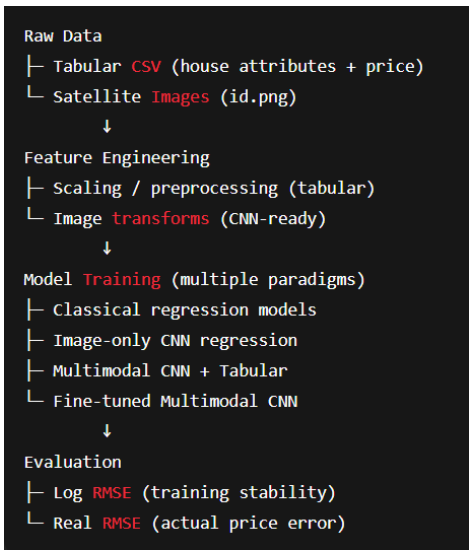
# Satellite-Aware Property Price Prediction using Multimodal Learning

## Overview and Architecture Diagram

My entire project flow is as follows




### Tabular Regression Models

The regression models were trained on the following features -

bedrooms, bathrooms, sqft_living, sqft_lot, sqft_living15, sqft_lot15, grade, condition, view, waterfront, lat, long

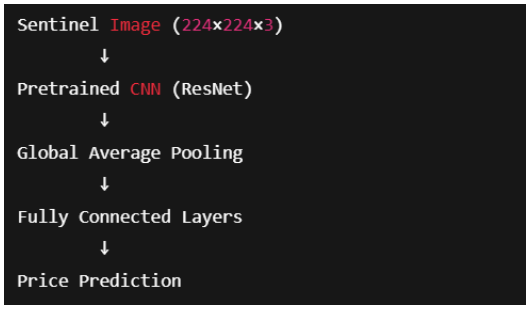
1. Linear Regression - Linear Regression provides a simple, interpretable baseline by assuming a linear relationship between these features and the target price.
2. Random Forest Regressor - Random Forest Regression improves upon this by modeling non-linear relationships through an ensemble of decision trees, capturing interactions between variables such as location and property size.
3. XGBoost Regressor -XGBoost further enhances performance by using gradient-boosted trees with regularization, allowing the model to learn complex patterns while avoiding overfitting.

Among these, XGBoost achieves the strongest performance, demonstrating that well-engineered tabular features alone can explain a significant portion of the variation in housing prices.

### Image-Only Model (CNN Regression)

Each satellite image is processed through a pretrained ResNet backbone, which extracts high-level visual features such as neighborhood density, road networks, proximity to water bodies, and surrounding land use. This model captures spatial and environmental context that is difficult to encode manually in tabular form.

Model Architecture-




Key Details

-CNN initialized with ImageNet weights

-Final regression head predicts log(price)

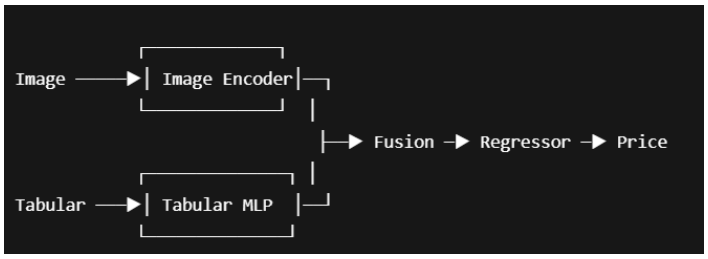
-Loss: MSE on log scale

## Multimodal Regression (Frozen CNN)

The multimodal regression model combines both satellite images and tabular property features to leverage complementary information from each modality. In the initial multimodal setup, the CNN backbone is largely frozen to stabilize training and prevent overfitting

Model Architecture-

The image branch uses a pretrained CNN to extract visual embeddings, while the tabular branch employs a multi-layer perceptron (MLP) encoder to transform structured features into a latent representation. These two embeddings are concatenated and passed through a multimodal regressor to produce the final price prediction.




**Components**

1. **Image Encoder**

-Pretrained ResNet

-Mostly frozen (feature extractor)

2. **Tabular Encoder**

-Small MLP (2–3 layers)

