Machine Learning Technologies (MUCEIM)

First Practical Laboratory: Deep Learning Architecture Experimentation

Author: Jordi Linares (jorlipel@upv.es)

Date: October 20, 2025

1. Introduction: From Theory to Practice

The theoretical study of neural network architectures provides a foundation for understanding why certain designs might be effective. However, the true test of any architecture lies in its empirical performance. In this first practical laboratory, you will transition from the role of a theoretical architect to that of a hands-on experimentalist. Your mission is to systematically design, implement, and evaluate a variety of deep learning architectures using TensorFlow/Keras.

This assignment is designed to build your practical proficiency in implementing and evaluating machine learning pipelines. You will confront real-world challenges such as overfitting, vanishing gradients, and the trade-offs between model complexity and performance. Through a structured process of experimentation, you will develop an intuition for how different architectural choices and training strategies interact to produce a successful model.

2. The Core Task: Systematic Experimentation and Analysis

Your task is to select a real-world dataset and conduct a series of controlled experiments to investigate the impact of different architectural and training parameters on a model's performance. You will implement your experiments in a Google Colab notebook using TensorFlow/Keras, ensuring your code is clean, reproducible, and well-documented. The primary deliverable is not just the code itself, but a complementary analytical report that explains your experimental process, presents your findings, and critically discusses the results.

You are encouraged to use AI assistants (e.g., Gemini, ChatGPT, Claude) as coding partners to help you generate boilerplate code, debug issues, and understand framework documentation. However, the final code and the analysis must be your own, and you must be able to explain every line and every decision.

3. Guidelines for the Experimentation

3.1. Dataset Selection

Choose one of the following datasets for your experiments. These have been selected for their balance of complexity and tractability, and are suitable for Google Colab's computational constraints. All datasets are available directly through TensorFlow/Keras, requiring no manual download.

Option 1: MNIST - Handwritten Digit Classification

The MNIST database contains 70,000 grayscale images of handwritten digits (0-9), each 28×28 pixels. It is split into 60,000 training images and 10,000 test images. This is an excellent starting point for understanding image classification with neural networks.

- **Task:** Multi-class classification (10 classes)
- **Dataset Size:** ~11 MB (very lightweight)
- **How to load:**

```python
from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Option 2: Fashion MNIST - Fashion Product Classification

Fashion MNIST is a drop-in replacement for MNIST, containing 70,000 grayscale images of fashion products (t-shirts, trousers, bags, etc.) across 10 categories. Each image is 28×28 pixels. It is slightly more challenging than MNIST while maintaining the same structure.

- **Task:** Multi-class classification (10 classes)
- **Dataset Size:** ~30 MB (lightweight)
- **How to load:**

```python
from tensorflow.keras.datasets import fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

Option 3: CIFAR-10 - Object Recognition

CIFAR-10 consists of 60,000 color images (32×32 pixels) across 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). It contains 50,000 training images

and 10,000 test images. This dataset is more challenging due to color channels and greater visual complexity.

