Plan & deliverables (what I will create & where)

Files I will add to GitHub and Canvas

1. data/data_set_large.csv
    - **~100k samples** (configurable) for smooth plots.
    - Columns (all numeric):
        - incidence_angle (rad)
        - num_ris_elements (discrete: e.g., 16,32,64,128)
        - snr_db (dB) and snr_linear
        - nakagami_m (shape parameter)
        - reflection_coeff (per-sample reflection factor from the “other” paper)
        - refraction_coeff (if applicable)
        - path_loss (dB or linear)
        - effective_snr (computed combining above)
        - achievable_rate = log2(1 + effective_snr)
        - outage (0/1 for achievable_rate < R_threshold)
    - Saved as CSV (Excel-compatible): data_set_large.csv.
2. scripts/generate_dataset.py
    - Script to reproduce the dataset in Colab (with seed, parameters at top).
    - Will include comments linking each generated variable to the corresponding equation/section in the two papers (so it’s traceable).
3. models/dnn_trio_duo.py
    - Baseline DNN (Trio/Duo style), configurable input size.
4. models/dnn_enhanced.py
    - Enhanced DNN that uses the extra channel/reflection/refraction features from Rahman et al.
5. training/train_compare.py

- Runs training for: baseline DNN, enhanced DNN; saves histories, model weights, and evaluation.
- Exposes hyperparams at the top (epochs, batch size, learning rate, dataset path).

6. analysis/plot_results.py
   - Produces publication-quality figures:
     - Training/validation/test loss (with test-loss overlay).
     - Outage Probability vs SNR curves with plt.ylim(1e0, 1e-6) (configurable), one curve per RIS configuration and per model (Sim / DNN / DNN-predicted).
     - Optionally smoothed OP curves (rolling average) and confidence bands.
   - Saves PNGs to results/.
7. README.md (updated)
   - Clear instructions: run order in Colab, where to open files, how to change parameters (e.g., number of RIS elements, m, R_threshold).
   - References to the two papers and which parts of the code correlate to which equations/sections.

Modeling details I will implement (how variables are computed)

- **Nakagami-m fading**: per-sample amplitude/power drawn as Gamma(shape=m, scale=Ω/m) as in Rahman et al.
- **RIS gain model**: simple aggregated gain proportional to number of RIS elements N and incidence angle factor cos(θ) (I'll include an option to use a more precise phase-alignment model if you want later).
- **Reflection / refraction**: add per-sample reflection_coeff and refraction_coeff (drawn from realistic ranges cited in Paper A). These are multiplied into path gain for the RIS-assisted and through-surface paths respectively, then combined (sum of powers) to form effective_snr. I'll include both additive and interference models (configurable).

- **Path loss**: Free-space / urban path-loss formula option; default is free-space + altitude term for UAV.
- **Effective SNR**: snr_linear * nakagami * RIS_gain * reflection_term / path_loss (exact formula commented and referenced).
- **Achievable rate**: R = log2(1 + effective_snr) → outage R < R_threshold.

I'll make the formulas and constants explicit and comment with pointers to paper sections (so reviewers/readers can follow).

Plots / Experiments included

- OP vs SNR for several RIS configurations (e.g., N = 16, 32, 64, 128), overlaying:
  - Analytical/simulation baseline (Nakagami model).
  - DNN (Trio/Duo) predicted OP.
  - Enhanced DNN predicted OP (with reflection/refraction).
  - (Later) BiLSTM predicted OP.
- Training/validation/test loss curves with test-loss horizontal line.
- Option: confidence intervals from repeated Monte-Carlo runs (configurable; slows runtime).

Performance / Colab practicality

- Default dataset size: **100k** samples — should run on Colab (I'll keep RAM usage in mind). I'll make num_samples a parameter; default 100k but I'll include recommended values and notes for Colab RAM/GPU.
- OP Monte Carlo uses num_op_samples parameter (default 100k for smooth curves; can be lowered for speed).

Next steps (I will do now unless you tell me otherwise)

1. Implement all files above and push to your GitHub RIS_NN_Simulations repo (under new folders data/, scripts/, models/, training/, analysis/, results/).

2. Also place the same scripts into Canvas (so you can copy into Colab).
3. Run a quick test in Canvas/Colab environment (dataset generation + one training epoch + OP curve with reduced samples) and report back the key outputs (file names, example plots).
4. After you confirm everything works and the plots look reasonable, I'll produce the BiLSTM implementation and run the full experiments for paper-quality figures.

If that plan looks good I'll proceed to implement and push the changes now. If you want any specific choices (e.g., default RIS element set, R_threshold value, Nakagami-m ranges, dataset size), tell me now — otherwise I'll use sensible defaults consistent with the two papers (I'll document them in the README).