Alireza Limooie

Geometric Noise Augmentation for Roof Graph Reconstruction

Date: 24/07/2025

1. Objective & Background

This report documents the continuation of efforts to solve the iterative refinement divergence identified in the previous report (21/07/2025). The primary objective was to validate the hypothesis that the static, initial confidence score was misleading the model in subsequent refinement iterations. This was tested by training and evaluating a "confidence-agnostic" model variant.

Concurrently, we investigated the impact of the model's output head architecture on performance. With insights from these experiments, a new, more sophisticated training paradigm—**Staged Deep Supervision**—is proposed. This new methodology aims to create a model with specialized layers, where shallower layers learn coarse, large-scale corrections and deeper layers learn fine-grained, precise adjustments, directly addressing the requirements of an iterative refinement process.

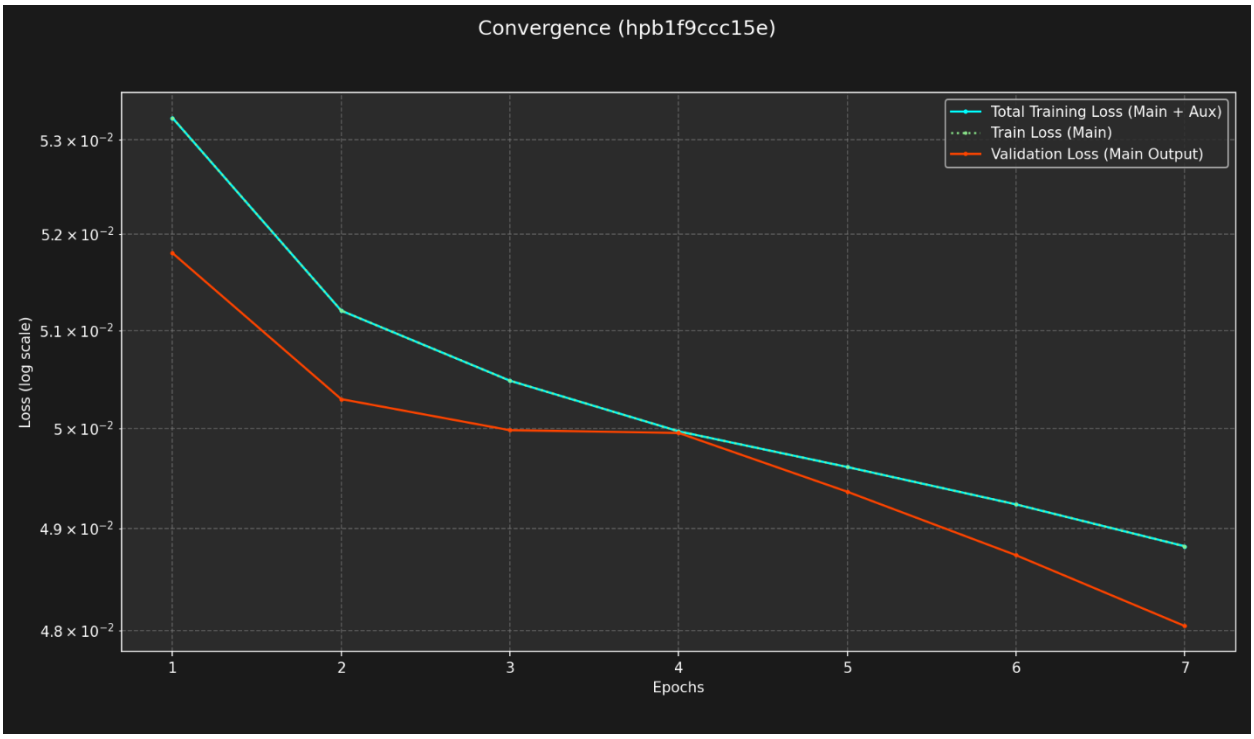
2. Experimental Results & Analysis

This work period focused on two key experiments: removing the confidence score feature and exploring different MLP head structures.

2.1. The Confidence-Agnostic Model: Validating the Hypothesis

Following the action plan from the previous report, a new model variant was trained from scratch with the --include_confidence flag disabled. This forced the model to rely exclusively on geometric and topological context to infer the quality and structure of the input graph.

