mGPS Algorithm Optimization

Course: Bioinformatics Research Project (BINP37), 15 credits

Student: Chandrashekar CR

(email: ch1131ch-s@student.lu.se)

Supervisor: Eran Elhaik

(email: eran.elhaik@biol.lu.se)

Lund University 2025

1. Materials and Methods

1.1 Dataset and Preprocessing

We utilized the MetaSUB dataset from the original mGPS study (Zhang et al., 2024), accessed via their GitHub repository. This dataset comprises 4,070 quality-controlled samples collected from subway stations in 40 cities across 7 continents between 2016 and 2017. Each sample contains taxonomic profiles with relative sequence abundances, generated by subsampling to 100,000 classified reads and processed using KrakenUniq with the NCBI/RefSeq Microbial database (Danko et al., 2021).

To maintain methodological consistency with previous mGPS work, we applied the same quality control and feature selection procedures. Specifically, cities with fewer than eight samples were excluded, and recursive feature elimination (RFE) with Random Forest was used to reduce the initial set of approximately 3,000 microbial features to the 200–300 most informative, using 5-fold cross-validation (Guyon et al., 2002). Class imbalance—particularly for underrepresented continents such as Oceania and Africa—was addressed using the Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002), achieving a 1:3 ratio between minority and majority classes. These steps ensured that our dataset and preprocessing pipeline remained directly comparable to the original mGPS study.

1.2 Model Development

We developed several modeling approaches to address the hierarchical geographic prediction problem, each offering distinct advantages and characteristics.

1.2.1 Neural Networks

Neural networks were chosen as a core modeling approach due to their capacity to learn complex, non-linear relationships and, crucially, their scalability with increasing data size (LeCun et al., 2015). The primary motivation was to develop a robust model that could not only perform well on the current dataset but also generalize effectively as more data becomes available in the future. This makes neural networks particularly suitable for scenarios where data volume is expected to grow, ensuring the modeling framework remains adaptable and performant.

Separate Neural Network Models In accordance with the previous study, which utilized a hierarchical approach with XGBoost (Zhang et al., 2024)(Chen and Guestrin, 2016), we constructed a set of independent neural networks to serve as baselines and to analyze error propagation at each prediction level. Specifically, we developed three specialized models: (1) a Continent Network that predicts continent labels from microbial features; (2) a City Network that incorporates both microbial features and continent probabilities to predict city labels; and (3) a Coordinate Network that leverages microbial features, continent, and city probabilities to perform coordinate regression.

Default parameters and the hyperparameter search space for these models are provided in Supplementary Tables 5 and 6.

Each network architecture follows a progressive dropout, a batch normalization, and ReLU activation functions.

Coordinate Transformation for Geographical Prediction: To appropriately model the spherical geometry of the Earth and avoid issues such as gradient explosion, vanishing gradients, and improper scaling, we transform latitude (ϕ) and longitude (λ) into 3D Cartesian coordinates for all neural network-based coordinate prediction models (Snyder, 1987; Aydin et al., 2016). This transformation ensures that points close on the globe (e.g., near the $-180°/+180°$ longitude boundary) are also close in the transformed space, which is not the case if standard scaling is applied directly to latitude and longitude. The transformation is defined as:

$$\begin{aligned} x &= \cos(\phi)\cos(\lambda) \\ y &= \cos(\phi)\sin(\lambda) \\ z &= \sin(\phi) \end{aligned} \tag{1}$$

For evaluation, we apply the inverse transformation to the predicted (x, y, z) values, converting them back to latitude and longitude in radians, and then to degrees. This allows for accurate geodesic error computation and ensures that the model predictions are interpretable in the original coordinate system.

Each neural network in the separate hierarchy is trained independently using a standard loss function for its task:

- **Continent and City Classification:** Cross-entropy loss is used for both continent and city classification tasks. Class weights are optionally applied to address class imbalance:

$$\mathcal{L}_{\text{classification}} = \text{CrossEntropyLoss}(\text{predictions}, \text{targets}, \text{weight} = w_{\text{class}}) \tag{2}$$

- **Coordinate Regression:** Mean squared error (MSE) loss is used for coordinate regression:

$$\mathcal{L}_{\text{regression}} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 \tag{3}$$

Each model is trained independently with its respective loss function, and no explicit weighting between tasks is used in this separate approach.

Table 1. Architecture and training parameters for separate neural networks.

Level	Task	Hidden Layers	Dropout	Batch Norm	Learning Rate	Batch Size	Epochs
1	Continent	[128, 64]	0.3–0.7	Yes	1×10^{-3}	128	400
2	City	[256, 128, 64]	0.3–0.7	Yes	1×10^{-3}	128	400
3	Coordinates	[256, 128, 64]	0.2–0.5	Yes	1×10^{-4}	64	600

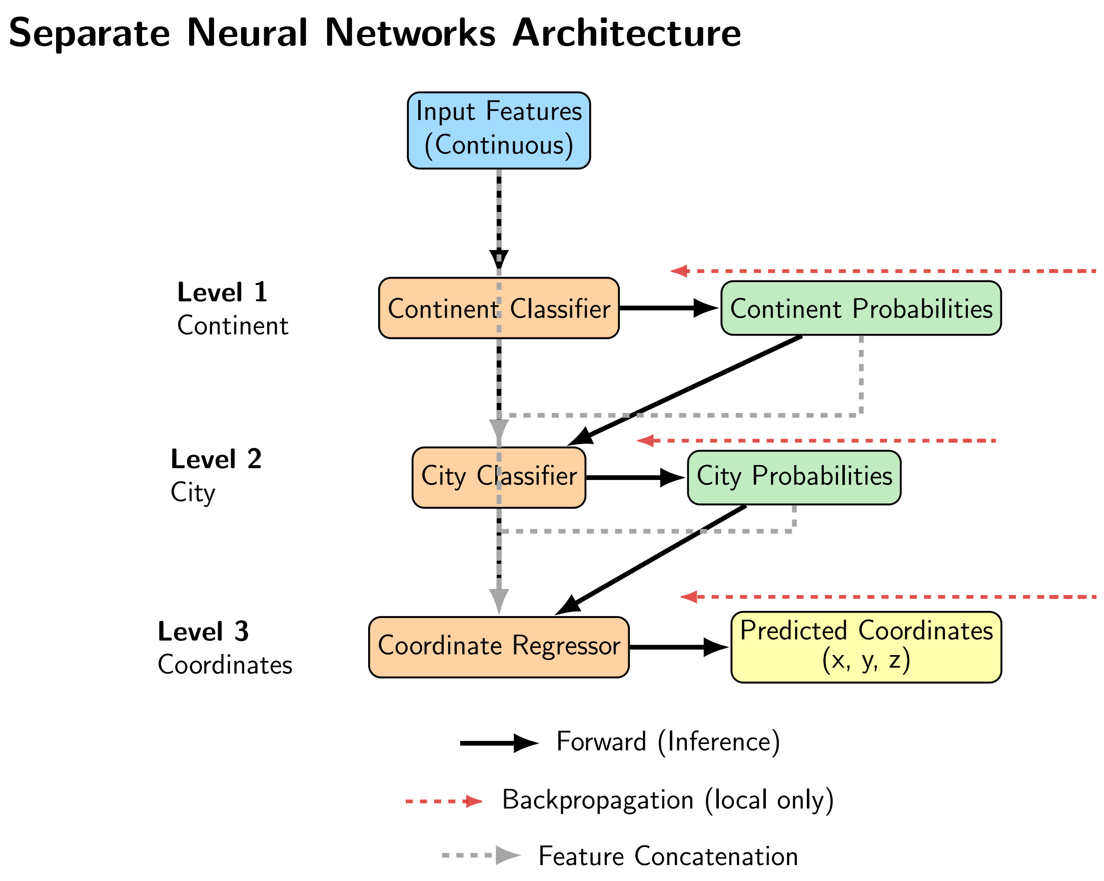

Figure 1. *Schematic of the separate neural network approach for hierarchical geographic prediction. Each prediction level (continent, city, coordinates) is modeled by an independent neural network. Outputs from each level are used as inputs for the next, but training and backpropagation are performed independently for each network.*

For each prediction level, the loss function is computed and backpropagated independently, ensuring that parameter updates for continent, city, and coordinate models remain decoupled.

