, implement and train:

- Vanilla RNN
- LSTM
- GRU

You may use high-level APIs (Keras layers, PyTorch nn.* modules). No need to implement cells from scratch.

B. Use Core Sequence Modeling Components and Explain

Use and briefly **analyze the effect** of:

- Sequence length

- Hidden size
- One vs multiple recurrent layers
- **Bidirectional** RNNs
- Dropout between recurrent layerss

C. Discussion Question

- Why are LSTMs and GRUs generally better than vanilla RNNs for long sequences?
  Explain the role of **gates** and how they help with **vanishing gradients** and **long-term dependencies**

4. Transformer Models — Attention-Based Architectures

In this section, you will **use** an existing transformer-based model or layer; you do **not** need to implement attention or transformer blocks from scratch.

A. Using a Transformer-Based Model

Choose one of the following approaches:

- Use a **pretrained Transformer model** (e.g., DistilBERT, BERT, TinyBERT) via a library such as HuggingFace Transformers.
- A **Transformer encoder layer** from your framework (e.g., PyTorch `nn.TransformerEncoder`, Keras `Transformer`/`MultiHeadAttention`)

Apply it to a classification or regression task (If you need, train or fine-tune your model).

Compare Transformer performance with RNN/LSTM.

C. Discussion Question

In your notebook, briefly explain:

- The main **advantages and disadvantages** of Transformer-based models
- Why do they **scale well** with data and model size?

- Why do they often require **large computational resources** compared to simpler models?
- What is **self-attention**, and what problem does it solve?
- Why can attention model **long-range dependencies** more effectively than simple RNNs?
- What is **multi-head attention**, and why does it help?
- What is the role of **positional encoding**?

5. Research about models (Bonus)

“Which Machine Learning Models Are Actually Used in Industry?”

In this section, you will conduct a small research review (Literature / Industry Review) and present your findings in a short written report (1–3 pages).

1. Investigate the Most Commonly Used Models in Industry

Using credible sources such as **Kaggle State of Data Science & ML reports**, **industry surveys**, **company technical blogs**, and **analytical articles**, investigate the following question:

“Given the rapid advancements in AI and deep learning, which machine learning model (or family of models) is currently *most widely used* in real industrial and organizational settings?”

2. Your Prediction for the Future

In a concluding section (2–3 paragraphs), answer the following:

“How will the distribution of widely used machine learning models change over the next 5–10 years?”

In your analysis, consider:

- Will classical models (e.g., Linear/Logistic Regression, Decision Trees) remain dominant?
- Will deep learning or LLM-based approaches gain more share?
- In which domains might the shift be strongest?

- Base your reasoning on current trends, industry behavior, and your own informed judgment.

Contact & Questions

If you have any questions about the assignment, feel free to ask in the **Telegram group**.
If you prefer to contact me directly, you can reach me through:

Telegram: t.me/peyman886
Email: peyman.75.naseri@gmail.com
You can usually find me in the **LLM Lab** during the **afternoons** :)

Final Note

This homework emphasizes **understanding, clarity, and thoughtful experimentation** not brute-force model training or heavy computation.

By the end of this assignment, you should be able to:

- Build neural networks
- Understand architectures
- Compare models
- Explain why each model behaves as it does
- And reason about modern deep learning systems with confidence

Due date: Mon, Dey 1, 23:59:59