CP: 217 Project 1: Are you in a safe building?

FGR: Falak Fatima, Ganga Nair B, Ranwin Kumar

Introduction:

Problem statement: Advances in data availability offer new ways to improve seismic risk assessment. This project uses image classification to evaluate building safety and structural integrity during seismic events with machine learning.

Dataset Description: The dataset comprises grayscale images (400 x 300 pixels) across five classes of buildings sourced from Google Street View. All images were confirmed to have the same shape. The dataset is imbalanced, with more images in classes D and C. The brightness vs contrast of the images were plotted and brightness of the dataset (123 ± 18), was acceptable, as it is centered within the 0-255 range with minimal deviation. The high contrast in the images aids in distinguishing features more effectively, so no additional preprocessing for colour correction was necessary. Approximately 10 outliers were identified, with a few examples shown below.

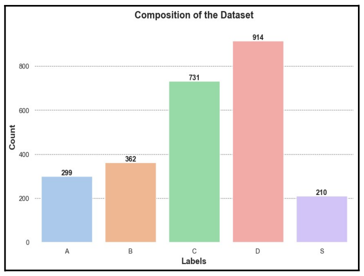

Fig 1: Distribution of data across classes

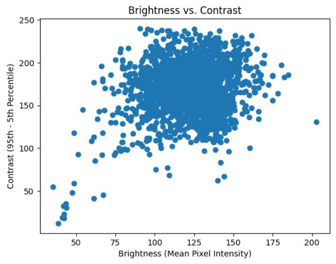

Fig 2: Depicts brightness vs contrast

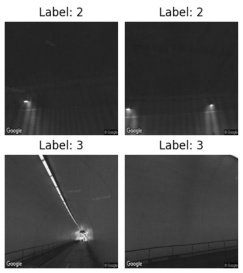

Fig 3: Outliers

Dataset Preprocessing:

Aim: Address class imbalance, generalize data, and extract features.

Data Augmentation: Applied via *ImageDataGenerator* and TensorFlow's *tf.keras.layers* with transformations (rotations, shifts, shearing, zoom, and flips). To address class imbalance, extra augmented images were generated for classes A, B, and S. However, to correct an imbalance ratio up to 900:200, more than ½ of the dataset was required to be augmented images which is not desirable and causes model overfitting. Using class weights during training proved more effective, so *ImageDataGenerator* was only utilized to diversify images. [3]
When comparing to *ImageDataGenerator*, *tf.keras.layers* has fewer options and are implemented in the pipeline during training. On comparing both on ResNet model, *ImageDataGenerator* gave a higher validation accuracy (Fig 4 below)

Segmentation: Initial attempts used the largest contour to segment images for feature extraction but failed to improve accuracy, as contours did not reliably represent buildings (Fig 5). Bounding box segmentation with pre-trained models was explored, but datasets like COCO and Pascal VOC lacked a specific "building" class. Ultimately, feature extraction was left to the CNN during model training.

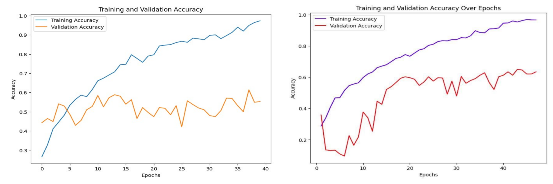

Fig 4: Comparison of 2 data augmentation techniques

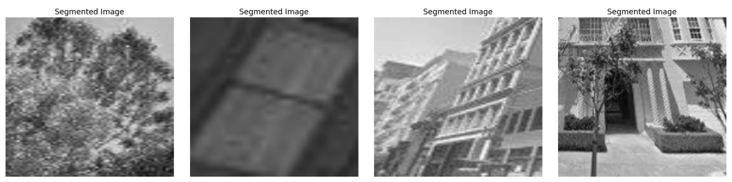

Fig 5: Examples of segmentation with contour

Model Selection

Classical models:

We explored several classical machine learning algorithms, including SVM with PCA and Random Forest, using feature extraction from the images. We faced a few challenges with classical models:

1. As this is a high complexity classification task, unsure if model captures complex patterns in the images
2. Extensive custom feature extraction required to help preserve spatial hierarchies and other complexities.
3. Lacked the ability to generalize well on given data especially with lower number of images per class.