Combined Neural Networks To enable end-to-end hierarchical learning, we developed the Combined Neural Networks, a unified multi-task neural network architecture with three sequential branches. This model shares feature representations across tasks while maintaining task-specific output heads. Training is performed using a weighted multi-task loss, combining cross-entropy for classification tasks and mean squared error (MSE) for coordinate regression. As with the separate models, coordinate prediction in this architecture also employs the Cartesian transformation described in Equation 1 (Snyder, 1987; Aydin et al., 2016).

Default parameters and the hyperparameter search space for the Combined Neural Networks are provided in Supplementary Tables 7 and 8.

The total weighted loss for the combined neural network is defined as:

$$\mathcal{L}_{\text{total}} = w_1\mathcal{L}_{\text{continent}} + w_2\mathcal{L}_{\text{city}} + w_3\mathcal{L}_{\text{coordinate}} \tag{4}$$

where w_1, w_2, w_3 are the task-specific weights. This joint optimization strategy encourages the model to learn representations that are robust to error propagation by penalizing errors at higher levels more strongly, reflecting the hierarchical structure of the problem. During backpropagation, gradients flow through all branches, but their magnitudes are modulated by these weights, promoting robust feature learning across the hierarchy.

Table 2. Architecture and training parameters for Combined Neural Networks.

Branch	**Hidden Layers**	**Dropout**	**Batch Norm**	**Loss**	**Learning Rate**
Continent	[128, 64]	0.3–0.7	Yes	Cross-entropy	1×10^{-3}
City	[256, 128, 64]	0.3–0.7	Yes	Cross-entropy	1×10^{-3}
Coordinates	[256, 128, 64]	0.2–0.5	Yes	MSE	1×10^{-3}

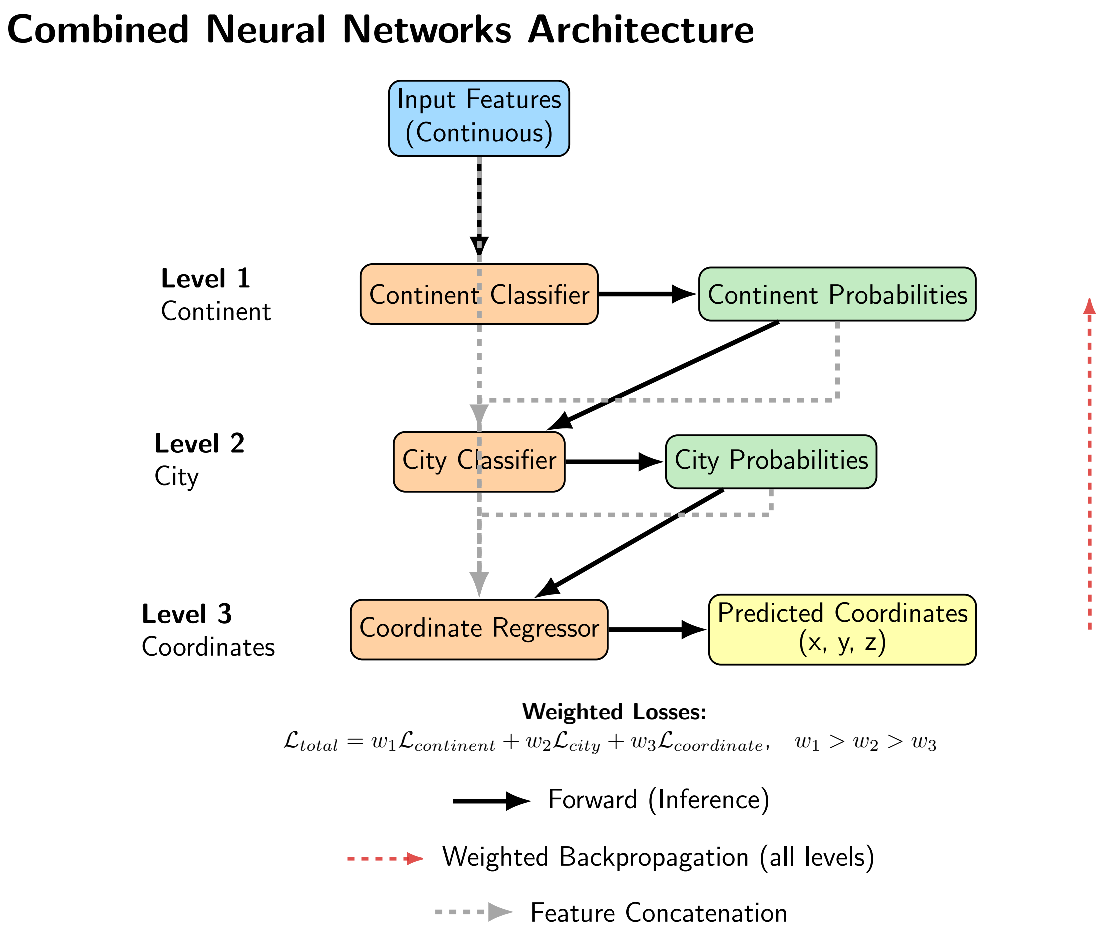
Combined Neural Networks Architecture
Input Features (Continuous)
Level 1
Continent
Continent Classifier
Continent Probabilities
Level 2
City
City Classifier
City Probabilities
Level 3
Coordinates
Coordinate Regressor
Predicted Coordinates (x, y, z)
Weighted Losses:
$\mathcal{L}_{total} = w_1\mathcal{L}_{continent} + w_2\mathcal{L}_{city} + w_3\mathcal{L}_{coordinate}, \quad w_1 > w_2 > w_3$
Forward (Inference)
Weighted Backpropagation (all levels)
Feature Concatenation

Figure 2. *Diagram of the Combined Neural Networks architecture. This unified multi-task neural network consists of sequential branches for continent, city, and coordinate prediction. Feature representations are shared, and predictions from higher levels are concatenated with features for downstream tasks. Training uses a weighted multi-task loss to reflect the hierarchy.*

In the separate neural network approach, each model is trained independently and the loss is propagated only within that level of the hierarchy, which limits the ability for the model to learn shared representations and may lead to error propagation. Alternatively, the combined neural network architecture provides end-to-end hierarchical learning, in which the loss function is propagated through the entire hierarchy. Joint optimization allows gradients to flow through all levels and encourages the model to learn feature representations that will minimize errors not just locally, but throughout the hierarchy. Therefore, the combined neural network approach is better equipped to handle task-level dependencies and limit errors compounding, which presented better performance overall.

