Parallel CNN Project Checklist

**Initial Setup and Planning**

1. **Select Topic Area**:
   * Choose a project directly related to your thesis or use a substitute process with similar input data.
2. **Define Inputs and Outputs**:
   * Identify the inputs and outputs for your problem.
3. **Data Acquisition**:
   * Decide whether to collect your own data, use an existing dataset, or simulate data.
   * Ensure the dataset is labeled and contains everything you need.

**Week 3 Deliverables**

1. **Meeting with Instructor**:
   * Prepare for a 10-minute discussion with the instructor about your topic, data acquisition, and ANN architecture.
   * Discuss feasibility of data collection or the adequacy of the existing/simulated dataset.
   * Define your ANN architecture (Dense, CNN, RNN, 1D ConvNet, etc.).
   * Establish a baseline and possible metrics for comparison.
   * Confirm hardware setup readiness.

**Data Preparation**

1. **Create Datasets**:
   * Ensure datasets are ready for use (either collected, existing, or simulated).
   * Prepare the data processing and splitting strategies.

**Development**

1. **Develop the ANN Architecture**:
   * Implement the chosen ANN architecture.
   * Ensure the architecture can scale with data quantity.

**Parallel Processing and Code Management**

1. **Implement Parallel Processing**:
   * Employ parallel processing in your algorithm.
   * Compare training speed without parallel processing and with parallel processing.
   * Print out processing times and computational requirements after running both ways on datasets of different sizes.
   * Use TensorFlow's MirroredStrategy for easy parallelization.
2. **Setup Git Repository**:
   * Store all code in a Git repository (e.g., GitHub, git.antcenter.net).
   * Configure the Git repository to execute unit tests on code changes.
   * Ensure the repository has a **.gitignore** file configured for the programming language used.
   * Specify dependencies through a **requirements.txt** or **pyproject.toml** file.

**Testing and Hyperparameter Tuning**

1. **Unit Tests**:
   * Ensure unit tests are successfully run for all capabilities without bugs.
2. **Hyperparameter Specification**:
   * Allow users to specify hyper-parameters for the module.
3. **Distributed Processing**:
   * Ensure the algorithm works in a distributed manner for extra points.

**Implementation and Flexibility**

1. **Implementation with Multiple Datasets**:
   * Demonstrate implementation with three datasets (two test datasets and one challenge dataset).
2. **Code Flexibility**:
   * Ensure the code works with other datasets that may have different numbers or types of variables.

**Methodology (Grading Emphasis)**

1. **Repeatable Architecture**:
   * Ensure the architecture is repeatable.
2. **Regularization and Hyperparameters**:
   * List regularization techniques used.
   * Specify learning rates, optimizer, and epochs.
3. **Model Fitting**:
   * Develop a model that fits the data shapes.
4. **Scaling Up**:
   * Scale up the model so it can learn the training data.
5. **Generalization**:
   * Regularize and tweak hyperparameters to generalize to validation data.

**Documentation and Presentation**

1. **Readme.md File**:
   * Include the following sections:
     + Algorithm Purpose
     + Hyperparameters
     + Background
     + History
     + Variations
     + Pseudo code
     + Example code to import and use the module
     + Visualization or animation of algorithm steps or results
     + Benchmark Results
     + Comparison of efficiency and effectiveness
     + Lessons Learned
     + Unit-testing strategy (steps tested individually and code-coverage measurement)
     + Checkpoint technique used
2. **Presentation**:
   * Prepare and record a 10-15 minute presentation on the project, covering the topic, input/output data, ANN used, and final results.
   * Focus on visual effects, unique problem set/solution, and best pilot retention solution.

**Final Deliverables**

1. **Submit Final Deliverables**:
   * Submit the paper in the style of a journal article.
   * Upload all software and data (if public) to the GitHub Classroom for submission.
   * Provide a hard drive or upload data to the ANT Center NAS on the CDN if not public.
   * Ensure the final project is configured to run on the established course image for reproducibility.