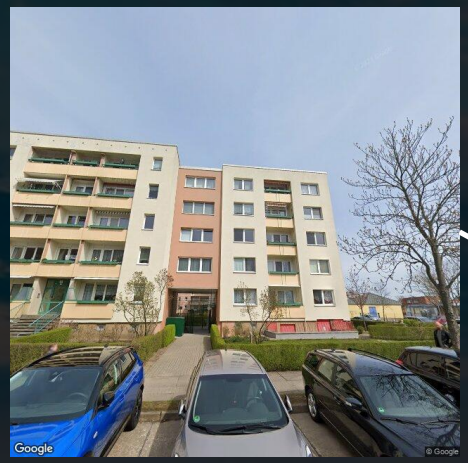
GeoValuator

MLHO – final project

by Leonardo da Camara Silva and Lukas Jehn

GeoValuator

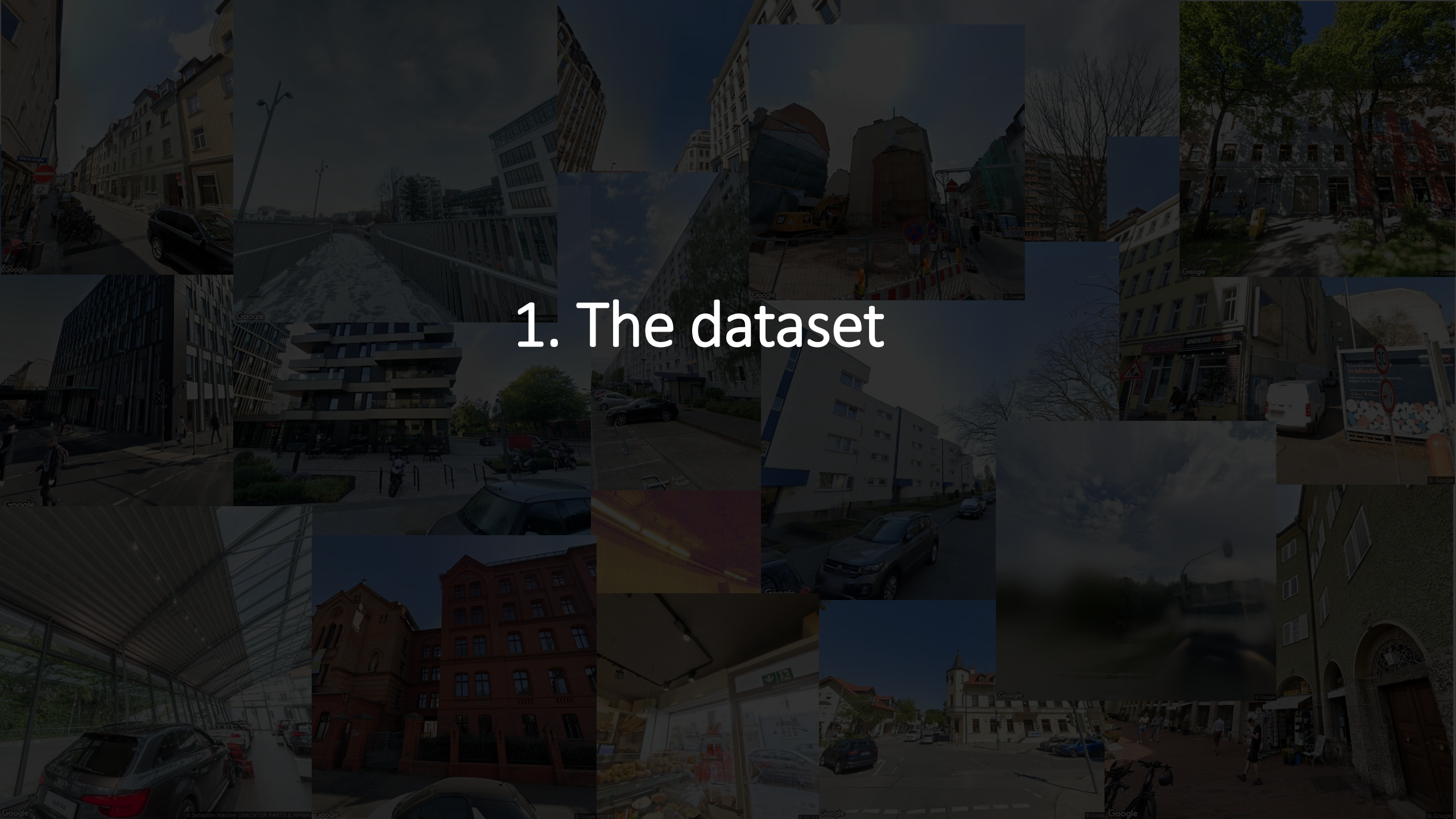


13,42€ per m^2

∞ € per m^2

Convolutional neural network (CNN) trained by Google Street View data of Berlin and Munich

Contents



1. The dataset

The dataset

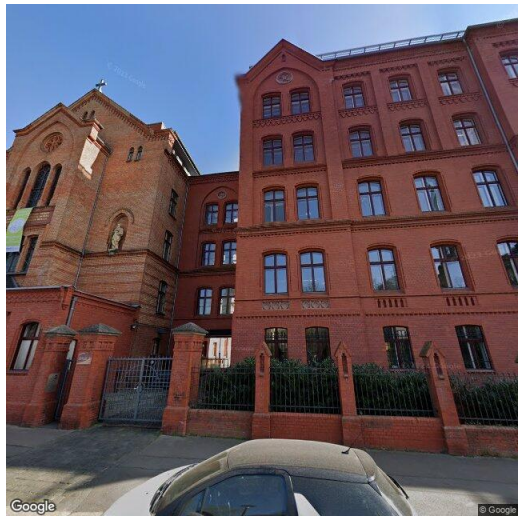
Machine Learning projects depend on complex & varied data

Goal: Create large-scale visual dataset for urban analysis that is labelled with a price tag

- Automatic data extraction pipeline

Berlin

20.11 €



22.39 €

Munich

23.03 €

27.40 €

Predict StreetView Image prices

Berlin, Mitte 22.39 €

Berlin, Marzahn 10.28 €

Predict StreetView Image prices

Berlin, Mitte 22.39 €

Berlin, Mitte 22.39 €

Problems with image price prediction

The appearance of a district with one price label can vary greatly

- Houses in poor & good condition
- Green regions in city center and scycraper outside the city
- Temporary construction sites
- Different weather conditions

➢ Use a large dataset to generate a wide range of variations

Munich, Schwantlerhöhe 25.35 €

Pipeline for data extraction

1) Geographic data processing
    - District data from webpages [1,2,3]: Boundary coordinates & district prices
    - Bin districts in price classes
    - Get coordinates by stratified sampling
    - Normalize the price labels to [0,1] -> less gradient problems

2) Get Google StreetView images
    - Google StreetView API integration with URL signing [4]
    - Checkpoint system for resumable downloads if connection gets lost
    - Retry coordinates if errors arrise

➢ Dataset: Images + Coordinates + Price Meta Data

Berlin, Mitte 22.39 €
(52.52206, 13.37647)

