- **Subject:** AIL304m
- **Author:** Hieu Nguyen

MACHINE LEARNING – NEURAL NETWORKS RUBRIC (3 STAGES)

Each stage is evaluated separately (0–100 points), focusing on:

- Planning & Design
- Model Training & Evaluation
- Application & Deployment

STAGE 1 – PROJECT PLANNING & DESIGN

Goal: Assess students' ability to define a problem, design a neural network approach, and plan the implementation effectively.

Criteria	Description	Performance Levels	Max Points
1. Problem Definition & Objectives	The ML problem is clearly identified with defined input/output and measurable goals.	☐ Unclear problem (0–25) ☐ Defined but lacks measurable metrics (26–50) ☐ Clear objectives with evaluation metrics (51–75) ☐ Well-structured, quantifiable, and practical objectives (76–100)	25
2. Data Collection & Description	Identifies data sources, data size, key features, and data quality issues.	☐ No data plan (0–25) ☐ Basic description only (26–50) ☐ Includes statistics or examples (51–75) ☐ Detailed EDA with visuals and justification (76–100)	25
3. Model & Pipeline Design	Proposes suitable NN architecture (MLP, CNN, RNN...) and defines the workflow from preprocessing to evaluation.	☐ Missing or incorrect (0–25) ☐ Basic structure (26–50) ☐ Logical design with diagrams (51–75) ☐ Well-justified architecture and performance expectations (76–100)	25

Criteria	Description	Performance Levels	Max Points
4. Implementation Plan & Tools	Defines timeline, task assignments, frameworks (TensorFlow, PyTorch, etc.), and computational resources.	□ Not defined (0–25) □ Partial timeline (26–50) □ Detailed plan with clear tool selection (51–75) □ Comprehensive plan with risk management (76–100)	25
		Stage 1 Total	**/100**

1. Problem Definition & Objectives (25 points)

What to include:

- A clear description of the real-world or academic problem.
- The input and output variables.
- What the model should predict or classify.
- Why a Neural Network is appropriate for this task (vs. traditional ML).
- The performance metric(s) to evaluate success (e.g., accuracy > 90%, MAE < 0.05).

Examples:

- Image classification: “Classify images of cats and dogs using CNN; target accuracy ≥ 90%.”
- Regression: “Predict house prices using DNN with RMSE < 10,000.”
- Time-series: “Forecast energy consumption using LSTM with MAE < 5%.”

Tips:

- Look for a specific and measurable goal.
- Deduct points if the objective is vague (“build a model to analyze data”) or doesn’t justify using NN.

2. Data Collection & Description (25 points)

Include:

- Data source (public dataset, API, self-collected, synthetic, etc.).
- Dataset size (rows, columns).
- Description of each key feature (type, meaning, range).
- Missing values, outliers, or imbalances.
- Plan for preprocessing (normalization, encoding, etc.).
- Visual exploration (histograms, boxplots, correlations).

Examples:

- Dataset: CIFAR-10 (60,000 images, 32×32 pixels, 10 classes).
- Source: Kaggle, UCI Repository, internal company data.
- Problem: Class imbalance (dog images = 20%, cat images = 80%).

- Preprocessing: Rescale pixel values to [0,1], augment images by flipping and rotation.

Tips:

- Reward students who include charts or tables summarizing their data.
- Deduct points for unclear or unverified data sources.

3. Model & Pipeline Design (25 points)

Include:

- Proposed Neural Network architecture (type + layer structure).
- Description of each component:
- Input layer (data shape)
- Hidden layers (number, activation)
- Output layer (activation, number of classes/values)
- Explanation of why this architecture suits the problem.
- Training pipeline: data loading => preprocessing => model training => evaluation.
- Optional: simple block diagram or flowchart.

Examples:

- CNN pipeline for image classification: Input (32x32x3) => Conv2D(32 filters, ReLU) => MaxPooling(2x2) => Conv2D(64 filters, ReLU) => Flatten => Dense(128, ReLU) => Dense(10, Softmax)
- Pipeline diagram: Data Collection => Preprocessing => Model Training => Evaluation => Save/Deploy

Tips: Give higher marks for teams that show understanding of layer design and reasoning, not just copying code from online examples.