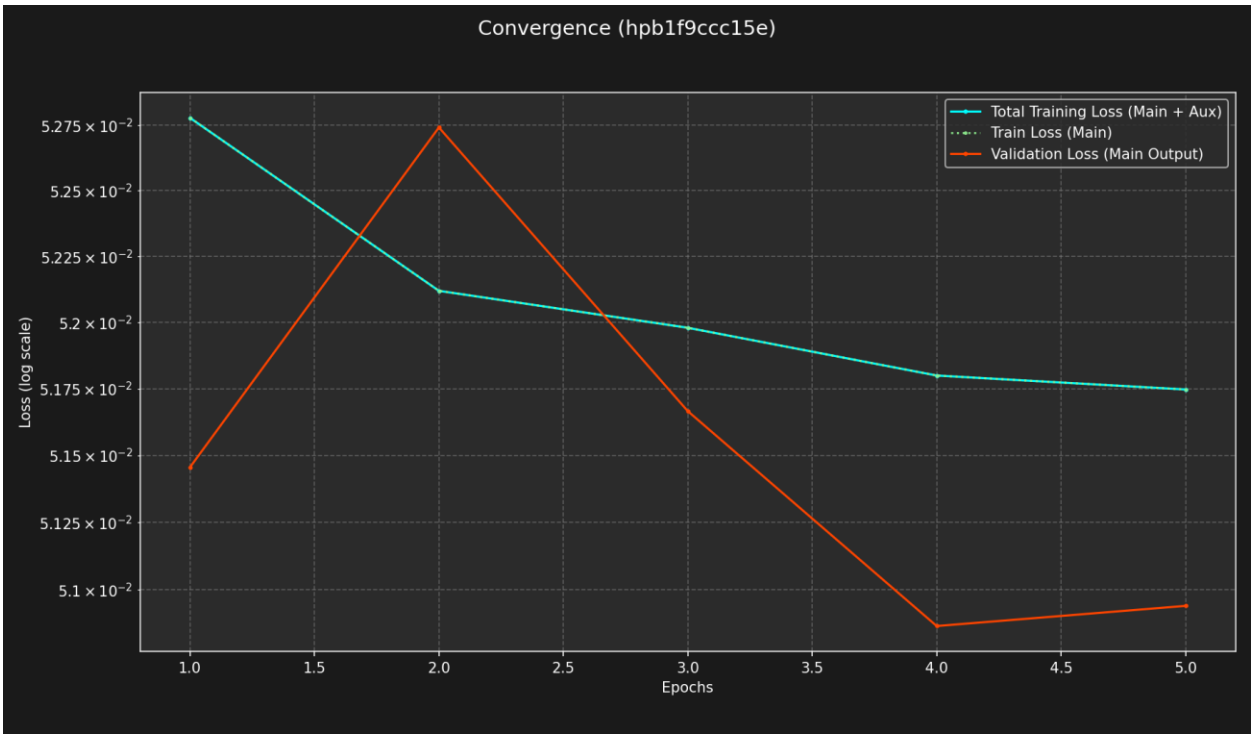
- **Results:** After 7 epochs of training, the confidence-agnostic model demonstrated stable behavior in the iterative refinement task.