3. **Fusion Layer**

-Concatenation of embeddings

4. **Regressor**

-Fully connected layers → price

### Fine-Tuned Multimodal Model

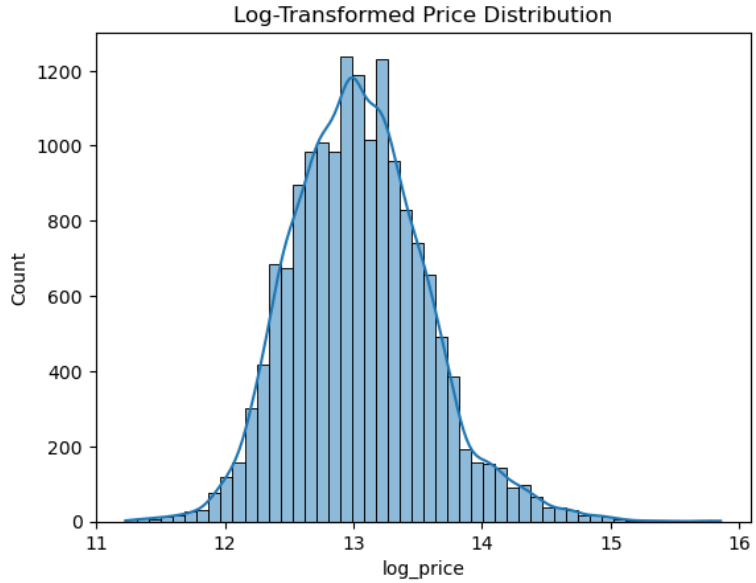
The fine-tuned multimodal model extends this approach by selectively unfreezing the later layers of the CNN, allowing high-level visual features to adapt specifically to the housing price prediction task. This fine-tuning improves the model's ability to align visual context with property attributes, leading to better performance compared to the frozen multimodal model.

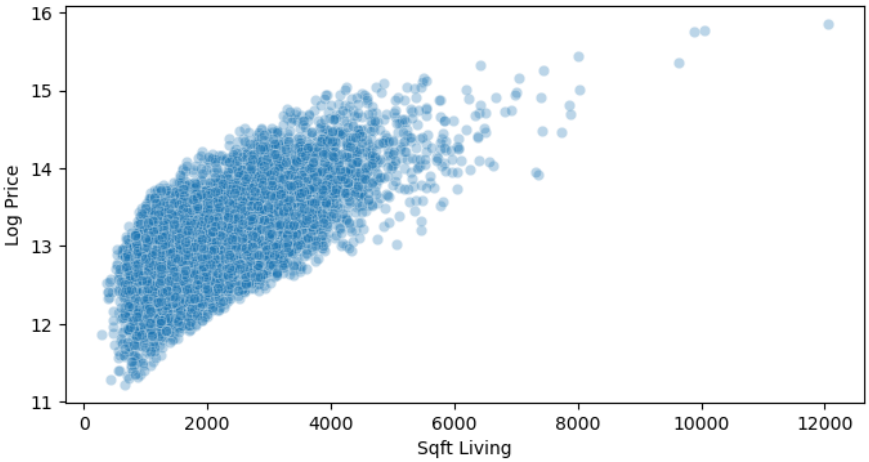
Model Architecture-

```
ResNet Backbone
 ├── Frozen early layers (edges, textures)
 └── Trainable layer4 (high-level semantics)
```

