UE22CS320B – Capstone Project Phase – 2

Project Progress Review #1

Project Title : Exploring the Applicability of the Lottery Ticket Hypothesis to Diffusion Models

Project ID : PW25_COP_01

Project Guide : Mr. Prakasha C O

Project Team with SRN : PW_25_COP_01_216_240_245_310

Outline

- Abstract and scope.
- Suggestions from phase-1
- Design Approach.
- Design constraints, Assumptions and Dependencies
- Proposed methodology/Approach
- Architecture
- Design Description
- Project Progress Plan for Phase 2
- References
- Any other Information

Abstract and Scope

- In this capstone project, we aim to explore the application of the Lottery Ticket Hypothesis (LTH) to diffusion models, a type of generative model that has gained prominence in areas such as image generation and denoising tasks.

- By applying LTH to these models, we seek to identify compact, high-performing subnetworks that retain the generative capabilities of the full model while reducing computational overhead.

- The LTH, introduced by Frankle and Carbin in 2018, posits that within large, pre-trained neural networks, there exist smaller sub-networks—referred to as "winning tickets"—that can be retrained from scratch to achieve comparable, or sometimes even better, performance relative to the full, original network.[1]

Suggestions from Phase 1

Provide the suggestions and remarks given by the panel members. (badri prasad sir asked how 99% sparsity is possible)

Sparsity is the number of zero or near-zero elements in a dataset or a model's parameters.

In the Lottery Ticket Hypothesis (LTH), a model can achieve strong performance despite being heavily sparse (99% of parameters being pruned) because of the key idea that there exists a smaller sub-network (the "winning ticket") within the larger network that can be trained to match or outperform the original model when trained in isolation.

Design Approach

Traditional LTH Design Approach

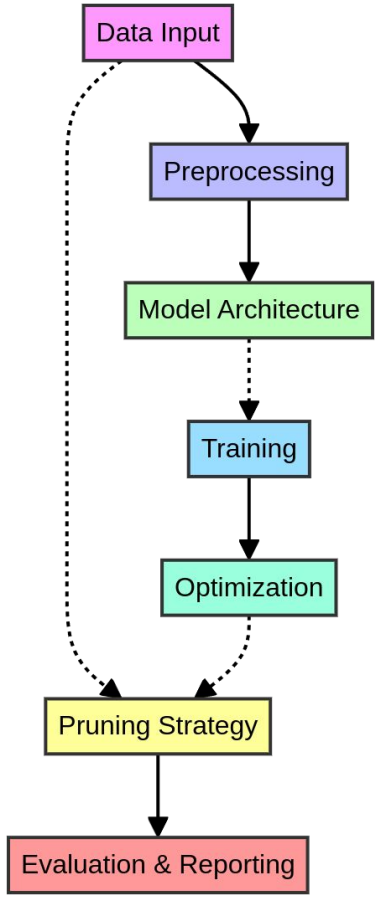
Step 1: Initialize the classification model

Step 2: Train the model to convergence or use a pretrained model

Step 3: Prune lowest magnitude weights(iterative magnitude pruning)

Step 4: Reset remaining weights to original values

Step 5: Retrain pruned network



Design Approach

Adapted Design for Diffusion Models

Step 1: Train a diffusion model or use a pretrained model.

Step 2: Apply iterative magnitude pruning (IMP) or Hilbert-Schmidt Independence Criterion (HSIC), or another technique to identify sparse subnetworks.

Step 3: Reinitialize pruned subnetwork to original weights.

Step 4: Retrain the sparse model to assess performance.

Step 5: Evaluate across different diffusion components (e.g., different layers with varying sparsity rather than a unified pruning ratio).

Design Constraints

Computational Efficiency: Given limited compute power, the design of the project must focus on optimizing the efficiency of computations. LTH's goal is to reduce computational overhead, which aligns with the project's constraints. The focus on finding smaller subnetworks within diffusion models should lead to a more manageable and efficient solution.

Diffusion Model Complexity: Diffusion models are often complex and require significant computational resources. Since the goal is to explore the Lottery Ticket Hypothesis (LTH) on these models, the assumption is that a trade-off exists between model performance and the number of parameters that can be pruned. The design must, therefore, prioritize pruning methods that maintain the diffusion model's essential characteristics.

Scalability of Experiments: The assumption that the experiments can be scaled down for initial tests is important. Given the computational limitations, testing on smaller subsets of data or reducing the size of the models in the early stages of the project will be necessary to evaluate the effectiveness of the pruning approach.

Dependencies and Their Impact on the Project

Dataset Availability and Preprocessing: The project depends on the availability of suitable datasets for training and testing the diffusion models. If high-quality image datasets for generative tasks are not available or accessible, the results of your experiments may be limited. Dataset preprocessing (such as normalization and augmentation) will also be critical to ensuring that the models generalize well.

Computational Resources for Retraining: Once the winning tickets are identified, retraining these subnetworks is required. Given the limited compute resources, the retraining process may need to be optimized. This dependency means that choosing smaller models will be necessary to prevent long training times that exceed available compute capabilities.

Performance Metrics for Comparison: The project will need reliable metrics to assess the performance of the pruned subnetworks. The dependencies here include having appropriate loss functions and evaluation protocols for generative tasks. If these metrics aren't well-defined or are not suited to diffusion models, the project may not be able to effectively evaluate whether pruning is successful or if the smaller subnetworks retain the generative capabilities of the full model.