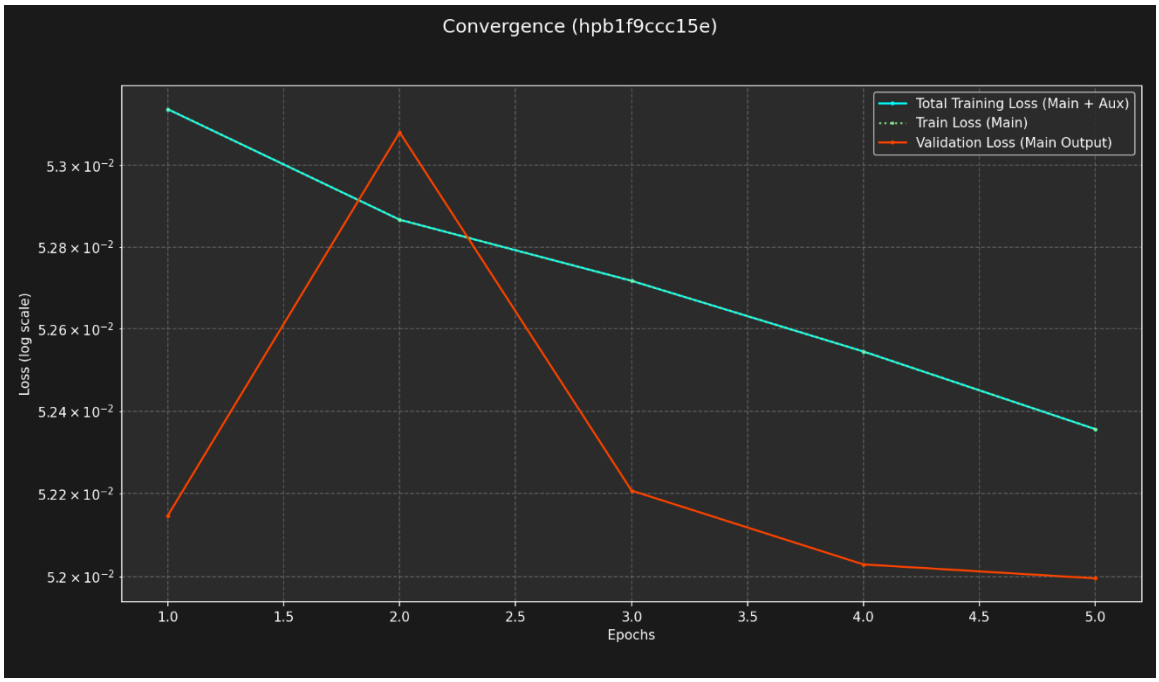
- **2.2. Exploring Output Head Architectures:**

A parallel experiment was conducted to determine if a more complex output head could improve performance. The default model uses a single linear layer to project the d_model (256) dimensional transformer output to the 4 target coordinates. We hypothesized that adding intermediate non-linear layers might allow the model to learn a more complex mapping. Several configurations were tested using the --mlp_head_dims and --mlp_head_activations arguments:

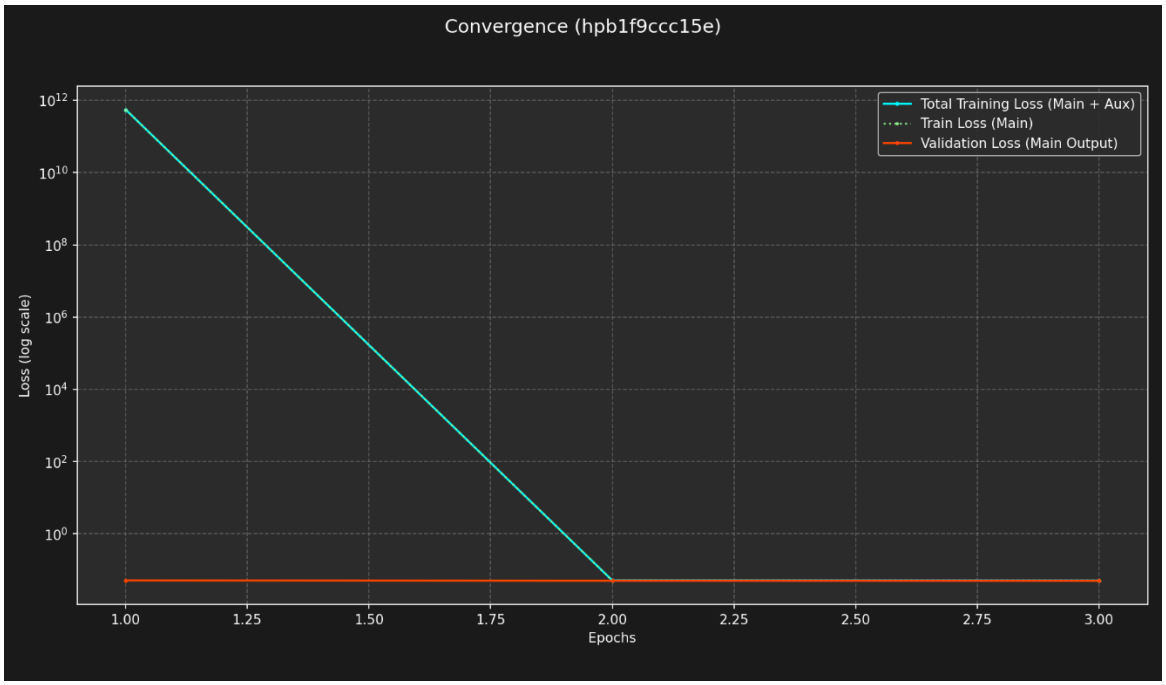
- --mlp_head_dims 64 16 --mlp_head_activations relu relu (256 -> 64 -> 16 -> 4)