Filter out unusual images by hand

Munich, Aidhausen

Interior

Underground

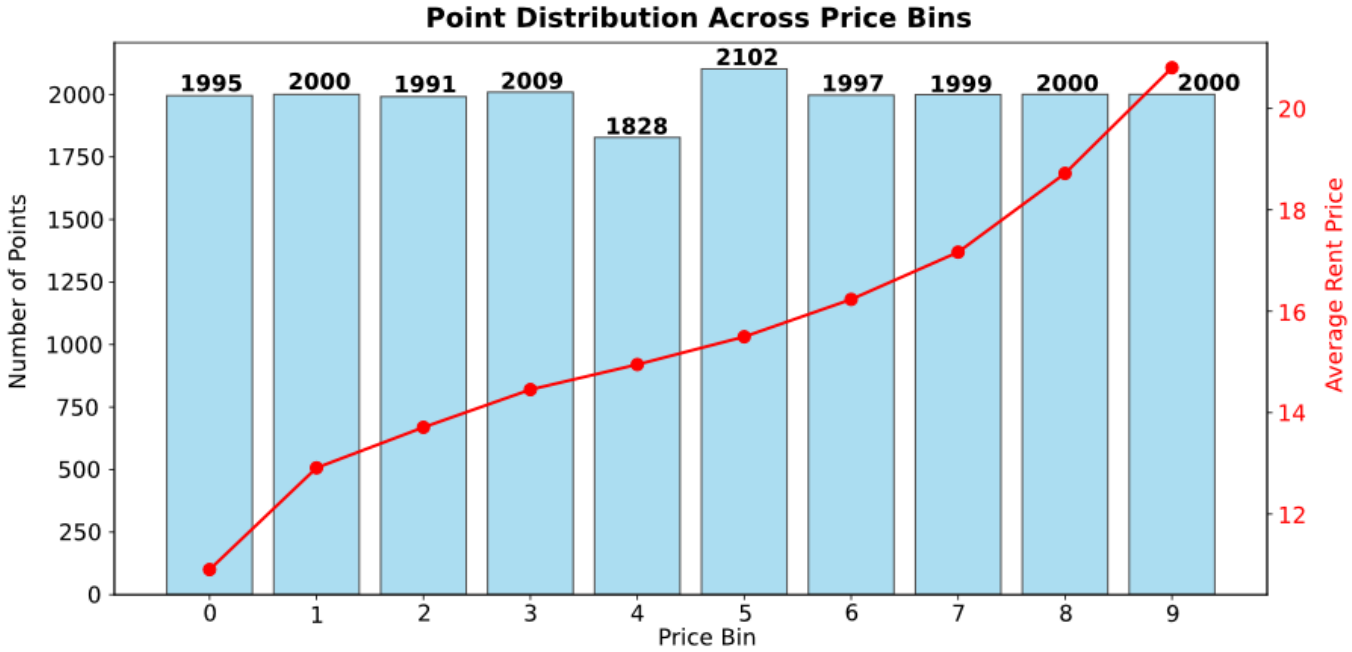
Interior

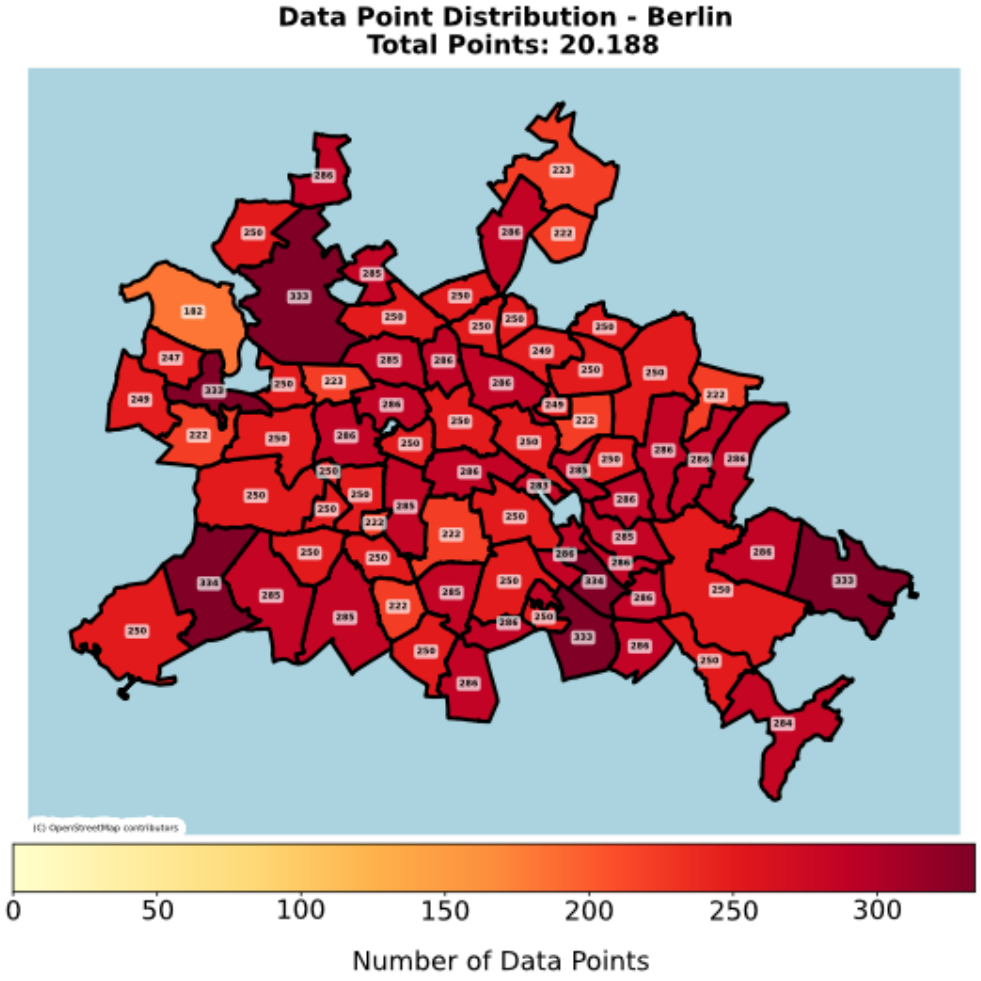
Blurred

Berlin Dataset

Statistics

- Total districts: 75
- Maximum coordinates in district: 334

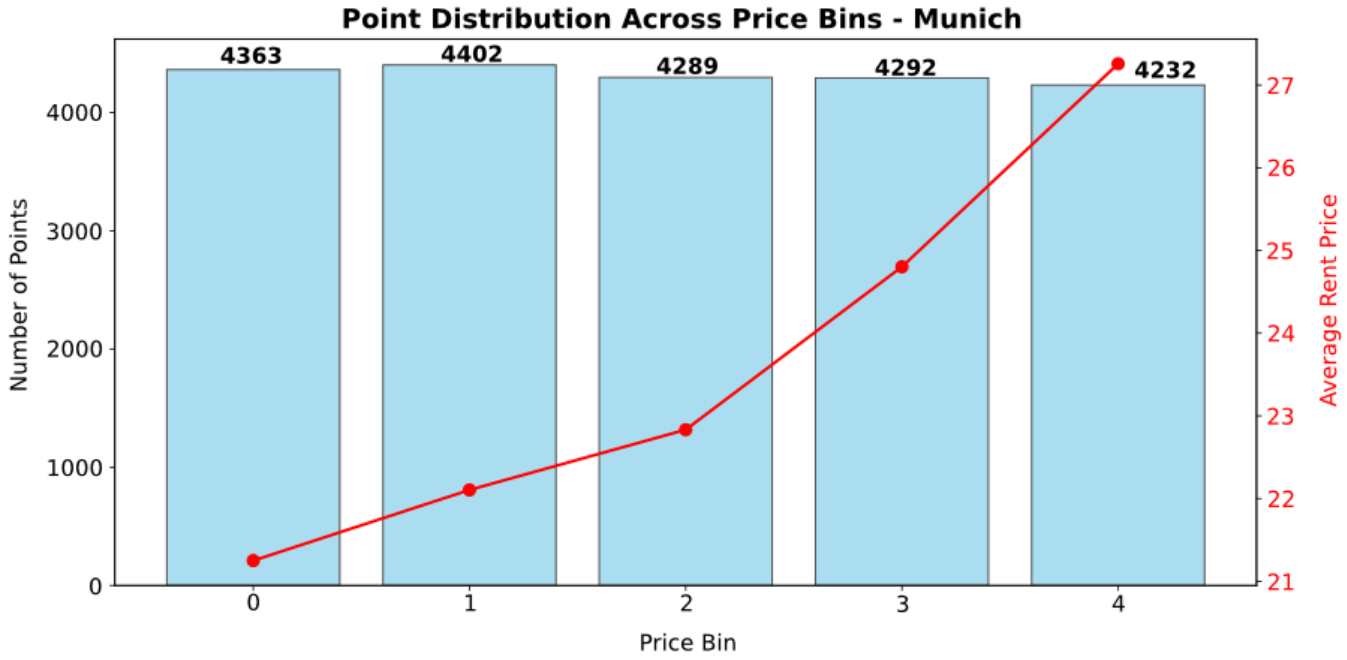


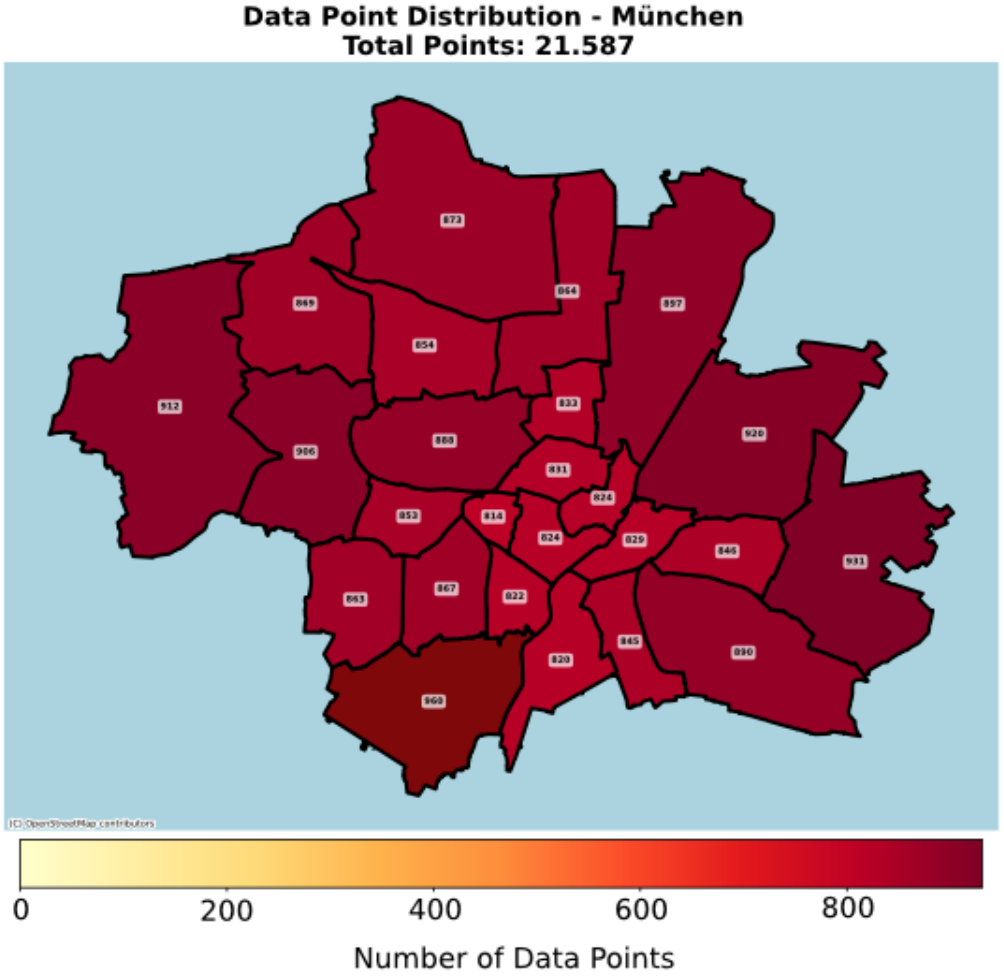


Munich Dataset

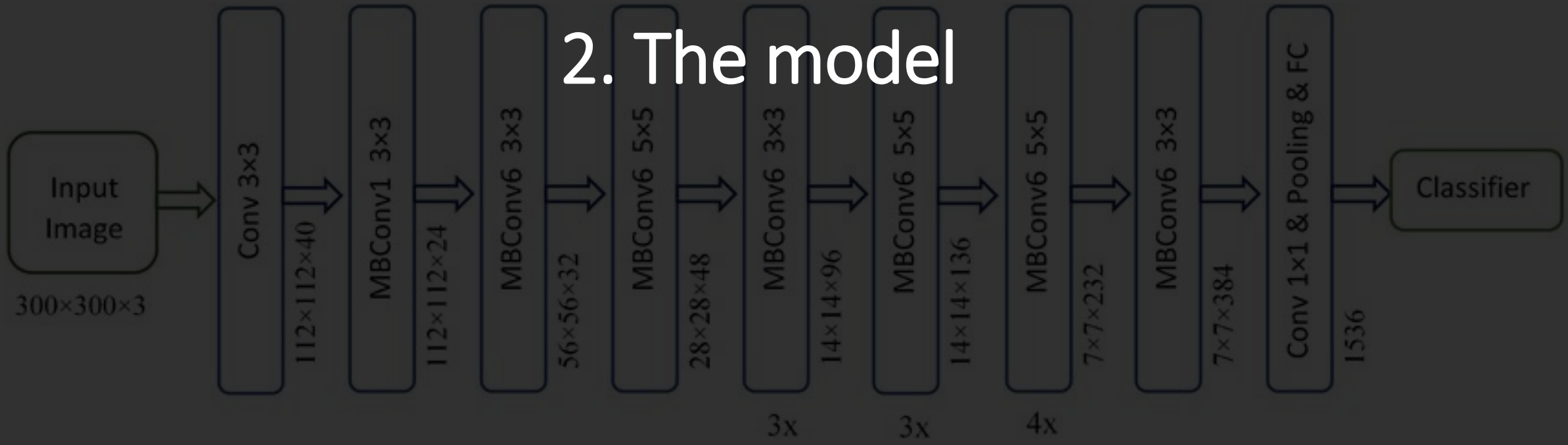
Statistics

- Total districts: 27
- Maximum coordinates in district: 961

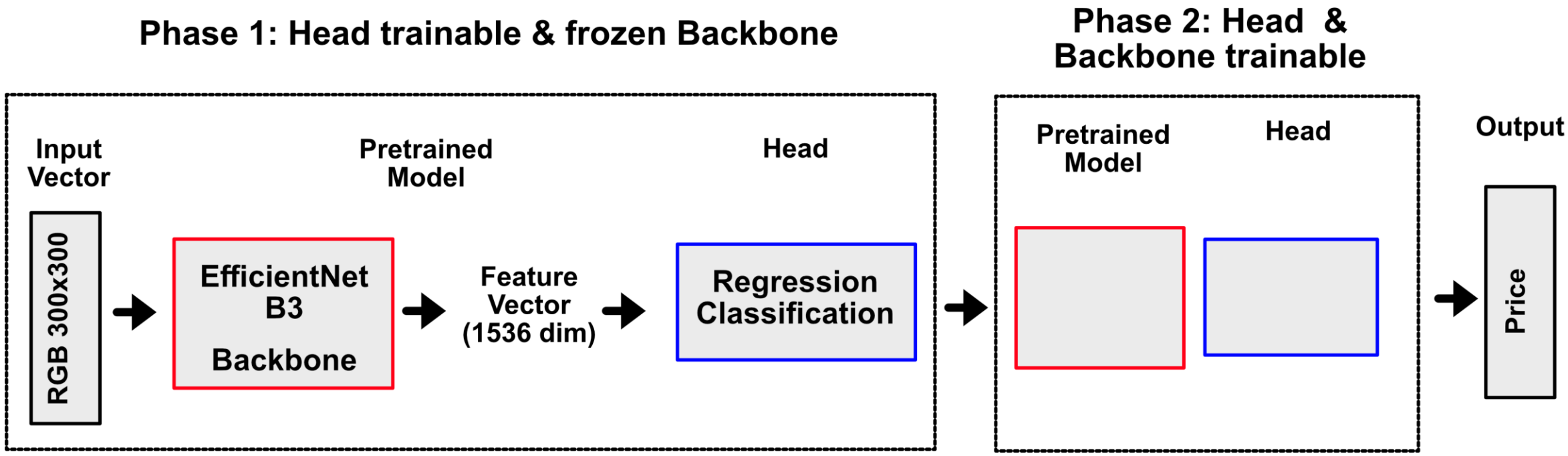




2. The model


Input Image
300×300×3
Conv 3×3
112×112×40
MBConv1 3×3
112×112×24
MBConv6 3×3
56×56×32
MBConv6 5×5
28×28×48
MBConv6 3×3
3x
14×14×96
MBConv6 5×5
3x
14×14×136
MBConv6 5×5
4x
7×7×232
MBConv6 3×3
7×7×384
Conv 1×1 & Pooling & FC
1536
Classifier

Model pipeline

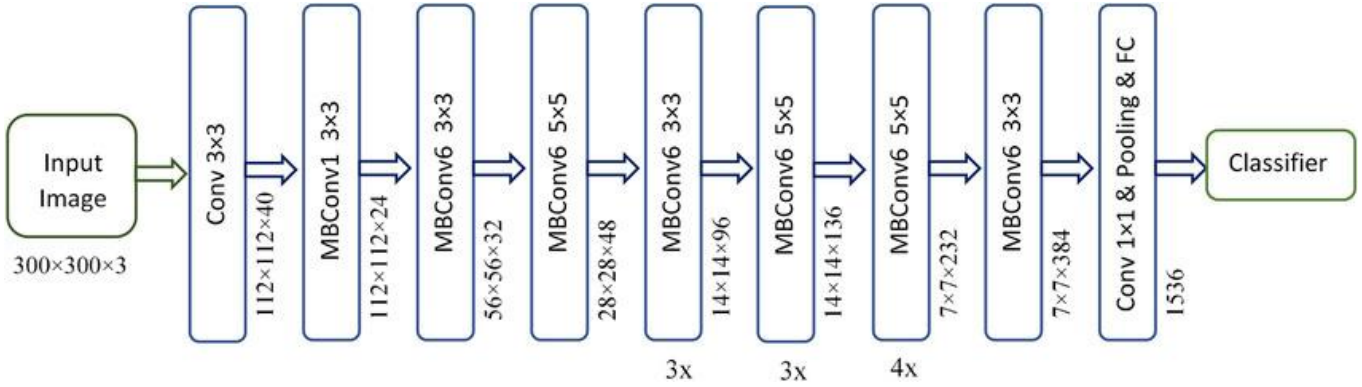


- Transfer Learning: Leverage ImageNet pretrained weights
- Progressive Unfreezing: Prevent forgetting
- Differential Learning Rates: Backbone learns slower than head

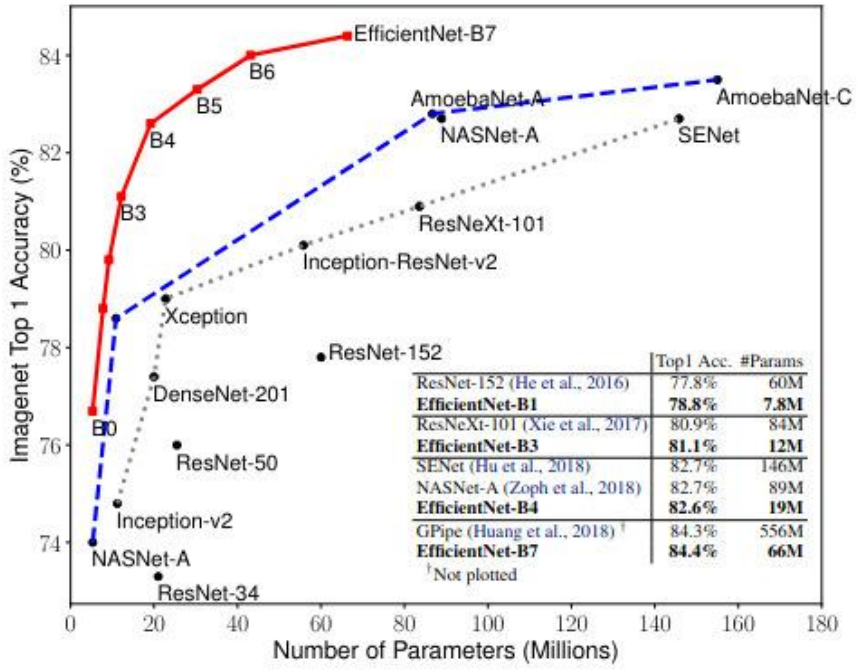
EfficientNet-B3

- Idea: Compound Scaling Method
  1) Depth – Number of **Layers** in the network
  2) Width – Number of **Channels** in the network
  3) Resolution – **Image Height** and **Width**

  ➢ Better performance and efficiency



[6] Sleimanipour et al., *Cultivar identification of pistachio nuts in bulk mode through EfficientNet deep learning mode*, (2022)



[7]Tan et al., *EfficientNet: Rethinking Model Scaling for Convolutional Neural Network,* (2019)