1.2.2 GrowNet Architecture

GrowNet was selected as it represents a state-of-the-art neural network-based boosting framework for classification and regression on tabular data. Its design allows it to match or exceed the performance of leading models such as XGBoost, while leveraging the flexibility of neural networks as weak learners (Feng et al., 2021). This makes GrowNet a strong candidate for hierarchical, multi-task problems where both accuracy and model adaptability are critical.

GrowNet is a gradient boosting framework that employs neural networks as weak learners for multi-task learning (Feng et al., 2021). The algorithm proceeds by sequentially adding shallow neural networks to the ensemble, each trained to correct the residuals (pseudo-residuals) of the previous learners, analogous to boosting in XGBoost (Chen and Guestrin, 2016). At each stage m, the pseudo-residuals $\mathbf{r}^{(m)}$ are computed as the negative gradient of the loss with respect to the current ensemble prediction, i.e., $\mathbf{r}^{(m)} = -\nabla_{F^{(m-1)}}\mathcal{L}(y, F^{(m-1)})$. Each weak learner h_m is then trained to fit these residuals.

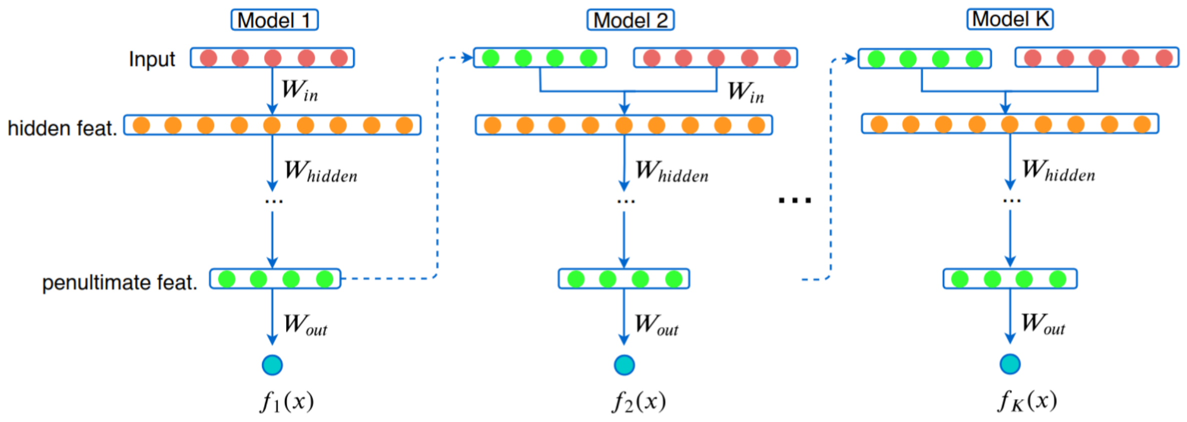

Figure 3. *Diagram of the GrowNet architecture. This framework utilizes a multi-task learning approach with neural networks as weak learners, enabling effective handling of hierarchical tasks.*

The hierarchical GrowNet training algorithm proceeds as follows:

1. **Input:** Training data $\{(\mathbf{x}_i, \mathbf{y}_{c,i}, \mathbf{y}_{city,i}, \mathbf{y}_{coord,i})\}_{i=1}^{N}$, hyperparameters M (number of stages), ρ (learning rate), λ (optimizer step size), and epochs_per_stage.

2. Initialize baseline predictions $F^{(0)}$.

3. For $m = 1$ to M:

   (a) Compute pseudo-residuals $\mathbf{r}^{(m)} = -\nabla_{F^{(m-1)}}\mathcal{L}(y, F^{(m-1)})$.

   (b) Initialize a new weak learner h_m.

   (c) For each epoch in epochs_per_stage:

      i. Sample a mini-batch B.

      ii. Compute gradients and update h_m parameters using $\nabla_\theta \mathcal{L}_{residual}(B; h_m)$.

   (d) Update ensemble: $F^{(m)} = F^{(m-1)} + \rho \cdot h_m$.

(e) Periodically, jointly fine-tune all weak learners via corrective optimization:

$$\{\theta_1, \ldots, \theta_m\} \leftarrow \arg\min_{\{\theta_i\}} \mathcal{L}_{\text{total}}(F^{(m)}; \{\theta_i\}_{i=1}^{m}) \tag{5}$$

(f) Evaluate on validation data and apply early stopping if necessary.

4. Return the final ensemble $\mathcal{F} = \{h_1, \ldots, h_M\}$.

Here, $F^{(m)}$ is the current ensemble prediction, h_m is the m-th weak learner, ρ is the learning rate, and $\mathcal{L}_{\text{total}}$ is the composite loss function (see Equation 4). Pseudo-residuals represent the direction and magnitude by which the current model's predictions should be adjusted to minimize the loss. The corrective optimization step enables earlier weak learners to adapt based on information acquired by subsequent learners, enhancing ensemble coherence and predictive performance.

In simple terms, GrowNet builds an ensemble of neural networks, each one learning to correct the mistakes of the previous ones. At each stage, the model computes how much its current prediction is wrong (the pseudo-residual), fits a new neural network to these errors, and adds it to the ensemble. This process continues for several stages, and occasionally all networks are jointly fine-tuned to further reduce the overall error. This approach allows GrowNet to combine the flexibility of neural networks with the boosting principle, resulting in strong performance for hierarchical, multi-task problems.

1.2.3 Ensemble Learning

Intuition for Model Selection in Ensemble Learning: The ensemble was constructed with careful consideration of each model's strengths and the overarching research objective of minimizing hierarchical error. State-of-the-art tree-based models (XGBoost, LightGBM, CatBoost) were included due to their proven effectiveness on tabular data and their success in classification and regression tasks (Chen and Guestrin, 2016)(Ke et al., 2017) (Prokhorenkova et al., 2018)(Grinsztajn et al., 2022). TabPFN, a transformer-based model pre-trained for small to medium tabular datasets, was incorporated for its superior performance in such settings (Hütter et al., 2022). Neural networks and GrowNet were retained in the ensemble to ensure scalability and robustness, especially as data volume increases (Feng et al., 2021)(LeCun et al., 2015)(Caruana et al., 2008)(Tang, 2024). This diverse selection ensures that the ensemble can adapt to varying data regimes and leverages the unique advantages of each model family.(Dietterich, 2000)(Opitz and Maclin, 1999)(Erickson et al., 2025)

Threshold Filtering and Best Model Selection: A key feature of the ensemble framework is the use of threshold filtering and best model selection, tailored to the nature of each prediction task. At each layer in the hierarchy, models are required to surpass predefined accuracy thresholds to be included in the ensemble. This ensures that only high-performing models contribute to the final predictions, enhancing the robustness and reliability of the ensemble. As the dataset grows, traditional gradient boosting models may struggle to scale, whereas neural networks and GrowNet can take over due to their scalability. (Tang, 2024) (Caruana et al., 2008) This dynamic selection mechanism allows the ensemble to maintain optimal performance across different data scenarios.

For classification tasks (continent and city prediction), each model may excel at different subsets of classes due to varying inductive biases and training dynamics. To leverage this diversity, the ensemble collects predicted probabilities from all high-performing models and passes them through a meta-model (XGBoost). The meta-model is trained to recognize which base models are most reliable for specific classes, resulting in more refined and robust predictions. This approach is particularly effective when some models are better at capturing certain classes (e.g., specific continents or cities) than others, allowing the ensemble to achieve superior overall classification performance.