4. Implementation Plan & Tools (25 points)

Include:

- Timeline (Gantt-style or weekly plan) – milestones such as data prep, model training, tuning, report, and deploy.
- Team roles (if group project): data engineer, model designer, tester, reporter, etc.
- Tools & frameworks: TensorFlow, Keras, PyTorch, NumPy, Matplotlib, etc.
- Hardware plan: GPU/Colab/Local CPU.
- Risk management: what happens if data is unavailable or model fails to converge.

STAGE 2 – MODEL TRAINING & EVALUATION

Goal: Assess ability to build, train, tune, and evaluate neural network models effectively.

Criteria	Description	Performance Levels	Max Points

Criteria	Description	Performance Levels	Max Points
1. Data Preparation & Preprocessing	Performs data cleaning, normalization, and splitting into train/test sets appropriately.	☐ Missing or wrong (0–25) ☐ Basic processing (26–50) ☐ Well-executed preprocessing with examples (51–75) ☐ Smart feature engineering and rationale (76–100)	25
2. Model Construction & Training	Builds correct architecture, selects appropriate optimizer, loss, batch size, learning rate, etc.	☐ Code errors or non-functional (0–25) ☐ Runs but not explained (26–50) ☐ Works well, includes logs/graphs (51–75) ☐ Thorough experimentation, multiple setups compared (76–100)	25
3. Model Evaluation	Uses multiple metrics (accuracy, F1-score, ROC, confusion matrix, etc.) to assess performance.	☐ Accuracy only (0–25) ☐ 1–2 additional metrics (26–50) ☐ Full evaluation with interpretation (51–75) ☐ Comprehensive comparison and analysis (76–100)	25
4. Optimization & Fine-tuning	Applies regularization, dropout, early stopping, hyperparameter tuning, etc.	☐ None (0–25) ☐ Minimal tuning (26–50) ☐ Multiple techniques applied (51–75) ☐ In-depth optimization with trade-off analysis (76–100)	25
		Stage 2 Total	**/100**

1. Data Preparation & Preprocessing

Description:
Performs data cleaning, normalization, and splitting into train/test sets appropriately.

Guidelines:

- Handle missing or inconsistent data properly (drop, fill, or impute).
- Apply normalization or standardization techniques (e.g., MinMaxScaler, StandardScaler).
- Split dataset correctly into training, validation, and testing sets.

- Provide clear explanation for chosen preprocessing methods.
- Conduct meaningful feature engineering with justification.
- Visualize data distributions before and after preprocessing to demonstrate improvements.

Hints / Tips:

- Always check for data imbalance and consider resampling or weighting if necessary.
- Use correlation matrices or feature importance plots to select relevant features.
- Keep a reproducible preprocessing pipeline (e.g., using scikit-learn's `Pipeline`).
- Document every data transformation for transparency.

---

2. Model Construction & Training

Description:
Builds correct architecture, selects appropriate optimizer, loss function, batch size, and learning rate.

Guidelines:

- Select a suitable neural network architecture (e.g., CNN, RNN, MLP, Transformer).
- Justify architectural decisions: number of layers, activation functions, neurons.
- Choose optimizer and loss function appropriate for the task.
- Show evidence of training progress: loss and accuracy curves.
- Experiment with multiple configurations and compare outcomes.
- Record and explain hyperparameter values used.

Hints / Tips:

- Start simple, then increase model complexity incrementally.
- Use learning rate schedulers to stabilize training.
- Save model checkpoints to avoid losing progress.
- Visualize training results regularly to detect overfitting or underfitting early.
- Consider using frameworks like PyTorch Lightning or Keras for cleaner code structure.

---

3. Model Evaluation

Description:
Uses multiple metrics (accuracy, F1-score, ROC, confusion matrix, etc.) to assess performance.

Guidelines:

- Select evaluation metrics relevant to the problem type (classification or regression).
- Include confusion matrix, ROC curve, precision, recall, F1-score, AUC, or MSE/RMSE.
- Use both quantitative and qualitative analysis.
- Compare against baseline or previous models.
- Interpret and discuss the meaning of results beyond raw numbers.
- Highlight limitations or unexpected outcomes.

Hints / Tips:

- For imbalanced datasets, prioritize metrics like F1-score or AUC over accuracy.
- Visualize results (e.g., ROC, PR curves) for better interpretation.
- Always compare your model with a simple baseline (e.g., logistic regression or dummy model).
- Discuss reasons for performance gaps, such as data bias or model complexity.