3. Performance evaluation

3.1 Classification

Classification

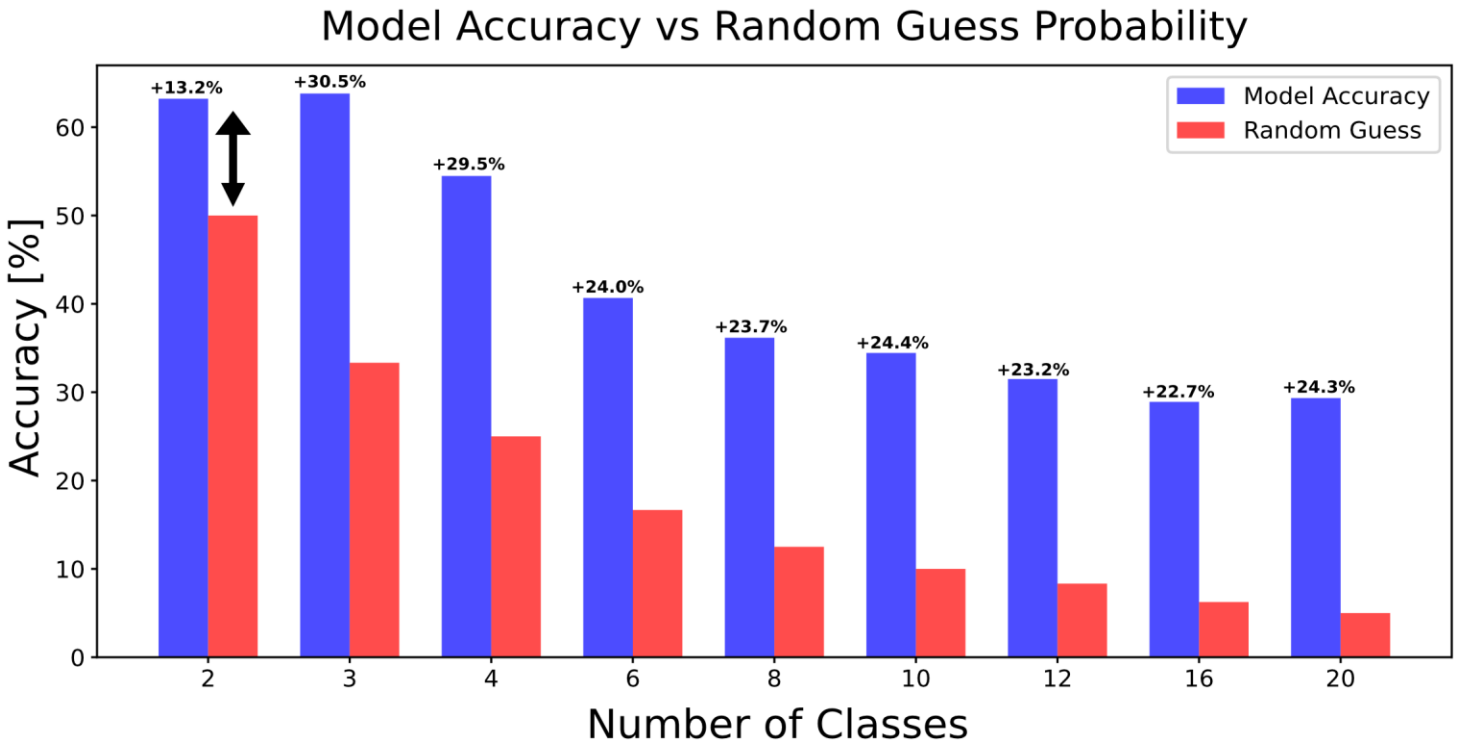
- Output: Probability vector with all classes
  [0.2, 0.7, 0.05, …]

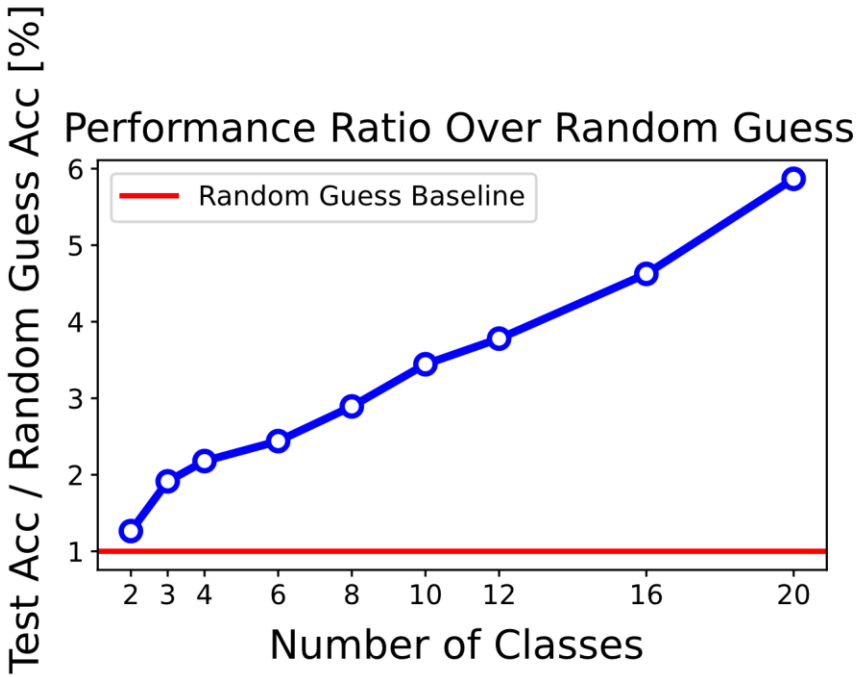
- BackBone Feature Extraction:
  - EfficientNet-B3
  - Input: RGB street view images
  - Transferlearning

- Classification Head:
  - Two fully-connected Layer
  - Dropout regularization (prevent overfitting)

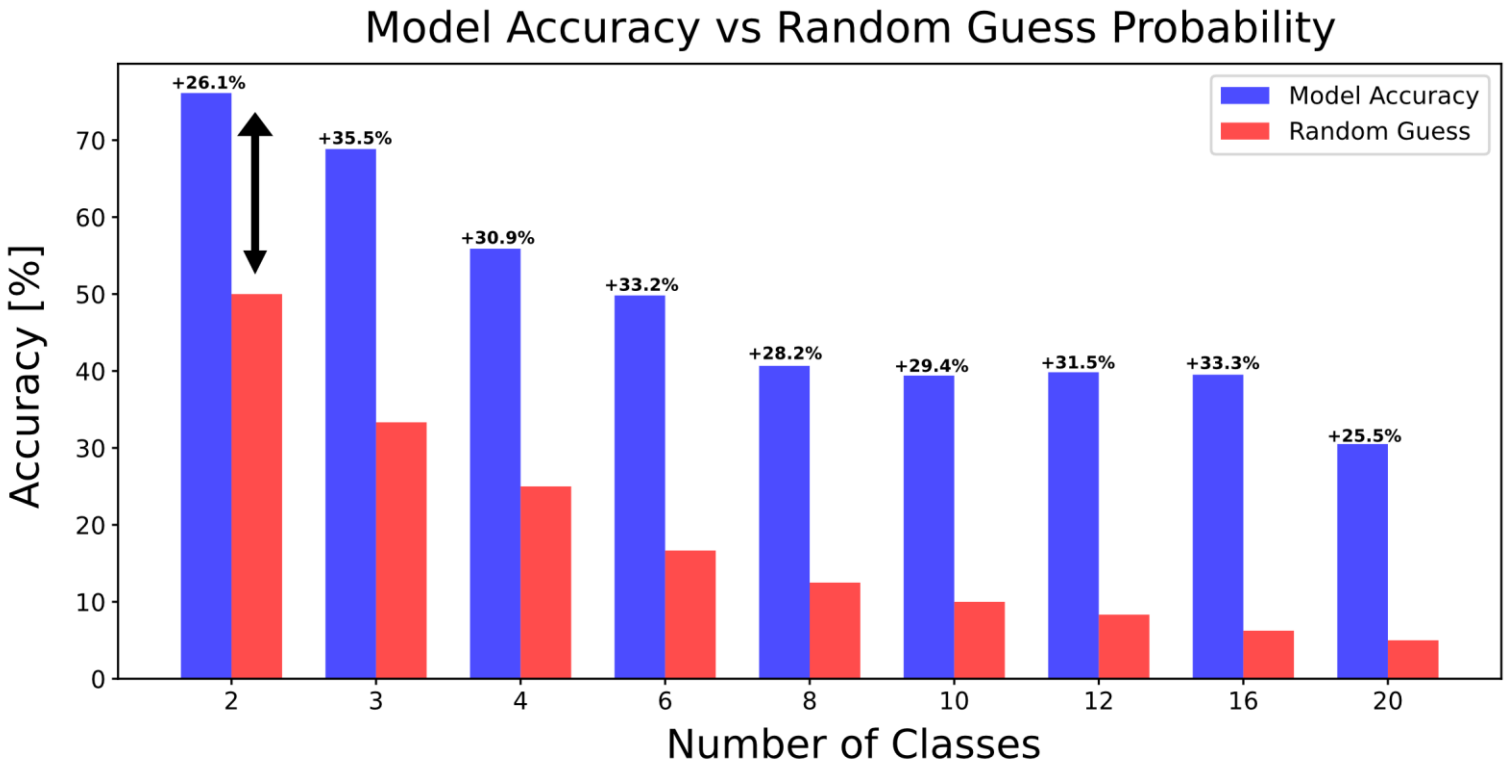
```python
class EfficientNetClassifier(nn.Module):
    def __init__(self, num_classes=10, model_name='efficientnet_b3'):
        super(EfficientNetClassifier, self).__init__()

        self.backbone = timm.create_model(model_name,
                                          pretrained=True,
                                          num_classes=0,
                                          global_pool='avg')

        self.classifier = nn.Sequential(
            nn.Dropout(0.2),
            nn.Linear(self.backbone.num_features, 512),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(512, num_classes)
        )

    def forward(self, x):
        features = self.backbone(x)
        output = self.classifier(features)
        return output
```

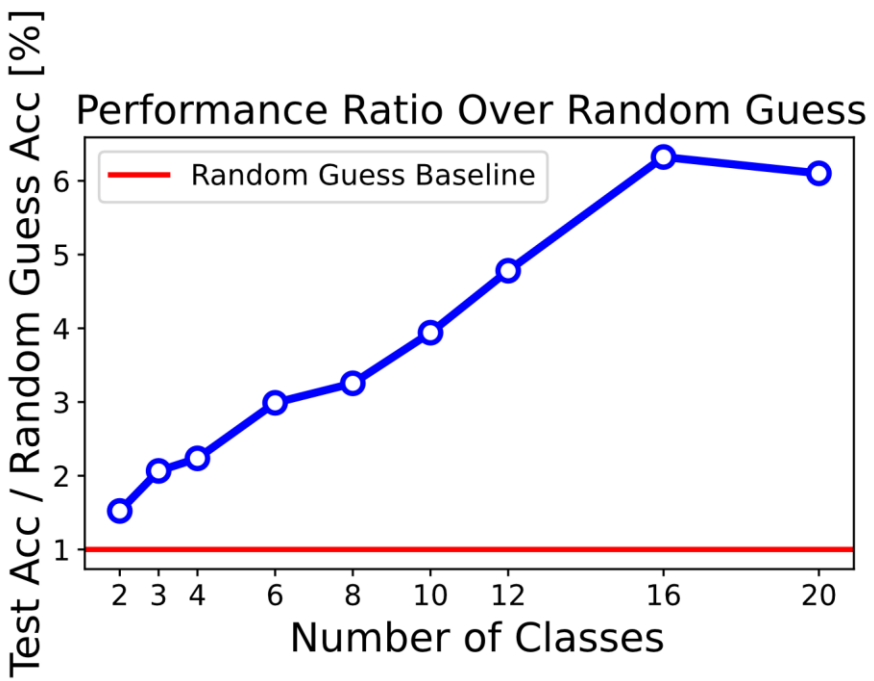
Classification results Berlin





Classification results Munich





3. Performance evaluation

3.2 Regression

Regression

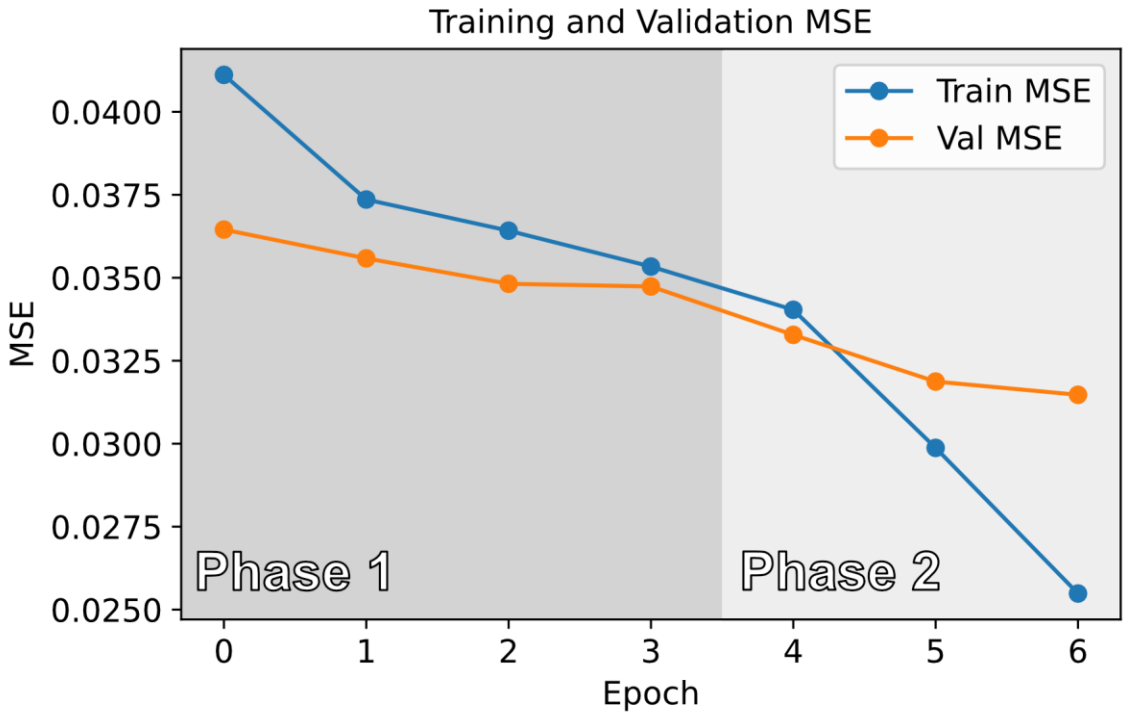
- Output: Sigmoid probability [0,1]