Proposed Methodology / Approach

Our current methodology assumes that the dataset is small enough to be able to run on platforms like Google Colab or Kaggle. This includes datasets like CIFAR-10, MNIST, Flowers102, SyntheticLungCT, etc.

The overall approach is as follows:

1. Train the unpruned model first.
2. Gradually begin pruning the model using different pruning techniques.
3. The goal is to find the best pruning technique that can prune the later models more efficiently.

Benefits

- Runs on free-to-use software platforms.
- Has previous research data for certain datasets to validate the accuracy of our model.

Disadvantages

- Time-consuming process due to having only low compute power.
- May need the ability to run bigger datasets later on to prove generalization.

Architecture

The goal is to identify high-performing subnetworks (or “winning tickets”) within pre-trained diffusion models used for generative tasks such as image generation and denoising. By pruning and retraining these subnetworks, the project aims to reduce computational overhead while maintaining or even improving performance.

Logical User Groups

Researchers and Developers:

1. **Primary Users:** Responsible for designing experiments, training models, applying various pruning techniques, and analyzing performance metrics.
2. **Secondary Users:** Individuals looking to extend or validate the research findings, such as academic peers or industry R&D teams.

End Users (Future Deployment):

Application Consumers: Users who benefit from the efficient, pruned models in real-world applications (e.g., faster inference on edge devices) once the research is mature.

System Architecture

The system is organized into distinct layers and components that interact to perform model training, pruning, evaluation, and eventual deployment. The architecture is designed to maximize modularity and allow independent development of each component

Application Components

- **Model Training Component:**
  - Responsible for training the full (unpruned) diffusion model using chosen datasets.
  - Frameworks: PyTorch or TensorFlow.
- **Pruning Component:**
  - Implements different pruning techniques (e.g., iterative magnitude pruning, structured pruning) to gradually reduce the network size.
  - Tracks the sparsity level across model layers and identifies “winning tickets.”

System Architecture

- **Retraining Component:**
  - After pruning, the winning ticket subnetworks are retrained .
  - Ensures that the pruned model recovers its generative performance.
- **Evaluation and Benchmarking Component:**
  - Measures performance using metrics such as Fréchet Inception Distance (FID), perceptual loss, inference speed, and memory footprint.
  - Compare between the full model and pruned subnetworks.
- **Visualization and Reporting Component:**
  - Uses tools like TensorBoard, Matplotlib, or Plotly to visualize training metrics, sparsity levels, and performance trade-offs.
  - Generates reports to support research findings.

System Architecture

Data Components

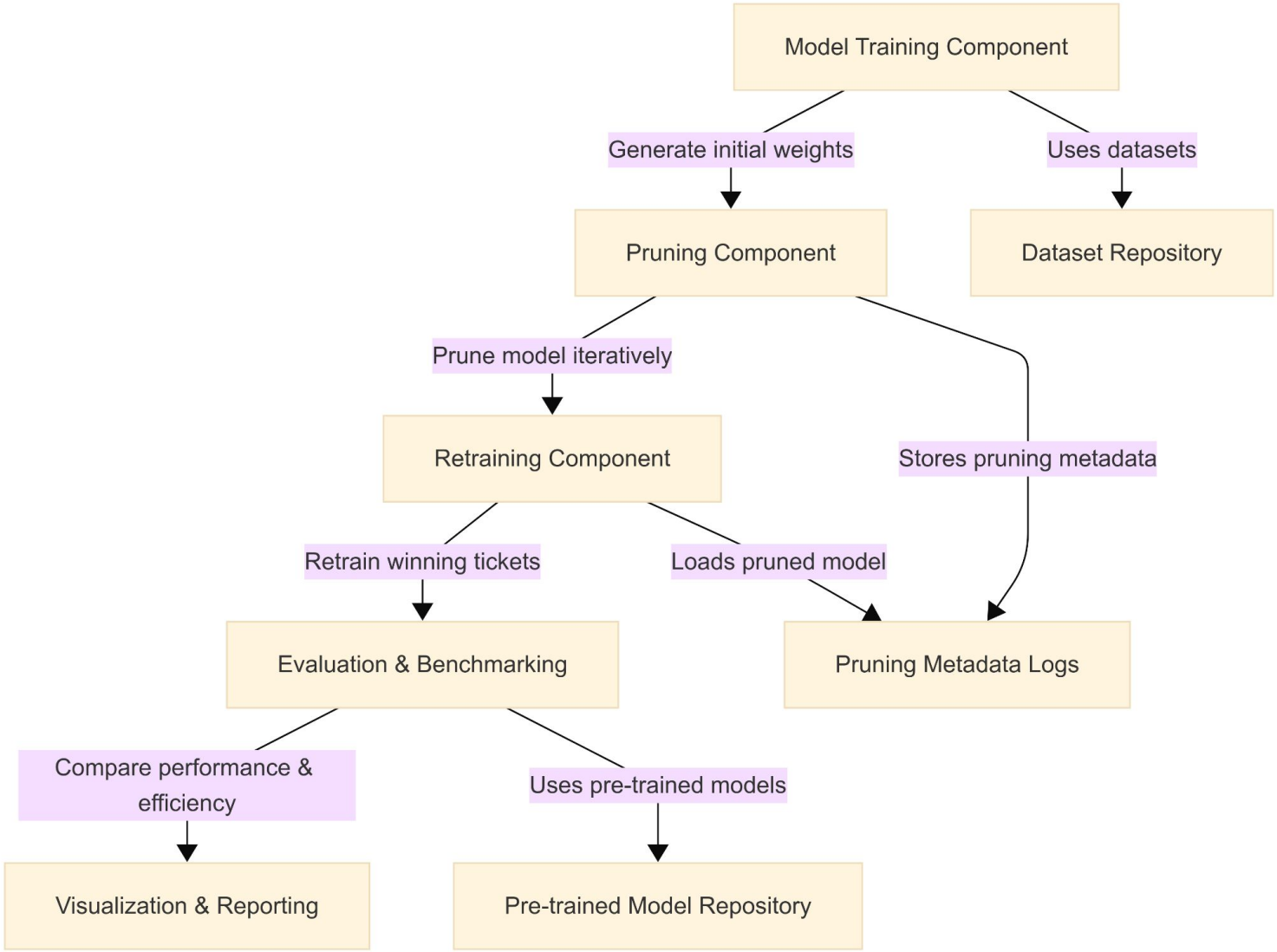
- **Datasets:**
  - Small-scale datasets (CIFAR-10, MNIST, etc.) stored locally or in cloud storage.
  - Includes data augmentation and normalization pipelines.
- **Model Checkpoints:**
  - Intermediate models stored after some pruning iteration.
  - Checkpoints facilitate rollback if performance degrades excessively.
- **Pruning Metadata:**
  - Logs detailing pruning percentages, layer-wise sparsity, and accuracy changes.
  - Useful for experimental analysis and reproducibility.