- **Task:** Multi-class classification (10 classes)
- **Dataset Size:** ~163 MB (moderate size, suitable for Colab)
- **How to load:**

```python
from tensorflow.keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

Option 4: A Public Dataset of Your Choice

You may select another dataset from repositories like Kaggle, the UCI Machine Learning Repository, or TensorFlow Datasets, provided it meets the following criteria:

- **Size:** Must be manageable in Google Colab (typically < 500 MB compressed)
- **Justification Required:** You must provide a clear explanation for your choice in your report
- **Documentation:** Include complete data loading and preprocessing code in your notebook

Recommended Sources:

- Kaggle Datasets - Filter by size and domain
- UCI Machine Learning Repository
- TensorFlow Datasets

3.2. Experimental Process

You must follow a systematic approach to experimentation. A template Google Colab notebook is provided to guide you. Your process should include:

1. **Data Loading and Preprocessing:** Load your chosen dataset and apply any necessary preprocessing steps (e.g., normalization, one-hot encoding).
2. **Baseline Model:** Implement and train a simple baseline model. This will serve as your point of comparison for all subsequent experiments.
3. **Architectural Experiments:** Conduct at least **three** of the following experiments, systematically varying one parameter at a time:
    - **Network Depth:** Compare the baseline to deeper architectures.
    - **Network Width:** Compare the baseline to wider architectures.

- **Activation Functions:** Compare the performance of different activation functions.
- **Regularization:** Investigate the effect of Dropout or L1/L2 regularization.
- **Optimization:** Compare different optimization algorithms (e.g., Adam, RMSprop, SGD).

4. **Results and Visualization:** For each experiment, you must collect and plot key metrics, such as training/validation accuracy and loss curves. These visualizations are essential for your analysis.
5. **Analysis and Commentary:** Within your Colab notebook, use markdown cells to document your observations, analyze the results of each experiment, and draw conclusions.

4. The Role of AI Assistants

You are permitted and encouraged to use AI assistants as a tool to enhance your productivity and learning. However, you are responsible for the final code and analysis.

Usage Guidelines:

- You can use AI assistants to generate code for model architectures, help with debugging, explain concepts, or suggest different approaches.
- **Attribution Requirement:** In your final report, you must include a section that briefly describes how you used AI assistants. While you do not need to provide exhaustive prompt logs, you should explain which parts of your code or analysis were influenced by an AI assistant.
- **Understanding Requirement:** You must be able to explain and justify any code that appears in your notebook. During evaluation, you may be asked to clarify your implementation choices.

5. Template Notebook

A comprehensive Jupyter notebook template has been prepared specifically for this assignment. The template includes:

- Structured sections for data loading, preprocessing, baseline model, and experiments
- Empty code cells with instructions for what to implement
- Markdown cells prompting you to document your hypotheses and analysis
- Example plotting code structures

The template is designed to work seamlessly in Google Colab with no additional setup required. You can access it, make a copy to your Google Drive, and begin working immediately.

6. Deliverables and Submission

You must submit **two items**:

6.1. PDF Report

A concise and professional report that includes:

- **Introduction:** Brief introduction to the problem and the dataset you selected.
- **Experimental Design:** Clear explanation of each experiment you conducted and the rationale behind it.
- **Results and Analysis:** Summary of your findings, supported by the visualizations from your notebook. Include comparison tables where appropriate.
- **AI Assistant Usage:** A section documenting how you used AI assistants in your work.
- **Conclusion:** Final insights about deep learning architecture based on your experiments.
- **Google Colab Link:** A publicly accessible link to your final Google Colab notebook. **Important:** Ensure that the sharing setting is set to "Anyone with the link can view."

6.2. Google Colab Notebook

- **Execution Requirement:** The notebook must be fully executed and runnable from the first cell to the last without errors.
- **Content:** It should contain all your code, markdown-based analysis, and the output of your cells (including plots).
- **Sharing:** The notebook must be shared with "Anyone with the link can view" permissions.
- **Link Inclusion:** The link to your notebook must be included in your PDF report.

7. Evaluation Criteria

Your work will be assessed according to the following criteria:

Criterion	Weight	Description
Experimental Rigor	40%	The systematic and controlled nature of your experim Clear hypotheses, proper controls, and methodical variation of parameters.
Code Quality and Reproducibility	20%	The clarity, documentation, and functionality of your notebook. Code should be well-commented and run without errors.
Analysis and Insight	30%	The depth of your analysis and the quality of your conclusions in the PDF report. Demonstrates understanding of why certain architectures perform b
Clarity and Professionalism	10%	The overall quality of your written report and the presentation of your findings. Proper formatting, clea writing, and professional presentation.

---

8. Important Notes

- **Computational Resources:** Google Colab provides free access to GPUs. To enable GPU acceleration, go to `Runtime > Change runtime type > Hardware accelerator > GPU`.
- **Session Limits:** Free Colab sessions have time and resource limits. Plan your experiments accordingly and save your work frequently.
- **Code Execution:** Ensure your entire notebook runs from top to bottom without errors before submission. Test this by selecting `Runtime > Restart and run all`.
- **Academic Integrity:** While AI assistants are permitted, plagiarism from other students or uncited sources is strictly prohibited.

Good luck with your experimentation!