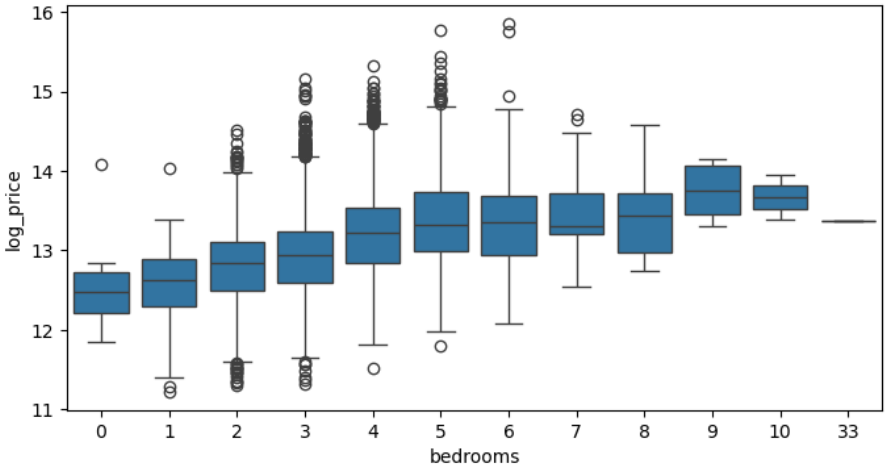
# EDA

### Distribution of House Prices (Histograms)

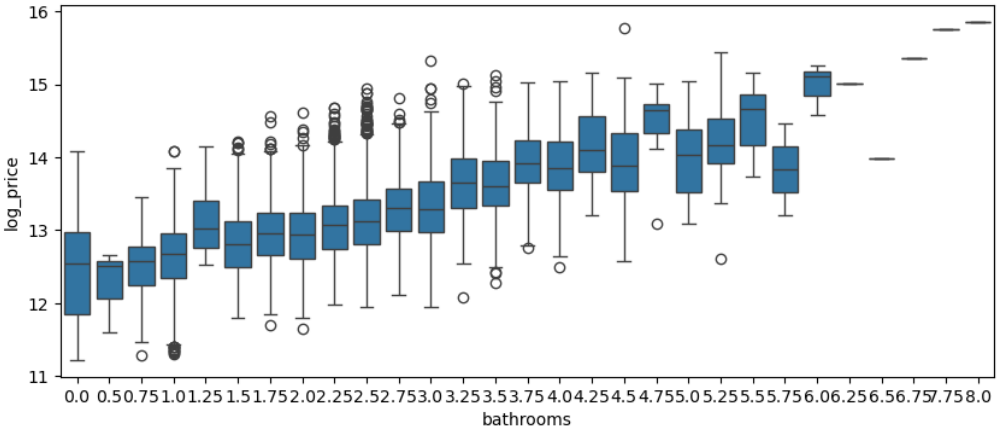





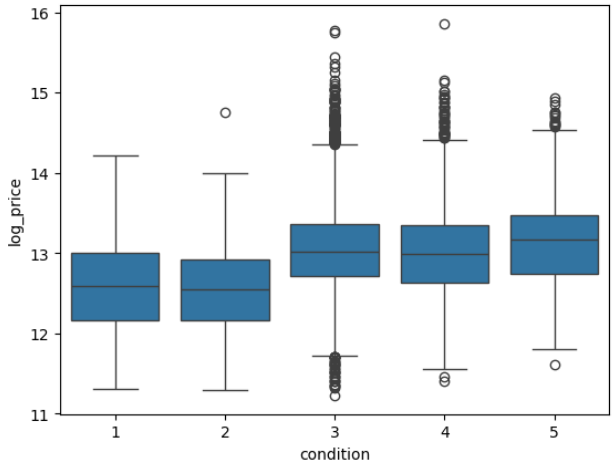

**Scatter plot**

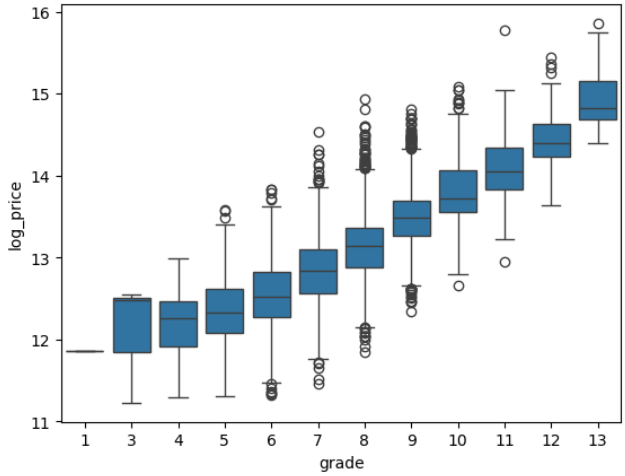



---

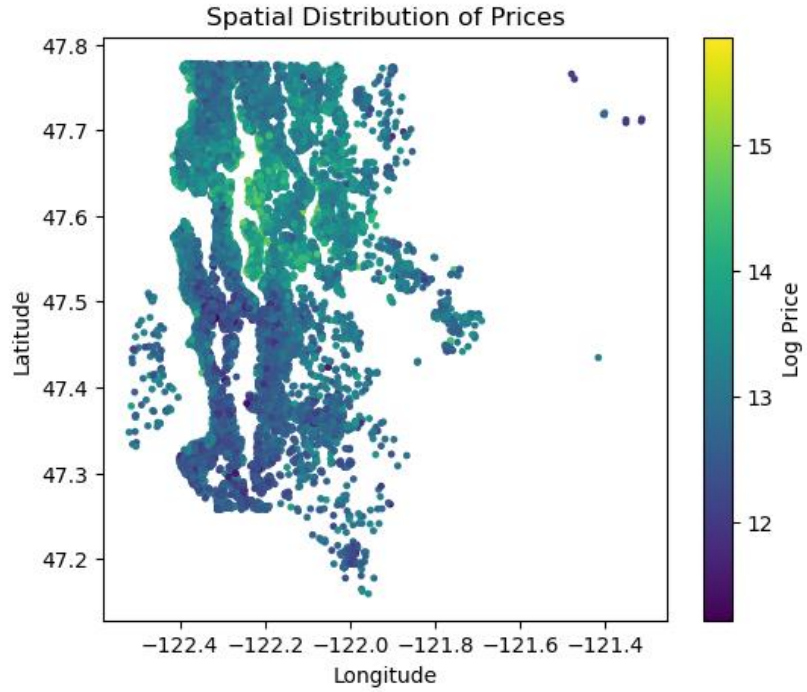



---