For the coordinate regression task (Layer 3), only the single best-performing model is selected for final predictions, rather than combining outputs from multiple models. This is because regression outputs are continuous and granular; averaging or stacking predictions from multiple models can sometimes degrade performance by smoothing out strong individual predictions, especially when one model clearly outperforms the others (Dietterich, 2000; Opitz and Maclin, 1999). Therefore, it is advisable to use the best model for coordinate prediction to preserve the highest possible accuracy.

Default parameters and the hyperparameter search space for the ensemble meta-models and all base models are provided in Supplementary Tables 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.

Model Selection and Integration The ensemble incorporates the following model families:

- **Gradient Boosting Models:** XGBoost (Chen and Guestrin, 2016) (see Supplementary Tables 11, 12), LightGBM (Ke et al., 2017) (see Supplementary Tables 13, 14), and CatBoost (Prokhorenkova et al., 2018) (see Supplementary Tables 15, 16), which are highly effective for capturing non-linear relationships in tabular data.

- **TabPFN:** A state-of-the-art prior-data fitted neural network specifically designed for small-to-medium tabular datasets (Hütter et al., 2022) (see Supplementary Table 21). TabPFN leverages meta-learning to rapidly adapt to new tasks, making it particularly suitable for our problem setting.

- **Neural Networks:** Standard multilayer perceptrons (MLPs) and hierarchical variants (see Supplementary Tables 19, 20), included for their capacity to model complex feature interactions, especially as dataset size increases.

- **GrowNet:** The aforementioned gradient boosting neural network architecture (see Supplementary Tables 17, 18), included as a robust alternative for scenarios with larger datasets or more intricate relationships.

Machine learning models were prioritized due to their strong empirical performance on tabular datasets (Grinsztajn et al., 2022). Neural network-based models and GrowNet were included as flexible alternatives for scenarios requiring greater model capacity.

Hierarchical Ensemble Architecture The ensemble is structured into three layers, each corresponding to a level in the geographic hierarchy:

Layer 1: Continent Classification Multiple base models predict continent probabilities from microbial features. Models are filtered based on cross-validation accuracy (threshold: 93%). SMOTE is applied to address class imbalance. Retained models generate out-of-fold predictions via 5-fold cross-validation, which are then used as meta-features for an XGBoost meta-model.

Layer 2: City Classification City prediction utilizes both the original microbial features and continent probability outputs from Layer 1. Models surpassing a 91% accuracy threshold are included in meta-learning, following the same protocol as Layer 1.

Layer 3: Coordinate Prediction Coordinate prediction leverages the full feature set: microbial abundances, continent probabilities, and city probabilities. Two approaches are considered:

- **Tree-based Models:** Latitude is predicted first, followed by longitude conditioned on the predicted latitude.

- **Neural Networks:** Direct prediction of 3D Cartesian coordinates 1 (Snyder, 1987; Aydin et al., 2016), which are subsequently converted to latitude and longitude.

The model with the lowest median Haversine distance error is selected for final predictions; no meta-model is used at this stage.

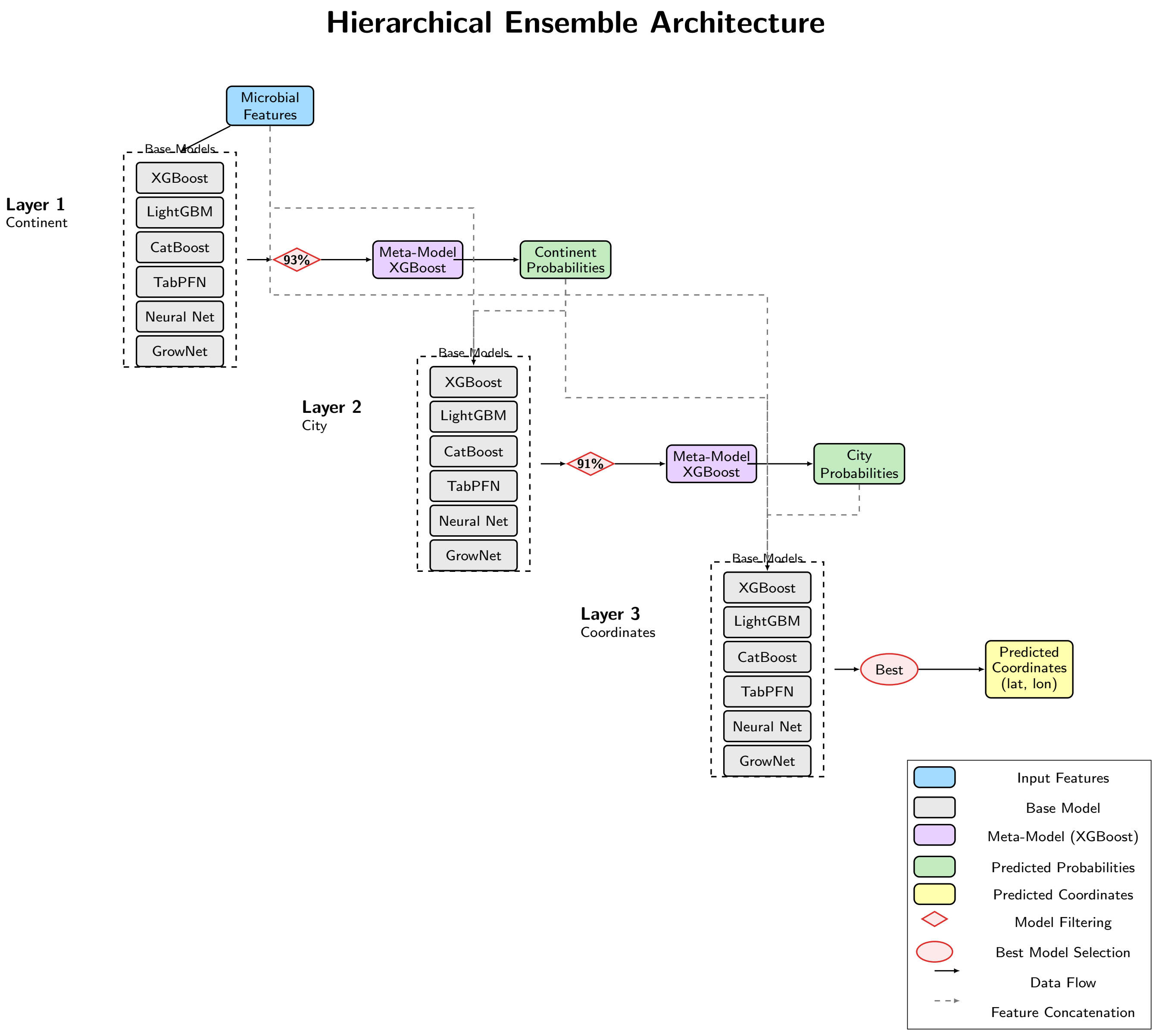

Figure 4. *Overview of the hierarchical ensemble learning workflow. The ensemble is organized in three layers: continent classification, city classification, and coordinate regression. At each stage, predictions from multiple base models are combined using meta-models, and probability outputs are used as augmented features for subsequent layers.*

Table 3. Meta-model configuration parameters.

