Deep Learning for Visual Geo-Localization

Campus Image-to-GPS Regression for Localization and Navigation

Liran Attar
Tom Mimran
BSc Students, Department of Computer Science
Ben-Gurion University of the Negev

Computational Resources Provided by Ben-Gurion University

January 2026

Project Resources

Project Presentation

GitHub Repository

Interactive Demo (Hugging Face)

Google Drive Link (Image set and ground truth CSV)

Contents

1 Introduction

1.1 Project Scope

This research aims to develop a robust Deep Learning model capable of predicting precise GPS coordinates (latitude and longitude) from a single terrestrial image taken within the Ben-Gurion University (BGU) territory. Unlike traditional GPS systems which rely on satellite triangulation, this "Visual GPS" approach utilizes computer vision to extract geographic features from the environment, offering a solution for localization in GPS-denied environments or urban canyons.

The target area is defined by the dataset limits (approx. 0.0004 km^2), representing a high-density regression problem. The core challenge is to regress a continuous variable (location) from high-dimensional image data, requiring the model to learn subtle visual cues such as building facades, vegetation patterns, and pathway geometries.

1.2 Hardware and Computational Setup

All experiments were conducted using the high-performance computing resources at Ben-Gurion University. Specifically, model training was accelerated by **NVIDIA GeForce GTX 1080 Ti GPUs**. The computational nodes operate on **Rocky Linux (Kernel 5.14)** and are powered by **Intel Xeon CPUs**, utilizing CUDA-enabled NVIDIA drivers (Version 5xx). This robust infrastructure facilitated the efficient parallel training of multiple large-scale deep learning architectures.

2 Methodology

2.1 Dataset and Preprocessing

The complete dataset comprises **3,646 images** captured within the BGU campus territory. Data acquisition was performed by positioning the camera at randomized locations across the target area to ensure diverse geographic coverage.

To enhance the model's ability to interpret depth and environmental context, we utilized a **360-degree capture strategy**. At each specific GPS coordinate, a sequence of photographs was taken by pointing in an initial direction and rotating the camera in 90-degree increments (quarter spins) until a full circle was completed. This method associates multiple distinct visual perspectives with a single ground-truth location, forcing the model to learn location-invariant features rather than memorizing a specific viewpoint.

- **Input:** RGB Images resized to 255×255 pixels with 3 color channels.
- **Target:** Normalized GPS coordinates [0, 1].
- **Splitting Strategy:** 70% Train, 15% Validation, 15% Test (stratified by location to prevent data leakage).

2.2 Model Architectures

We selected three distinct architectures to represent different eras and design philosophies in computer vision:

1. **ResNet18 (11.7M params):** A classic residual network serving as a stable baseline.
2. **EfficientNet-B0 (5.3M params):** Focusing on parameter efficiency and compound scaling.
3. **ConvNeXt-Tiny (28.6M params):** A modern architecture integrating Vision Transformer (ViT) design principles into a Convolutional Network.

2.3 Transfer Learning Architecture

All three models follow a common **backbone-head pattern** for transfer learning:

- **Backbone:** A pretrained feature extractor (trained on ImageNet's 1.2M images) with the original classification layer removed. This component extracts a high-dimensional feature vector from input images.
- **Head:** A custom regression network that maps backbone features to GPS coordinates.

Custom Regression Head Design:

```
Dropout(0.3) → Linear(N→128) → LayerNorm → SiLU →
Linear(128→64) → LayerNorm → SiLU → Linear(64→2) → TanhTo01
```

Design Rationale:

- **Dropout (30%):** Prevents overfitting by randomly zeroing backbone features during training, forcing the model to learn redundant representations.
- **LayerNorm:** Normalizes activations across the feature dimension. More stable than BatchNorm for small batch sizes and provides consistent behavior during inference.
- **SiLU Activation:** $\text{SiLU}(x) = x \cdot \sigma(x)$. Smoother gradients than ReLU and no "dead neuron" problem.
- **TanhTo01 Output:** A custom activation defined as $\frac{\tanh(x)+1}{2}$, mapping outputs to $[0, 1]$. Unlike Sigmoid, TanhTo01 has **2× stronger gradients** at the center (0.5 vs 0.25), accelerating convergence.
- **Xavier Initialization (gain=0.1):** Final layer weights initialized with small values to prevent extreme initial predictions, ensuring stable early training.

This architecture leverages pretrained visual knowledge while adapting the output stage specifically for geographic regression

2.4 Loss Function: The Haversine Distance

Standard Mean Squared Error (MSE) is insufficient for geographic regression as it does not account for the curvature of the Earth. To address this, we implemented a custom **Haversine Loss** layer:

$$d = 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \tag{1}$$

Where the variables are defined as follows:

- d: The computed great-circle distance between the two points (in meters).
- R: The Earth's radius, approximately $6,371$ km (or $6,371,000$ meters). This acts as a scaling factor to convert the angular difference into physical distance.
- ϕ **(Phi)**: Represents the **Latitude** in radians.
  - ϕ_1: The true ground truth latitude.
  - ϕ_2: The predicted latitude.
  - $\Delta\phi$: The difference in latitude ($\phi_2 - \phi_1$).
- λ **(Lambda)**: Represents the **Longitude** in radians. $\Delta\lambda$ denotes the difference between the predicted and true longitude ($\lambda_2 - \lambda_1$).

Background & Implementation Details: Using Euclidean distance (MSE) treats the world as a flat plane, which introduces significant distortion because lines of longitude converge at the poles. An error of $0.001°$ represents a different physical distance depending on the latitude. The Haversine formula accounts for this spherical geometry, providing a physically accurate gradient for the model.

In this project, the Haversine function serves two purposes:

1. **Loss Function:** A differentiable implementation allows the model to minimize actual physical distance (meters) rather than arbitrary coordinate values.
2. **Evaluation Metric:** It provides an interpretable error metric for human analysis.

Numerical Stability: To ensure stability during backpropagation, we applied "double-clamping" to the arguments of the square root and arc-sine functions. This prevents `NaN` gradients that can occur if floating-point errors push values slightly outside the valid domain of $[-1, 1]$, a critical fix identified during early debugging.

