"Are you in a safe building: Street Vision based Automated Building Classification"

Praveen Kottari(praveenkotta@iisc.ac.in)

1 ABSTRACT

The project started with analysing supervised ML models for a small, complex dataset of street-view images. After experimenting with traditional models like Logistic Regression, KNN and SVM, we transitioned to DL techniques. CNN-based models like ResNet-50 showed moderate improvement, while transformer architectures such as ViT, DeiT, and SWin Transformer offered better results. While monitoring possible overfitting issues and model performance, we implemented techniques like early stopping, weight decay and data augmentation. Our final submission scored 0.606 public and 0.61 private respectively.

Notations: CV- Computer Vision, DL- Deep Learning, ML- Machine Learning, NN- Neural Network, TL- Transfer Learning

2 INTRODUCTION

Managing earthquake risks is crucial because of the high potential for loss of life and economic damage. To prevent collapses, the stability of buildings is of great interest. Imagine walking down a street, surrounded by towering buildings, each made from different materials and designs. The key question arises: how can we efficiently assess which buildings are structurally sound during seismic activity? More importantly, how can we identify potentially unsafe buildings across vast urban landscape without inspecting buildings one by one. With technology advancing, we have access to new types of data like Streetview images. However, these images are noisy and collected images might not be evenly distributed. These challenges motivate our project, where we focus on data cleaning and model development to create a supervised ML model for accurate building classification and risk assessment.

3 DATASET

The dataset used for this project originally consists of 2,516 grayscale, texture rich image samples with 300x400 resolution. These image capture five distinct building classes, defined based on the materials and other structural characteristics shown in Table [2]. Most of the images were taken from different streets and provide a wide range of building perspectives and conditions. Each building class exhibits certain unique structural properties which replicates in the pattern of surface textures. However, many of the images also share common features across different classes, adding a layer of complexity to the classification task. The annotation highlights key characteristics for model training. However, some images are misclassified and the number of images of different classes are imbalanced. This motivates for thorough data cleaning.

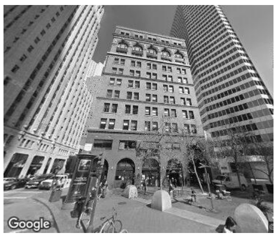

Figure 1 Class 1

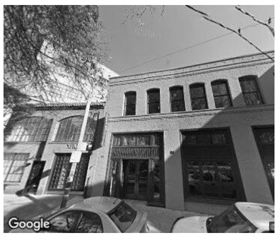

Figure 2 Class 2

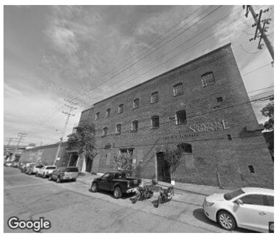

Figure 3 Class 3

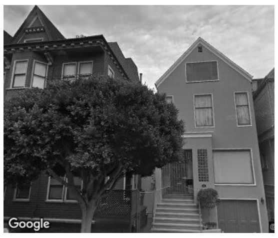

Figure 4 Class 4

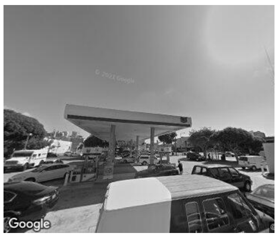

Figure 5 Class 5

Table 1 Sample Images of five building classes

Class 1: Steel Buildings, large presence of glass, significant slenderness.
Class 2: Concrete, boxy aspect, smooth texture, persistent opening.
Class 3: Masonry, bricks exposed, no soft story, decorative patterns.
Class 4: Wooden framed, wooden bow windows, slanted roof.
Class 5: Steel with panel, industrial/retail, max 2 floors, non-slanted roof.

Table 2 Class Definitions

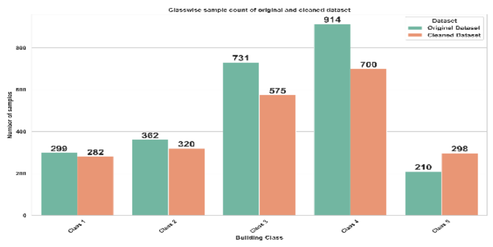

Figure 6 Class wise count for original and cleaned dataset

3.1 Exploratory Data Analysis (EDA)

For the original dataset, classical models were not able to capture the complex features and high dimensional properties, so we did EDA to extract the clean dataset by manual inspection of the samples and analysing the class distributions. During this process, we identified some images that violated key class properties, which were then replaced with samples that better matched the correct characteristics. Additionally, we identified blurred images and images without buildings as outliers and removed them. We also addressed duplicates, ensuring that multiple identical copies were eliminated. To preserve diversity, images with shared features were retained unless they significantly deviated from the expected class properties. After cleaning, we were left with 2175 image samples. Figure [6] refers the counts of samples.

3.2 Filtering Technique and Feature extraction

Before diving directly into NN-based feature extraction, we tried applying some classical CV-based algorithms and passed the features to models. We tried extracting only the edges of the frame using the Canny edge filter and applying the Gabor filter to extract the rich texture from the image. But we did not achieve significant improvement, as these features were very much overlapping between classes, and the model often got confused while predicting the actual class.