Untrained CNN:
CNNs **automatically learn features** from raw images, capturing complex patterns, unlike SVM with PCA. This is vital for tasks like classifying building types from street-view images.
CNNs **preserve spatial hierarchies** by scanning small image sections, enabling them to detect local patterns like windows and doors, which SVM + PCA struggles with.
Non-linear activation functions like ReLU help CNNs model complex patterns, improving performance on datasets with high variability.

Custom features used with CNN and random forest models:
Predefined filters were used to extract relevant features.
1. Texture analysis using ‘Local binary Pattern’ (**LBP**)filter: Can help identify material, typer of architecture,
2. Edge detection with Sobel filter: To identify presence of windows, steel frames
3. Canny Filter to filter between 2 threshold values.

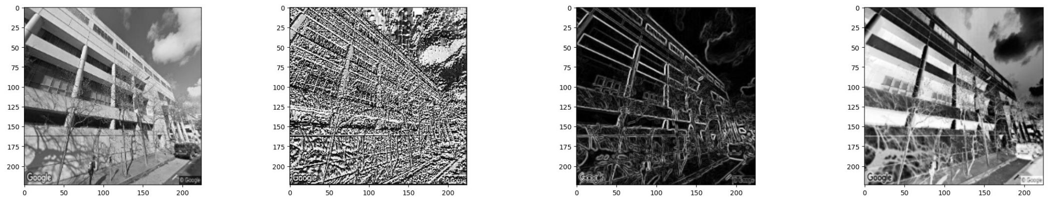

Fig 5: Details on custom filters. (i)Original image, (ii) LBP, (iii) Sobel Filter, Canny edge detection

Training a neural network from scratch requires a large dataset for each class. To overcome this challenge, we leveraged pre-trained networks, using their lower-layer weights to extract useful features. While not specifically trained for building classification, these pre-trained models help provide useful features as they are trained on very large amount of data.

Implementation of models

Trial no.	Model name	Additional features	Accuracy from Kaggle
1	Efficient NetB0, EfficientNetB3	- Pretrained EfficientNet + 6 layers - Class weights to reduce imbalance	43%, 42.5%
2	Ensemble of ResNet50 and random forest	- ResNet50 for feature extraction - Random forest for classification	33%
3	CNN + Random Forest	- Features from CNN input to random forest	33.4%
4	CNN - Pretrained ResNet50 with pre-processing, Renet50(125*125)	- Optimised image size to 256 x 256 + image data generator	54.7%,45%

Pre-trained NN: Details on selected model

We selected **ResNet50**[1] for image classification due to its superior accuracy, achieving 54% compared to **EfficientNet**[2] (40-45%) and **SVM with PCA** (30%). ResNet50's 50-layer architecture, pre-trained on ImageNet, provides excellent feature extraction for complex patterns, making it ideal for our dataset. While EfficientNet focuses on computational efficiency, it struggled to capture the intricate features required here, likely due to its emphasis on minimizing resource use, which can limit detail detection. Fine-tuning ResNet50's top layers with regularization further improved its adaptability and generalization, outperforming the other models

Model Summary:

Layer (type)	Output Shape	Param #
resnet50 (Functional)	None, 4, 4, 2048	23,587,712
flatten (Flatten)	None, 32768	0
dense (Dense)	None, 1024	33,555,456
dropout (Dropout)	None, 1024	0
dense_1 (Dense)	None, 512	524,800

dropout_1 (Dropout	None, 512	0

Tuning of Optimiser: We tested AdamW and Momentum optimizers. AdamW, with its adaptive learning rates and weight decay, achieved 49% accuracy, outperforming Momentum's 33%, which lacked adaptive rates.
Overfitting reduction: We used ReduceLROnPlateau to adjust learning rates when validation loss plateaued, and checkpoints saved the best model. Running for 40 epochs, with early stopping, ensured sufficient training without overfitting.

Experiment with unfreeze layers and handling class imbalance: Performance differences arise from balancing general features and task-specific adaptation. Unfreezing too many layers (e.g., 30) exposes too many parameters, causing overfitting and reducing accuracy to 22.4%. Here we are unable to take advantage of the pretrained weights in ResNet50. Unfreezing too few layers (e.g., 10) restricts the model's flexibility, dropping accuracy to 11%. Unfreezing 20 layers strikes the right balance, preserving general features while allowing task-specific learning, achieving the best accuracy of 54.8%. This balance optimizes generalization and adaptation for the dataset. Class imbalance, was addressed by using class weights which are inversely proportional to number of instances in the class. It is calculated as $Class\ Weight\ (i) \ = \ Avg\ no.\ of\ data\ in\ a\ class / \ (\#\ of\ data\ in\ (i))$.