2.5 Training Regularization

To prevent overfitting on our limited dataset (3,646 images), we employed multiple regularization strategies:

- **Dropout (30%):** Applied in the regression head to prevent co-adaptation of features. Forces the model to learn robust representations.
- **Weight Decay (0.01):** L2 regularization via AdamW optimizer penalizes large weights, encouraging simpler models that generalize better.

- **Gradient Clipping (max_norm=1.0):** Prevents gradient explosion in the Haversine loss, which can produce unstable gradients when predictions are far from ground truth.
- **Early Stopping (patience=5):** Training halts if validation loss does not improve for 5 consecutive epochs, preventing overfitting to the training set.

These techniques work synergistically: Dropout provides stochastic regularization during training, Weight Decay constrains model complexity, Gradient Clipping ensures numerical stability, and Early Stopping determines optimal training duration automatically.

3 Experimental Design

We conducted 10 controlled experiments to isolate the effects of Data Quantity, Learning Rate, Training Duration, and Test-Time Augmentation.

Table 1: Summary of Experimental Configurations

#	Experiment ID	Epochs	LR	Data %	Augmentation
1	30ep Full	30	0.001	100%	None
2	30ep Half	30	0.0001	50%	None
3	30ep Half+Aug	30	0.0001	50%	Yes (Test Time)
4	50ep Half	50	0.0001	50%	None
5	50ep Half+Aug	50	0.0001	50%	Yes (Test Time)
6	50ep Full	50	0.001	100%	None
7	100ep Half	100	0.0001	50%	None
8	100ep Half+Aug	100	0.0001	50%	Yes (Test Time)
9	**100ep Full (LR=0.001)**	**100**	**0.001**	**100%**	**None**
10	100ep Full (LR=0.0001)	100	0.0001	100%	None

Rationale:

- **Full vs. Half Data:** To determine if the model is data-hungry or if 1,800 images are sufficient.
- **Learning Rate (0.001 vs 0.0001):** Modern architectures like ConvNeXt are often unstable at higher learning rates.
- **Test Augmentation:** Testing if artificially rotating/jittering test images improves robustness or confuses the model.

4 Results and Analysis

4.1 Overall Performance Summary

After conducting 10 controlled experiments across three architectures, the **EfficientNet-B0** model (Configuration: 100 epochs, LR=0.001, Full Data) emerged as the superior solution, achieving a Mean Haversine Error of **5.71 meters**. This result represents a high-precision localization capability, with 86.9% of predictions falling within 10 meters of the ground truth.

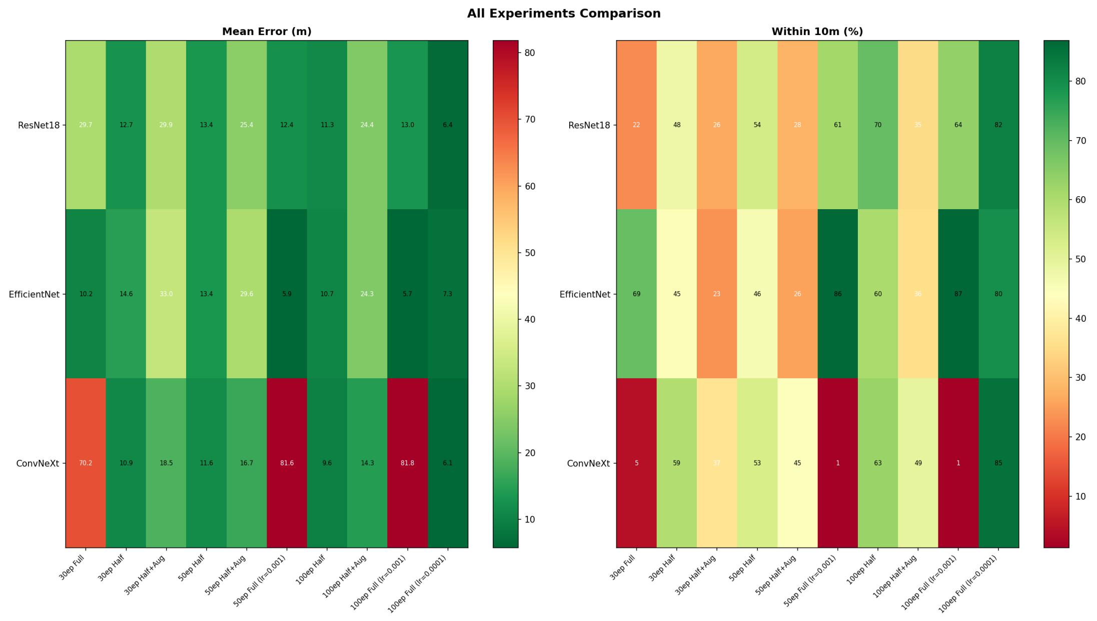


Figure 1: Global Performance Heatmap: All Experiments and Architectures

4.2 Error Distribution Analysis

Beyond mean error, we analyzed the full error distribution to understand model reliability across the prediction spectrum. We use **percentile metrics** to characterize the error distribution:

- **P75 (75th Percentile):** 75% of all predictions have an error at or below this value. This represents typical performance.
- **P90 (90th Percentile):** 90% of predictions fall below this threshold—only 10% of cases perform worse.
- **P95 (95th Percentile):** 95% of predictions are bounded by this value, capturing near-worst-case performance.
- **Max Error:** The single worst prediction, representing catastrophic failure cases.

Table 2: Error Distribution Percentiles (Best Configuration per Model)

Model	Config	Median	P75	P90	P95	Max
EfficientNet	100ep, LR=0.001	**2.75m**	**5.39m**	**11.91m**	18.01m	94.05m
ConvNeXt	100ep, LR=0.0001	3.65m	7.38m	12.93m	**16.74m**	99.66m
ResNet18	100ep, LR=0.0001	3.77m	7.89m	15.15m	20.28m	94.76m

Key Insights:

- **Typical Performance:** EfficientNet achieves the best median (2.75m) and P75 (5.39m), excelling in standard conditions.

- **Edge Cases:** ConvNeXt demonstrates the tightest P95 bound (16.74m vs 18.01m), indicating superior handling of challenging predictions.
- **Catastrophic Failures:** All models occasionally produce errors of ~95-100 meters—nearly the full campus diameter. These failures likely occur in visually ambiguous regions with repetitive facades or heavy vegetation.