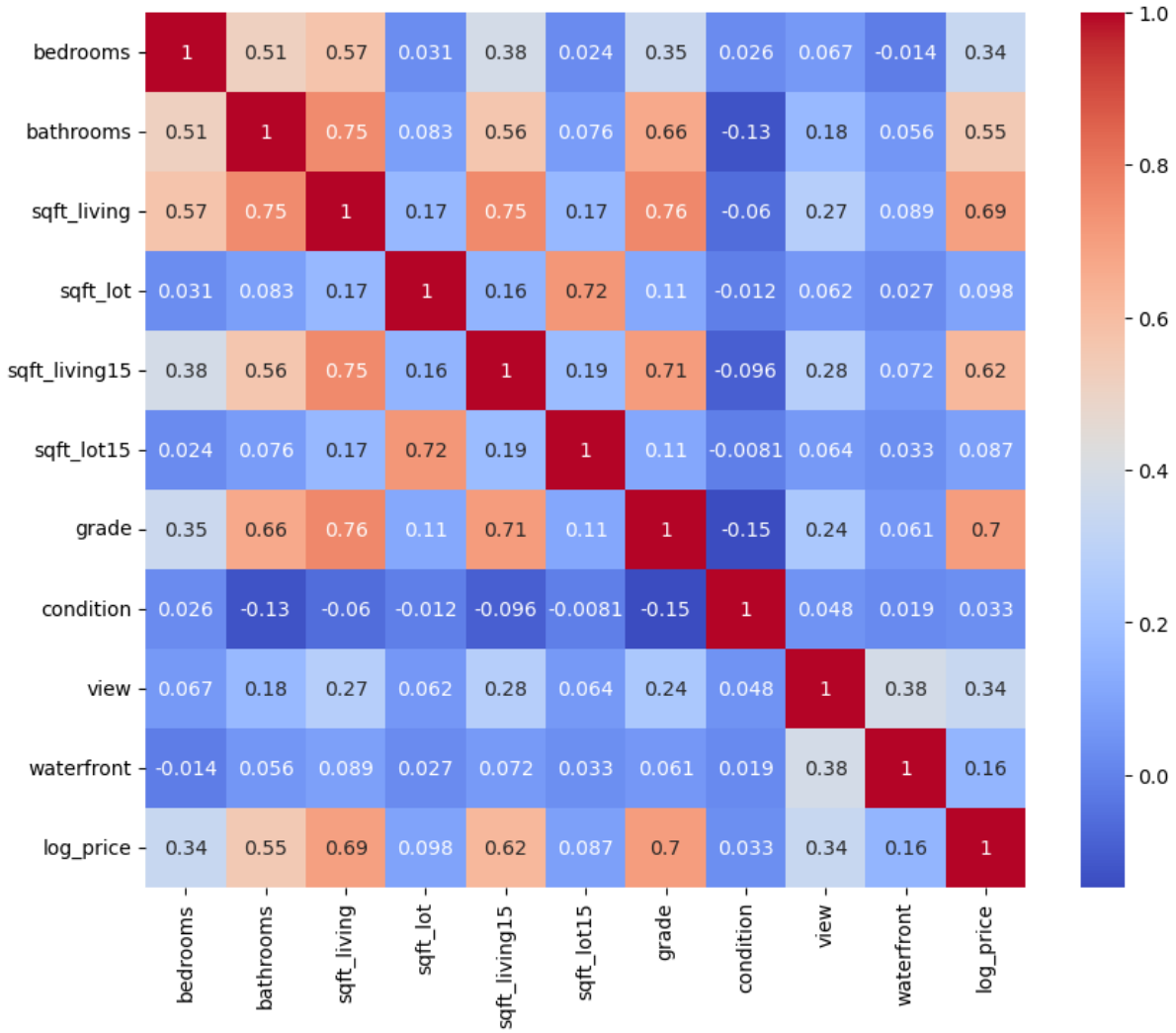

log_price
condition
16
15
14
13
12
11
1
2
3
4
5


log_price
grade
16
15
14
13
12
11
1
3
4
5
6
7
8
9
10
11
12
13

**Box Plots**


Spatial Distribution of Prices
Latitude
Longitude
Log Price
47.8
47.7
47.6
47.5
47.4
47.3
47.2
−122.4
−122.2
−122.0
−121.8
−121.6
−121.4
15
14
13
12

