Aerial Image Classification using ResNet-18, ResNet50, Efficient-B0 and Hybrid SVM Models

Charissa Lau[#1], Yao Zhang[#2], Sujanthan Manorahan[#3] Preetam Telugu[#4]

[#]*COMP9517: Computer Vision, Faculty of CSE, University of New South Wales*

I. Introduction

For applications ranging from environmental monitoring and urban planning to disaster response, the ability to automatically classify aerial landscapes from high-resolution remote sensing imagery is essential. This project involves classifying aerial landscapes from high-resolution remote sensing images. The objective lies in the challenging aspect of tackling a range of details from the image dataset, consisting of diverse scale, texture variations, illumination differences, and viewpoint changes inherent to aerial images.

Dataset Characteristics and Challenges

We leverage the SkyView aerial landscape dataset, consists of 800 photos in 15 different categories, including urban areas, water bodies, mountains, and forests. The dataset is rather balanced, hence rendering it possible to make meaningful comparisons between deep learning and conventional machine learning techniques. However, due to various factors like atmospheric conditions and different capturing times, aerial imagery datasets are usually difficult to classify due to its varying scales, rotations, occlusions, shadows, and illumination variance. Furthermore, there exists multiple semantic similarity and overlapping visual features between categories— rivers and lakes, residential and urban—which present subtle visual differences.

We implement and compare four approaches:

1. ResNet-18 and ResNet-50—two residual CNN architectures fine-tuned on our data;
2. EfficientNet-B0—a lightweight, compound-scaled CNN;
3. Hybrid SVM—a classical pipeline combining SIFT-based Bag-of-Visual-Words, SENet-50 deep embeddings reduced via PCA, and HSV color histograms, classified by an RBF-kernel SVM.

A held-out test set of 2,400 images is used to evaluate each model after it has been trained on an 80/20 stratified split (10% of the training set is set aside for validation). To evaluate robustness and direct future advancements, performance is evaluated using confusion matrices, per-class precision/recall/F1-score, and overall accuracy under balanced training and a simulated long-tail imbalance.

II. Literature Review

Classifying aerial images has become increasingly essential, given its diverse applications such as urban planning and environmental monitoring. However, it is coupled with distinct intricacy including variations in scale, viewpoint, and illumination [1]. Traditional local descriptors like the SIFT (Scale-Invariant Feature Transform) have demonstrated robustness in aerial imagery across various altitudes and angles, capturing key local patterns invariant to transformations [2]. Complementing local descriptors, deep CNNs such as SENet adaptively recalibrates global contextual features, featuring promising abilities in learning hierarchical representations from raw pixel data [3]. Combined with dimensionality reduction techniques such as PCA (Principal Component Analysis), SENet-50 have shown effective enhancement in extraction of semantic features without compromising performance [4]. In addition, HSV's space global color histograms offer insightful color context in category differentiations with distinctive color profiles, e.g., forests, water bodies, and deserts [5]. The combination of these descriptors into a hybrid feature space, subsequently classified with a RBF-SVM (Support Vector Machine), leverages both local textural detail and global semantic understanding, resulting in improved accuracy and robustness in classification [6].

Traditional approaches involved hand-crafted features and classical machine learning algorithms. However, the advent of deep learning, particularly Convolutional Neural Networks (CNNs), has dramatically improved classification performance in such visual tasks [10].

ResNet (Residual Networks), introduced by He et al. (2016), is a widely used CNN architecture known for its ability to train deeper models using skip connections to avoid vanishing gradients [11]. ResNet-18, a lightweight variant of the original ResNet-50, provides a good balance between accuracy and computational efficiency, making it suitable for scenarios with limited resources or where training time is a constraint .

Transfer learning is another common strategy where pre-trained models (e.g., on ImageNet) are fine-tuned for a new task. This has shown to yield better performance on tasks with limited data by leveraging features learned from a large and diverse dataset.