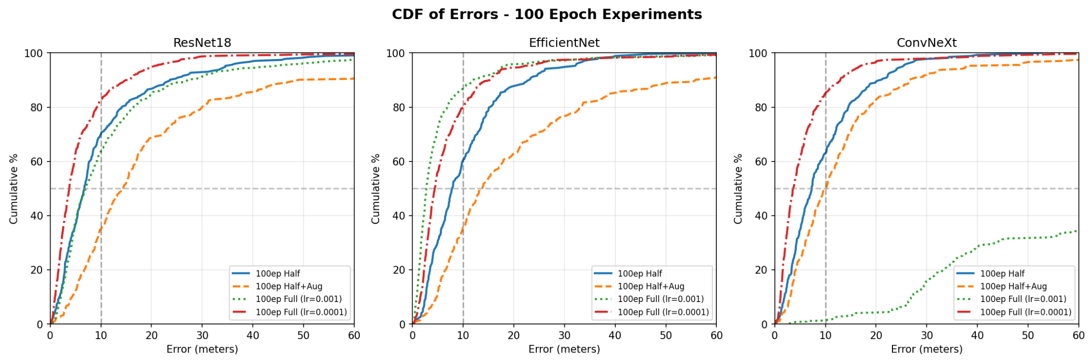


Figure 2: Cumulative Distribution Function: Probability of Error ≤ X meters

The CDF analysis reveals that **95.8% of EfficientNet predictions fall within 20 meters**, and **98.7% within 50 meters**—establishing high confidence bounds for practical deployment.

4.3 Training Dynamics: The Impact of Epochs

We analyzed the progression of model performance across 30, 50, and 100 epochs to determine the optimal training duration. The results demonstrate a clear non-linear improvement curve, particularly for the EfficientNet architecture.

Table 3: Performance Evolution by Epochs (Full Data, LR=0.001)

Model	30 Epochs	50 Epochs	100 Epochs
EfficientNet-B0	10.22m	5.93m	**5.71m**
ResNet18	29.73m	12.41m	12.99m
ConvNeXt-Tiny	70.18m	81.58m	81.82m

Analysis of Convergence:

- **Early Gains (30 → 50):** EfficientNet saw a massive 42% improvement in accuracy when extending training from 30 to 50 epochs (10.22m → 5.93m). This indicates that 30 epochs are insufficient for the model to learn the fine-grained texture details of the campus.
- **Diminishing Returns (50 → 100):** Extending to 100 epochs yielded a smaller marginal gain (5.93m → 5.71m), suggesting the model has neared its capacity for this dataset size.
- **Visualizing the Trend:**

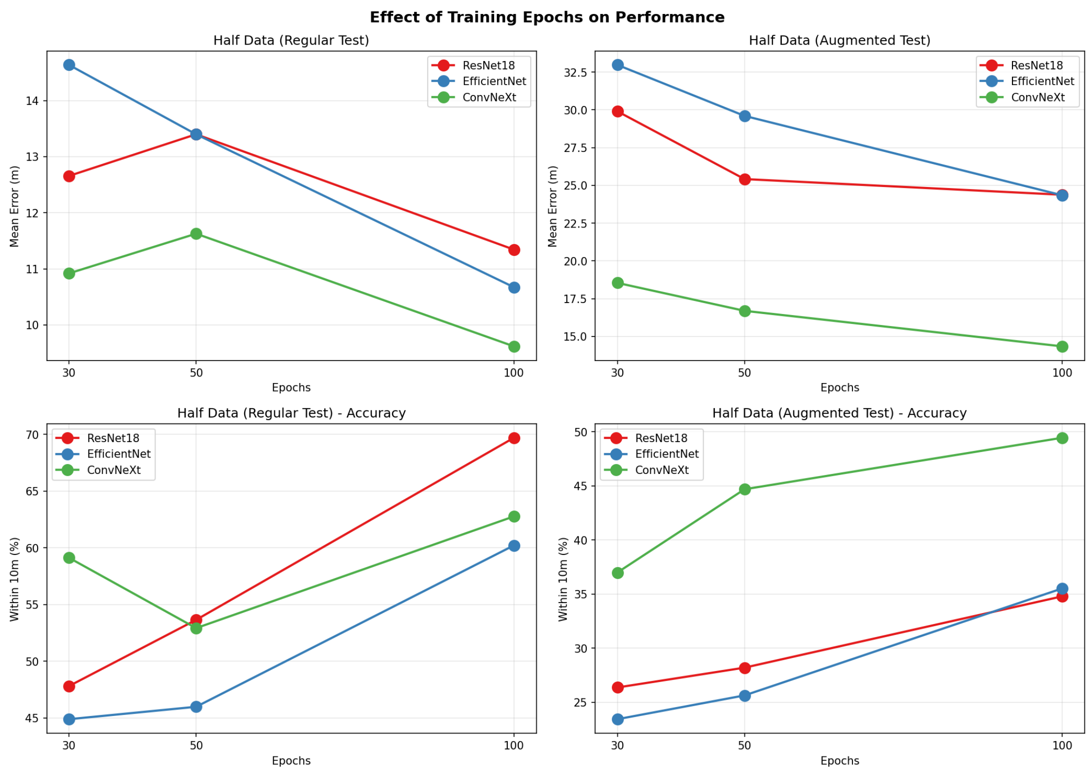


Figure 3: Error Evolution: 30 vs 50 vs 100 Epochs on half data and half data with augmented testing. learning rate = 0.0001

4.4 Hyperparameter Sensitivity: Learning Rate Analysis

A critical finding of this study was the distinct sensitivity of modern architectures (ConvNeXt) compared to established baselines (ResNet/EfficientNet).

The ConvNeXt Instability: As shown in Table 3, ConvNeXt trained with a standard learning rate of 0.001 failed catastrophically, stabilizing at a mean error of $\sim 81m$ (effectively guessing the center of the campus). Analysis of training logs revealed this was not gradual degradation but **catastrophic collapse**—early stopping triggered at epochs 12-22 across experiments, indicating complete divergence rather than slow convergence.