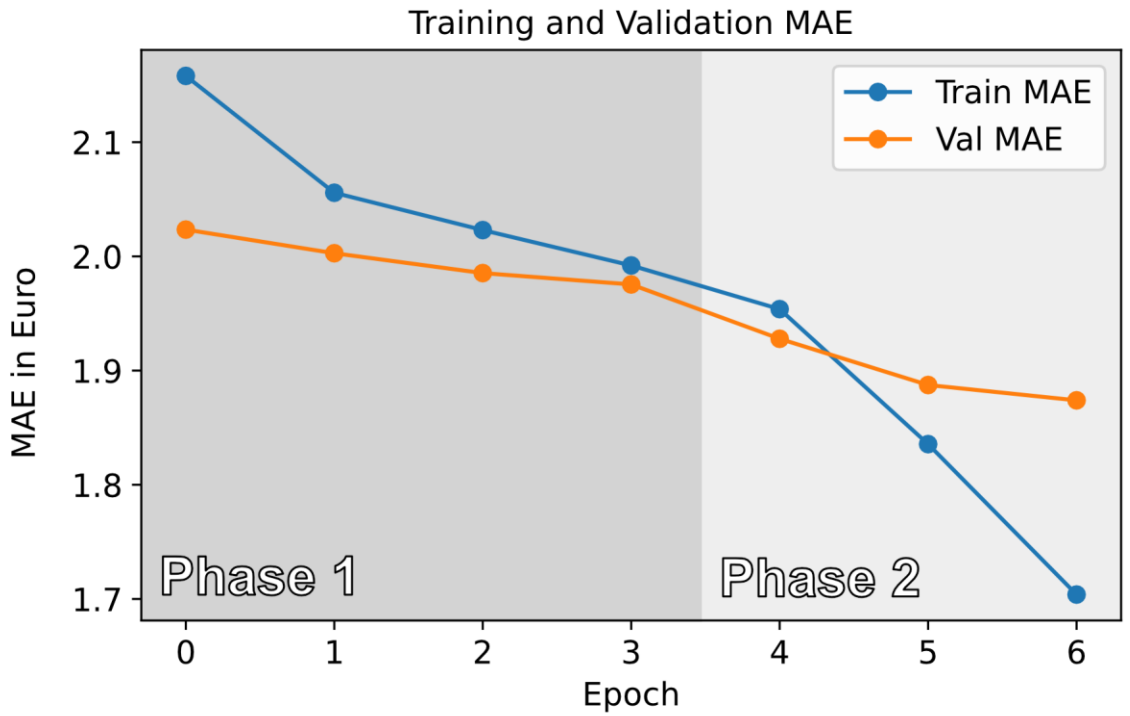
- BackBone Feature Extraction:
  - EfficientNet-B3
  - Input: RGB street view images
  - Transferlearning

- Regression Head:
  - Five fully-connected layers
  - Progressive dimensionality reduction
  - Batch Normalization
  - LeakyReLu activation: Prevents dead neurons max(0.01*x,x)
  - Progressive Dropout

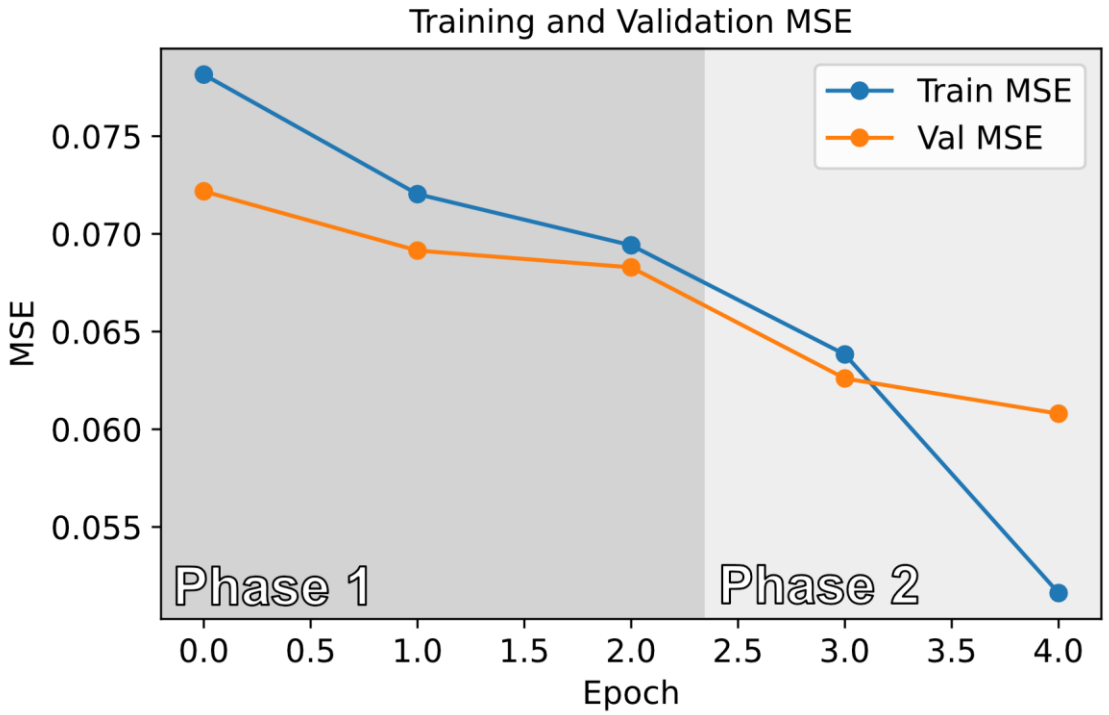
```python
class EfficientNetRegressor(nn.Module):
    def __init__(self, model_name='efficientnet_b3'):
        super(EfficientNetRegressor, self).__init__()

        self.backbone = timm.create_model(model_name,
                                          pretrained=True,
                                          num_classes=0,
                                          global_pool='avg')

        # Single output neuron for regression
        self.regressor = nn.Sequential(
            nn.Dropout(0.3),
            nn.Linear(self.backbone.num_features, 1024),
            nn.BatchNorm1d(1024),
            nn.LeakyReLU(0.1),

            nn.Dropout(0.4),
            nn.Linear(1024, 512),
            nn.BatchNorm1d(512),
            nn.LeakyReLU(0.1),

            nn.Dropout(0.3),
            nn.Linear(512, 256),
            nn.BatchNorm1d(256),
            nn.LeakyReLU(0.1),

            nn.Dropout(0.2),
            nn.Linear(256, 128),
            nn.BatchNorm1d(128),
            nn.LeakyReLU(0.1),

            nn.Dropout(0.1),
            nn.Linear(128, 1),
            nn.Sigmoid()
        )

    def forward(self, x):
        features = self.backbone(x)
        return self.regressor(features)
```

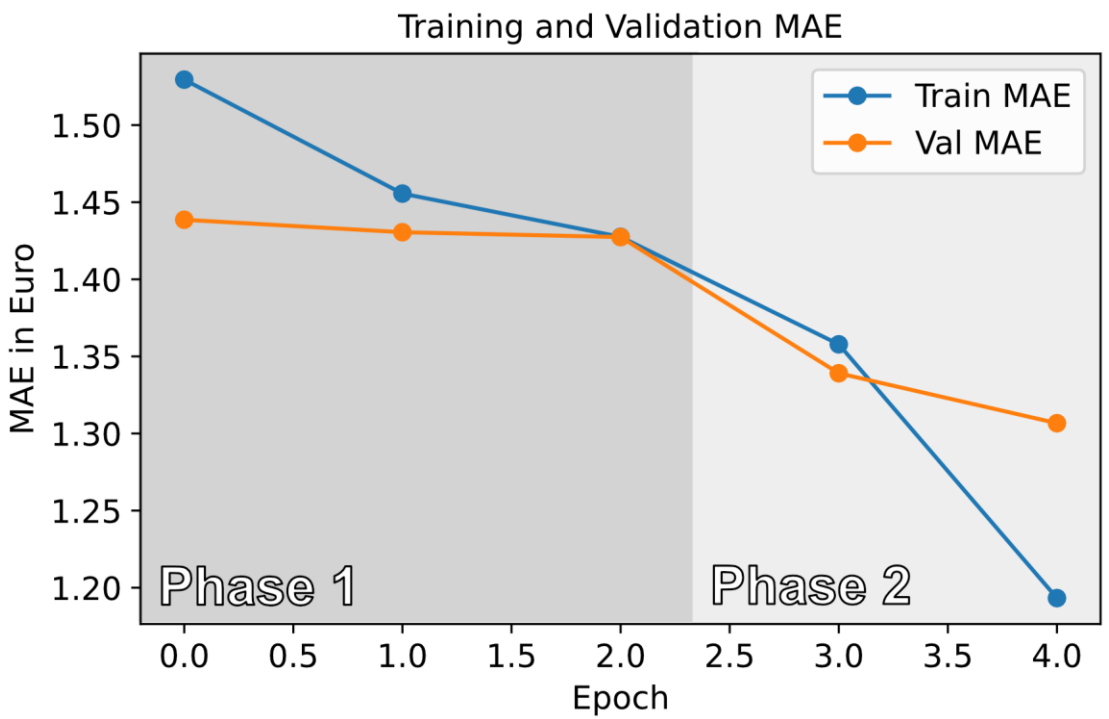
Regression results Berlin





Regression results Munich





3. Performance evaluation

3.3 Image augmentation

Augmentation – improving the accuracy?

- **Idea:** change brightness, contrast, saturation and hue of some of the training images

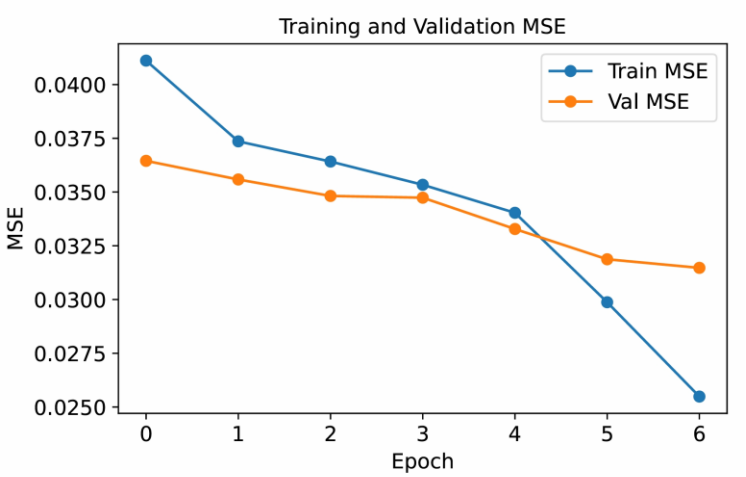
- **Goal** [8]: - Enhance the size and quality of training data sets
  - Simulate different lighting and weather conditions
  - Prevent overfitting and improve model performance

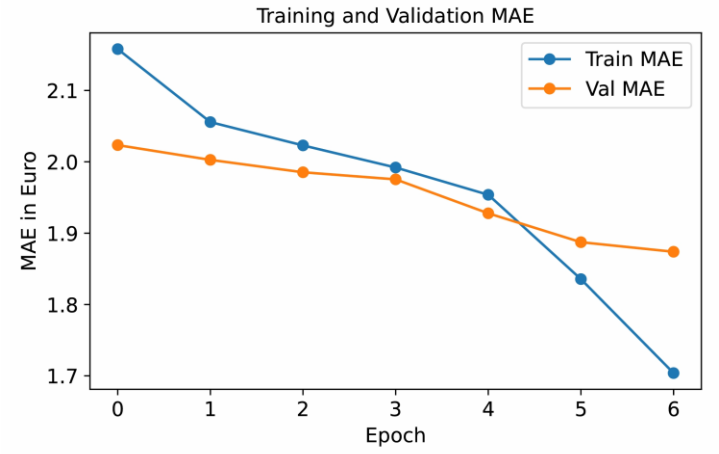
- **Realisation** [9]:
  torchvision.transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.1, hue=0.05)

[9] https://docs.pytorch.org/vision/main/generated/torchvision.transforms.ColorJitter.html