## Scatter Plot



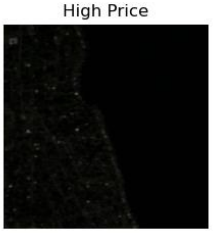

**Heatmap of different parameters**

## Satellite images

High Price

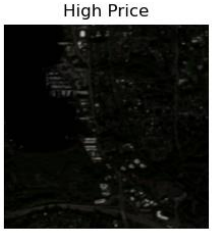
High Price

High Price



High Price

High Price



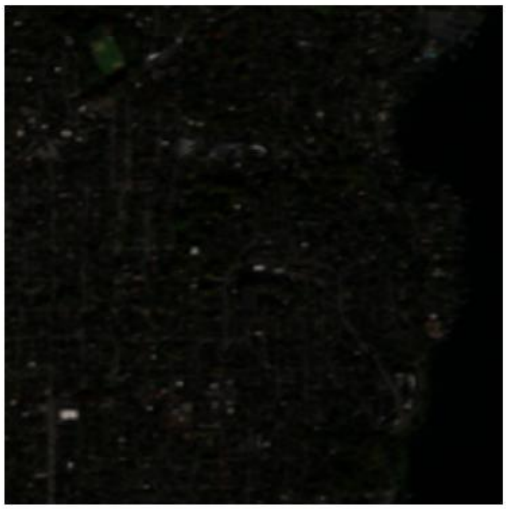
Low Price

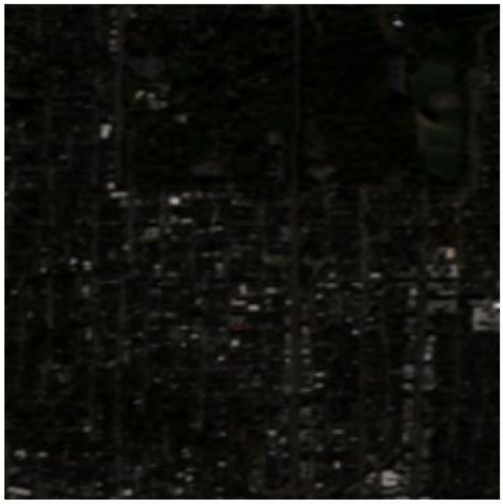
Low Price

Low Price

Low Price

Low Price





# Results

**Tabular Data based results(RMSE is measured)**

| | Model | RMSE |
|---|---|---|
| 0 | Linear | 0.261496 |
| 1 | Random Forest | 0.177452 |
| 2 | XGBoost | 0.166699 |