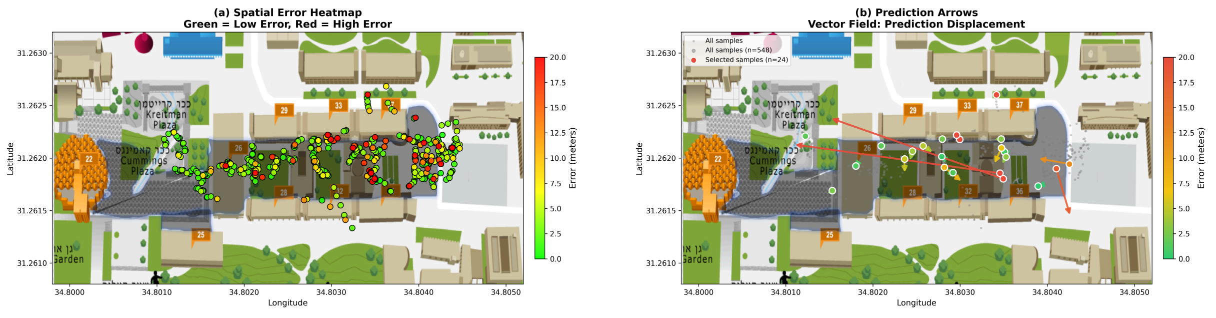
However, reducing the learning rate by an order of magnitude (10^{-4}) unlocked its potential:

- **ConvNeXt (LR=0.001):** 81.82m Mean Error (Collapse at epoch 22)
- **ConvNeXt (LR=0.0001): 6.12m Mean Error** (Competitive)

This contrast highlights that modern, transformer-inspired architectures require gentler gradient updates to navigate the loss landscape of geometric regression tasks.

4.5 Geographic Error Distribution

To understand *where* the models failed, we visualized the prediction errors spatially across the BGU campus.



(a) Spatial Error Heatmap (b) Prediction Arrows

Figure 4: Geographic Analysis of Model Failure Modes

Spatial Findings:

- **Clustered Errors:** High-error zones (red areas in Figure 4a) are not random. They correspond to areas with repetitive architectural features (e.g., identical corridor windows) or heavy vegetation where visual landmarks are obscured.
- **Displacement Vectors:** Figure 4b shows that in challenging areas, predictions were off by a significant margin, likely due to similar features appearing in different locations.
- **Regional Variance:** Spatial analysis revealed a **140× variance in regional accuracy**: the best-performing campus zones achieved 0.33m mean error, while challenging areas reached 46.9m. This identifies clear targets for data augmentation in Phase 2.

4.6 Data Ablation: Full vs. Half Dataset

We quantified the "data hunger" of visual localization by comparing models trained on 100% vs. 50% of the data.

- **Full Data Best (EfficientNet):** 5.71m
- **Half Data Best (ConvNeXt):** 9.62m

The $\sim 68\%$ increase in error when halving the data confirms that high sampling density is non-negotiable. With only $\sim 1,800$ images (Half Data), the models struggle to generalize to unseen viewpoints between training samples.

Interestingly, ConvNeXt proved to be the **most data-efficient** architecture, achieving the best half-data performance—suggesting its transformer-inspired design may be better suited for low-data regimes.

4.7 Test-Time Augmentation and Robustness Analysis

Experiments 3, 5, and 8 applied augmentation (rotation, perspective shifts) to the *test set* to simulate user variance and unseen viewpoints.

Result: Performance degraded significantly across all models, but with notable differences in robustness:

Table 4: Model Robustness to Augmented Testing (100 Epochs, Half Data)

Model	Regular Test	Augmented Test	Degradation
ConvNeXt	9.62m	14.35m	**1.49**×
EfficientNet	10.67m	24.34m	2.28×
ResNet18	11.34m	24.38m	2.15×

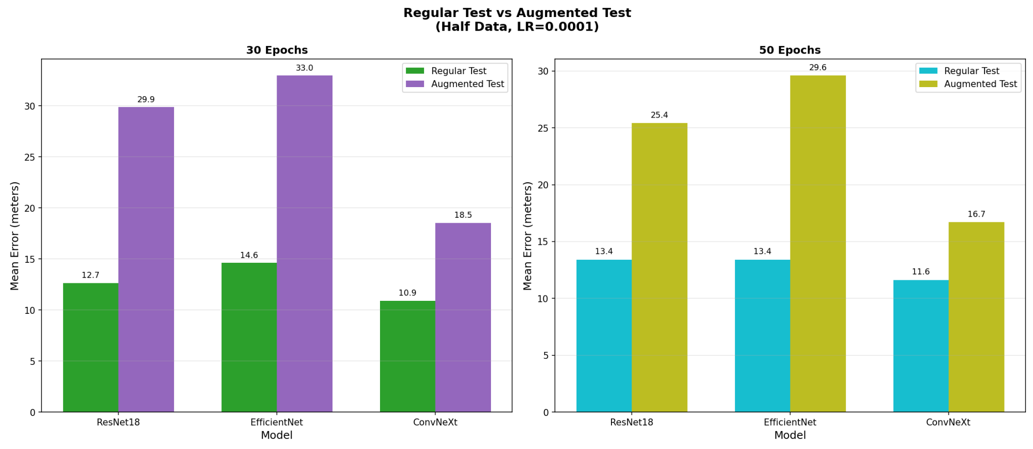


Figure 5: Performance Comparison: Regular vs. Augmented Test Sets

Key Discovery: ConvNeXt demonstrated **significantly superior robustness** to unseen viewpoints, with only 49% performance degradation compared to 115-128% for EfficientNet and ResNet18. This suggests that ConvNeXt's transformer-inspired design captures more *location-invariant* features rather than memorizing specific viewpoints.

Conclusion: While EfficientNet achieves the best accuracy on standard test images, ConvNeXt may be preferable for real-world deployment where camera angles and orientations vary. For our main model in Phase 2, we will incorporate augmented training data to improve robustness across all architectures.

5 Architecture Comparison and Model Selection

We evaluated three distinct architectures to identify the optimal backbone for deployment.