System Architecture

Interfacing Systems

- **Compute Platforms:**
  - Google Colab / Kaggle Notebooks for training and experimentation.
  - Local development environment for prototype testing.
- **Deep Learning Frameworks:**
  - PyTorch/TensorFlow for model development, training, and pruning implementation.
- **Visualization Tools:**
  - TensorBoard, Matplotlib, and Plotly for tracking model performance and resource usage.
- **Pre-trained Model Sources:**
  - External repositories (e.g., Hugging Face, GitHub) for accessing initial diffusion model weights.

System Architecture



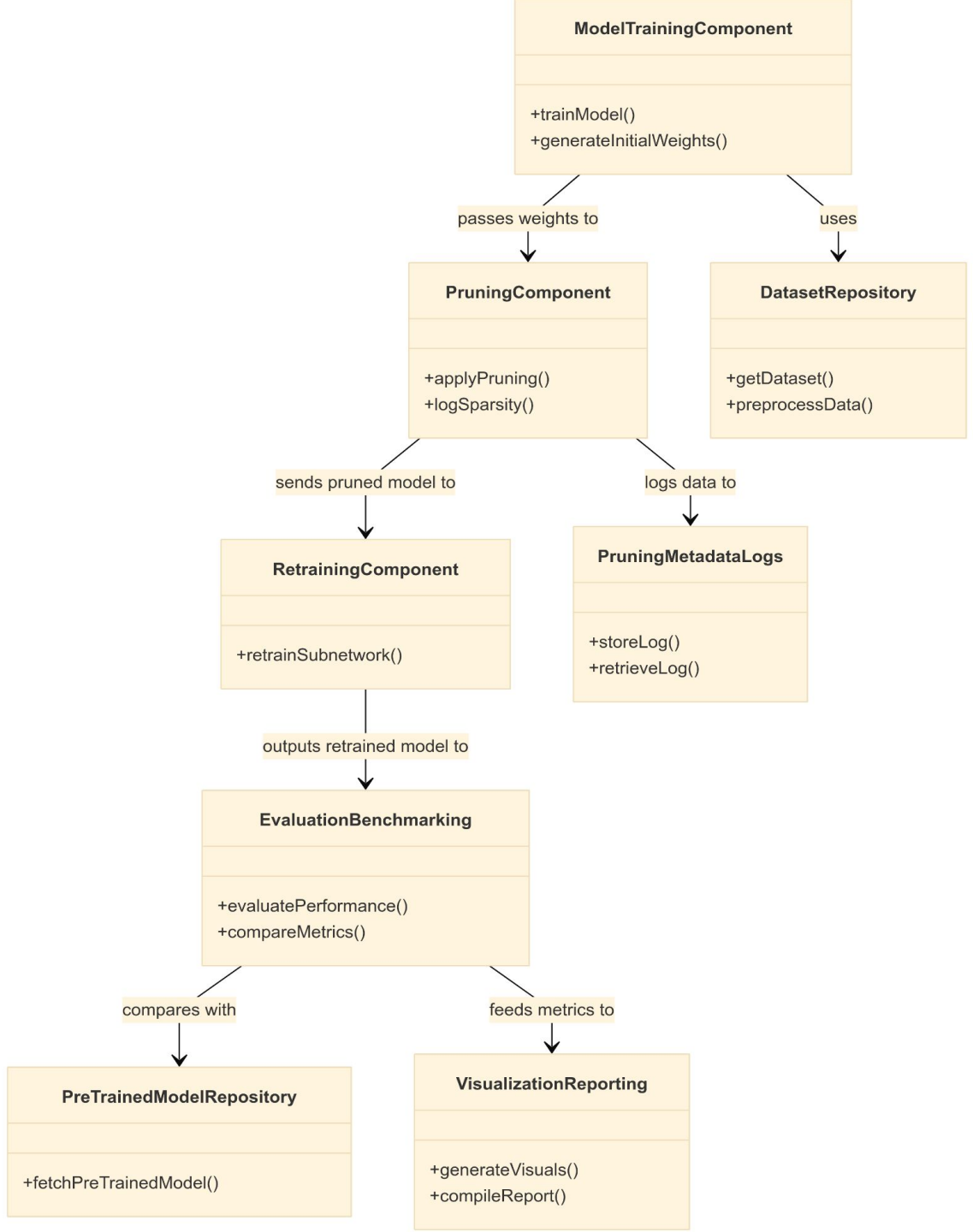
System Architecture

Design Description

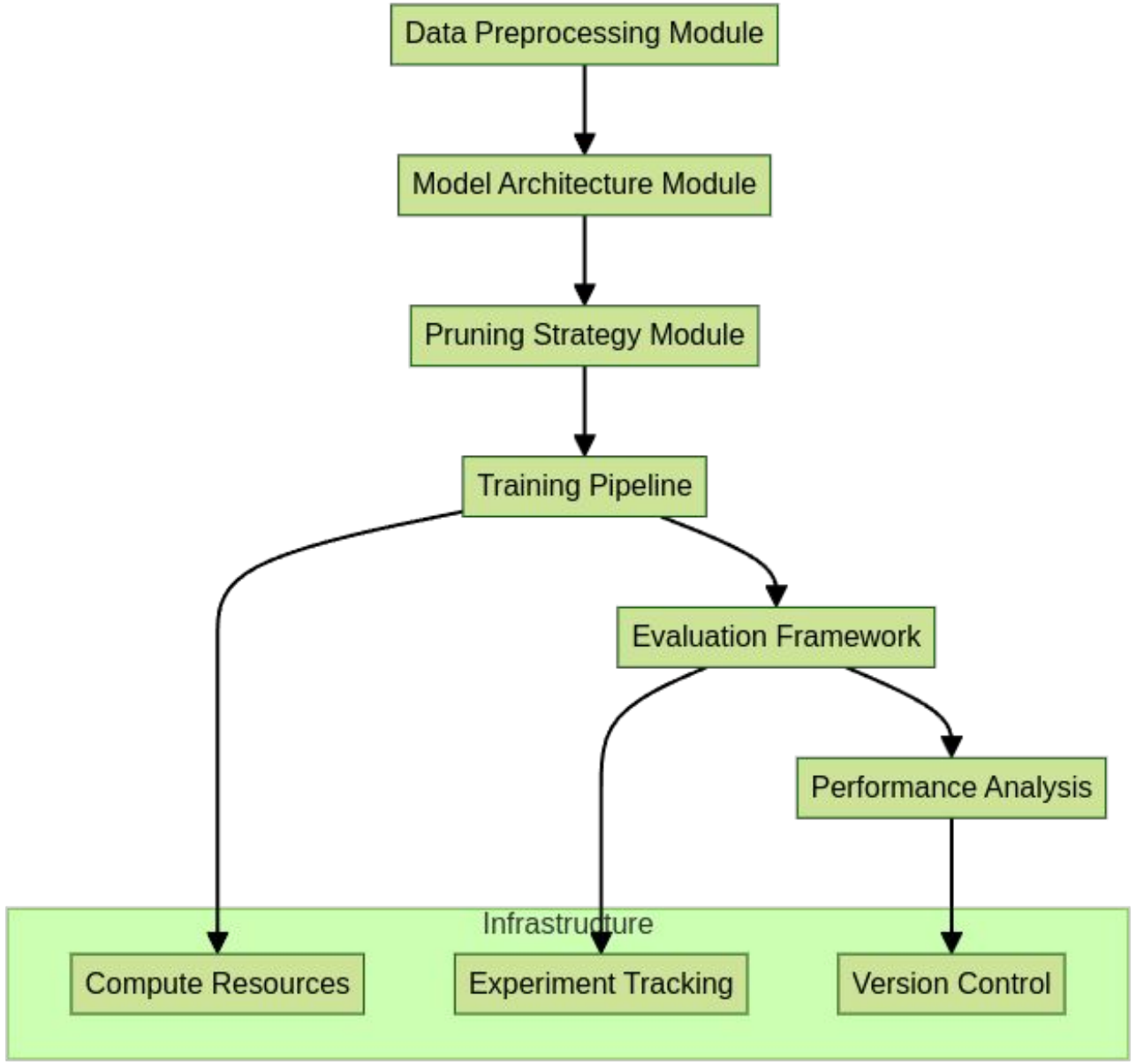
Add as many slides as required to cover the following aspects:

- Master class diagram(done)
- ER Diagram(done)
- User Interface Diagrams/ Use Case Diagrams(done)
- Report Layouts
- External Interfaces(done)