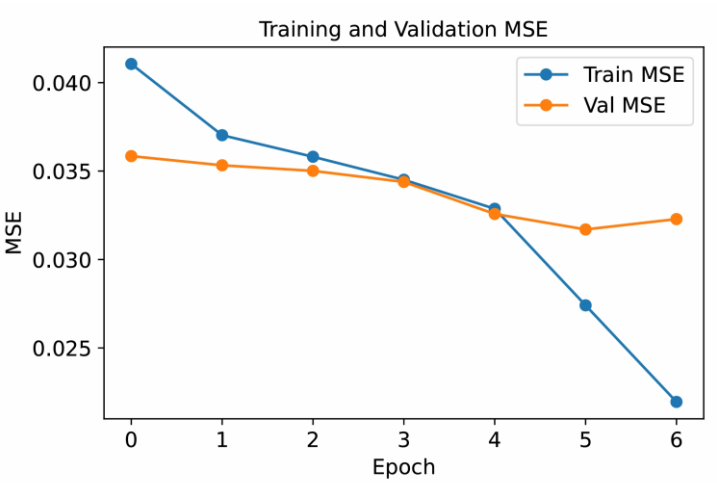
Augmentation – results Berlin

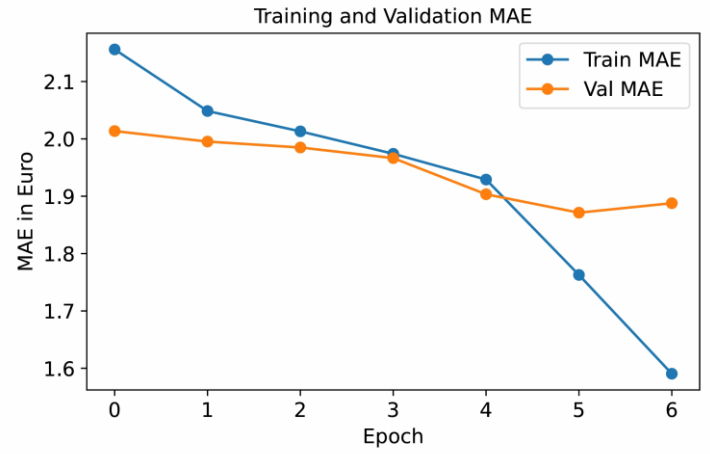
With augmentation





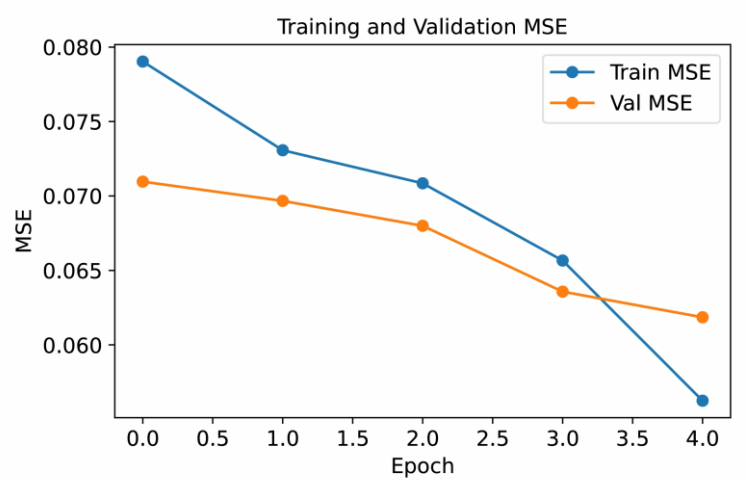
Without augmentation

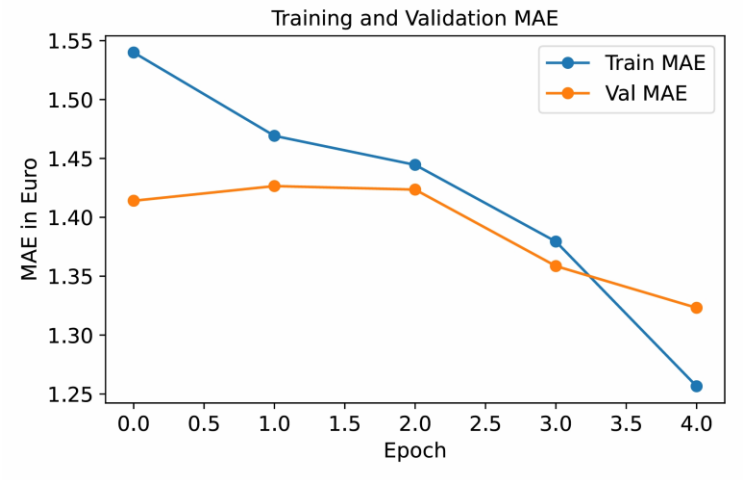




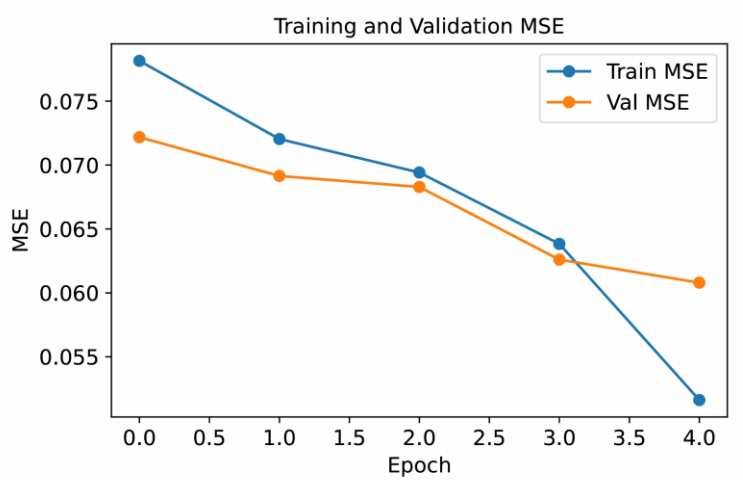
Augmentation – results Munich

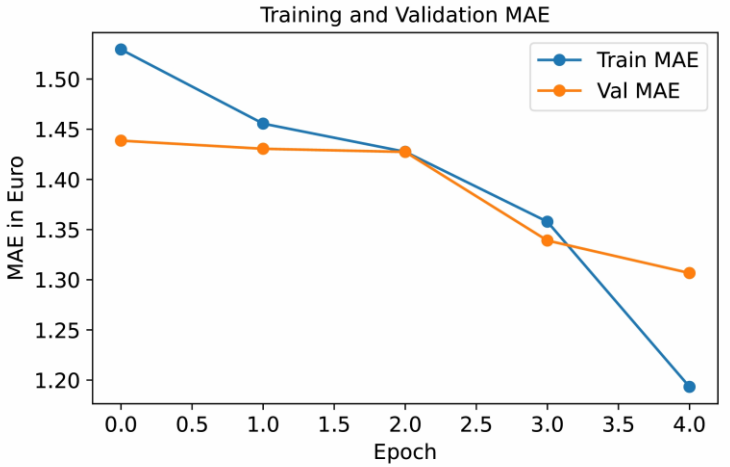
With augmentation





Without augmentation



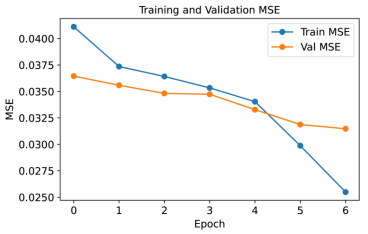


Augmentation – summary

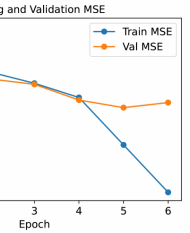
- Augmentation increases the variety of images

- Berlin data set already has a broad variety of different types of neighbourhoods and lighting/weather conditions
  → only small effect

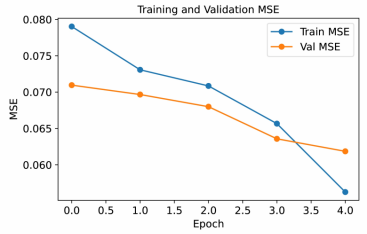
With augmentation

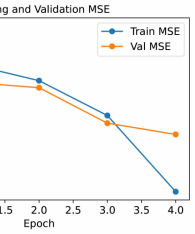


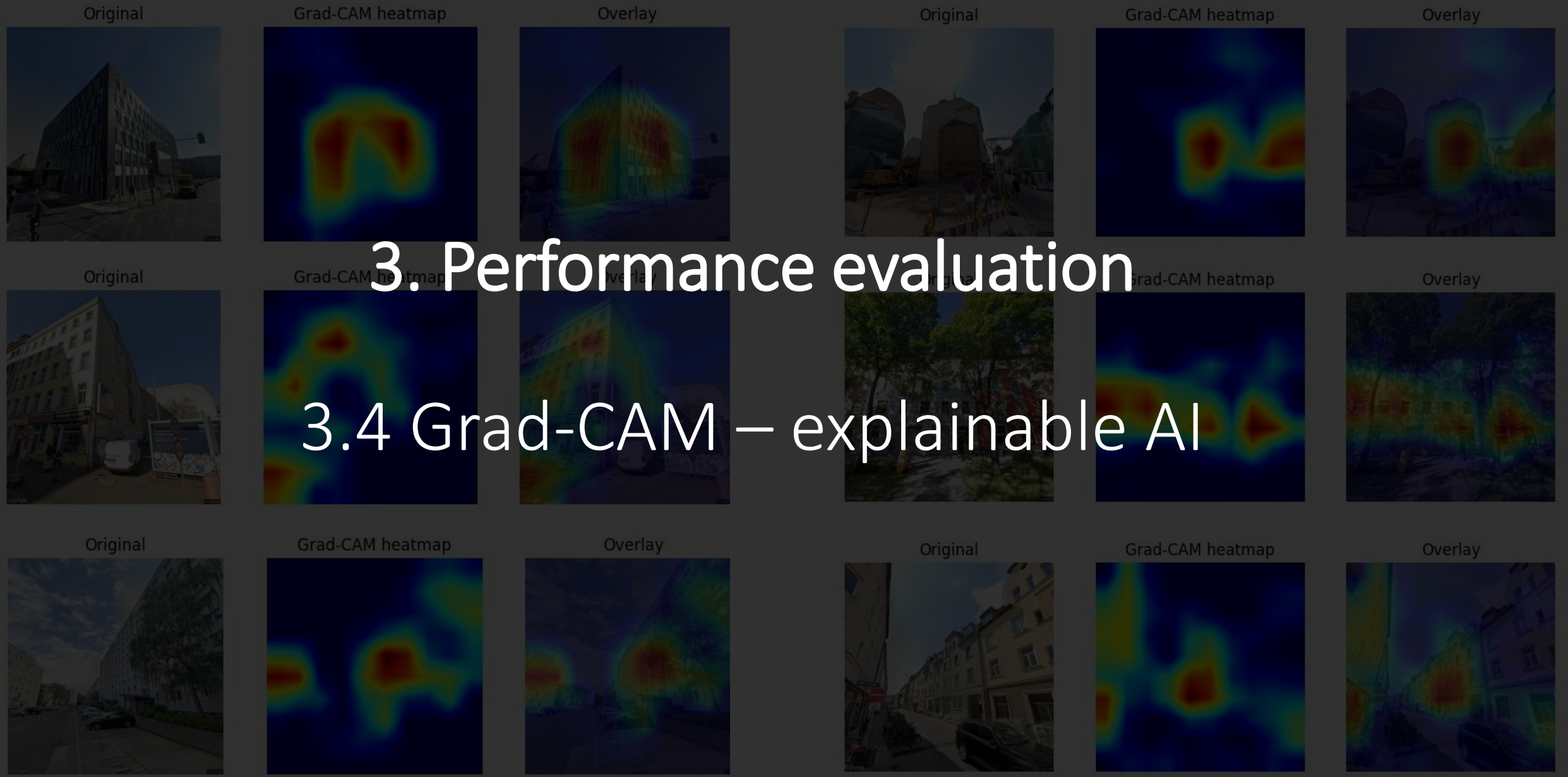
Without augmentation



- Munich model has a worse performance than the Berlin model and some problem with images without buildings (more on later slides)
  → augmentation cannot save the limitations of this model



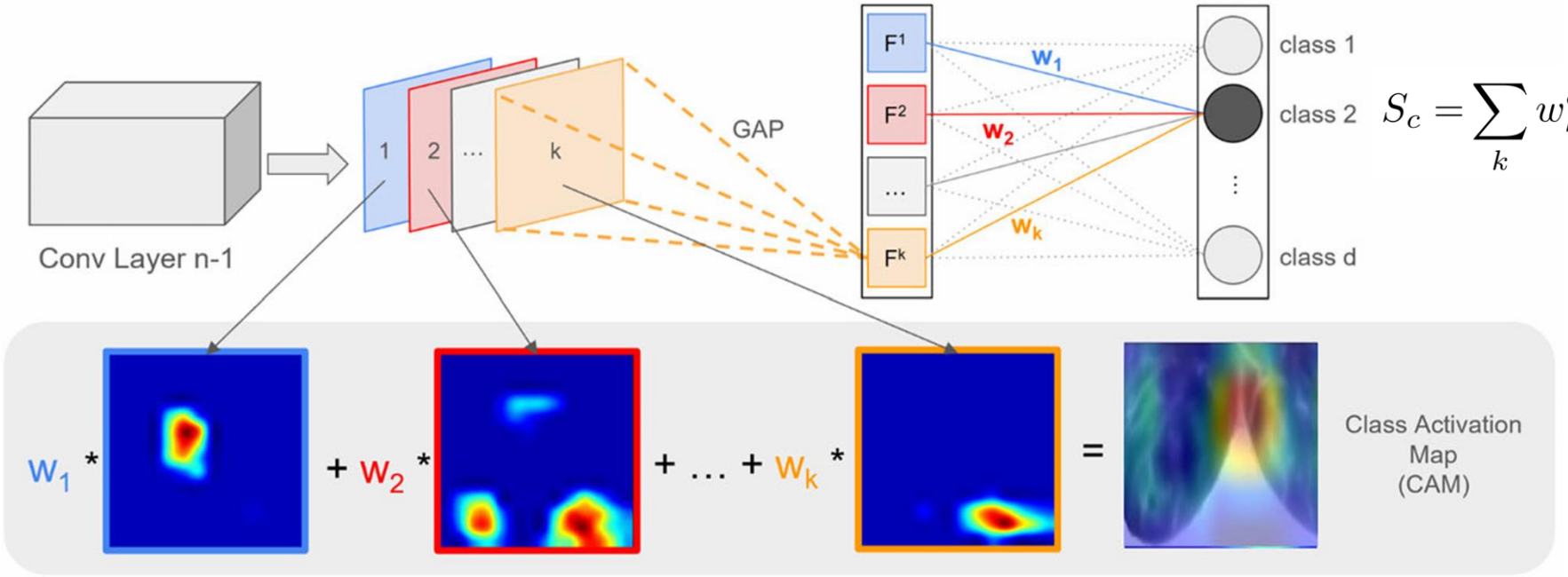




3. Performance evaluation

3.4 Grad-CAM – explainable AI

Class activation mapping (CAM)



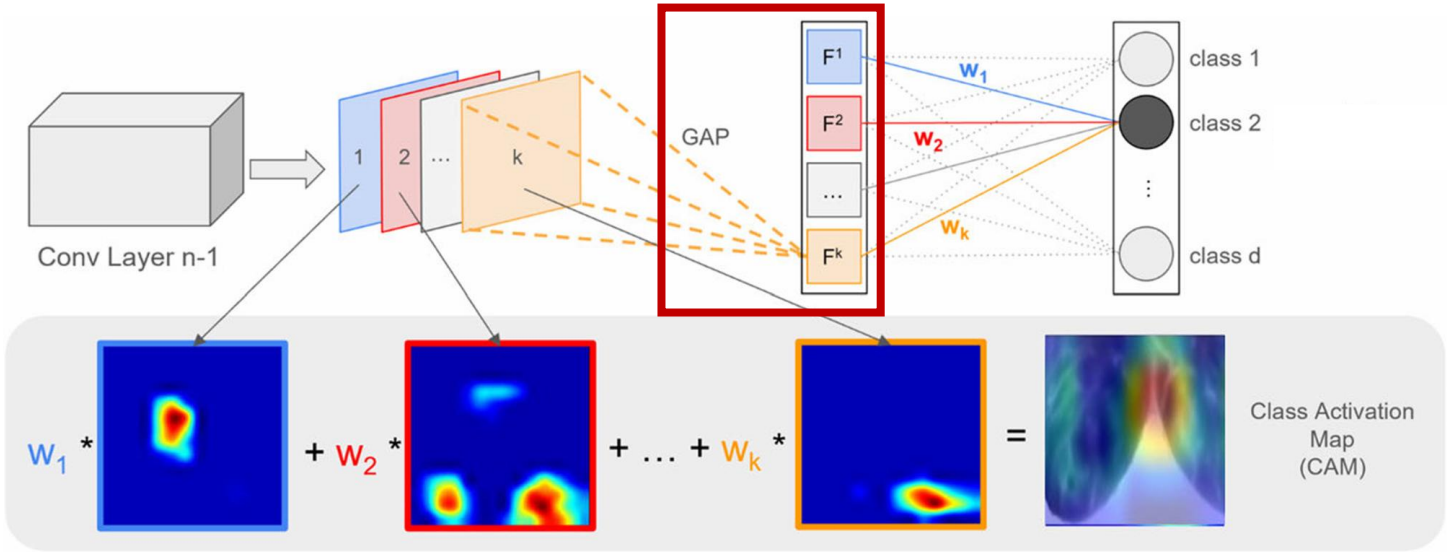
$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c f_k(x,y)$$

$$M_c(x,y) = \sum_k w_k^c f_k(x,y)$$

[10,11] Raghavan et al., *Attention guided grad-CAM: an improved explainable artificial intelligence model for infrared breast cancer detection,* Multimedia Tools and Applications (2024)

w_k^C: weight of GAP activation map for feature k and for class C
$f_k(x,y)$: pixel (x,y) of activation map for feature k
S_C: class score of class C
$M_C(x,y)$: pixel (x,y) of class activation map for class C