- --mlp_head_dims 32 --mlp_head_activations relu (256 -> 32 -> 4)



- --mlp_head_dims 16 --mlp_head_activations none (A smaller linear projection: 256 -> 16 -> 4)



- **Results:** Contrary to our expectations, all tested configurations performed worse than the simple 256 -> 4 linear head. The training was less stable, and the final validation loss was higher. The deeper, non-linear heads seemed to struggle to converge effectively. Even though the model's weight initialization was updated to use Kaiming (He) initialization, which is better suited for deep, ReLU-activated networks, performance did not improve.

2.3. Analysis of Current Bottlenecks

While progress has been significant, the experimental cycle is slow. Each training run, even for a limited number of epochs, takes a substantial amount of time (over 30 minutes per epoch on available hardware). This bottleneck makes exhaustive hyperparameter tuning impractical and necessitates the development of more targeted and efficient training strategies.

3. A New Methodological Proposal: Staged Deep Supervision

The core challenge of iterative refinement is that the model must be adept at handling inputs of vastly different quality—from heavily corrupted initial data to nearly perfect graphs in later iterations. A single training objective struggles to optimize for both scenarios simultaneously. To address this, we propose a new training methodology: **Staged Deep Supervision**.

3.1. The Rationale

The idea is to structure the model itself to mirror the refinement process. We will train the model in stages, where each stage adds depth and specializes in a finer level of correction.

- **Shallow layers** will be trained to perform coarse, large-scale corrections on highly noisy inputs.
- **Deeper layers** will be trained to perform fine, precise adjustments on already-clean inputs.

This is made possible by the model's new capability to generate auxiliary outputs from intermediate encoder layers, each with its own, separately weighted loss function, configurable via the --aux_output_layers and --aux_loss_target_vals arguments.

3.2. The Implementation Plan

The training will proceed in two distinct stages:

- **Stage 1: Coarse Correction Training**
    1. **Model Configuration:** A model with a small number of encoder layers will be instantiated (e.g., --num_layers 3).
    2. **Loss Configuration:** The loss will be balanced for a large error margin. We will use --balance-losses with a high main_loss_target_val (e.g., 1.0) and a large balance_distance (e.g., 0.5). This teaches the initial layers to focus on bringing distant endpoints closer, without penalizing small errors heavily.
    3. **Training:** Train this shallow model until convergence.
- **Stage 2: Fine Refinement Training**
    1. **Model Configuration:** Load the weights from the converged Stage 1 model. Freeze the weights of the initial 3 layers (param.requires_grad = False). Increase the total number of layers (e.g., --num_layers 6).
    2. **Loss Configuration:** Configure two loss outputs.
        - A new **main output** from the final new layer (layer 6). This main output will be trained with a much stricter loss target: a low main_loss_target_val (e.g., 0.05) and a small balance_distance (e.g., 0.1).

3. **Training:** Train the model, where gradients will only update the newly added layers (4, 5, and 6). This stage will exclusively teach the new layers how to perform precise, small-scale adjustments on inputs that are already expected to be mostly correct.

3.3. Expected Benefits

This staged approach is expected to yield several advantages:

1. **Specialization:** Creates a model with layers explicitly specialized for different scales of correction.
2. **Improved Iterative Performance:** The model should be inherently better at the refinement task, as its architecture and training directly mirror the multi-step process.
3. **Training Efficiency:** Each stage focuses on a simpler, more targeted task, which may lead to faster convergence per stage and a more effective overall model.

4. Immediate Next Steps & Future Work

The immediate priority is to implement and validate the proposed Staged Deep Supervision methodology.

- **Primary Action:** Execute the two-stage training plan outlined in Section 3.2.
    1. Train a Stage 1 model (3 layers, coarse loss target).
    2. Use the Stage 1 model as a base to train a Stage 2 model (6 layers total, fine loss target on new layers).
- **Future Direction:** If the Staged Deep Supervision model proves successful, it will become the new foundational architecture. Future work would then involve optimizing this paradigm, such as exploring the ideal loss targets for each stage, and potentially adding a third, even more precise refinement stage.