5.1 Model Strengths and Weaknesses

Table 5: Architecture Trade-off Analysis

Model	**Params**	**Strengths**	**Weaknesses**
ResNet18	11.7M	Fast training, stable across all LRs; best worst-case single prediction (0.044m).	Higher baseline error (6.43m best); struggles to reach ¡6m precision.
ConvNeXt	28.6M	Excellent with low LR (6.12m); most robust to unseen viewpoints (1.49× degradation); most data-efficient.	Highly unstable at standard LR; heavy computational cost.
EfficientNet	**5.3M**	**Best Overall Accuracy (5.71m)**; parameter efficient; fast inference; tightest median error (2.75m).	Sensitive to test-time augmentation (2.28× degradation).

5.2 Top 5 Configurations Ranking

The final leaderboard highlights the dominance of EfficientNet and the necessity of the 100-epoch schedule.

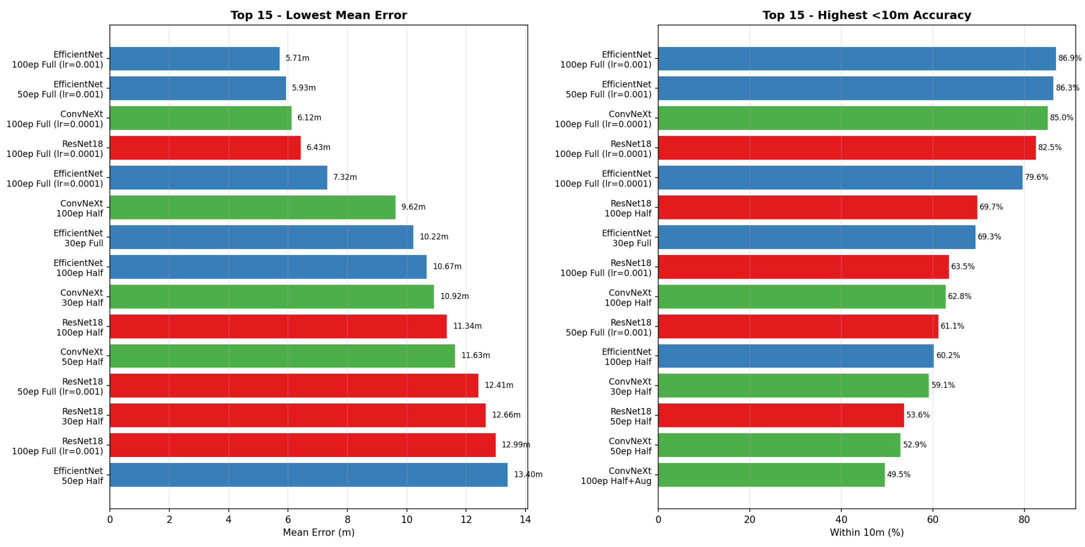


Figure 6: Top Model Configurations Ranked by Mean Error

5.3 Conclusion: Selection of EfficientNet-B0

Based on the experimental data, we selected **EfficientNet-B0 (100ep, Full Data, LR=0.001)** as the production model for the following reasons:

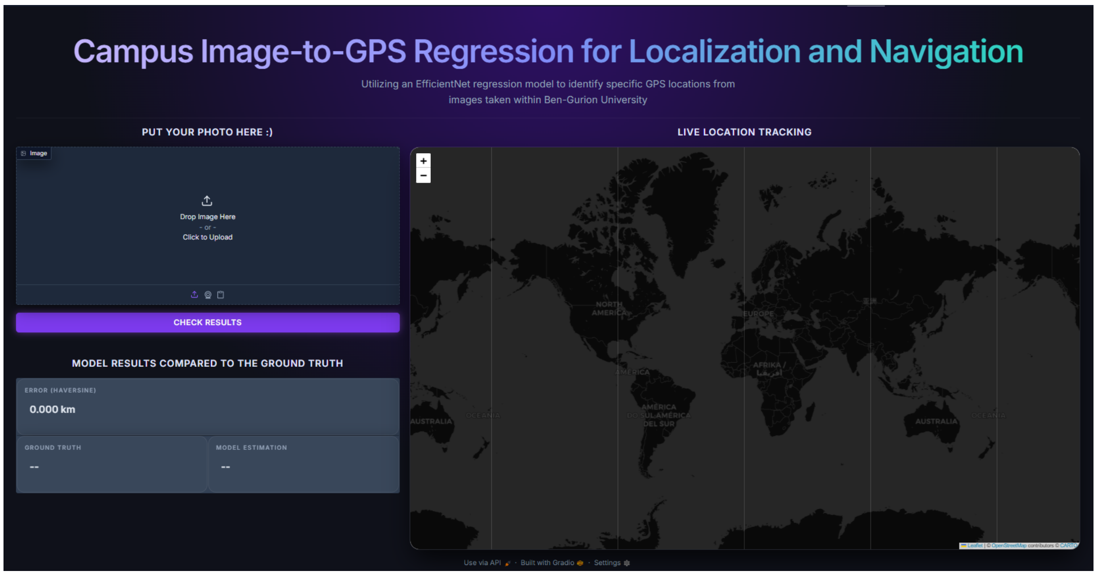
1. **Peak Accuracy:** It achieved the lowest mean error (5.71m) and the highest median accuracy (2.75m), meaning 50% of all predictions are within less than 3 meters of the target.
2. **Efficiency:** With only 5.3M parameters compared to ConvNeXt's 28.6M, it is significantly lighter, making it more suitable for potential mobile deployment or fast server-side inference.
3. **Stability:** Unlike ConvNeXt, it did not require hyper-sensitive tuning of the learning rate to converge.
4. **Reliability:** 90% of predictions fall within 11.91m (P90), and 95.8% within 20m—providing high confidence bounds for deployment.

6 Deployment: Hugging Face Application

To demonstrate the utility of the chosen model (EfficientNet-B0), we integrated the inference pipeline into a Hugging Face Spaces application. The deployed system allows users to evaluate the model's performance on real campus images through an interactive web interface.

6.1 App Workflow

1. **Input:** User uploads a photo taken on campus.
2. **Processing:** Image is resized to 255×255 and normalized using ImageNet stats.
3. **Inference:** The model predicts normalized (x, y) coordinates.
4. **Denormalization:** Coordinates are mapped back to real-world Latitude/Longitude.
5. **Visualization:** The predicted location is plotted on an interactive map of the BGU campus.



