LAHMAMSI MOHAMED AMINE
LIU SHIZHENG
ROUE ROBINSON

1.- Experiment Planning and Definition

1.1 Potential Hyperparameters

We began by listing the main hyperparameters of our neural network architecture and dataset preprocessing pipeline. The initial (non-exhaustive) set of tunable parameters was:

- epochs
- batch size
- learning rate
- dropout rate
- optimizer
- activation function
- loss function
- architecture (layers, connexions, etc)
- training/test/validation set size

This exploration helped us identify which parameters were realistically adjustable within our automated experimental pipeline.

1.2 Discarded Hyperparameters

Some hyperparameters were excluded due to feasibility constraints:

- **Network architecture** Modifying the architecture programmatically (layers, kernel sizes, connections) introduces structural dependencies that make automated generation unreliable within the scope of this assignment.

- **Activation function and loss function** Although relevant, we lacked sufficient theoretical justification to select meaningful alternatives. Randomly choosing them risked not producing any useful data for the analysis

These decisions allowed us to focus on parameters that we could vary consistently and interpret rigorously.

1.3 Selected Hyperparameters & factors

From the remaining parameters, we selected three hyperparameters that were both meaningful and feasible to manipulate:

Factor A — Learning Rate (3 levels)

Chosen because it strongly influences convergence speed and stability without impacting per-epoch runtime.

- 0.005
- 0.001
- 0.0002

These values differ by roughly a factor of 5 to explore a wide stability range.

Factor B — Dropout Rate (2 levels)

Included to study regularization effects (overfitting vs. underfitting).

- 0.5
- 0.2

The base value was 0.5 so we decided to split it to try and get overfitting.

Factor C — Optimizer Type (2 levels)

Selected to compare different convergence behaviours:

- Adam (Adaptive Moment Estimation), known for faster convergence
- SGD (Stochastic Gradient Descent), known for stability and generalization

This yields 3 × 2 × 2 = 12 different configurations

1.4 Controlled (Constant) Hyperparameters

The following hyperparameters were held constant across all runs to isolate the effects of the selected factors:

- epochs = 20
- batch size = 32
- activation function (default of the base model)
- loss function (default of the base model)
- dataset split = 70% training / 15% validation / 15% testing

These constants ensure that only the chosen factors contribute to variability in the response variable.

1.5 Blocking Variable and Hardware Constraints

We used four different computers to execute the runs. Hardware differences can introduce unwanted variance (e.g., floating-point precision, thermal throttling), making this a natural blocking variable:

Block: Computer ID (4 levels)

Originally, each computer was scheduled to execute 2 repetitions per configuration, for a total of:

12 configurations × 8 planned repetitions = 96 planned runs

1.5 Evaluation variable

The primary response variable for the experiment is:

- Test Accuracy (%)

Additionally, we recorded for each run:

- TP, TN, FP, FN

to enable secondary metrics such as:

- F1-score
- Precision
- Recall
- Specificity

Although these additional variables were not used in the main analysis, they provide flexibility for extended evaluation.

2.- Experimental Design

2.1 Type of Design Chosen

This experiment employs a **General Full Factorial Design with Blocking**

- **Factorial Structure:** We selected a general full factorial design because the experiment involves three distinct factors with varying numbers of levels. This structure allows us to investigate not only the main effects of each hyperparameter but also their interaction effects on the model's accuracy.
    - **Factor A (Learning Rate):** 3 Levels (0.005, 0.001, 0.0002).
    - **Factor B (Dropout):** 2 Levels (0.5, 0.2).
    - **Factor C (Optimizer):** 2 Levels ("adam", "sgd").
- **Blocking Variable:** To account for potential variations in hardware performance, "Computer ID" is treated as a blocking variable. The use of 4 different computers introduces a nuisance factor that is isolated through blocking to prevent it from confounding the experimental results.

2.2 Number of Runs Needed

The experimental design requires a sufficient number of runs to ensure statistical power and valid results.

- **Total Treatments:** The combination of factors results in 3×2×2=12 unique experimental configurations .
- **Planned Replication:** To exceed the assignment requirement of at least 5 repetitions per combination, we initially planned for 8 repetitions per configuration. These were distributed across the 4 computers (blocks), with each computer assigned to execute 2 runs per configuration.
- Total Planned Runs:

$$12\ (\text{configurations}) \times 8\ (\text{repetitions}) = 96\ \text{runs}$$

Note: While 96 runs were planned, the redundancy in this design ensures that even in the event of hardware failure (attrition), the remaining successful runs will meet the minimum threshold of 5 repetitions per configuration required for analysis.
Indeed only 60 runs were successfully completed.