Parameter	Continent Meta-Model	City Meta-Model
Algorithm	XGBoost	XGBoost
Objective	Multi-class log-loss	Multi-class log-loss
Max depth	3	4
Learning rate	0.1	0.1
N-estimators	100	150
Subsample	0.8	0.8
Colsample bytree	0.8	0.8

Feature Augmentation and Data Flow The hierarchical ensemble implements systematic feature augmentation at each stage:

$$X_{cont} = \text{RFE}(X_{microbial}) \tag{6}$$

$$\hat{P}_{cont} = \text{MetaModel}_{cont}(\{f_i(X_{cont})\}_{i=1}^{N}) \tag{7}$$

$$X_{city} = [X_{cont}; \hat{P}_{cont}] \tag{8}$$

$$\hat{P}_{city} = \text{MetaModel}_{city}(\{f_j(X_{city})\}_{j=1}^{M}) \tag{9}$$

$$X_{coord} = [X_{cont}; \hat{P}_{cont}; \hat{P}_{city}] \tag{10}$$

$$\hat{Y}_{coord} = f_{best}(X_{coord}) \tag{11}$$

Table 4. Ensemble layer specifications and selection criteria.

Layer	Input Features	Selection Threshold	Meta-Model
Continent	Microbial (200-300)	93% accuracy	XGBoost
City	Microbial + continent probabilities	91% accuracy	XGBoost
Coordinates	Microbial + all probabilities	Best median distance	None

Training Protocol and Meta-Learning Ensemble training proceeds in three stages:

- **Stage 1: Model Filtering and Meta-Feature Generation.** All base models undergo 5-fold stratified cross-validation to generate out-of-fold predictions. Out-of-fold predictions are generated by training each model on $k-1$ folds and predicting on the held-out fold, ensuring that predictions for each sample are made by a model that has not seen that sample during training. This prevents information leakage and provides unbiased meta-features for the meta-model. Only models meeting predefined performance thresholds are retained for meta-learning.

- **Stage 2: Hyperparameter Optimization.** Retained models are further optimized using Bayesian optimization (Optuna (Akiba et al., 2019)) with model-specific search spaces.

- **Stage 3: Meta-Model Training.** XGBoost meta-models are trained on concatenated probability outputs from the selected base models, enabling a learned ensemble strategy that outperforms simple averaging. In the final protocol, after hyperparameter tuning, the best models are retrained on the full training set and used to generate predictions on the test set. These predictions are then used as input features for the meta-model, which is trained to combine them optimally.

Suppose we have 1000 samples and use 5-fold cross-validation. For each base model, we train on 800 samples and predict on the 200 held-out samples, repeating this for all folds. This gives us out-of-fold predictions for all 1000 samples, which are then used as features for the meta-model. The meta-model learns to combine the strengths of each base model, improving overall performance. This stacking approach ensures that the meta-model does not overfit to the training data and can generalize well to new, unseen samples [REF-about-oof-logic].

1.3 Error Propagation and Geodesic Error Calculation

To provide a more nuanced understanding of coordinate prediction error, we compute the expected coordinate error $E[D]$ as a weighted sum over all possible combinations of continent and city prediction correctness:

$$E(D) = P_{cc,zc}\,E_{cc,zc} + P_{cc,zi}\,E_{cc,zi} + P_{ci,zc}\,E_{ci,zc} + P_{ci,zi}\,E_{ci,zi} \tag{12}$$

where:

- $P_{cc,zc} = P(C = C^*, Z = Z^*)$ is the probability of predicting both the correct continent and correct city,

- $P_{cc,zi} = P(C = C^*, Z \neq Z^*)$ is the probability of predicting the correct continent but incorrect city,

- $P_{ci,zc} = P(C \neq C^*, Z = Z^*)$ is the probability of predicting the incorrect continent but correct city,

- $P_{ci,zi} = P(C \neq C^*, Z \neq Z^*)$ is the probability of predicting both the incorrect continent and incorrect city,

- $E_{cc,zc} = E(D|C = C^*, Z = Z^*)$ is the expected geodesic error when both continent and city are correct,
- $E_{cc,zi} = E(D|C = C^*, Z \neq Z^*)$ is the expected error when continent is correct but city is incorrect,
- $E_{ci,zc} = E(D|C \neq C^*, Z = Z^*)$ is the expected error when continent is incorrect but city is correct,
- $E_{ci,zi} = E(D|C \neq C^*, Z \neq Z^*)$ is the expected error when both continent and city are incorrect.

This decomposition quantifies how errors at the continent and city levels propagate to the final coordinate prediction.

Geodesic Error Calculation (Haversine Formula) Geodesic error is computed as the great-circle distance between predicted and true coordinates using the Haversine formula:

$$d = 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \tag{13}$$

where:

- d is the geodesic distance (in kilometers),
- R is the Earth's radius (mean value $R = 6371$ km),
- ϕ_1, ϕ_2 are the latitudes (in radians) of the true and predicted points,
- λ_1, λ_2 are the longitudes (in radians) of the true and predicted points,
- $\Delta\phi = \phi_2 - \phi_1$,
- $\Delta\lambda = \lambda_2 - \lambda_1$.

This formula accurately measures the shortest distance over the Earth's surface between two points, and is used throughout this work to quantify spatial prediction error.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631.

Aydin, C. C., Demir, C., and Yilmaz, E. (2016). Capability of artificial neural network for forward conversion of geodetic coordinates (phi, lambda, h) to cartesian (x,y,z) coordinates. *Environmental Earth Sciences*, 75(7):1–10.

Caruana, R., Karampatziakis, N., and Yessenalina, A. (2008). An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 96–103, New York, NY, USA. Association for Computing Machinery.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Danko, D., Bezdan, D., Afshin, E. E., Ahsanuddin, S., Bhattacharya, C., Butler, D. J., Chng, K. R., Donnellan, D., Hecht, J., Jackson, K., Kuchin, K., Karasikov, M., Lyons, A., Mak, L., Meleshko, D., Mustafa, H., Mutai, B., Neches, R. Y., Ng, A., Nikolayeva, O., Nikolayeva, T., Png, E., Ryon, K. A., Sanchez, J. L., Shaaban, H., Sierra, M. A., Thomas, D., Young, B., Abudayyeh, O. O., Alicea, J., Bhattacharyya, M., Blekhman, R., Castro-Nallar, E., Cañas, A. M., Chatziefthimiou, A. D., Crawford, R. W., De Filippis, F., Deng, Y., Desnues, C., Dias-Neto, E., Dybwad, M., and Elhaik, E. (2021). A global metagenomic map of urban microbiomes and antimicrobial resistance. *Cell*, 184(13):3376–3393.e17.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *Multiple Classifier Systems*, 1857:1–15.

Erickson, N., Purucker, L., Tschalzev, A., Holzmüller, D., Desai, P. M., Salinas, D., and Hutter, F. (2025). Tabarena: A living benchmark for machine learning on tabular data.

Feng, J., Wang, Y., Wang, Y., Wang, Y., and Liu, Y. (2021). Grownet: Refuel boosting with concatenation and forward propagation. In *Advances in Neural Information Processing Systems*, volume 34, pages 22237–22249.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on tabular data?

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1):389–422.

Hütter, F., Zimmer, L., Probst, P., Hees, J., Krämer, N., and Hutter, F. (2022). Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, pages 3146–3154.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). Catboost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31:6638–6648.

Snyder, J. P. (1987). *Map Projections—A Working Manual*. U.S. Geological Survey Professional Paper 1395. U.S. Government Printing Office, Washington, DC.

Tang, L. (2024). Comparison the performances for distributed machine learning: Evidence from xgboost and dnn. *Applied and Computational Engineering*, 103:209–215.

Zhang, Y., McCarthy, L., Ruff, E., and Elhaik, E. (2024). Microbiome geographic population structure (mgps) detects fine-scale geography. *Genome Biology and Evolution*, 16(11):evae209.