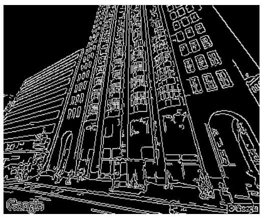
Figure 7 Canny filter image

4 BENCHMARKING

To keep uniform standard input size, all images were resized to 224x224. For better diversity, the training data was shuffled. For experiments, we initially implemented classical algorithms in Sklearn and utilized DL models in PyTorch from TL. This enabled fine-tuning of pre-trained models by modifying the final layer for 5-class prediction. After trying with multiple loss functions, we decided to use CrossEntropy loss as it gives well-calibrated probability outputs, penalizes confident wrong predictions and yields smooth gradients. For faster convergence and efficient noisy gradients handling, we used Adam optimizer. During multiple trial, we observed that the introduction of RandomCrop showed improvements in the performance. We have learned that this technique improves the network by focusing on the texture of the surface**.**

4.1 Traditional ML models

The Logistic Regression model performed poorly with an accuracy of 0.23, as it treated each pixel independently and struggled with high dimensionality (120,000 features), leading to overfitting. KNN model improved the accuracy slightly to 0.34, but still suffered from the curse of dimensionality and sensitivity to noise. A SVM with a linear kernel performed better, achieving an accuracy of 0.41, outperforming the RBF kernel due to the dataset's high dimensionality and limited samples. Here, we also used Gabor filter to reduce dimension of data samples, but even after these attempts, traditional models' performance remained limited. This motivated us to explore DL models, first focusing on the widely used ResNet50 architecture. While it improved the F1-score to 0.51, performance was still below expectations. After trying multiple CNN variants with minimal gains, we shifted to transformer-based architectures to capture image features more effectively for better performance.

4.2 Transformer based Architecture

We chose transformer architectures due to their ability to effectively capture both local and global features, which is essential given the small and complex nature of the dataset. The building images often share structural similarity across classes, making it challenging for traditional models to differentiate. Transformer models are characterized by learning complex patterns, textures and features through self-attention mechanisms. Thus, they can capture relationships between all input tokens regardless of their position, enabling them to efficiently model long-range dependencies.

- **ViT (Vision Transformer)** [1]: Treats images as a sequence of patches, like tokens in NLP and applies transformer layers directly. This allows to capture long-range dependencies and global context across the image.
- **DeiT (Data-efficient Image Transformer)** [2]: Uses ViT as backbone that improves data efficiency by using a knowledge distillation approach, allowing it to perform well even with smaller datasets.
- **SWin Transformer** [3]: Introduces a hierarchical design with shifted windows, allowing it to capture both local and global features efficiently. Divides image into smaller regions and progressively builds up the feature map.

4.3 Model Training and Regularization

During training, class imbalance is one of the main concerns to address. We incorporated class weights into the loss function such that all classes get an equal importance while updating weights. SWin Transformer with hyperparameter tuning performed slightly better than the others. However, after a certain number of epochs, all models exhibited overfitting, prompting us to explore additional techniques to mitigate these issues. Below, we listed some of the logic we applied to learn the dataset better.

- **Early Stopping:** We terminate training when the validation loss stopped improving for 5 epochs.
- **Limited Epochs:** The models were trained for a smaller number of epochs (based on the learning rate).
- **Weight Decay:** We used weight decay to penalize large weights, keeping penalty at 1e-5.
- **Batch Size Tweaking:** We adjusted the batch size during training to optimize the learning process.
- **Data Augmentation (224x224):** We introduced random cropping to create variability in the training data.

5 RESULTS

Model	Precision	Recall	F-1 Score	Test Acc.	Remark
Classical models	-	-	-	0.297(SVM)	Sensitivity to high-dimensional data, difficult to find a clear hyperplane for separation.
Resnet50 [5]	0.51	0.51	0.50	0.460	Lacks data to capture global patterns relying heavily on local feature extraction.
EfficentNet [4]	0.52	0.51	0.51	0.481	Sensitive to insufficient data, scaling approach may not capture intricate global features.
ViT [1]	0.57	0.58	0.57	0.539	Not able to fully understand global relationships.
DeiT [2]	0.63	0.64	0.63	0.531	Not effectively capture subtle differences in features, leading to suboptimal performance.
SWin-base [3]	0.67	0.66	0.66	0.606	Efficiently captured both local and global feature still require more data /fine-tuning for optimal results.

Table 3 Comparison of model performance

6 CONCLUSION

The project aimed to develop a more generalizable model, and despite experimenting with various techniques including classical ML, CNNs, and transformer-based architectures, we were able to capture many class relationships. However, the limited dataset posed challenges in capturing intricate features, leading to stagnation in validation loss improvement. While SWin Transformer outperformed traditional models, overfitting and feature extraction limitations persisted, ultimately constraining the models' ability to generalize effectively due to the dataset's size and complexity.

References

[1] Attention Is All You Need http://arxiv.org/abs/1706.03762
[2] Training data-efficient image transformers {\&} distillation through attention https://arxiv.org/abs/2012.12877
[3] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows https://arxiv.org/abs/2103.14030
[4] EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks http://arxiv.org/abs/1905.11946
[5] Deep Residual Learning for Image Recognition http://arxiv.org/abs/1512.03385