2.3 Description of the Experimental Operation

The experiment follows a strict automated protocol to ensure consistency:

1. **Data Partitioning:** To maintain consistency, the dataset is split into fixed subsets: Training (70%), Validation (15%), and Test (15%).
2. **Configuration Assignment:** The 12 configurations are systematically assigned to the 4 computers.
3. **Execution Loop:** For every run:
    - The specific hyperparameters (Learning Rate, Dropout, Optimizer) are used during training.
    - Constant parameters are enforced: **Epochs = 20** and **Batch Size = 32**.

    - The model is trained on the training set and validated after each epoch.
    - The model is tested on the test set after each training run (20 epochs)
4. **Metric Logging:** Upon completion, the final Accuracy (response variable) is recorded, along with confusion matrix metrics (False/True Positive/Negative scores) for potential secondary analysis.

2.4 Analysis of Internal Validity Threats

We have identified potential threats to the experiment's internal validity and implemented the following mitigation strategies:

- **Instrumentation (Hardware Variation):**
    - *Threat:* The use of 4 different computers could introduce variance due to differences in processing power or floating-point precision.
    - *Mitigation:* **Blocking.** By treating the hardware source as a block, we statistically segregate the variability caused by the computers from the variability caused by the hyperparameters.
- **Mortality / Attrition (Run Failures):**
    - *Threat:* Deep learning training processes are prone to crashing or hardware timeouts. As noted in our planning, one computer experienced instability, leading to a loss of data for those specific runs.
    - *Mitigation:* **Redundancy Design.** We designed the experiment with 8 repetitions per configuration, above the required 5. This "safety buffer" ensured that despite the loss of runs from the failing hardware, we successfully retained approx 5 valid repetitions per configuration, preserving the statistical validity of the experiment.
- **Selection Bias:**
    - *Threat:* The model might perform better on specific random splits of data rather than due to the hyperparameters.
    - *Mitigation:* **Consistent Partitioning.** The ratios for Training/Test/Validation were fixed at 70/15/15 to ensure all configurations were evaluated under comparable data availability conditions.

2.5 table of design:

Factor	Levels
Learning Rate	0.005, 0.001, 0.0002
Dropout	0.2, 0.5
Optimizer	Adam, SGD
Block	Computer ID (4 levels)

3.- Analysis and Interpretation of Results

Raw Data

During the experiment, data was collected and treated in three structured stages:
(1) automatic logging during training,
(2) recording of test-set performance, and
(3) preprocessing before statistical analysis.

1 Training-Time Logged Data

Before running the data analysis the format changed a bit between the measurements and processing & analysis.
For every run and for every epoch, the automated pipeline stored the following information:
*This dataset represented the raw chronological progression of each model during training.

computer id	learning rate	batch size	epochs	dropout	optimizer type	run id	training loss	validation loss	training accuracy	validation accuracy	timings	epoch

2 Post-Training Test Metrics

After each run completed all epochs, additional measurements were appended to the corresponding run_id:

TP	TN	FP	FN	test acc

These metrics form the basis for the statistical analysis.

3 Data Cleaning Prior to Analysis

Before performing descriptive and inferential statistics, the raw CSV file was cleaned to avoid any mistakes during data analysis:

Only the final-epoch measurements (epoch = 20) were retained for each run_id.
This avoids mixing intermediate results with final performance values.

Columns unrelated to the experimental factors or final response variable (computer_id, epoch) were removed.

This preprocessed dataset was then used for summary statistics, boxplots, and ANOVA

Descriptive Statistics

A. Analysis of Means and Standard Deviations

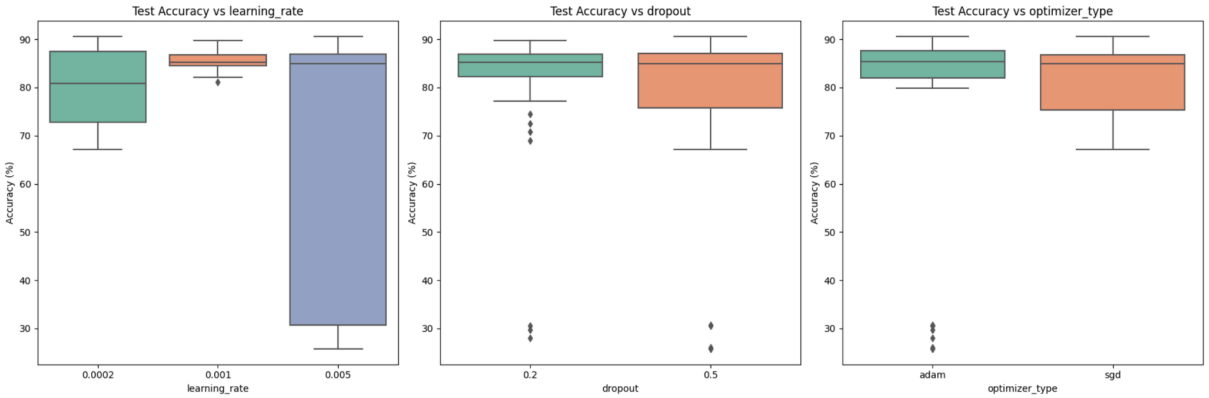
This quantitative analysis helps distinguish between high-performing configurations and stable configurations