2. Supplementary Materials

2.1 Separate Neural Network Parameters

Table 5. Default parameters for separate neural network models

Parameter	**Continent Model**	**City Model**	**Coordinate Model**
Hidden dimensions	[128, 64]	[256, 128, 64]	[256, 128, 64]
Batch normalization	True	True	True
Initial dropout	0.3	0.3	0.2
Final dropout	0.7	0.7	0.5
Learning rate	1e-3	1e-3	1e-4
Weight decay	1e-5	1e-5	1e-5
Batch size	128	128	64
Epochs	400	400	600
Early stopping steps	20	20	30
Gradient clip	1.0	1.0	1.0

Table 6. Hyperparameter search space for neural network tuning

Hyperparameter	**Search Space**
Hidden dimensions	[64], [128], [128, 64], [256, 128, 64], [256, 128], [512, 256, 128, 64]
Initial dropout	0.1 to 0.3
Final dropout	0.5 to 0.8
Learning rate	1e-4 to 1e-2 (log uniform)
Batch size	64, 128, 256
Weight decay	1e-6 to 1e-3 (log uniform)
Gradient clip	0.5 to 2.0

2.2 Combined Neural Network Parameters

Table 7. Default parameters for combined neural network model

Parameter	**Value**
Architecture parameters	
Continent branch hidden dimensions	[128, 64]
City branch hidden dimensions	[256, 128, 64]
Coordinate branch hidden dimensions	[256, 128, 64]
Continent branch dropout (initial, final)	(0.3, 0.7)
City branch dropout (initial, final)	(0.3, 0.7)
Coordinate branch dropout (initial, final)	(0.2, 0.5)
Batch normalization	True
Training parameters	
Learning rate	1e-3
Weight decay	1e-5
Batch size	128
Epochs	600
Early stopping steps	50
Continent loss weight	1.0
City loss weight	0.5
Coordinate loss weight	0.2

Table 8. Hyperparameter search space for combined neural network tuning

Hyperparameter	**Search Space**
Continent branch hidden dimensions	[128, 64] or [256, 128, 64]
City branch hidden dimensions	[128, 64] or [256, 128, 64]
Coordinate branch hidden dimensions	[128, 64] or [256, 128, 64]
Continent dropout initial	0.2 to 0.5
Continent dropout final	0.6 to 0.8
City dropout initial	0.2 to 0.5
City dropout final	0.6 to 0.8
Coordinate dropout initial	0.1 to 0.3
Coordinate dropout final	0.4 to 0.6
Learning rate	1e-4 to 1e-2 (log uniform)
Weight decay	1e-6 to 1e-3 (log uniform)
Batch normalization	True or False
Batch size	64, 128, 256
Continent loss weight	1.0 to 2.0
City loss weight	0.5 to continent_weight
Coordinate loss weight	0.05 to city_weight

2.3 GrowNet Parameters

Table 9. Default parameters for hierarchical GrowNet model

Parameter	**Value**
Architecture parameters	
Hidden size	256
Input feature dimension	200
Coordinate dimension	3
Dropout rates (2 layers)	0.2, 0.4
Boosting parameters	
Number of weak learners	30
Boost rate	0.4
Epochs per stage	20
Corrective epochs	5
Training parameters	
Learning rate	1e-3
Weight decay	1e-4
Batch size	128
Early stopping steps	5
Gradient clip	1.0
Loss weights	
Continent loss weight	2.0
City loss weight	1.0
Coordinate loss weight	0.5

Table 10. Hyperparameter search space for GrowNet tuning

Hyperparameter	**Search Space**
Hidden size	128, 256, 512
Number of weak learners	10 to 30
Boost rate	0.1 to 0.8
Learning rate	1e-4 to 1e-2 (log uniform)
Batch size	64, 128, 256
Weight decay	1e-6 to 1e-3 (log uniform)
Epochs per stage	5 to 10
Gradient clip	0.5 to 2.0
Hierarchical loss weights	
Continent loss weight	1.0 to 2.0
City loss weight	0.5 to (continent_weight - 0.05)
Coordinate loss weight	0.05 to (city_weight - 0.05)

2.4 Ensemble Meta-Model Parameters

2.4.1 XGBoost Parameters

Table 11. Default parameters for XGBoost models

Parameter	**Classification**	**Regression**
Objective	multi:softprob	reg:squarederror
Eval metric	mlogloss	rmse
Learning rate	0.1	0.1
Max depth	6	6
Min child weight	1	1
Gamma	0	0
Subsample	0.8	0.8
Colsample bytree	0.8	0.8
Lambda	1.0	1.0
Alpha	0.0	0.0
n_estimators	300	300

Table 12. Hyperparameter search space for XGBoost tuning

Hyperparameter	Search Space
Learning rate	1×10^{-3} to 0.3 (log uniform)
Max depth	3 to 12
Min child weight	1 to 10
Gamma	0 to 5
Subsample	0.5 to 1.0
Colsample bytree	0.5 to 1.0
Lambda	1×10^{-3} to 10 (log uniform)
Alpha	1×10^{-3} to 10 (log uniform)
n_estimators	100 to 400

2.4.2 LightGBM Parameters

Table 13. Default parameters for LightGBM models

Parameter	Classification	Regression
Objective	multiclass	regression
Metric	multi_logloss	rmse
Learning rate	0.1	0.1
Max depth	6	6
Num leaves	31	–
Min child samples	20	20
Subsample	0.8	0.8
Colsample bytree	0.8	0.8
Reg alpha	0.1	0.0
Reg lambda	1.0	1.0
n_estimators	300	300

Table 14. Hyperparameter search space for LightGBM tuning

Hyperparameter	Search Space
Learning rate	1×10^{-3} to 0.3 (log uniform)
Max depth	3 to 12
Num leaves	15 to 256 (classification only)
Min child samples	5 to 100
Subsample	0.5 to 1.0
Colsample bytree	0.5 to 1.0
Reg lambda	1×10^{-3} to 10 (log uniform)
Reg alpha	1×10^{-3} to 10 (log uniform)
n_estimators	100 to 400

337 ### 2.4.3 CatBoost Parameters

Table 15. Default parameters for CatBoost models

Parameter	Classification	Regression
Loss function	MultiClass	RMSE
Eval metric	–	RMSE
Iterations	300	300
Learning rate	0.1	0.1
Depth	6	6
L2 leaf reg	3.0	3
Random strength	–	1
Bagging temperature	–	1
Border count	–	254
Random seed	42	42
Verbose	False	False

Table 16. Hyperparameter search space for CatBoost tuning

Hyperparameter	Search Space
Iterations	100 to 400 (classification), 100 to 500 (regression)
Learning rate	1×10^{-3} to 0.3 (log uniform)
Depth	3 to 10
L2 leaf reg	1 to 10
Random strength	1×10^{-9} to 10 (log uniform, regression only)
Bagging temperature	0 to 10 (regression only)
Border count	1 to 255 (regression only)

338 ### 2.4.4 GrowNet Parameters

Table 17. Default parameters for GrowNet models (ensemble context)

Parameter	**Classification**	**Regression**
Hidden size	256	256
Num weak learners	10	10
Boost rate	0.4	0.4
Learning rate	1e-3	1e-3
Weight decay	1e-5	1e-5
Batch size	128	128
Epochs per stage	30	30
Early stopping steps	7	7
Gradient clip	1.0	1.0
n_outputs	–	3

Table 18. Hyperparameter search space for GrowNet tuning (ensemble context)