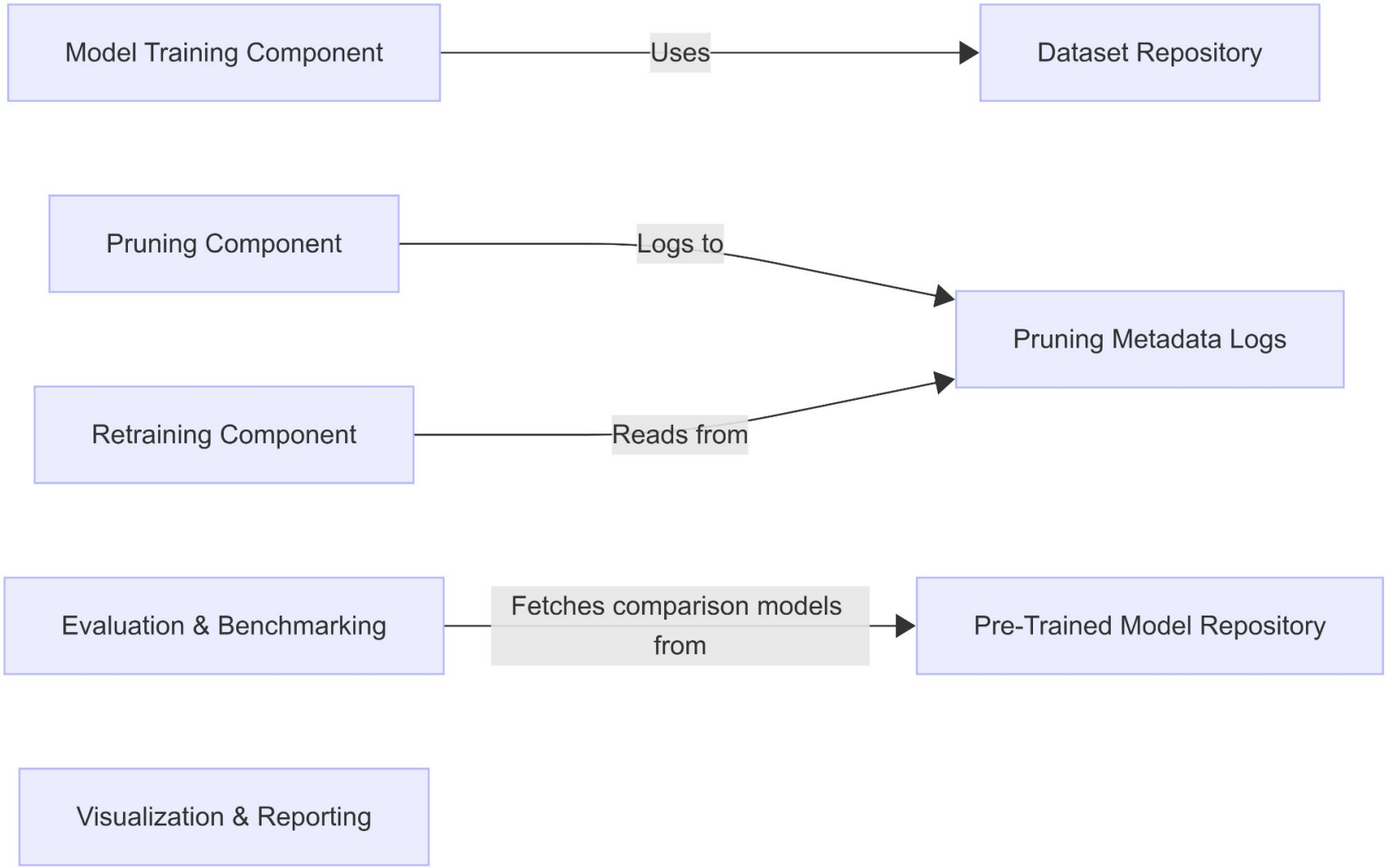
Master Class Diagram



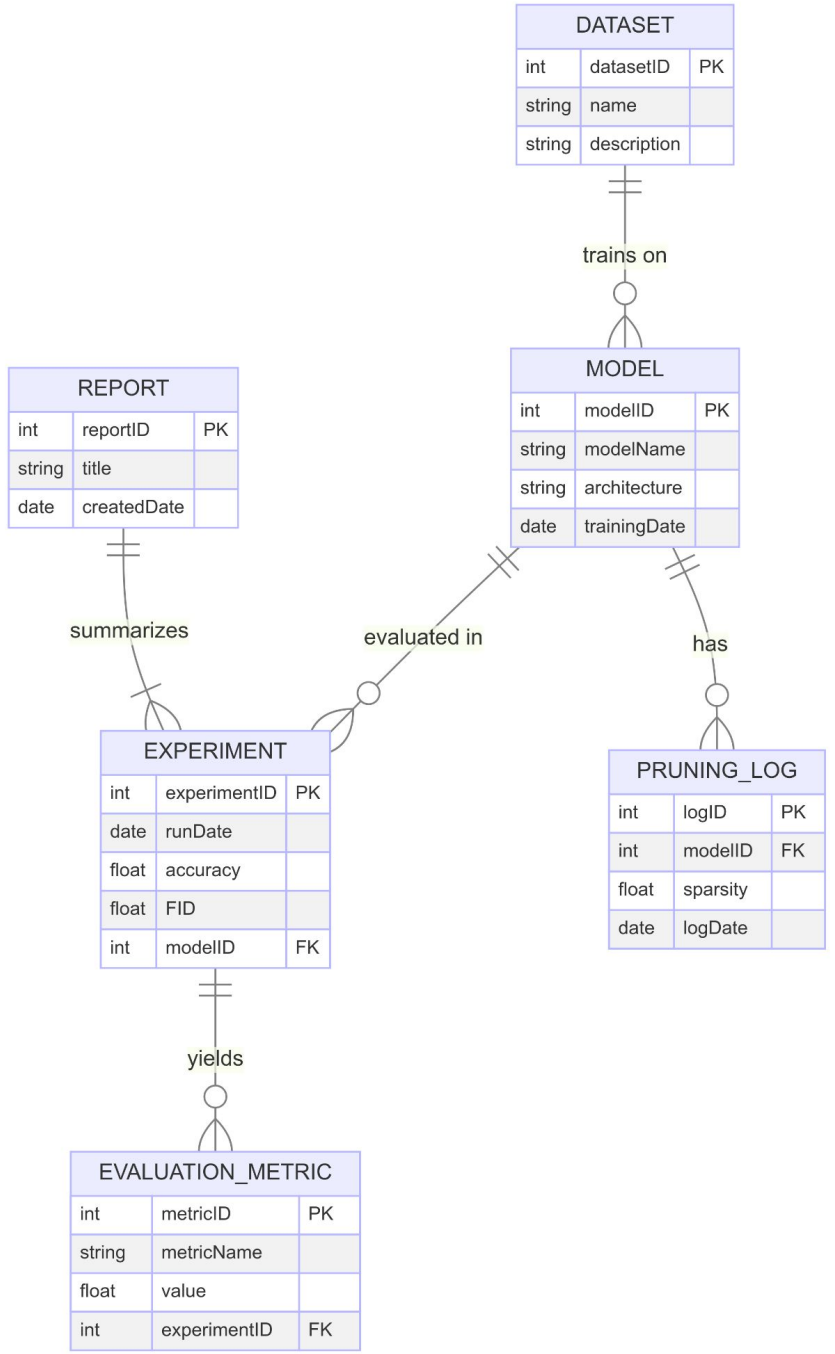
Use Case Diagram



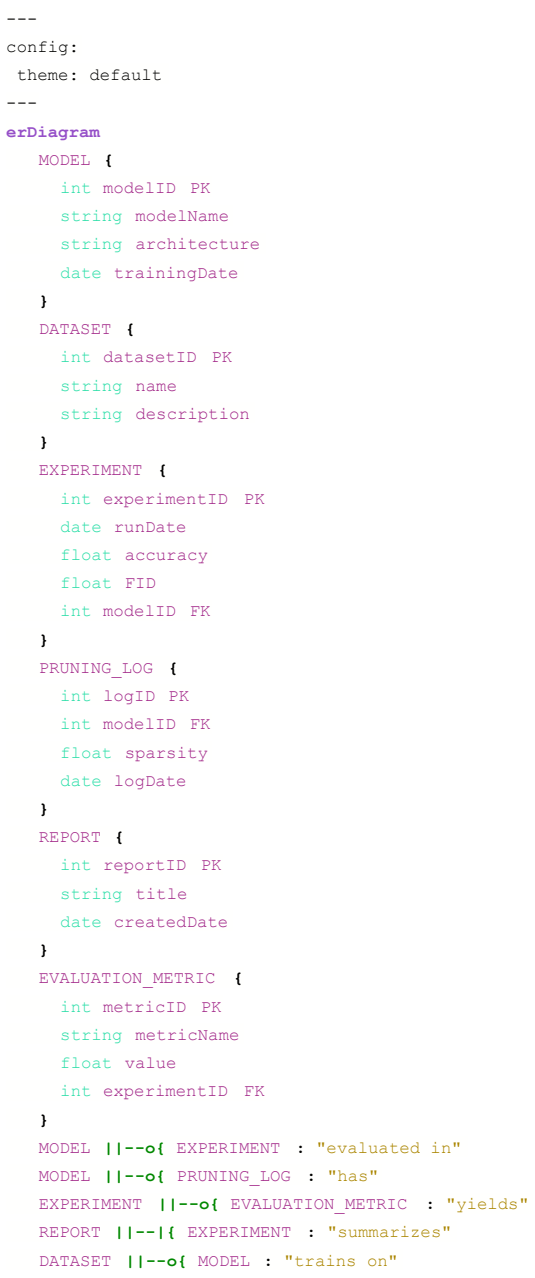
External Interfaces Diagram



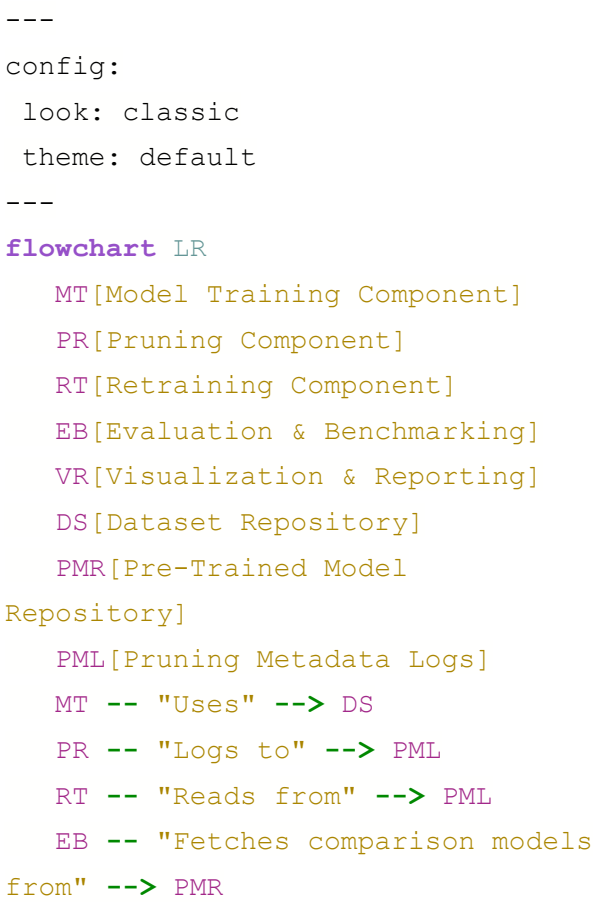
ER Diagram


DATASET
int datasetID PK
string name
string description
trains on
MODEL
int modelID PK
string modelName
string architecture
date trainingDate
REPORT
int reportID PK
string title
date createdDate
summarizes
evaluated in
has
EXPERIMENT
int experimentID PK
date runDate
float accuracy
float FID
int modelID FK
PRUNING_LOG
int logID PK
int modelID FK
float sparsity
date logDate
yields
EVALUATION_METRIC
int metricID PK
string metricName
float value
int experimentID FK

ER Diagram


---
config:
theme: default
---
erDiagram
MODEL {
int modelID PK
string modelName
string architecture
date trainingDate
}
DATASET {
int datasetID PK
string name