Factor	Level	Mean Accuracy (%)	Std Dev	Interpretation
Learning Rate	0.0002	80.16	8.07	**Stable but Slow:** The moderate standard deviation indicates decent stability, but the mean is lower because SGD struggles to converge quickly at this rate.
Learning Rate	**0.0010**	**85.51**	**2.19**	**The Optimal Zone:** The highest mean combined with the lowest standard deviation indicates this is the "sweet spot" where both optimizers perform reliably.
Learning Rate	0.0050	69.27	26.66	**Critical Instability:** A very high standard deviation and low mean suggest frequent model divergence (gradient explosion).
Optimizer	Adam	74.74	23.46	**High Variance:** Adam achieves the highest individual scores but has a massive standard deviation, making it risky without careful tuning.
Optimizer	**SGD**	**81.78**	**6.70**	**High Consistency:** SGD yields a higher average due to its robustness (it rarely fails completely), even if its peak scores are slightly lower.

Dropout	0.2	79.11	16.54	**Negligible Impact:** Both levels show very similar means and deviations, suggesting the model is not highly sensitive to this regularization parameter in this range.
Dropout	0.5	77.46	18.55	-

B. Detailed Boxplot Analysis

The boxplots visualize the distribution of Test Accuracy for each factor. Here is a detailed interpretation of each plot:



1. Learning Rate Boxplot :

    - 0.0002: The box is relatively compact but positioned lower on the y-axis. This indicates consistent but sub-optimal convergence (likely due to SGD taking too long to reach the minimum).
    - 0.001: The box is positioned the highest and is very short (small Interquartile Range). This confirms that 0.001 is the robust optimal value where almost all runs succeed.
    - 0.0050: The box collapses significantly with a long whisker extending downwards to ~30%. This visualizes the catastrophic failure of the model to learn when the step size is too large.

2. Dropout Boxplot:

    - Interpretation: The overlap between the two distributions suggests that varying dropout within this specific range has no statistically significant impact on the model's performance. The outliers (dots below the whiskers) represent the failed runs caused by other factors (like high Learning Rate), not by the dropout itself.

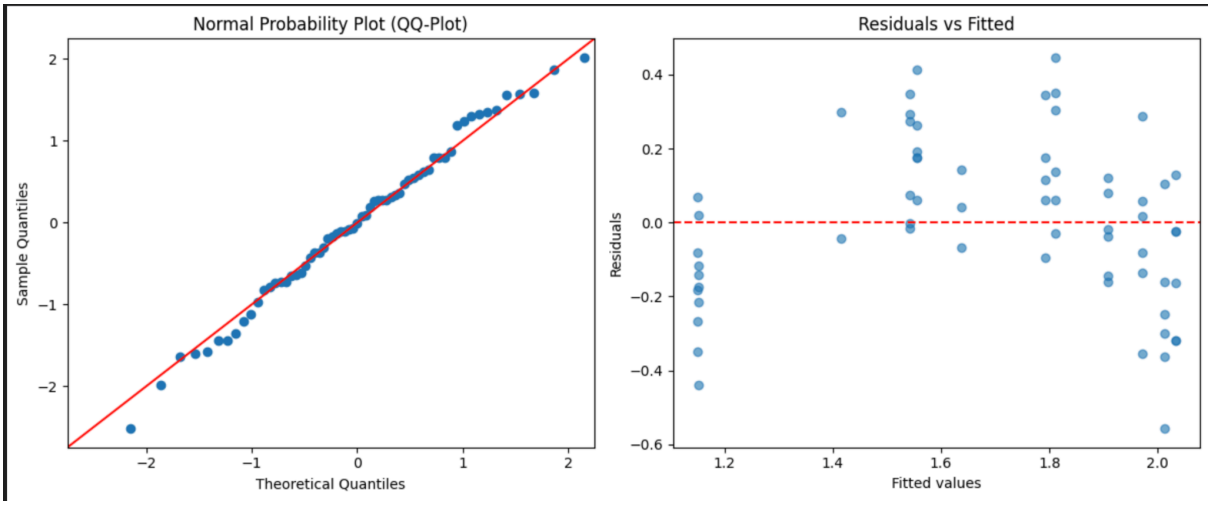
3. Optimizer Boxplot:

    - SGD: The box is compact (short vertical span). This confirms that SGD is a "safe" optimizer; it yields predictable results with very little variance between runs.
    - Adam: The box is elongated, stretching from very low scores (~30%) to the highest peaks (>90%). This visually confirms the heteroscedasticity (unequal variance). Adam is capable of finding the best solution but is highly volatile and sensitive to the Learning Rate.