Class activation mapping (CAM)

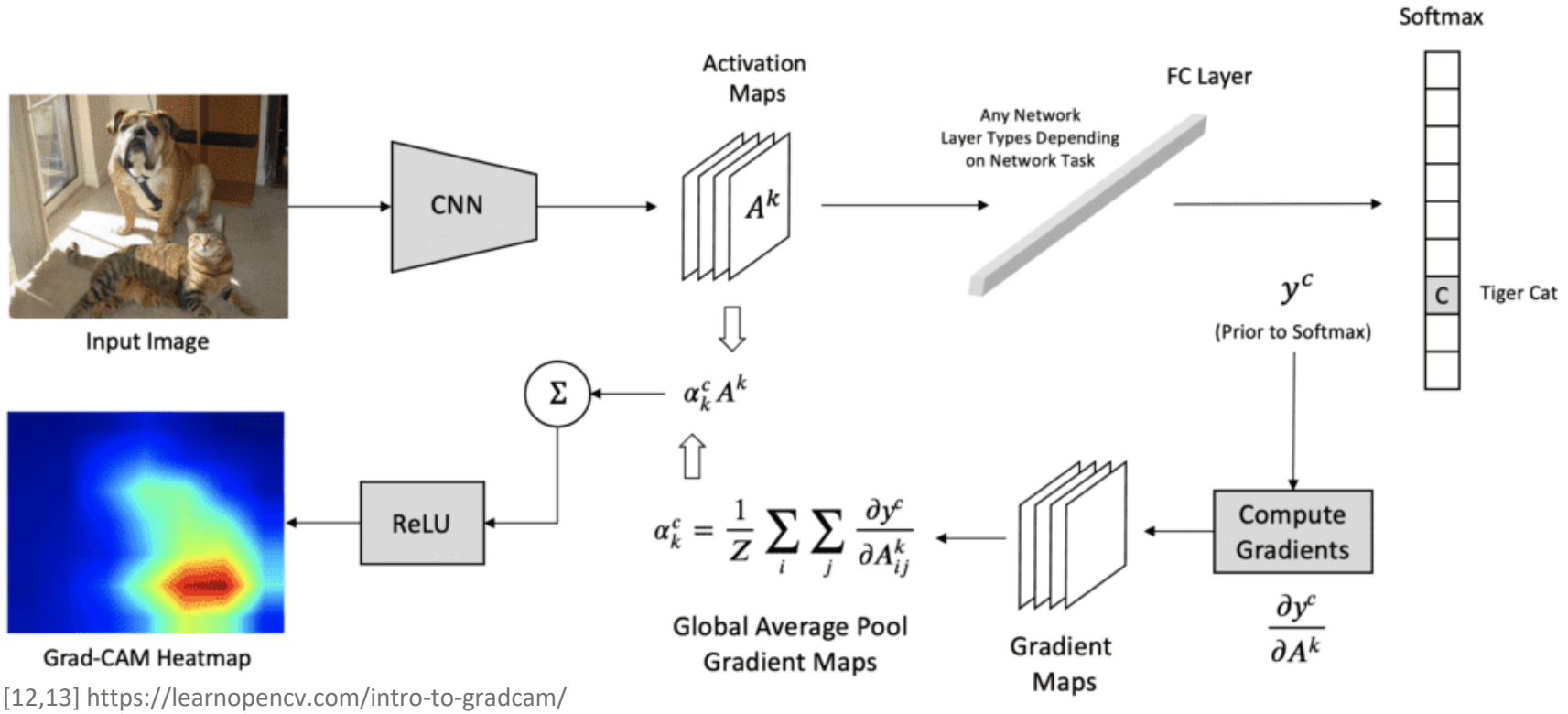


Problem: We have a regression based model without global average pooling (GAP)

Solution: gradient-based class activation mapping (Grad-CAM)

- Generalization of CAM
- No global average pooling (GAP) after last convolutional layer needed
- Compute weights of feature map A^k via **backpropagation** from the input image instead of using the existing weights after GAP

Gradient-based class activation mapping (Grad-CAM)



$$L^c_{\text{Grad-CAM}} = ReLU\underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$

[12,13] https://learnopencv.com/intro-to-gradcam/

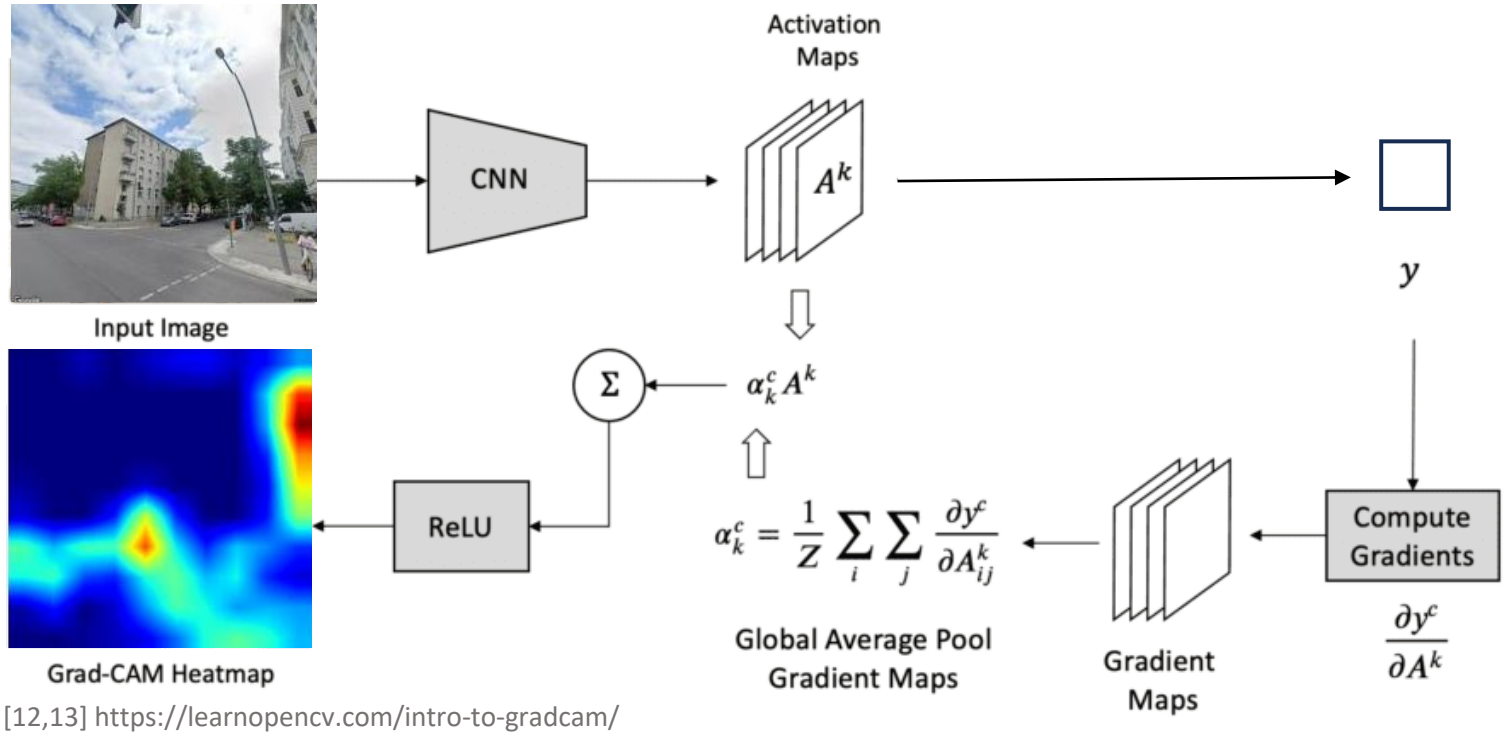
$A^k(i,j)$: pixel (i,j) of activation map for feature k

y^C: pre-softmax output for class C

α_k^C: average weight of activation map for feature k and for class C

$L_C(i,j)$: pixel (i,j) of gradient-based class activation map for class C

Grad-CAM for regression



[12,13] https://learnopencv.com/intro-to-gradcam/

$$L^c_{\text{Grad-CAM}} = ReLU\underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$

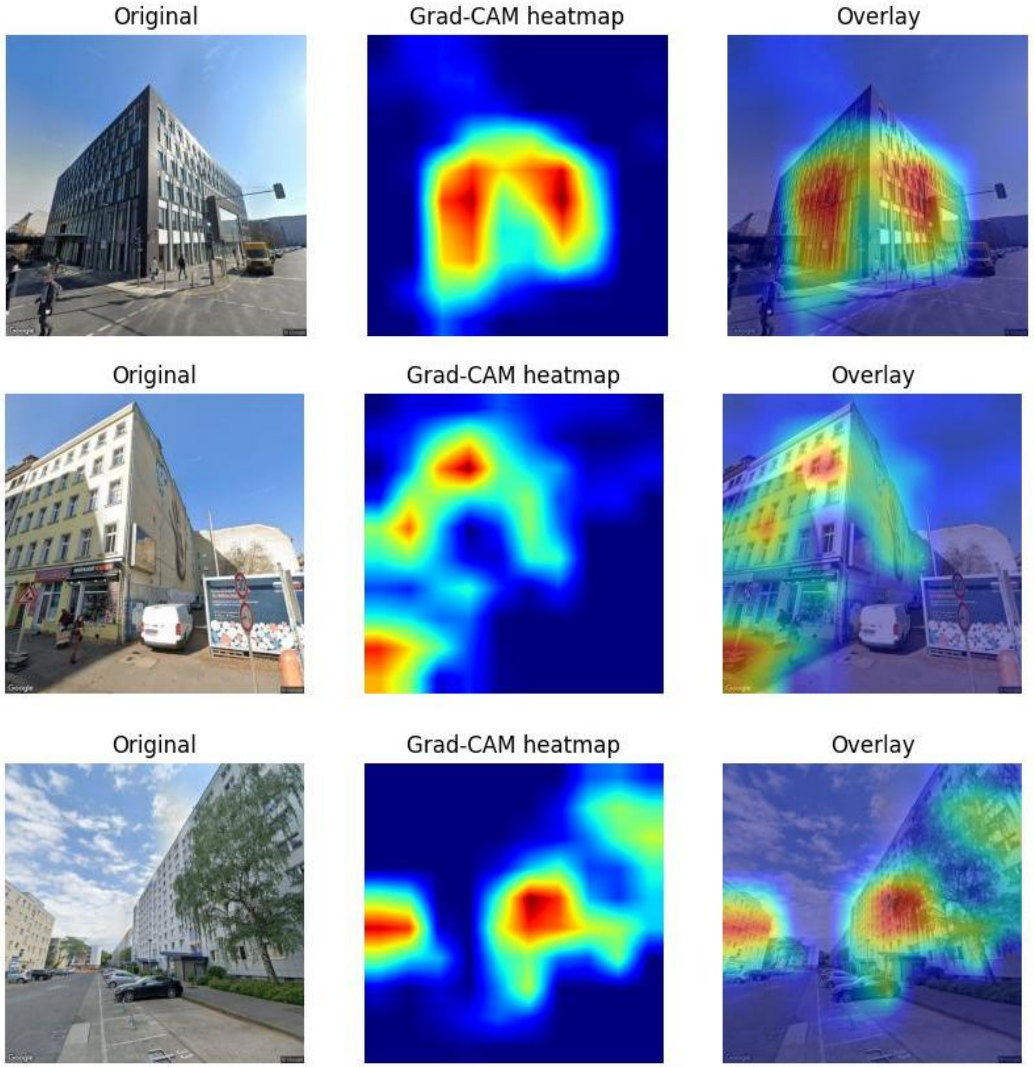
$A^k(i,j)$: pixel (i,j) of activation map for feature k

y^C: pre-softmax output for class C

α_k^C: average weight of activation map for feature k and for class C

$L_C(i,j)$: pixel (i,j) of gradient-based class activation map for class C

Grad-CAM results - Berlin

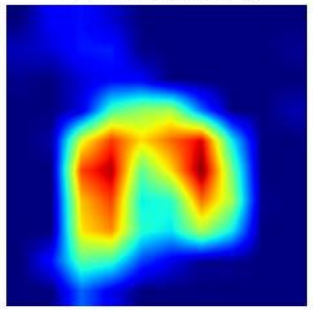


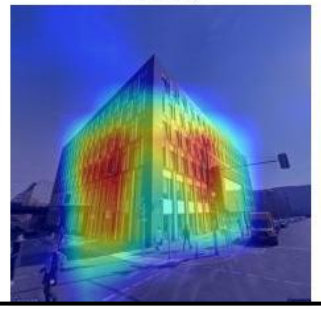
Images from district Mitte,
average rent: 22.39 € per m^2

Predicted rent:	**Error:**
19.65 €	2.74 €
18.61 €	3.78 €
19.33 €	3.06 €

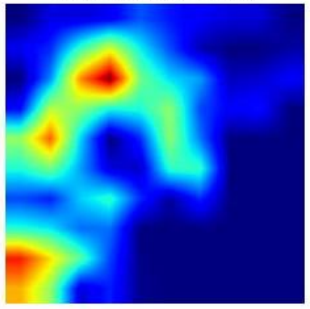
Grad-CAM results - Berlin

Images from district Mitte,
average rent: 22.39 € per m^2





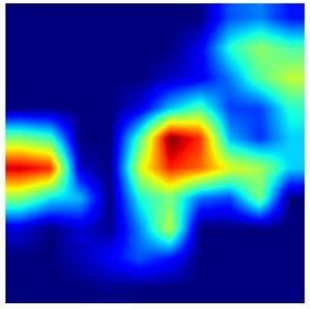
Predicted rent:	Error:
19.65 €	2.74 €

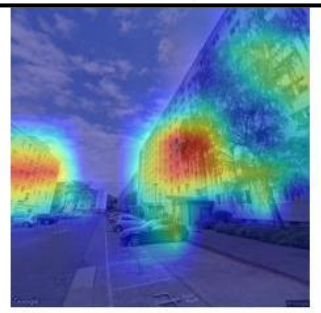


18.61 €	3.78 €

Which features are important for the Berlin model?