Hyperparameter	**Search Space**
Hidden size	128, 256, 512
Num weak learners	10 to 30
Boost rate	0.1 to 0.8
Learning rate	1×10^{-4} to 1×10^{-2} (log uniform)
Batch size	64, 128, 256
Weight decay	1×10^{-6} to 1×10^{-3} (log uniform)
Epochs per stage	5 to 10
Gradient clip	0.5 to 2.0

339 ### 2.4.5 Neural Network (MLP) Parameters

Table 19. Default parameters for neural network (MLP) models (ensemble context)

Parameter	Classification	Regression
Input dimension	200	200
Hidden dimensions	[128, 64]	[128, 64]
Output dimension	7	3
Batch normalization	True	True
Initial dropout	0.3	0.2
Final dropout	0.8	0.5
Learning rate	1e-3	1e-3
Weight decay	1e-5	1e-5
Batch size	128	128
Epochs	400	400
Early stopping steps	20	50
Gradient clip	1.0	1.0

Table 20. Hyperparameter search space for neural network (MLP) tuning (ensemble context)

Hyperparameter	Search Space
Hidden dimensions	[64], [128], [128, 64], [256, 128, 64], [256, 128], [512, 256, 128, 64]
Initial dropout	0.1 to 0.3
Final dropout	0.5 to 0.8
Learning rate	1×10^{-4} to 1×10^{-2} (log uniform)
Batch size	64, 128, 256
Weight decay	1×10^{-6} to 1×10^{-3} (log uniform)
Gradient clip	0.5 to 2.0

340 ### 2.4.6 TabPFN Parameters

Table 21. TabPFN model configuration

Parameter	Value
Model	Pre-trained TabPFN
Hyperparameter tuning	Max time

2.5 Continent Classification: Separate Neural Network

Table 22. Continent Classification Report (Separate Neural Network)

Continent	Precision	Recall	F1-score	Support
east_asia	0.93	0.89	0.91	278
europe	0.86	0.82	0.84	283
middle_east	0.93	0.93	0.93	15
north_america	0.74	0.85	0.79	149
oceania	0.31	0.44	0.36	9
south_america	0.75	0.71	0.73	21
sub_saharan_africa	0.88	0.88	0.88	59
Accuracy	0.85 (814 samples)			
Macro avg	0.77	0.79	0.78	814
Weighted avg	0.86	0.85	0.85	814

2.6 Contienent Classification: Combined Neural Network

Table 23. Continent Classification Report (Combined Neural Network)

Continent	Precision	Recall	F1-score	Support
east_asia	0.90	0.90	0.90	278
europe	0.89	0.74	0.81	283
middle_east	0.70	0.93	0.80	15
north_america	0.72	0.85	0.78	149
oceania	0.33	0.44	0.38	9
south_america	0.65	0.81	0.72	21
sub_saharan_africa	0.80	0.90	0.85	59
Accuracy		0.83 (814 samples)		
Macro avg	0.71	0.80	0.75	814
Weighted avg	0.84	0.83	0.83	814

343 ## 2.7 Continent Classification: Hierarchical GrowNet

Table 24. Continent Classification Report (GrowNet)

Continent	Precision	Recall	F1-score	Support
east_asia	0.94	0.94	0.94	278
europe	0.87	0.81	0.84	283
middle_east	0.70	0.93	0.80	15
north_america	0.75	0.87	0.80	149
oceania	0.29	0.22	0.25	9
south_america	1.00	0.81	0.89	21
sub_saharan_africa	0.89	0.85	0.87	59
Accuracy	0.86 (814 samples)			
Macro avg	0.78	0.78	0.77	814
Weighted avg	0.87	0.86	0.86	814

344 ## 2.8 Continent Classification: Ensemble Learning

Table 25. Continent Classification Report (Ensemble Learning)

Continent	Precision	Recall	F1-score	Support
east_asia	0.95	0.97	0.96	278
europe	0.95	0.94	0.95	283
middle_east	0.93	0.93	0.93	15
north_america	0.93	0.97	0.95	149
oceania	0.67	0.44	0.53	9
south_america	1.00	0.86	0.92	21
sub_saharan_africa	0.98	0.95	0.97	59
Accuracy	0.95 (814 samples)			
Macro avg	0.92	0.87	0.89	814
Weighted avg	0.95	0.95	0.95	814

345 ## 2.9 City Classification: Separate Neural Network

Table 26. City-level classification report for Separate Neural Network on the test set.

City	Prec.	Rec.	F1	Sup.
auckland	0.00	0.00	0.00	1
baltimore	0.33	1.00	0.50	1
barcelona	0.96	1.00	0.98	23
berlin	0.50	0.93	0.65	15
bogota	0.67	0.50	0.57	4
brisbane	0.40	0.80	0.53	5
denver	0.54	0.87	0.67	15
doha	0.93	0.93	0.93	15
europe	0.59	0.83	0.69	12
fairbanks	0.50	0.24	0.32	21
hamilton	0.25	0.33	0.29	3
hanoi	0.75	0.60	0.67	5
hong_kong	0.98	0.86	0.92	148
ilorin	0.87	0.62	0.72	55
kuala_lumpur	0.69	0.90	0.78	10
kyiv	0.42	0.50	0.45	20
lisbon	0.38	0.25	0.30	12
london	0.91	0.64	0.75	125
marseille	0.80	0.80	0.80	5
minneapolis	1.00	0.33	0.50	3
naples	0.67	0.67	0.67	3
new_york_city	0.72	0.83	0.77	105
offa	0.10	0.50	0.17	4
oslo	0.52	0.94	0.67	17
paris	0.00	0.00	0.00	1
rio_de_janeiro	0.83	0.71	0.77	7
sacramento	0.50	1.00	0.67	2
san_francisco	0.25	0.50	0.33	2
santiago	0.83	1.00	0.91	5
sao_paulo	0.40	0.40	0.40	5
sendai	0.33	1.00	0.50	4
seoul	0.77	0.89	0.83	19
singapore	0.45	0.31	0.37	32
sofia	0.50	0.67	0.57	3
stockholm	0.64	0.29	0.40	24
taipei	0.76	1.00	0.86	19
tokyo	0.67	0.53	0.59	38
vienna	0.00	0.00	0.00	4
yamaguchi	0.00	0.00	0.00	3
zurich	0.46	0.58	0.51	19
accuracy			0.70	814
macro avg	0.55	0.62	0.55	814
weighted avg	0.75	0.70	0.71	814

2.10 City Classification: Combined Neural Network

Table 27. City-level classification report for Combined Neural Network on the test set.