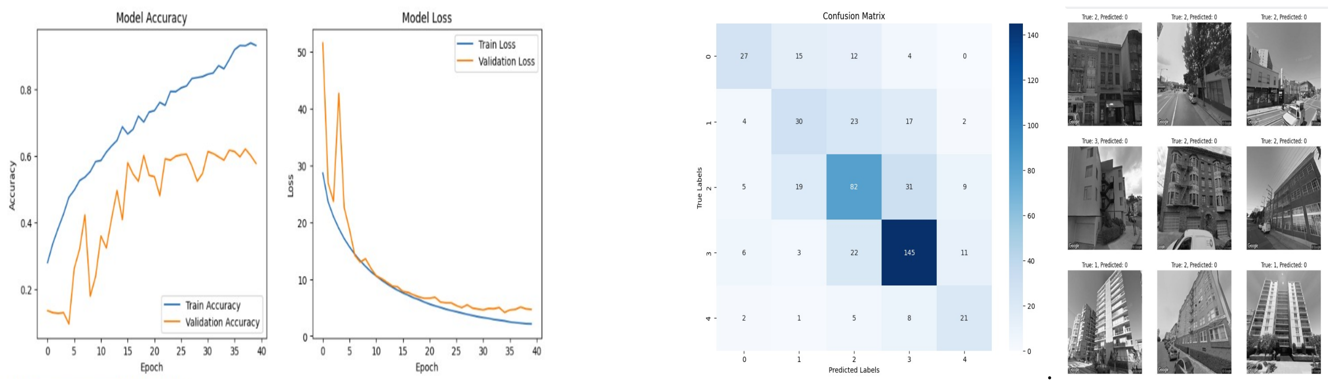

Fig 6: Model characteristics; (i)Plot of accuracy and loss vs epochs; (ii)Confusion matrix; (iii) Misclassified predictions

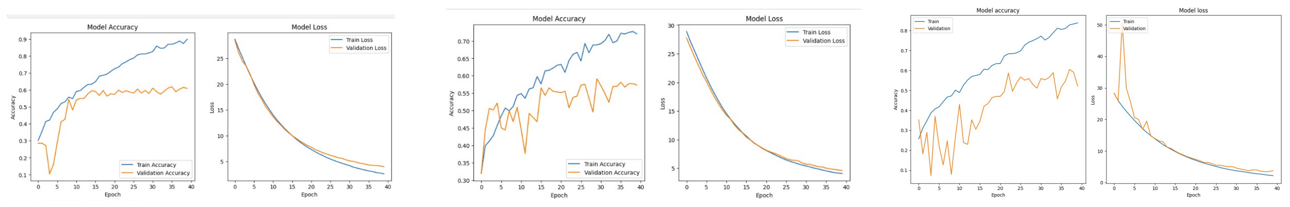

Fig 7: Accuracy and loss vs epoch plot of (i)EfficientNetB0; (ii) EfficientNetB3; (iii)ResNet50(125*125)

Conclusion:

ResNet50 outperformed other models, achieving 88% training accuracy, 62% validation accuracy, and 54.8% on Kaggle by capturing complex image patterns.
Key takeaways: While *EfficientNet* models focused on computational efficiency, they lacked the necessary detail extraction capabilities for this task. The best performance was achieved by fine-tuning *ResNet50*, applying class weights, and using moderate data augmentation.
Challenges & Improvements: Class imbalance was a persistent challenge. Although class weighting helped, further work could involve experimenting with more targeted augmentation or advanced segmentation techniques tailored to building features. Additionally, the use of more specialized pre-trained models for building detection and segmentation combined with pre-trained models for prediction could improve accuracy.

References:

1. Theckedath, D., & Sedamkar, R. R. (2020). Detecting affect states using VGG16, ResNet50 and SE-ResNet50 networks. SN Computer Science, 1(2), 79.
2. Koonce, B., & Koonce, B. (2021). EfficientNet. Convolutional neural networks with swift for Tensorflow: image recognition and dataset categorization, 109-123.
3. Chaki, J., & Dey, N. (2018). A beginner's guide to image preprocessing techniques. CRC Press.