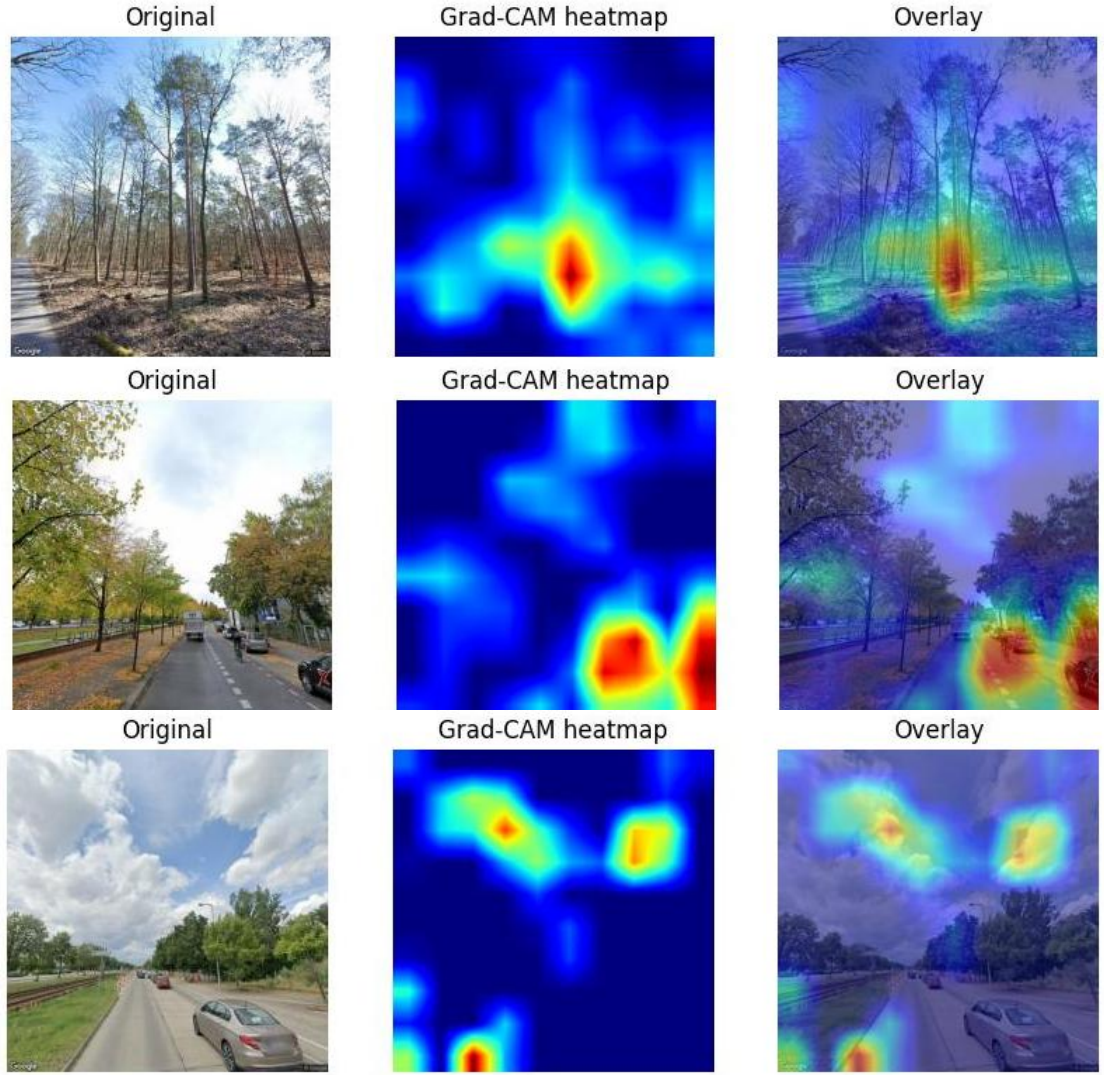
Mainly facades and windows





19.33 €	3.06 €

Grad-CAM limitations - Berlin



Predicted rent:	Actual rent:	Error:
16.22 €	14.49 €	1.73 €
18.69 €	22.39 €	3.70 €
13.01 €	13.37 €	0.36 €

Grad-CAM limitations - Berlin

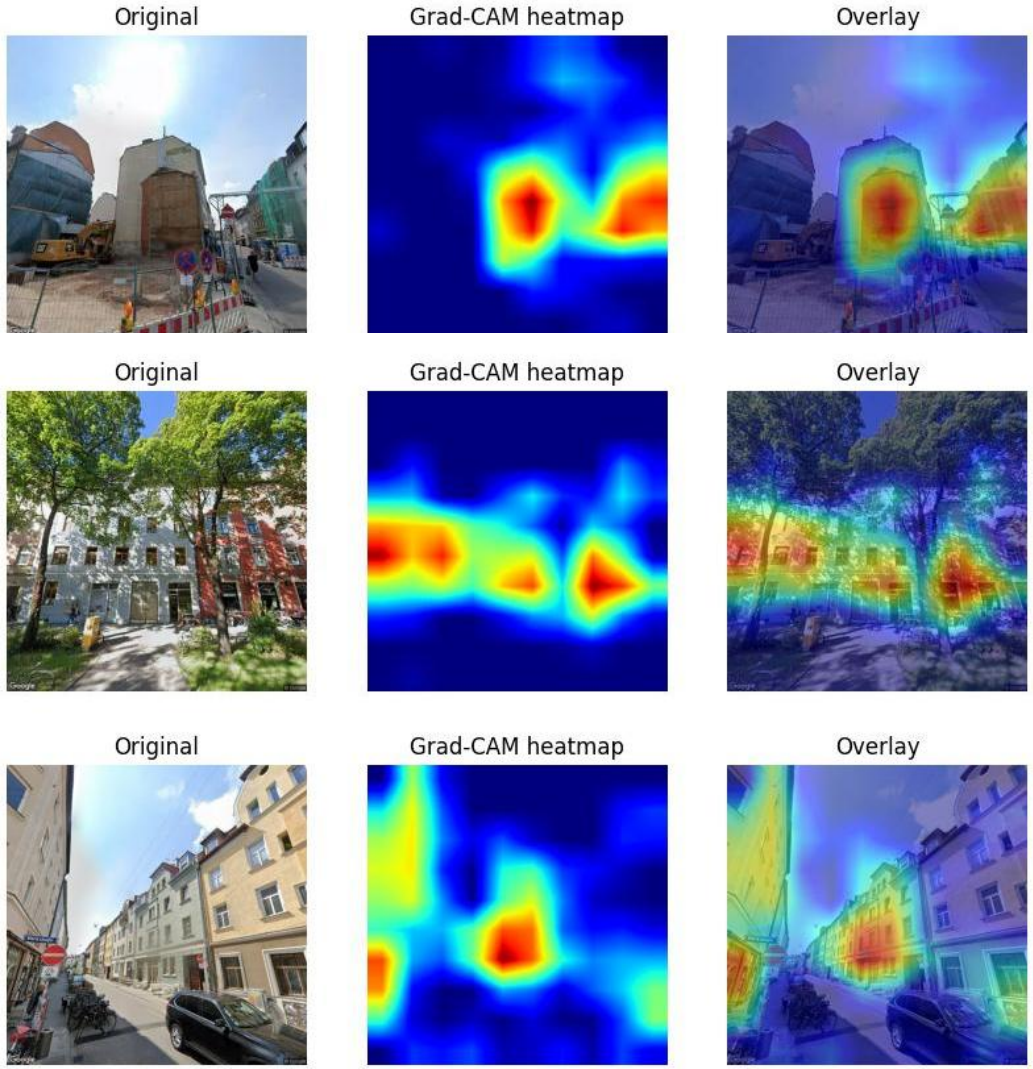
If typical houses are missing, the model decides based on …

… trees, bicycles and clouds

Predicted rent:	Actual rent:	Error:
16.22 €	14.49 €	1.73 €
18.69 €	22.39 €	3.70 €
13.01 €	13.37 €	0.36 €

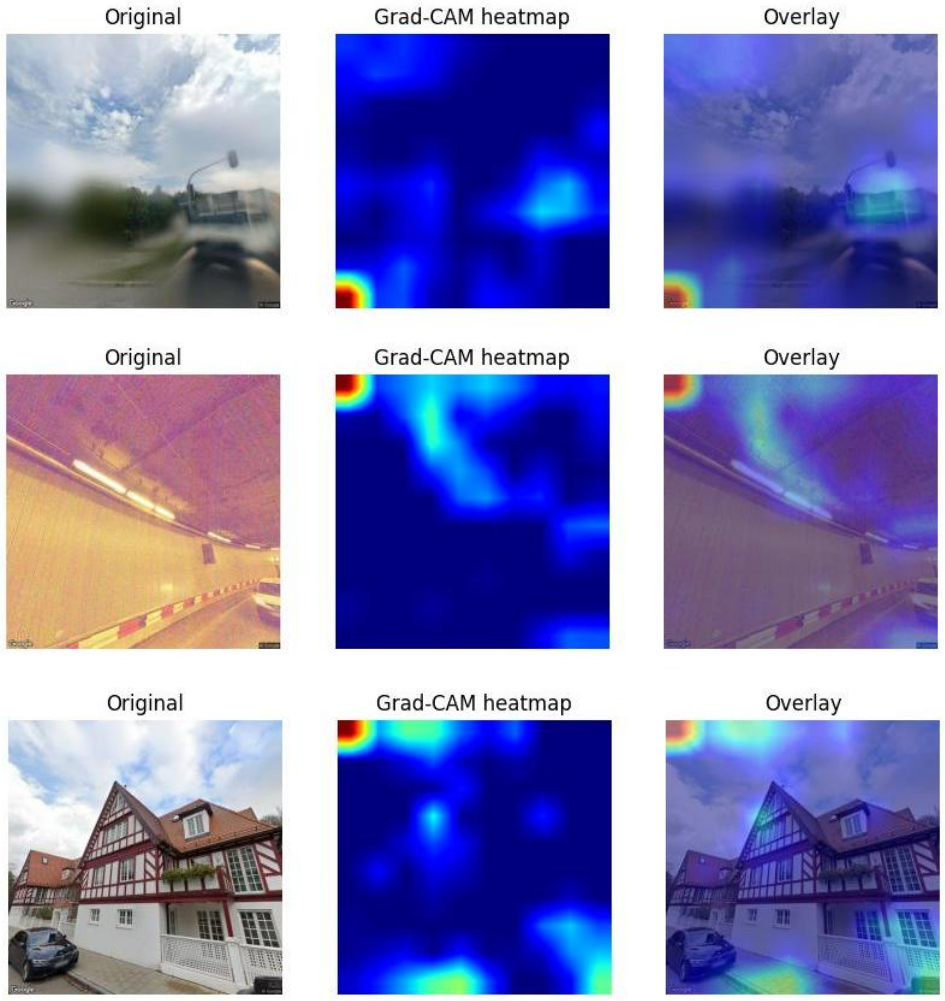
Grad-CAM results - Munich

Images from district Schwanthalerhöhe, average rent: 25.35 € per m^2



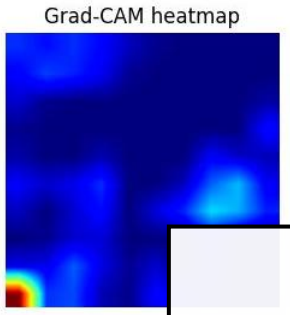
Predicted rent:	Error:
25.75 €	0.40 €
25.05 €	0.30 €
25.99 €	0.64 €

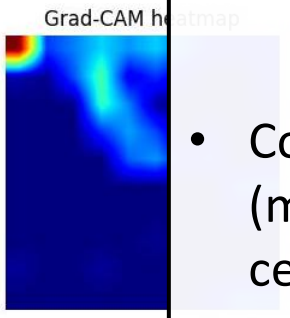
Grad-CAM limitations - Munich

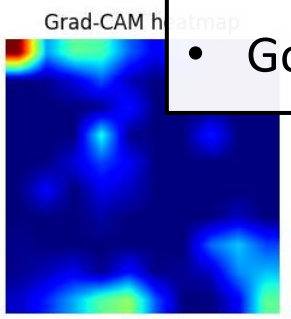


Predicted rent:	Actual rent:	Error:
27.03 €	26.00 €	1.03 €
26.14 €	26.00 €	0.14 €
26.49 €	27.62 €	1.13 €