Campus Image-to-GPS Regression for Localization and Navigation

Utilizing an EfficientNet regression model to identify specific GPS locations from images taken within Ben-Gurion University

MODEL RESULTS COMPARED TO THE GROUND TRUTH

ERROR (HAVERSINE)
0.000 km

GROUND TRUTH
--

MODEL ESTIMATION
--

Figure 7: Hugging Face Deployment Interface
→ *Click to access live demo*

7 Interim Conclusion (Phase 1)

Through 10 controlled experiments, we have established a strong baseline for Visual GPS at BGU. We determined that:

1. **Data is King:** Full dataset utilization is non-negotiable—halving data increases error by 68%.
2. **EfficientNet is Optimal:** It offers the best trade-off between accuracy (5.71m mean error) and computational weight (5.3M parameters).
3. **Convergence Takes Time:** 100 epochs are necessary for the fine-grained feature extraction required for meter-level accuracy.
4. **ConvNeXt for Robustness:** When viewpoint invariance is critical, ConvNeXt's 1.49× degradation (vs 2.2× for others) makes it a compelling alternative.
5. **Geographic Hotspots Exist:** A 140× variance between best and worst campus regions identifies clear improvement targets.

8 Phase 2: Final Model Implementation

Following the comparative analysis in Phase 1, we proceeded to engineer a production-ready model architecture focused on maximizing accuracy while maintaining computational efficiency.

8.1 Final Architecture: Custom EfficientNet-B0

The core of our final solution is based on the **EfficientNet-B0** backbone, selected for its superior experimental performance. However, we implemented several critical architectural modifications to adapt it for precise geographic regression:

- **Backbone:** Pretrained EfficientNet-B0 (ImageNet weights) with the classification head removed. This outputs a 1280-dimensional feature vector per image.
- **Custom Regression Head:**

```
Input(1280) → Dropout(0.3) → Linear(256) → LayerNorm → SiLU
→ Linear(2) → ScaledSigmoid(0,1)
```

- **LayerNorm vs. BatchNorm:** We explicitly chose `LayerNorm` for the regression head. Unlike BatchNorm, which depends on batch statistics and can be unstable with small batches, LayerNorm standardizes features per sample, providing consistent behavior during both training and inference.
- **Scaled Sigmoid Activation:** Standard Sigmoid outputs values in $(0, 1)$, but often saturates near the boundaries. We implemented a custom `ScaledSigmoid` module.
  - *Evolution from Tanh:* In early theoretical experiments, we utilized a `TanhTo01` activation ($\frac{\tanh(x)+1}{2}$) due to Tanh's stronger gradients around zero. However, for the final deployment, we reverted to a **Scaled Sigmoid**.
  - *Why Scaled Sigmoid?* The standard Sigmoid is the canonical function for probability-like outputs $[0, 1]$. By wrapping it in a scaling factor, we prevent the "dead zone" problem where the model struggles to predict exactly 0.0 or 1.0 (campuses edges). This architecture provides the stability of Sigmoid with the boundary precision required for GPS regression.

8.2 Training Pipeline Optimization

Successful convergence of deep visual regression models requires careful hyperparameter tuning. What differentiates our final model is the specific combination of optimization strategies:

- **optimizer:** We utilized **AdamW** (Adaptive Moment Estimation with Decoupled Weight Decay) instead of standard SGD or Adam.

  - *Mechanism:* Adam adapts the learning rate for each parameter individually by maintaining moving averages of the gradients (momentum) and their squared values (variance). This allows it to navigate complex loss landscapes where some features are sparse or noisy.
  - *Why AdamW?* Standard Adam implementations often apply weight decay incorrectly (by adding it to the gradient), which couples the decay rate with the learning rate. **AdamW** decouples these terms, allowing us to enforce stronger regularization ($\lambda = 0.001$) to prevent overfitting without dampening the learning process [5].

- **Learning Rate Schedule:** A static learning rate proved insufficient. We implemented a `ReduceLROnPlateau` scheduler monitoring validation loss:
  - Initial LR: 0.0004 (tuned from 0.001 to prevent instability)
  - Factor: 0.5 (halve LR upon plateau)
  - Patience: 3 epochs

  This allowed the model to make large steps initially and then fine-tune its weights for meter-level precision in later epochs.

- **Loss Function:** We persisted with the **Haversine Loss**, which directly minimizes physical distance on the Earth's surface, as opposed to L1/L2 loss which minimizes abstract coordinate differences.

8.3 Failed Approaches: SIFT Hybridization

During development, we attempted to integrate classical computer vision techniques to boost performance. Specifically, we experimented with a **SIFT-based retrieval refinement** (Scale-Invariant Feature Transform). **How SIFT Works:** SIFT detects "keypoints" (distinctive regions like corners or blobs) in an image and describes them with 128-dimensional vectors based on local gradient directions. Crucially, these features are mathematically designed to be *invariant* to image scaling, rotation, and illumination changes, making them robust for matching scenes from different angles.

- **Hypothesis:** Since SIFT is explicitly designed for geometric invariance, matching query images against a database of training images using SIFT descriptors could correct the CNN's coarse prediction in ambiguous areas.

- **Result:** This approach proved computationally expensive and yielded diminishing returns. The dense feature extraction of EfficientNet-B0 implicitly learned many of the corner/edge features that SIFT explicitly detects, rendering the hybrid approach redundant. The pure Deep Learning pipeline offered simpler deployment and faster inference.