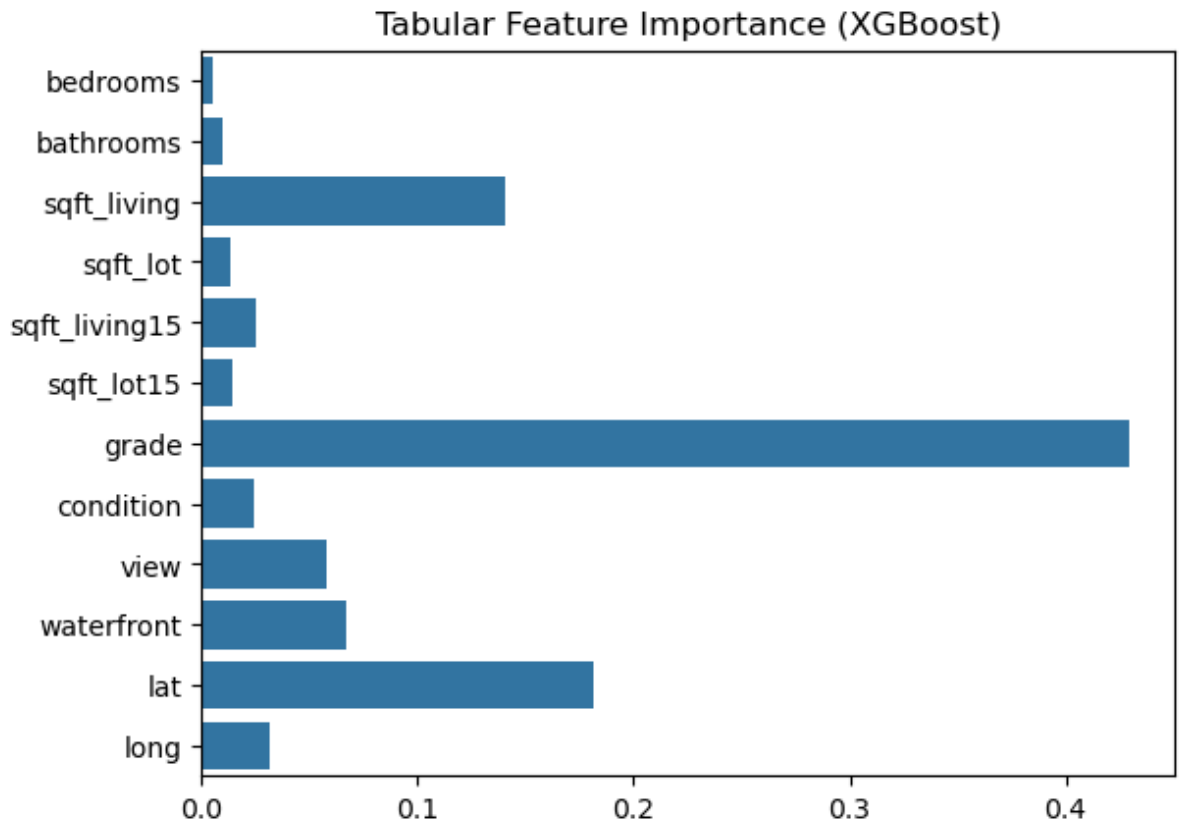



**Image only model results**

```
100%|██████████| 406/406 [04:02<00:00,  1.67it/s]
Epoch 1/20 | Train MSE: 22.53 | Val RMSE: 0.46
Val RMSE (real prices): 326,056
100%|██████████| 406/406 [01:47<00:00,  3.78it/s]
Epoch 2/20 | Train MSE: 0.17 | Val RMSE: 0.38
Val RMSE (real prices): 296,331
100%|██████████| 406/406 [01:43<00:00,  3.93it/s]
Epoch 3/20 | Train MSE: 0.14 | Val RMSE: 0.36
Val RMSE (real prices): 270,946
100%|██████████| 406/406 [01:48<00:00,  3.74it/s]
Epoch 4/20 | Train MSE: 0.13 | Val RMSE: 0.36
Val RMSE (real prices): 256,567
100%|██████████| 406/406 [01:44<00:00,  3.88it/s]
Epoch 5/20 | Train MSE: 0.12 | Val RMSE: 0.34
Val RMSE (real prices): 244,429
100%|██████████| 406/406 [01:46<00:00,  3.81it/s]
Epoch 6/20 | Train MSE: 0.11 | Val RMSE: 0.36
Val RMSE (real prices): 266,081
100%|██████████| 406/406 [01:44<00:00,  3.87it/s]
Epoch 7/20 | Train MSE: 0.10 | Val RMSE: 0.33
Val RMSE (real prices): 247,117
100%|██████████| 406/406 [01:45<00:00,  3.85it/s]
Epoch 8/20 | Train MSE: 0.09 | Val RMSE: 0.33
Val RMSE (real prices): 250,827
100%|██████████| 406/406 [01:44<00:00,  3.88it/s]
Epoch 9/20 | Train MSE: 0.09 | Val RMSE: 0.33
Val RMSE (real prices): 247,793
100%|██████████| 406/406 [01:44<00:00,  3.89it/s]
Epoch 10/20 | Train MSE: 0.08 | Val RMSE: 0.32
Val RMSE (real prices): 239,164
100%|██████████| 406/406 [01:45<00:00,  3.83it/s]
Epoch 11/20 | Train MSE: 0.07 | Val RMSE: 0.32
Val RMSE (real prices): 238,142
100%|██████████| 406/406 [01:45<00:00,  3.86it/s]
Epoch 12/20 | Train MSE: 0.07 | Val RMSE: 0.32
Val RMSE (real prices): 245,936
100%|██████████| 406/406 [01:46<00:00,  3.82it/s]
Epoch 13/20 | Train MSE: 0.06 | Val RMSE: 0.43
Val RMSE (real prices): 288,469
100%|██████████| 406/406 [01:43<00:00,  3.94it/s]
Epoch 14/20 | Train MSE: 0.06 | Val RMSE: 0.35
Val RMSE (real prices): 261,161
100%|██████████| 406/406 [01:43<00:00,  3.94it/s]
Epoch 15/20 | Train MSE: 0.05 | Val RMSE: 0.32
Val RMSE (real prices): 239,537
Early stopping triggered.
```

**Multimodal Regression (Frozen CNN)**

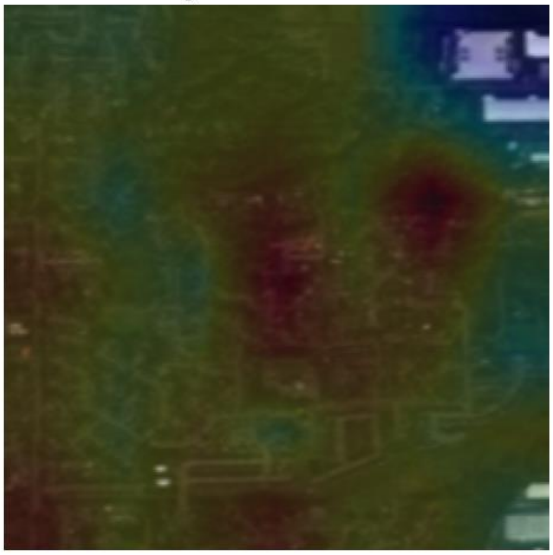
```
Epoch 1 | Train Loss (log MSE): 10.7016 | Log RMSE: 0.6645 | Real RMSE: 356,983
Epoch 2 | Train Loss (log MSE): 2.1261 | Log RMSE: 0.5351 | Real RMSE: 353,840
Epoch 3 | Train Loss (log MSE): 2.0324 | Log RMSE: 0.7889 | Real RMSE: 406,495
Epoch 4 | Train Loss (log MSE): 1.9596 | Log RMSE: 0.9549 | Real RMSE: 436,962
Epoch 5 | Train Loss (log MSE): 1.8340 | Log RMSE: 0.5044 | Real RMSE: 374,792
Epoch 6 | Train Loss (log MSE): 1.7126 | Log RMSE: 0.5653 | Real RMSE: 562,613
Epoch 7 | Train Loss (log MSE): 1.6890 | Log RMSE: 0.4570 | Real RMSE: 294,050
Epoch 8 | Train Loss (log MSE): 1.6325 | Log RMSE: 0.4937 | Real RMSE: 318,614
Epoch 9 | Train Loss (log MSE): 1.6421 | Log RMSE: 0.5133 | Real RMSE: 381,979
Epoch 10 | Train Loss (log MSE): 1.5897 | Log RMSE: 0.4829 | Real RMSE: 355,568
Epoch 11 | Train Loss (log MSE): 1.5612 | Log RMSE: 0.5068 | Real RMSE: 326,922
Early stopping triggered at epoch 11
```