Grad-CAM limitations - Munich





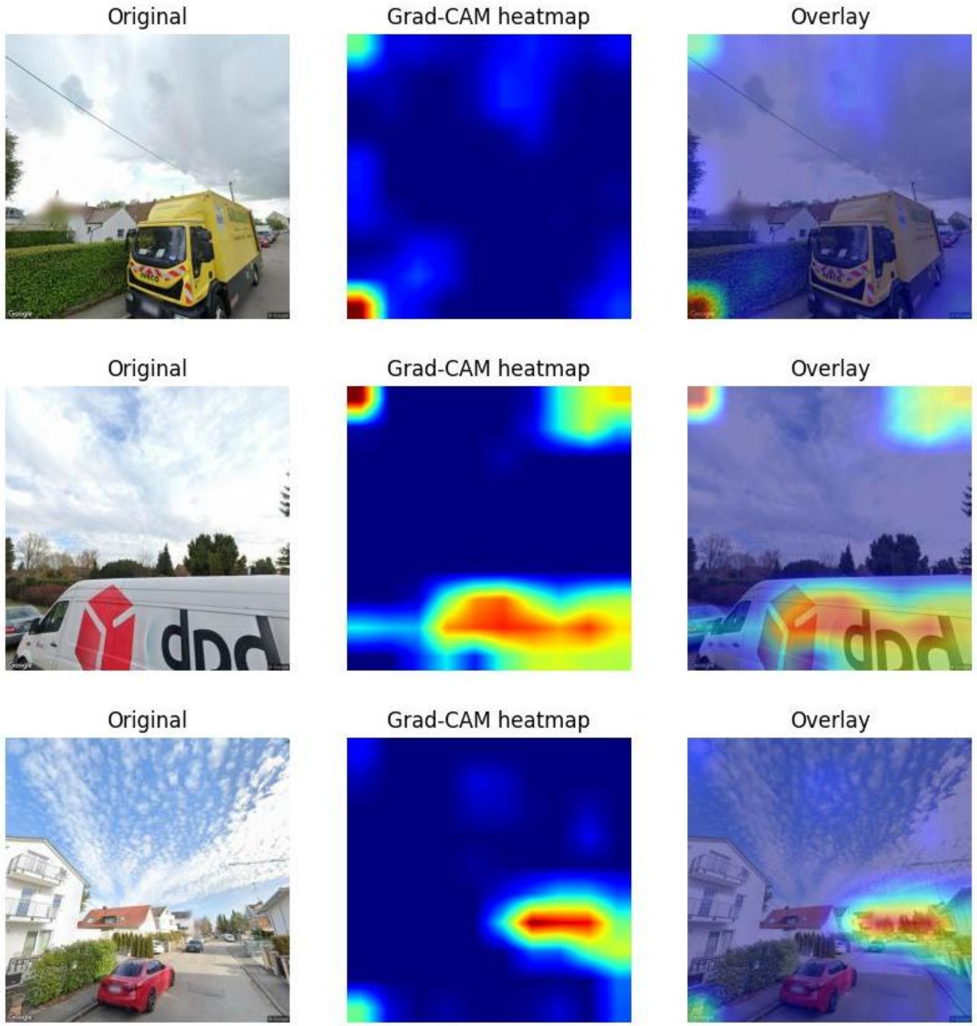


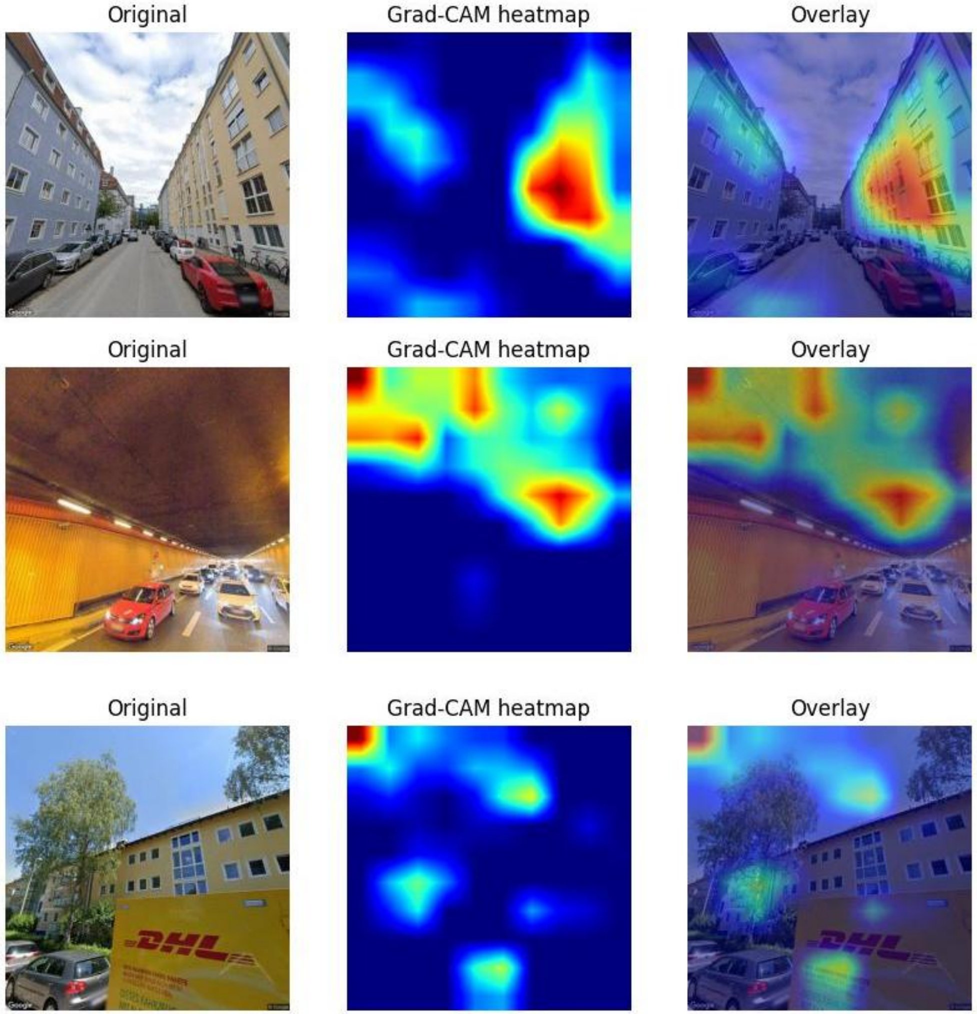
If typical houses are not clearly visible, the Munich model predicts based on

- Color of top left corner (maybe Street View car covers a district at a certain day with a certain weather?)
- Google logo in bottom left corner

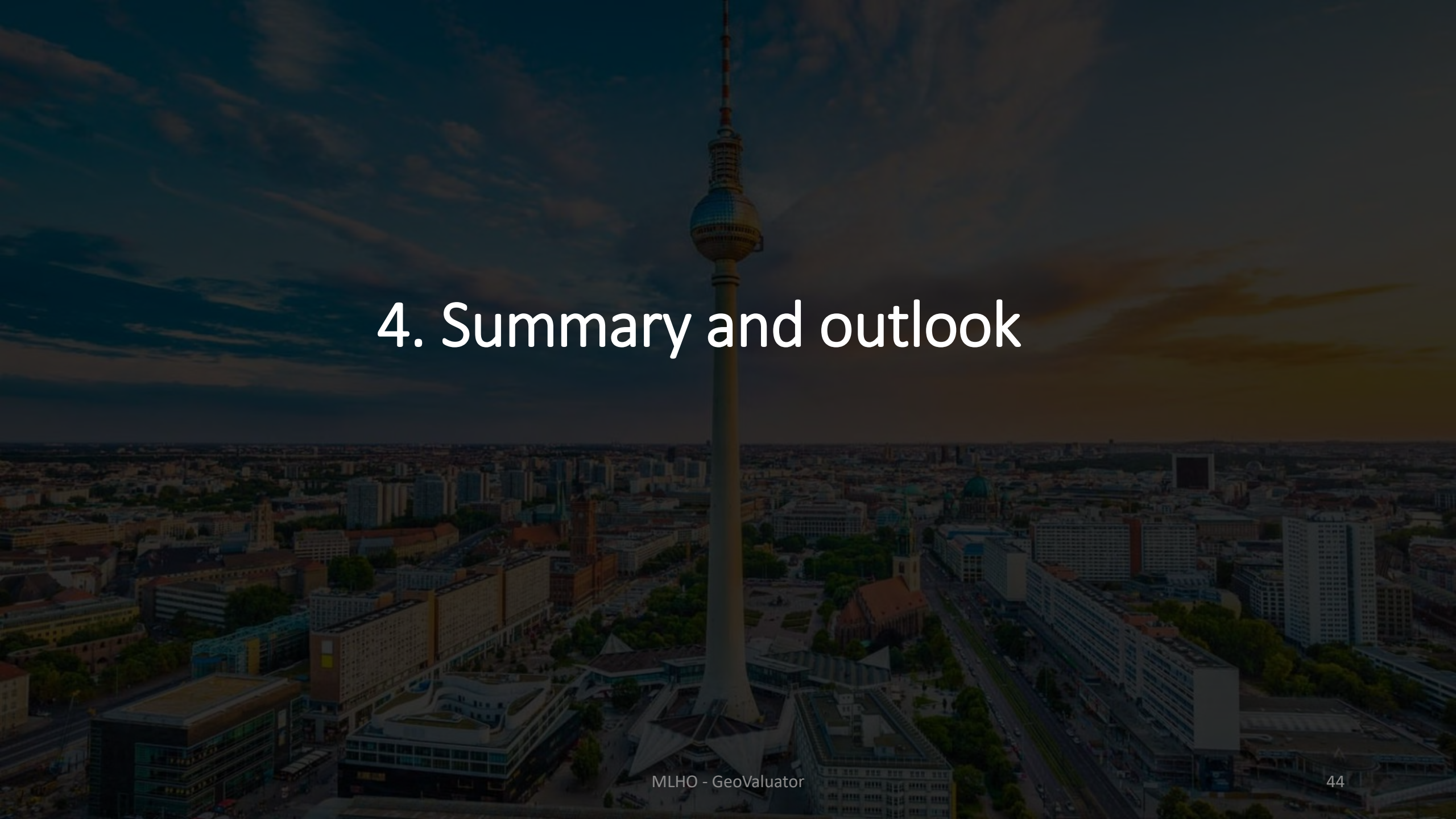
Predicted rent:	Actual rent:	Error:
27.03 €	26.00 €	1.03 €
36.14 €	26.00 €	0.14 €
26.49 €	27.62	1.13 €

Grad-CAM results – What about cars?





Grad-CAM results – What about cars?



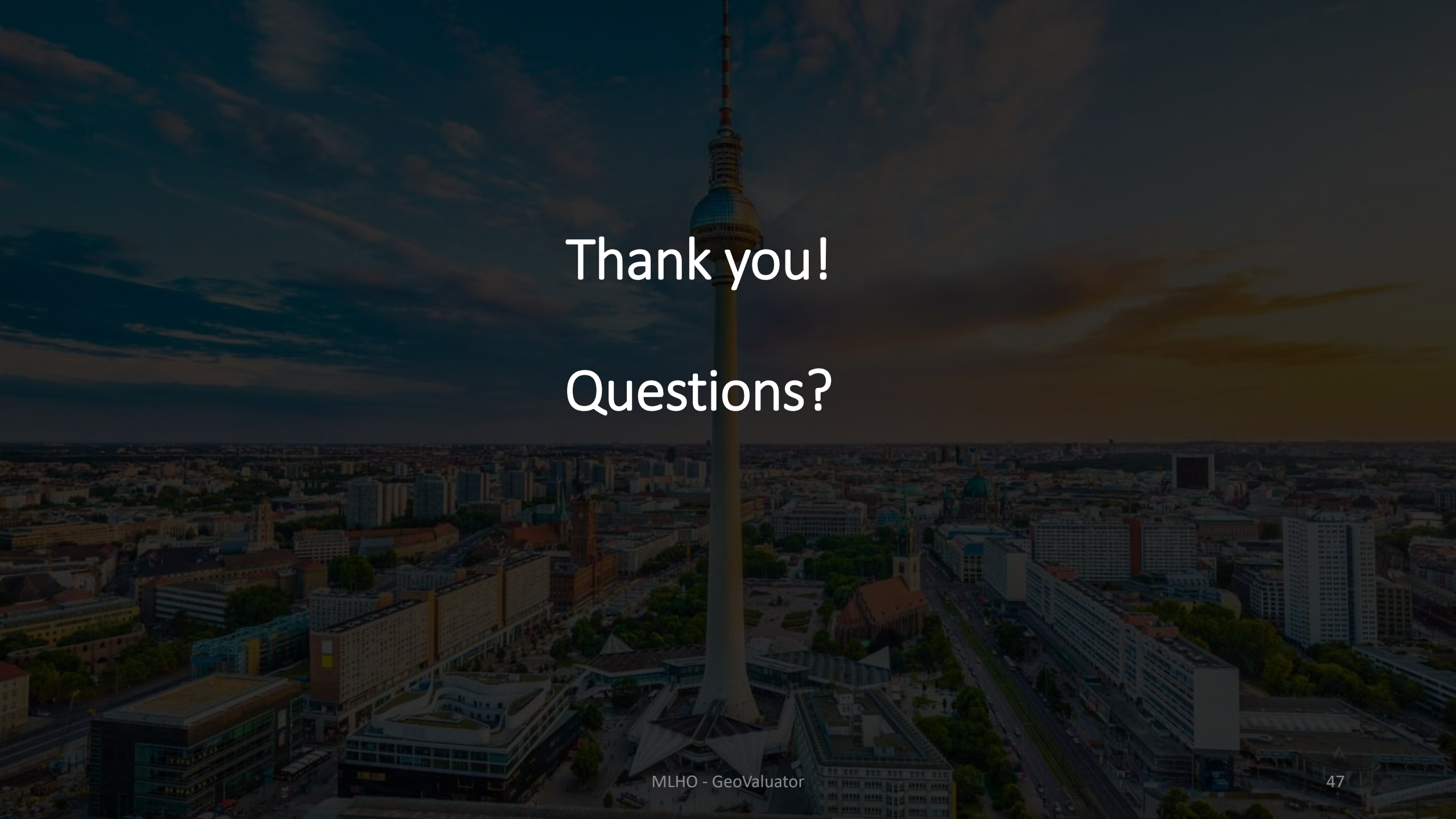
4. Summary and outlook

Summary

- Regression is superior to classification for our task
- Berlin model performs quite well
- Munich model is worse, likely due to less differences between the districts
- Effect of augmentation is negligable
- Both models mainly predict based on the buildings and facades, but the Munich model also uses lighting and weather conditions and the Google logo

Outlook – What next?

- More data! (millions to billions of images instead of 10000s)
- Images of many different cities in one model
- Other models instead of EfficientNet-B3
  (e.g. StreetClip OpenAI, NLP model)
- Finetuned dataset, exclude bad images
- Distinguish between city and rural areas


Thank you!
Questions?

References

[1] opendata.muenchen.de

[2] daten.berlin.de

[3] wiki.openstreetmap.org

[4] cloud.google.com

[5] Cabello, *An introduction to log-linearizations,* (2004)

[6] Sleimanipour et al., *Cultivar identification of pistachio nuts in bulk mode through EfficientNet deep learning mode*, (2022)

[7] Tan et al., *EfficientNet: Rethinking Model Scaling for Convolutional Neural Network,* (2019)

[8] Shorten and Khoshgoftaar, *A survey on Image Data Augmentation for Deep Learning,* Journal of Big Data (2019)

[9] https://docs.pytorch.org/vision/main/generated/torchvision.transforms.ColorJitter.html

[10] Zhou et al., *Learning Deep Features for Discriminative Localization,* IEEE Conference on Computer Vision and Pattern Recognition (2016)

[11] Raghavan et al., *Attention guided grad-CAM: an improved explainable artificial intelligence model for infrared breast cancer detection,* Multimedia Tools and Applications (2024)

[12] Selvaraju et al., *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization,* IEEE International Conference on Computer Vision (2017)

[13] https://learnopencv.com/intro-to-gradcam/

GitHub: https://github.com/Dacasil/GeoValuator.git

Compare Berlin and Munich Models

Spearman Correlation:

- Non-parametric: Measures monotonic relationships
- Rank-focused: Compares ordinal positions rather than raw values

Berlin Model with Munich Data
Spearman Correlation: 0.432

Munich Model with Berlin Data:
Spearman Correlation: 0.275

District prize normalization

Data Normalization [5]: Normalize price to [0,1]

- $\log(price)$
- Min-Max Scaling: $\frac{\log(price) - \min\_\log(price)}{max\_\log(price) - min\_\log(price)}$

Data Renormalization to Euro €

- Mean Average Norm = mean(|log(price_predict) – log(price_label|)
- Error in log space: Δlog = MAE × $(max\_\log - min\_\log)$
- Local linear approximation: Taylor Expansion around midpoint a of the log data
- Δ_€ = exp(a) x Δlog