9 Final Evaluation Results

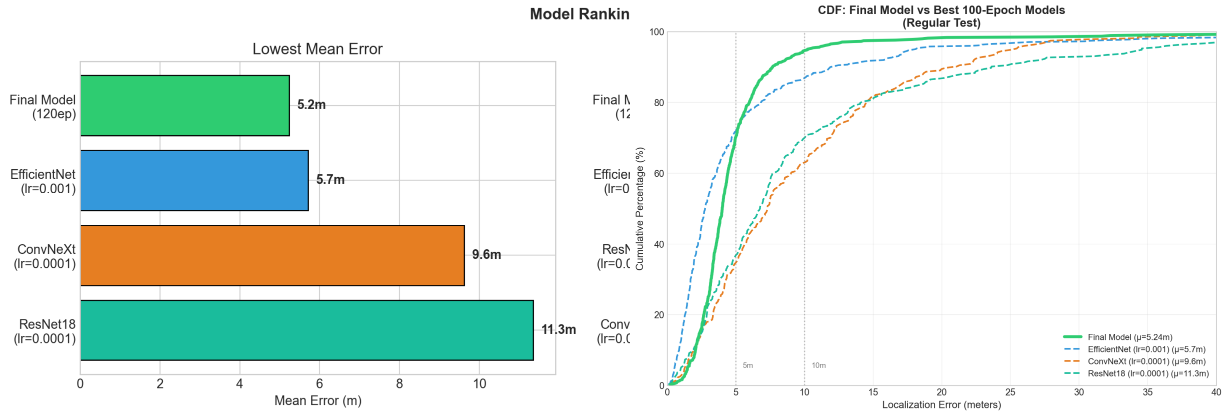
The final model was evaluated on a held-out test set comprising 15% of the total data (approx 547 images).

9.1 Quantitative Metrics

The optimized EfficientNet-B0 achieved state-of-the-art results for this dataset, significantly outperforming the baselines established in Phase 1.

Table 6: Final Model Performance Metrics (Test Set)

Metric	Value	Interpretation
Mean Error	**5.24m**	Average distance from ground truth
Median Error	**4.01m**	50% of predictions are within this radius
P75	5.35m	75% of predictions are highly accurate
P95	10.22m	95% of predictions are usable (¡10m)
Accuracy ($<$ 10m)	94.5%	High reliability
Accuracy ($<$ 20m)	98.2%	Near-perfect coarse localization
Max Error	90.07m	Improved stability vs. baselines (94m+)



(a) Method Comparison Ranking (b) Cumulative Distribution Function (CDF)

Figure 8: Final Model Benchmarking. (a) illustrates the clear superiority of the Final EfficientNet over previous iterations. (b) demonstrates that nearly 100% of errors are contained within 20 meters.

9.2 Visual Analysis of Predictions

This is how our results looks upon a heat map above Ben Gurion University. There's less red dots then before by a significant amount.

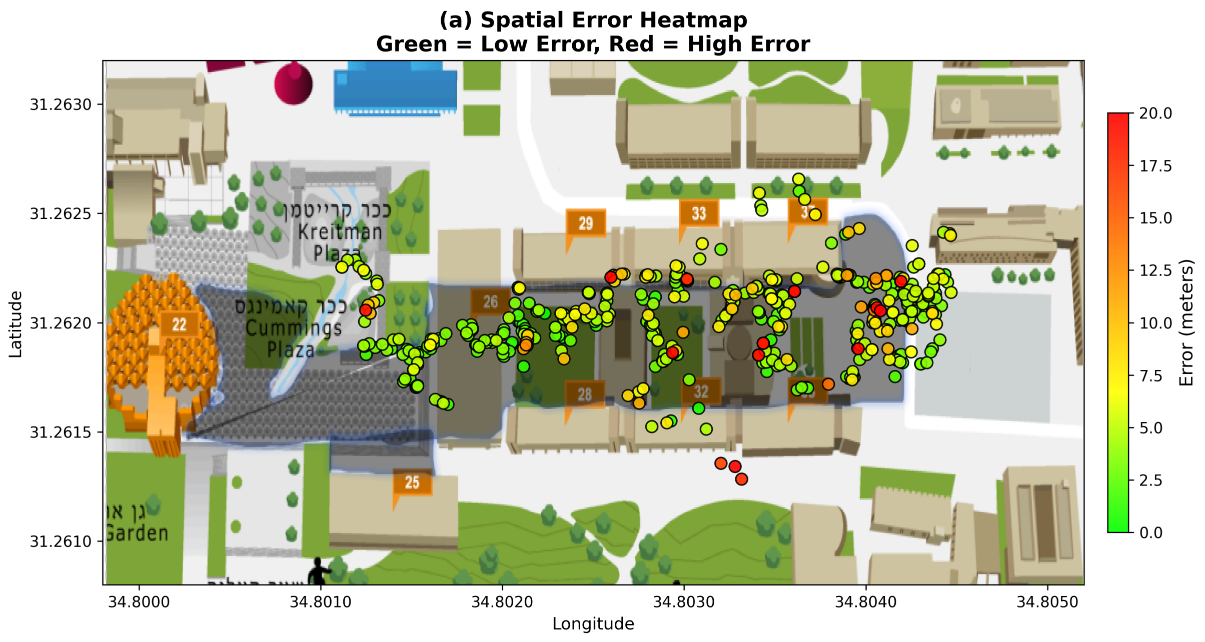


Figure 9: Final Spatial Error Heatmap (0-20m Scale). Green indicates high precision (<5m), while Red indicates local failure modes. Note the high consistency across most of the campus pathways.