**Fine Tune Multimodal Model**

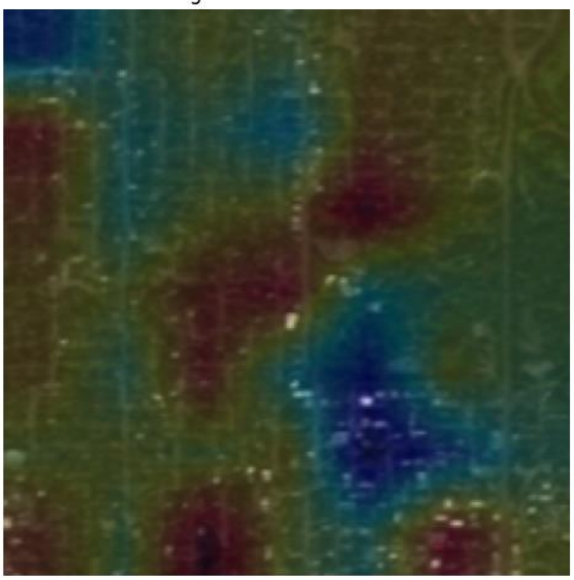
```
[Day7] Epoch 1 | Train Loss (log MSE): 1.5558 | Log RMSE: 0.4713 | Real RMSE: 286,978
[Day7] Epoch 2 | Train Loss (log MSE): 1.4809 | Log RMSE: 0.5247 | Real RMSE: 320,007
[Day7] Epoch 3 | Train Loss (log MSE): 1.4257 | Log RMSE: 0.5102 | Real RMSE: 312,695
[Day7] Epoch 4 | Train Loss (log MSE): 1.4028 | Log RMSE: 0.5020 | Real RMSE: 285,655
[Day7] Epoch 5 | Train Loss (log MSE): 1.3664 | Log RMSE: 0.4240 | Real RMSE: 282,475
[Day7] Epoch 6 | Train Loss (log MSE): 1.2931 | Log RMSE: 0.5024 | Real RMSE: 306,464
[Day7] Epoch 7 | Train Loss (log MSE): 1.2401 | Log RMSE: 0.4067 | Real RMSE: 266,295
[Day7] Epoch 8 | Train Loss (log MSE): 1.1853 | Log RMSE: 0.3990 | Real RMSE: 253,901
[Day7] Epoch 9 | Train Loss (log MSE): 1.1668 | Log RMSE: 0.4242 | Real RMSE: 274,599
[Day7] Epoch 10 | Train Loss (log MSE): 1.1282 | Log RMSE: 0.3888 | Real RMSE: 263,560
[Day7] Epoch 11 | Train Loss (log MSE): 1.0569 | Log RMSE: 0.3836 | Real RMSE: 247,480
[Day7] Epoch 12 | Train Loss (log MSE): 1.0405 | Log RMSE: 0.4350 | Real RMSE: 302,735
[Day7] Epoch 13 | Train Loss (log MSE): 0.9910 | Log RMSE: 0.3907 | Real RMSE: 251,414
[Day7] Epoch 14 | Train Loss (log MSE): 0.9691 | Log RMSE: 0.4926 | Real RMSE: 322,332
[Day7] Epoch 15 | Train Loss (log MSE): 0.9354 | Log RMSE: 0.4426 | Real RMSE: 297,943
[Day7] Epoch 16 | Train Loss (log MSE): 0.8707 | Log RMSE: 0.3790 | Real RMSE: 259,318
[Day7] Epoch 17 | Train Loss (log MSE): 0.8603 | Log RMSE: 0.4193 | Real RMSE: 269,556
[Day7] Epoch 18 | Train Loss (log MSE): 0.7944 | Log RMSE: 0.3995 | Real RMSE: 269,119
[Day7] Epoch 19 | Train Loss (log MSE): 0.7716 | Log RMSE: 0.4604 | Real RMSE: 298,968
[Day7] Epoch 20 | Train Loss (log MSE): 0.7414 | Log RMSE: 0.3726 | Real RMSE: 245,746
```

# Financial/Visual Insights

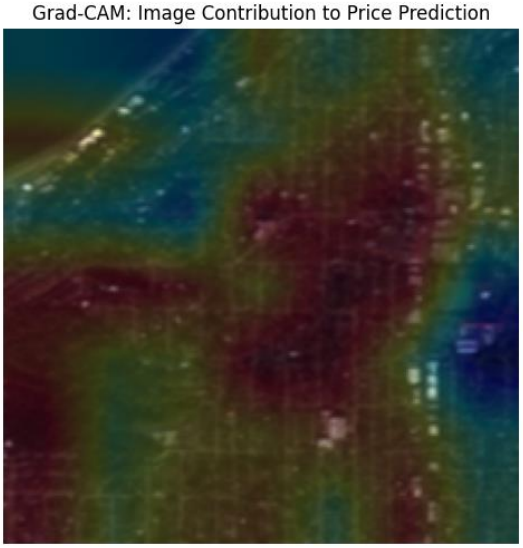
Grad-CAM: Image Contribution to Price Prediction