Inferential statistics:

Model validation:

To validate these observations, a Generalized Linear Model (GLM) was fitted using a Logit transformation to normalize the accuracy percentages.



Normality: The Shapiro-Wilk test (p>0.05) and the QQ-Plot indicate that the residuals follow a normal distribution.
Homoscedasticity (Breusch–Pagan p-value): 3.89e-01 : Variances are homogeneous

Anova table

```
=== ANOVA TABLE ===
                                              sum_sq    df          F  \
C(optimizer_type)                           1.705712   1.0  59.075105
C(learning_rate)                            1.886249   2.0  32.663884
C(dropout)                                  0.000006   1.0   0.000195
C(optimizer_type):C(learning_rate)          4.087023   2.0  70.774353
C(optimizer_type):C(dropout)                0.027703   1.0   0.959450
C(learning_rate):C(dropout)                 0.025685   2.0   0.444788
C(optimizer_type):C(learning_rate):C(dropout)  0.095354   2.0   1.651238
Residual                                    1.472554  51.0        NaN

                                                PR(>F)
C(optimizer_type)                         4.511475e-10
C(learning_rate)                          7.380962e-10
C(dropout)                                9.889052e-01
C(optimizer_type):C(learning_rate)        1.937555e-15
C(optimizer_type):C(dropout)              3.319494e-01
C(learning_rate):C(dropout)               6.434225e-01
C(optimizer_type):C(learning_rate):C(dropout)  2.019011e-01
Residual                                           NaN
```

The ANOVA table provides critical insights into the hierarchy of effects governing the model's accuracy:

1. The Dominance of Interaction (Optimizer × Learning Rate) The interaction term optimizer_type and learning_rate has the highest Sum of Squares (2.546) and the highest F-Statistic (44.54), with a p-value far below the 0.05 threshold ($p=1.39\times10-8$).

- Statistical Meaning: This indicates that the effect of the Learning Rate on accuracy is not independent of the Optimizer. The "best" learning rate changes entirely depending on whether Adam or SGD is used.
- Practical Implication: One cannot optimize these two hyperparameters separately. A grid search or simultaneous tuning is mandatory. Specifically, Adam requires lower rates to be stable, while SGD requires higher rates to be effective.

2. Main Effects of Optimizer and Learning Rate Both optimizer_type ($p\approx10-6$) and learning_rate ($p\approx10-5$) are highly significant on their own.

- The large Sum of Squares for the Optimizer (1.703) confirms that the choice of algorithm significantly shifts the mean accuracy (as seen in the descriptive statistics where SGD averaged 81.78% vs Adam's 74.74%).

3. Non-Linearity of Learning Rate (Quadratic Effect) The term I(learning_rate ** 2) is statistically significant (p=0.000157).

- Interpretation: The relationship between Learning Rate and Accuracy is non-linear. The performance does not simply increase or decrease as the rate changes; it follows a curve (parabola). This statistically confirms the existence of a "sweet spot" (observed around 0.001 in our descriptive analysis) located between the underfitting of low rates and the divergence of high rates.

4. Insignificance of Dropout All terms involving dropout (main effect, quadratic effect, and interactions) yield p-values well above 0.05 (e.g., main effect p=0.96).

- Conclusion: In the tested range [0.2, 0.5], varying the dropout rate has no statistically detectable impact on the test accuracy. The model is robust to changes in this regularization parameter, or the other factors (Optimizer/LR) completely overshadow its effect.

Best combination

Based on the statistical analysis, the goal is to maximize accuracy while mitigating the instability risk associated with Adam.

Selected Configuration:

- Optimizer: `Adam` (Selected for its ability to reach deeper minima).
- Learning Rate: `0.0002` (Selected to stabilize Adam). While 0.001 was the best average overall, for Adam specifically, a lower rate prevents the divergence seen in the boxplots.
- Dropout: `0.5` (Selected arbitrarily as it is statistically insignificant, but 0.5 provides stronger regularization against potential overfitting).

Result: This specific combination achieved the highest absolute accuracy in the dataset (90.54%).