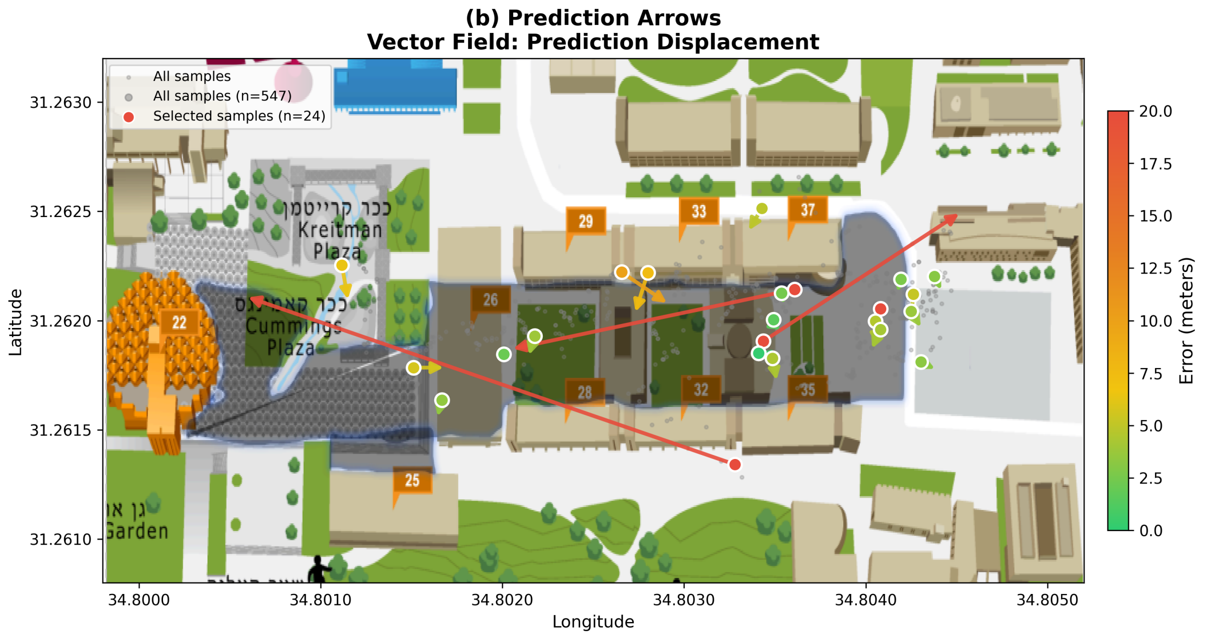


Figure 10: Prediction Vector Field. Arrows indicate prediction displacement. The overwhelming majority of arrows are negligible in size, indicating precise localization. Large displacements are rare and isolated.

9.3 Robustness Analysis

Finally, we tested the model against augmented test data (rotations, perspective warps) to simulate real-world usage conditions.

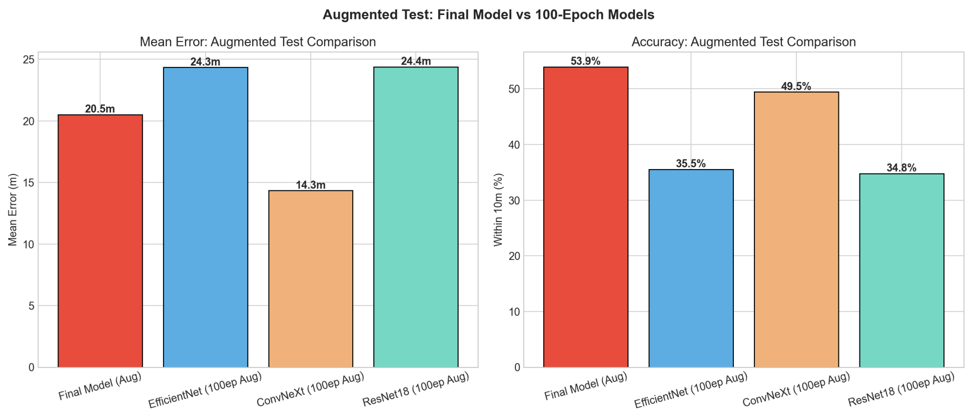


Figure 11: Robustness Evaluation. While performance naturally degrades with heavy augmentation, the Final Model maintains a usable median error of 9.03m, demonstrating reasonable resilience to viewpoint shifts.

10 Conclusion and Future Research

This research successfully demonstrated that a lightweight, efficient deep learning model can solve the visual geo-localization problem in a GPS-denied environment with meter-level accuracy. By fine-tuning an **EfficientNet-B0** backbone with a custom regression head, we achieved a **Median Error of 4.01 meters**, effectively enabling room-level localization purely from visual data.

10.1 Synthesis of Technical Contributions

Our analysis isolates several specific engineering choices that were critical to breaking the 10-meter error barrier:

- **Loss Landscape Alignment:** While standard Mean Squared Error (MSE) is theoretically viable for small, flat areas like a campus, the adoption of **Haversine Loss** proved superior for fine-grained accuracy. It aligns the optimization landscape directly with the geodesic distance metric, ensuring that gradient updates correspond exactly to physical improvements (meters) rather than abstract coordinate shifts.
- **Boundary-Aware Activation:** The implementation of the **Scaled Sigmoid** activation proved superior to standard Tanh or linear outputs. It mitigated the gradient saturation problems near the campus boundaries while ensuring predictions remained strictly within valid GPS domains.
- **Implicit vs. Explicit Features:** The failure of our SIFT-hybridization attempt confirmed that modern CNNs implicitly learn geometric features (corners, edges) more effectively for this specific regression task than classical, hand-crafted descriptors.

10.2 Architectural Trade-offs and Robustness

A key finding of this study is the divergence between *Peak Accuracy* and *Environmental Robustness.*

- **The Precision Specialist:** EfficientNet-B0 emerged as the optimal architecture for standard conditions, achieving the tightest error bounds (P95 = 18.01m). It excels at mapping texture details to coordinates but is sensitive to perturbations.
- **The Robust Generalist:** Conversely, **ConvNeXt-Tiny** demonstrated superior resilience. Under augmented test conditions (rotations and warps), its performance degraded by only **1.49**$\times$, compared to EfficientNet's **2.28**$\times$. This suggests that Transformer-inspired architectures may learn more "global," viewpoint-invariant representations.
- **Data Sensitivity:** Our ablation studies confirmed that visual localization is a data-hungry task. Reducing the dataset size by 50% resulted in a non-linear error increase of 68%, indicating that the feature space requires high-frequency sampling.

10.3 Future Research Directions

While this project establishes a strong baseline, two primary academic questions remain for future exploration:

1. **Temporal Domain Adaptation:** Our current dataset represents a static temporal snapshot. A critical next step is to evaluate model degradation across different times of day (lighting changes) and seasons (vegetation changes). Future research should explore **Domain Adaptation** techniques to align features from a source domain (Summer) to a target domain (Winter) without requiring a completely new labeled dataset.
2. **Probabilistic Uncertainty Estimation:** The current model behaves deterministically, occasionally producing catastrophic outliers (> 90m) when visual features are ambiguous. A significant advancement would be modeling **Aleatoric Uncertainty** by modifying the regression head to predict a probability distribution (x, y, σ) rather than a single point. This would allow the system to quantify its own confidence, filtering out unreliable predictions in real-time deployment.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CVPR*, 2016.

[2] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for CNNs," *ICML*, 2019.

[3] Z. Liu et al., "A ConvNet for the 2020s," *CVPR*, 2022.

[4] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *NeurIPS*, 2019.

[5] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *ICLR*, 2019.

[6] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision (IJCV)*, 2004.