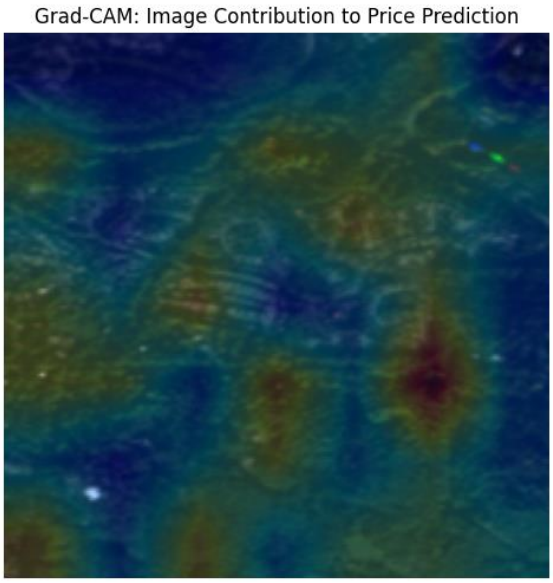


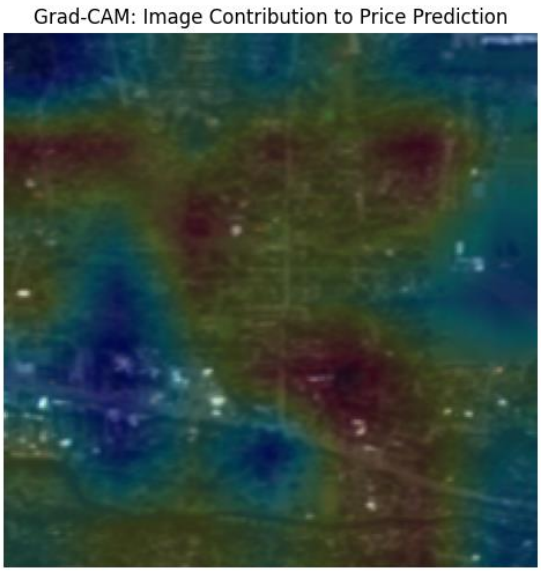
Grad-CAM: Image Contribution to Price Prediction

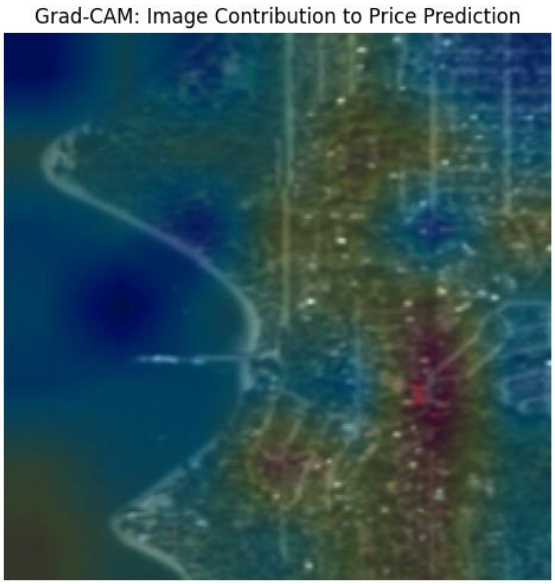


Grad-CAM: Image Contribution to Price Prediction






Grad-CAM (Gradient-weighted Class Activation Mapping) is a visualization technique used to understand which regions of an input image influence a model's prediction

## Grad-CAM in Image-Based Regression Tasks

Although Grad-CAM is commonly used for classification, it is equally applicable to **regression problems**, such as price prediction. In this setting, the activation map highlights image regions that **most affect the predicted numerical value**. Instead of explaining why an image belongs to a class, Grad-CAM explains **how different spatial regions contribute to increasing or decreasing the output value**.

## Meaning of Highlighted Regions in Grad-CAM

The heatmap produced by Grad-CAM uses color intensity to indicate importance. **Warmer colors (red/yellow)** represent regions that have a stronger influence on the prediction, while **cooler colors (blue)** indicate areas with less influence. These regions do not directly represent real-world value; rather, they show **where the model is focusing its attention** when making a decision.

## Importance of Grad-CAM for Satellite and Aerial Images

In satellite or aerial imagery, there are no explicit labels for individual objects. Grad-CAM helps verify that the model is using **meaningful spatial cues**, such as layout, density, and structure, rather than irrelevant textures or noise. This is especially important when images are used to infer complex attributes like socioeconomic or environmental indicators.

## Role of Grad-CAM in Multimodal Models

When image data is combined with tabular data, Grad-CAM helps assess whether the **image branch contributes useful information** to the final prediction. By visualizing attention over image regions, Grad-CAM confirms that the visual modality complements numerical features rather than being ignored or overshadowed.

## Interpretability and Model Trust

Grad-CAM increases confidence in model predictions by providing a **human-interpretable explanation** of image usage. If important regions consistently align with meaningful visual patterns, it suggests that the model has learned relevant representations. This interpretability is crucial for validating models used in real-world or high-stakes applications.