**Report on Model Selection and Performance Analysis for Building Classification**

1. Exploratory Data Analysis (EDA) and Initial Observations

We noticed significant issues, such as difficulty in distinguishing building types visually due to similarities across classes. Additionally, some images had obstructions like trees or were tilted, further complicating the classification task. This analysis highlighted the need for careful data handling and informed our approach moving forward.

2. Traditional Machine Learning Models: A Baseline Approach

We started by experimenting with traditional machine learning models—Logistic Regression, Random Forest, and SVM—to establish a baseline.

* **Logistic Regression:** Initially yielded a 31% validation accuracy. After adding HOG feature extraction, accuracy improved to 44%, highlighting the importance of specialized feature extraction for images.
* **Random Forest:** Achieved 43% validation accuracy.
* **SVM:** Began with 32% accuracy. After tuning parameters like C and kernel types, accuracy improved to 50%, indicating non-linear structures in the data that required more complex modelling.

3. Challenges with Initial Models

Despite some improvement with HOG and hyperparameter tuning, our results remained below expectations. The primary reasons were:

* Data Imbalance: The classes were imbalanced, leading to biased predictions.
* Small Dataset: With only ~2,000 images, models were unable to generalize well without overfitting.
* Model Complexity: Traditional machine learning models like Logistic Regression and Random Forest struggled with the complexity of image data.

4. Moving Towards Deep Learning: Transfer Learning

Given the challenges with traditional models, we turned to transfer learning using pre-trained models like ResNet50, VGG16, and Vision Transformer (ViT). These models already have learned features from large datasets, which are useful for smaller datasets like ours.

* ResNet50: After applying basic data augmentations, ResNet50 achieved a validation accuracy of 47%. This was an improvement over traditional methods, but not enough to move forward.
* VGG16: VGG16 slightly outperformed ResNet with a validation accuracy of 47.6%. Despite better feature extraction, both ResNet and VGG struggled with overfitting due to the small dataset size.
* ViT: The Vision Transformer model gave us a training accuracy of 57%, but the validation accuracy dropped to 43.85%. Transformers tend to require large datasets, and the lack of sufficient data hampered performance.

These results led us to conclude that the dataset's size was the primary limiting factor. Augmentations and more sophisticated models were necessary to improve the performance further.

5. A Strategic Shift: Trying Multiple Models

To address the limitations, we decided to try a range of popular models to determine which one would give the best baseline performance. We tested EfficientNet, DenseNet, and Inception, alongside additional data augmentation techniques. This approach allowed us to compare model performances directly and choose the best candidate for further tuning.

* EfficientNet: Achieved the best results with a training accuracy of 86.33% and validation accuracy of 58.13%. EfficientNet's scaling mechanisms helped capture features even with a smaller dataset.
* DenseNet: Underperformed with a validation accuracy of 19.44%, likely due to its complexity relative to the dataset size.
* Inception: Achieved a validation accuracy of 32.14%, slightly better than DenseNet but still far behind EfficientNet.

At this point, we visualized the data using PCA and observed that EfficientNet produced the best clustering of classes, confirming it as the best model for our project.

6. Overcoming Data Imbalance and Overfitting

To address overfitting and data imbalance, we applied several techniques:

* Data Augmentation: We implemented various transformations like horizontal flips, small rotations, brightness/contrast adjustments, and Gaussian blur to artificially increase our dataset's diversity. This helped improve the model's robustness.
* Batch Normalization and Dropout: In our EfficientNet model, we added Batch Normalization and Dropout layers to reduce overfitting, particularly useful when working with small datasets.
* Focal Loss: We also used a custom Focal Loss function to handle the class imbalance. This loss function focuses more on harder-to-classify samples, helping the model learn from minority classes more effectively.
* Weighted Sampling: Stratified KFold and WeightedRandomSampler were used during training to ensure that all classes were represented fairly during each epoch.

7. Fine-Tuning EfficientNet and Achieving Success

Using Optuna for hyperparameter tuning, we optimized key parameters such as learning rate, number of epochs, and the gamma value in our Focal Loss. After a series of trials, we reached an accuracy of 89%. This was a significant jump from earlier models.

8. Final Improvements: Feature Extraction and PCA

To further improve our model, we extracted features from the last hidden layer of EfficientNet and applied PCA to reduce the dimensionality. We found that 1,500 features were sufficient to explain 95% of the variance in the data. PCA revealed that the data was highly non-linear, and linear classifiers like SVM with RBF kernel performed well in this space, giving us an outstanding accuracy of 97%.

9. Conclusion

Key takeaways include:

* EfficientNet was the most suitable model for our small dataset, particularly with data augmentation and advanced loss functions.
* Handling Imbalanced Data: Using Focal Loss and stratified sampling was crucial for dealing with imbalanced classes.
* PCA allowed us to uncover the non-linear structure of the data, leading to significant performance improvements with non-linear classifiers.

Our next step involves exploring additional non-linear classifiers and experimenting with ensembling methods to push the performance further.

 