string description
}
EXPERIMENT {
int experimentID PK
date runDate
float accuracy
float FID
int modelID FK
}
PRUNING_LOG {
int logID PK
int modelID FK
float sparsity
date logDate
}
REPORT {
int reportID PK
string title
date createdDate
}
EVALUATION_METRIC {
int metricID PK
string metricName
float value
int experimentID FK
}
MODEL ||--o{ EXPERIMENT : "evaluated in"
MODEL ||--o{ PRUNING_LOG : "has"
EXPERIMENT ||--o{ EVALUATION_METRIC : "yields"
REPORT ||--|{ EXPERIMENT : "summarizes"
DATASET ||--o{ MODEL : "trains on"

External Interfaces Diagram



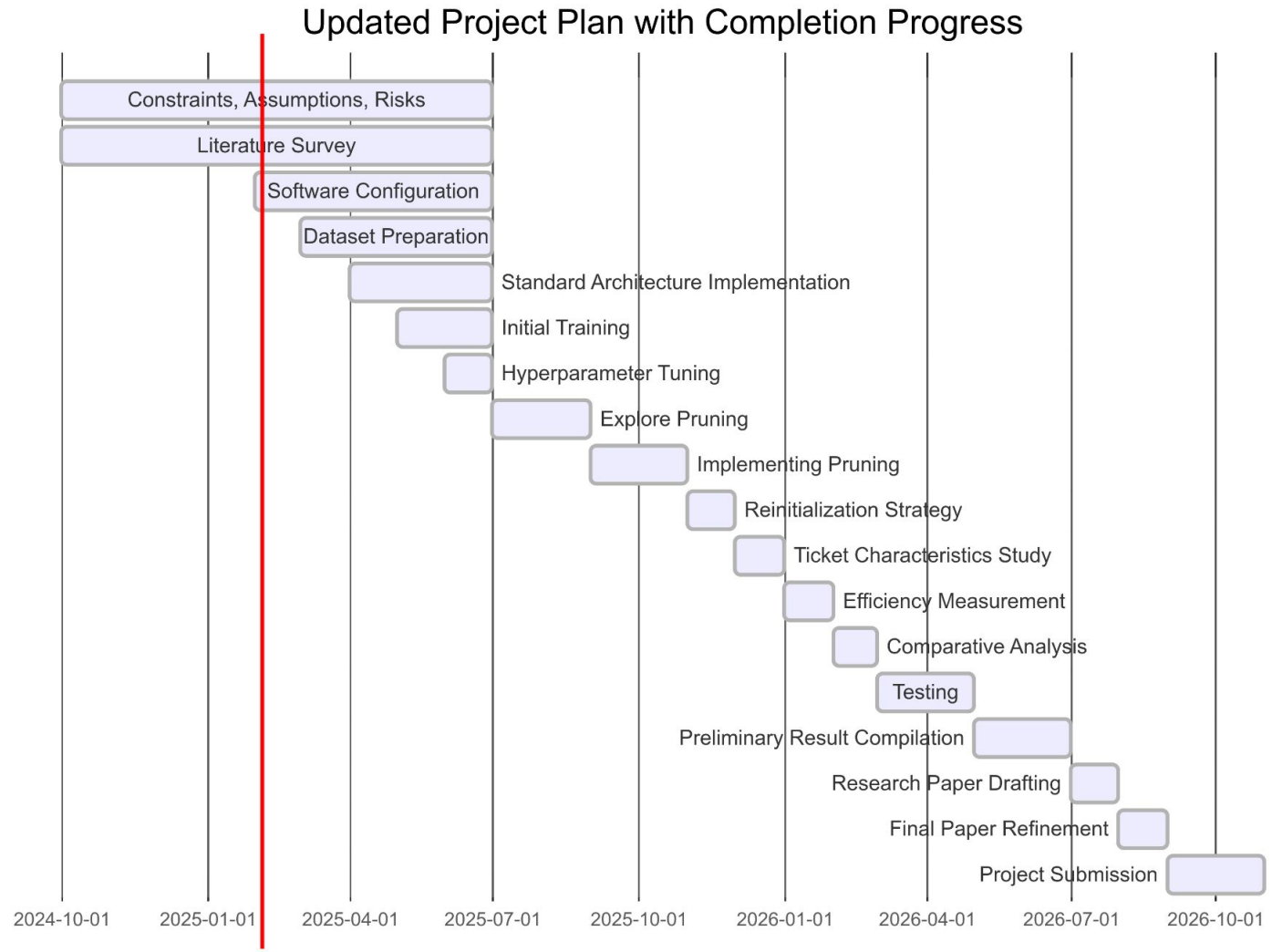
Master Class Diagram

```
---
config:
 theme: base
 themeVariables:
   nodeSpacing: 150
   rankSpacing: 200
 layout: dagre
---
classDiagram
   class ModelTrainingComponent{
     +trainModel()
     +generateInitialWeights()
   }
   class PruningComponent{
     +applyPruning()
     +logSparsity()
   }
   class RetrainingComponent{
     +retrainSubnetwork()
   }
   class EvaluationBenchmarking{
     +evaluatePerformance()
     +compareMetrics()
   }
   class VisualizationReporting{
     +generateVisuals()
     +compileReport()
   }
   class DatasetRepository{
     +getDataset()
     +preprocessData()
   }
   class PruningMetadataLogs{
     +storeLog()
     +retrieveLog()
   }
   class PreTrainedModelRepository{
     +fetchPreTrainedModel()
   }
   ModelTrainingComponent--> DatasetRepository: uses
   ModelTrainingComponent--> PruningComponent: passes weights to
   PruningComponent--> PruningMetadataLogs: logs data to
   PruningComponent--> RetrainingComponent: sends pruned model to
   RetrainingComponent--> EvaluationBenchmarking: outputs retrained model to
   EvaluationBenchmarking--> PreTrainedModelRepository: compares with
   EvaluationBenchmarking--> VisualizationReporting: feeds metrics t
```