---

4. Optimization & Fine-tuning

Description:
Applies regularization, dropout, early stopping, hyperparameter tuning, etc.

Guidelines:

- Implement regularization methods such as L1/L2 or dropout.
- Use early stopping and batch normalization to improve generalization.
- Tune hyperparameters like learning rate, number of layers, and optimizer settings.
- Compare results before and after optimization.
- Provide trade-off discussion (e.g., accuracy vs training time).
- Document tuning steps clearly.

Hints / Tips:

- Use grid search or random search for hyperparameter optimization.
- Log all experiments with timestamps and settings.
- Track validation loss separately to avoid overfitting.
- Test small learning rates before large ones; minor adjustments can make big differences.
- Apply cross-validation if dataset size allows.

STAGE 3 – APPLICATION & DEPLOYMENT

Goal: Evaluate ability to integrate, deploy, and present the trained model as a usable product.

Criteria	Description	Performance Levels	Max Points
1. Model Application	Integrates the model into a usable system (API, GUI, or product demo).	□ Not applied (0–25) □ Simple local demo (26–50) □ Working application with clear inputs/outputs (51–75) □ Fully functional real-world application (76–100)	25
2. Deployment	Deploys the model on a cloud or server (e.g., Flask, FastAPI, Streamlit, Docker).	□ Not deployed (0–25) □ Manual/local deployment (26–50) □ Automated deployment with documentation (51–75) □ Full CI/CD or public web deployment (76–100)	25

Criteria	Description	Performance Levels	Max Points
3. Visualization & Results Presentation	Includes visual dashboards, charts, or performance reports.	□ Missing visuals (0–25) □ Basic plots only (26–50) □ Clear dashboard or graphs (51–75) □ Professional visualization and insights (76–100)	25
4. Final Report & Presentation	Final report or slides explain pipeline, results, limitations, and future work.	□ Incomplete (0–25) □ Adequate but lacks clarity (26–50) □ Well-structured report with visuals (51–75) □ Professional, logical, and insightful presentation (76–100)	25
		Stage 3 Total	**/100**

1. Model Application

Description:
Integrates the trained model into a usable system (API, GUI, or product demo).

Guidelines:

- Provide a functional interface for the model (command line, GUI, or web API).
- Ensure inputs and outputs are clear, validated, and meaningful.
- Demonstrate that the model can process new data correctly.
- Include instructions for users to interact with the application.

Hints / Tips:

- Start with a simple interface and gradually enhance usability.
- Include error handling for invalid inputs.
- Use frameworks like Streamlit, Flask, or FastAPI for quick deployment.
- Document how to run the application and required dependencies.

---

2. Deployment

Description:
Deploys the model on a server, cloud platform, or containerized environment.

Guidelines:

- Choose a suitable deployment method (local server, Docker, cloud platform).
- Provide clear instructions or scripts to launch the application.

- Ensure the application is accessible and runs without errors.
- Include necessary environment specifications (Python version, dependencies).

Hints / Tips:

- Use Docker to create a reproducible deployment environment.
- Consider cloud services like AWS, GCP, or Heroku for public access.
- Automate deployment if possible using CI/CD pipelines.
- Test the deployed application thoroughly before submission.

---

3. Visualization & Results Presentation

Description:
Includes visual dashboards, charts, or performance reports for user understanding.

Guidelines:

- Visualize predictions, performance metrics, or comparisons effectively.
- Use interactive dashboards if possible to enhance user experience.
- Clearly label charts and tables for readability.
- Provide interpretation of results to guide decision-making.

Hints / Tips:

- Use libraries such as Matplotlib, Seaborn, Plotly, or Dash.
- Highlight key findings visually rather than showing raw numbers only.
- Include comparisons between models or baseline results.
- Ensure visualizations are responsive and easy to understand.

---

4. Final Report & Presentation

Description:
Provides a final report or slides explaining the pipeline, results, limitations, and future work.

Guidelines:

- Summarize problem statement, model design, training process, and deployment.
- Include performance metrics and visualizations.
- Discuss limitations and potential improvements.
- Prepare clear presentation slides or document for reviewers.

Hints / Tips:

- Structure the report logically: Introduction → Method → Results → Discussion → Conclusion.
- Use visuals and tables to summarize key results.
- Highlight insights gained and practical applications.
- Keep slides concise; use bullet points and diagrams to communicate clearly.