City	Prec.	Rec.	F1	Sup.
auckland	0.00	0.00	0.00	1
baltimore	0.00	0.00	0.00	1
barcelona	0.96	1.00	0.98	23
berlin	1.00	0.13	0.24	15
bogota	0.00	0.00	0.00	4
brisbane	0.00	0.00	0.00	5
denver	0.76	0.87	0.81	15
doha	0.70	0.93	0.80	15
europe	0.39	1.00	0.56	12
fairbanks	0.75	0.43	0.55	21
hamilton	0.00	0.00	0.00	3
hanoi	0.00	0.00	0.00	5
hong_kong	0.94	0.99	0.96	148
ilorin	0.76	0.95	0.85	55
kuala_lumpur	0.78	0.70	0.74	10
kyiv	1.00	0.05	0.10	20
lisbon	0.33	0.17	0.22	12
london	0.94	0.74	0.83	125
marseille	0.00	0.00	0.00	5
minneapolis	0.00	0.00	0.00	3
naples	0.00	0.00	0.00	3
new_york_city	0.70	0.91	0.79	105
offa	0.00	0.00	0.00	4
oslo	0.58	0.82	0.68	17
paris	1.00	1.00	1.00	1
rio_de_janeiro	0.57	0.57	0.57	7
sacramento	0.50	0.50	0.50	2
san_francisco	0.50	1.00	0.67	2
santiago	0.83	1.00	0.91	5
sao_paulo	0.43	0.60	0.50	5
sendai	1.00	0.25	0.40	4
seoul	0.85	0.89	0.87	19
singapore	0.43	0.75	0.55	32
sofia	0.00	0.00	0.00	3
stockholm	0.87	0.54	0.67	24
taipei	0.90	1.00	0.95	19
tokyo	0.63	0.71	0.67	38
vienna	0.00	0.00	0.00	4
yamaguchi	0.00	0.00	0.00	3
zurich	0.53	0.53	0.53	19
accuracy			0.75	814
macro avg	0.49	0.48	0.45	814
weighted avg	0.75	0.75	0.72	814

2.11 City Classification: Hierarchical GrowNet

Table 28. City-level classification report for Hierarchical GrowNet on the test set.

City	Prec.	Rec.	F1	Sup.
auckland	0.00	0.00	0.00	3
baltimore	0.00	0.00	0.00	0
barcelona	1.00	0.95	0.97	19
berlin	0.64	0.93	0.76	15
bogota	1.00	0.50	0.67	2
brisbane	0.33	0.50	0.40	4
denver	0.62	0.62	0.62	13
doha	0.74	0.93	0.82	15
europe	0.76	0.72	0.74	18
fairbanks	0.32	0.39	0.35	18
hamilton	0.00	0.00	0.00	2
hanoi	0.38	1.00	0.55	3
hong_kong	0.97	0.86	0.91	179
ilorin	0.91	0.74	0.81	53
kuala_lumpur	0.85	0.92	0.88	12
kyiv	0.19	0.46	0.27	13
lisbon	0.26	0.31	0.29	16
london	0.88	0.76	0.82	123
marseille	0.71	1.00	0.83	5
minneapolis	0.25	1.00	0.40	1
naples	1.00	0.20	0.33	5
new_york_city	0.75	0.74	0.75	105
offa	0.00	0.00	0.00	6
oslo	0.77	0.85	0.81	20
paris	0.33	0.50	0.40	2
rio_de_janeiro	1.00	0.67	0.80	6
sacramento	1.00	0.67	0.80	6
san_francisco	0.56	0.83	0.67	6
santiago	0.80	0.80	0.80	5
sao_paulo	1.00	0.75	0.86	8
sendai	0.67	1.00	0.80	6
seoul	0.71	1.00	0.83	15
singapore	0.59	0.42	0.49	24
sofia	0.33	0.50	0.40	2
stockholm	0.88	0.85	0.87	27
taipei	0.87	1.00	0.93	13
tokyo	0.73	0.70	0.71	23
vienna	0.50	1.00	0.67	1
yamaguchi	0.33	0.33	0.33	3
zurich	0.71	0.59	0.65	17
accuracy			0.75	814
macro avg	0.61	0.65	0.60	814
weighted avg	0.79	0.75	0.76	814

2.12 City Classification: Ensemble Learning

Table 29. City-level classification report for Ensemble Learning on the test set.

City	Prec.	Rec.	F1	Sup.
auckland	0.33	1.00	0.50	1
baltimore	0.00	0.00	0.00	1
barcelona	1.00	1.00	1.00	23
berlin	0.94	1.00	0.97	15
bogota	1.00	0.75	0.86	4
brisbane	1.00	0.60	0.75	5
denver	0.94	1.00	0.97	15
doha	1.00	0.93	0.97	15
fairbanks	0.83	0.95	0.89	21
hamilton	1.00	0.67	0.80	3
hanoi	1.00	0.80	0.89	5
hong_kong	0.99	0.99	0.99	148
ilorin	0.98	0.93	0.95	55
kuala_lumpur	0.91	1.00	0.95	10
kyiv	0.58	0.70	0.64	20
lisbon	0.92	0.92	0.92	12
london	1.00	0.97	0.98	125
marseille	0.75	0.60	0.67	5
minneapolis	0.60	1.00	0.75	3
naples	0.67	0.67	0.67	3
new_york_city	0.95	0.97	0.96	105
offa	0.67	1.00	0.80	4
oslo	1.00	0.94	0.97	17
paris	0.00	0.00	0.00	1
porto	0.92	1.00	0.96	12
rio_de_janeiro	1.00	0.86	0.92	7
sacramento	1.00	1.00	1.00	2
san_francisco	0.67	1.00	0.80	2
santiago	1.00	1.00	1.00	5
sao_paulo	1.00	0.60	0.75	5
sendai	1.00	1.00	1.00	4
seoul	0.86	0.95	0.90	19
singapore	0.73	0.84	0.78	32
sofia	1.00	0.67	0.80	3
stockholm	0.96	1.00	0.98	24
taipei	0.90	1.00	0.95	19
tokyo	0.85	0.87	0.86	38
vienna	0.60	0.75	0.67	4
yamaguchi	0.00	0.00	0.00	3
zurich	0.91	0.53	0.67	19
accuracy			0.93	814
macro avg	0.81	0.81	0.80	814
weighted avg	0.93	0.93	0.92	814

2.13 Coordinate Regression: Separate Neural Network

Table 30. Error Group Analysis (Separate Neural Network)

Group	Count	Mean Error (km)	Median Error (km)	Proportion	Weighted Error
C_correct Z_correct	565	3994	3255	0.694	2772
C_correct Z_wrong	126	5333	3703	0.155	826
C_wrong Z_correct	6	7668	8555	0.007	57
C_wrong Z_wrong	117	9098	7532	0.144	1308

Notes: C = Continent, Z = City. Groups indicate correctness of continent and city predictions.

2.14 Coordinate Regression: Combined Neural Network

Table 31. Error Group Analysis (Combined Neural Network)

Group	Count	Mean Error (km)	Median Error (km)	Proportion	Weighted Error
C_correct Z_correct	581	502	274	0.714	358
C_correct Z_wrong	92	2101	1523	0.113	237
C_wrong Z_correct	29	3434	2252	0.036	122
C_wrong Z_wrong	112	6637	5377	0.138	913

Notes: C = Continent, Z = City. Groups indicate correctness of continent and city predictions.

2.15 Coordinate Regression Metrics: Hierarchical GrowNet

Table 32. Error Group Analysis (GrowNet)

Group	Count	Mean Error (km)	Median Error (km)	Proportion	Weighted Error
C_correct Z_correct	604	904	599	0.742	671
C_correct Z_wrong	99	2215	1710	0.122	269
C_wrong Z_correct	7	4501	4324	0.009	39
C_wrong Z_wrong	104	7090	5896	0.128	906

Notes: C = Continent, Z = City. Groups indicate correctness of continent and city predictions.

2.16 In-Radius Accuracy Metrics

Table 33. In-Radius Accuracy Metrics for Separate Neural Network, Combined Neural Network, and Hierarchical GrowNet on the test set.

Radius	Separate NN (%)	Combined NN (%)	GrowNet (%)
<1 km	0.00	0.00	0.00
<5 km	0.00	0.00	0.00
<50 km	0.00	0.37	0.98
<100 km	0.00	9.46	2.70
<250 km	0.00	30.34	12.78
<500 km	0.86	49.75	30.96
<1000 km	1.84	66.34	57.37
<5000 km	55.65	89.31	89.07