Project Progress Plan for Phase 2

- Performed training and pruning successfully for MNIST and CIFAR-10 datasets with LeNet and ResNet-18 models.
- Used different pruning models like MP, IMP and DiffPruning.
- Verified the expected results corroborated by the referred papers.

Project Progress Plan for Phase 2 - Gantt Chart



Project Progress Plan for Phase 2

```
gantt
    title Updated Project Plan with Completion Progress
    dateFormat  YYYY-MM-DD
    Constraints, Assumptions, Risks : 100%, 2024-10-01, 2025-07-01
    Literature Survey : 100%, 2024-10-01, 2025-07-01
    Software Configuration : 100%, 2025-02-01, 2025-07-01
    Dataset Preparation : 100%, 2025-03-01, 2025-07-01
    Standard Architecture Implementation : 100%, 2025-04-01, 2025-07-01
    Initial Training : 100%, 2025-05-01, 2025-07-01
    Hyperparameter Tuning : 90%, 2025-06-01, 2025-07-01
    Explore Pruning : 70%, 2025-07-01, 2025-09-01
    Implementing Pruning : 80%, 2025-09-01, 2025-11-01
    Reinitialization Strategy : 60%, 2025-11-01, 2025-12-01
    Ticket Characteristics Study : 20%, 2025-12-01, 2026-01-01
    Efficiency Measurement : 10%, 2026-01-01, 2026-02-01
    Comparative Analysis : 10%, 2026-02-01, 2026-03-01
    Testing : 10%, 2026-03-01, 2026-05-01
    Preliminary Result Compilation : 10%, 2026-05-01, 2026-07-01
    Research Paper Drafting : 10%, 2026-07-01, 2026-08-01
    Final Paper Refinement : 10%, 2026-08-01, 2026-09-01
    Project Submission : 10%, 2026-09-01, 2026-11-01
```

References

- [1] J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," arXiv preprint arXiv:1803.03635, 2018. [Online]. Available: https://arxiv.org/abs/1803.03635
- [2] C. Jiang, B. Hui, B. Liu, and D. Yan, "Successfully Applying Lottery Ticket Hypothesis to Diffusion Model," arXiv preprint arXiv:2310.18823, 2023. [Online]. Available: https://arxiv.org/abs/2310.18823.
- [3] G. Fang, X. Ma, and X. Wang, "Successfully Applying Lottery Ticket Hypothesis to Diffusion Model," in *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023. [Online]. Available: https://papers.neurips.cc/paper_files/paper/2023/file/35c1d69d23bb5dd6b9abcd68be005d5c-Paper-Conference.pdf

Any other information

Provide any other information you wish to add on.

Note: Changes can be made in the template, with the consent of the guide for inclusion of any other information.

Thank You