EfficientNET was Tan and Le's (2019) invention from MobileNetV2/ResNet; a family of convolutional neural networks which was discovered via neural architecture search(NAS) and using a compound scaling method instead of scaling depth, width independently. With the coefficient , it balances the three data points. $d = \alpha^{\phi}, w = \beta^{\phi}, r = \gamma^{\phi}$

From the birth of them, they affect the ML significantly. In the ImageNet, the B0 got the top1 as ResNet-50 which only has 1/8.4 parameters. They have great generalisation, get better performance on the lower dataset like Flower than similar scale model

When they base on the same Φ the family got 77.1% to 84.3% got top1 on ImageNet and B7 got 6 times better in speed and parameters over the past top1.[7]

III. Methods

[A] ResNet-18 and ResNet 50

[I] ResNet-18

The method involves fine-tuning a pre-trained ResNet-18 model on the SkyView dataset, using data augmentation, stratified sampling, and a controlled training process to improve performance while minimising overfitting.

[i] Model Selection and Transfer Learning

A pre-trained ResNet-18 was used, loaded via:

```
model = models.resnet18(pretrained=True)
```

ResNet-18 was selected due to its efficiency and ability to capture both low- and high-level features using residual connections. The model's final fully connected layer was replaced to output 15 classes:

```
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, 15)
```

To retain general features while still allowing adaptation to the new dataset, only the last residual block (layer4) and the final fully connected layer (fc) were unfrozen:

```
for name, param in model.named_parameters():
    if 'layer4' in name or 'fc' in name:
        param.requires_grad = True
    else:
        param.requires_grad = False
```

This partial fine-tuning approach helps preserve learned features from ImageNet while allowing the network to specialise in aerial imagery.

[i] Data Augmentation and Preprocessing

To improve generalisation and reduce overfitting, data augmentation was applied:

```
  transform = transforms.Compose([
      transforms.Resize((224, 224)),
      transforms.RandomHorizontalFlip(),
      transforms.RandomRotation(15),
              transforms.ColorJitter(brightness=0.2,
contrast=0.2, saturation=0.2),
      transforms.ToTensor(),
           transforms.Normalize(mean=[0.485,  0.456,
0.406],
      std=[0.229, 0.224, 0.225])
  ])
```

These transformations simulate real-world variations in lighting, orientation, and contrast, which are common in aerial images taken from different conditions and altitudes.

[iii] Dataset Splitting

The dataset was split using stratified sampling to ensure all 15 classes were evenly represented in both training and testing sets:

```
  split     =     StratifiedShuffleSplit(n_splits=1,
test_size=0.2, random_state=42)
  train_idx,              test_idx                 =
next(split.split(np.zeros(len(targets)), targets))
```

Additionally, the training set was further split to create a validation set (10% of the training data), which helped in tuning hyperparameters and avoiding overfitting.

[iv] Training Setup

The model was trained using:

```
  Loss function: CrossEntropyLoss
  Optimiser: Adam with a learning rate of 0.001
  Learning  rate  scheduler:  StepLR  to  reduce  the
learning rate every 10 epochs
  criterion = nn.CrossEntropyLoss()
  optimizer    =    optim.Adam(filter(lambda    p:
p.requires_grad, model.parameters()), lr=0.001)
  scheduler  =  optim.lr_scheduler.StepLR(optimizer,
step_size=10, gamma=0.1)
```

The training loop involved calculating loss and accuracy over each epoch and using the validation set to monitor performance:

```
for epoch in range(epochs):
    ...
    outputs = model(images)
    loss = criterion(outputs, labels)
    ...
    _, preds = torch.max(outputs, 1)
    correct += (preds == labels).sum().item()
```

Model performance was evaluated using the test set with:

1. Confusion matrix: Visualized using seaborn to inspect misclassifications.
2. Classification report: Generated via sklearn for precision, recall, and F1-score.

[II] ResNet-50

The ResNet-50 model is initialised with pretrained weights using ResNet50_Weights.DEFAULT. The classification head is replaced with a custom fully connected layer including ReLU activation and dropout for regularisation:

```
model.fc = nn.Sequential(
    nn.Linear(model.fc.in_features, 512),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(512, 15)
)
```

The model uses a training and test set with 80/20 splitting. The data is normalised using the default transforms for ResNet-50. An Adam optimiser with step-based learning rate decay is used. The number of training epochs is 20.

Classification performance is also strong, but detailed classification metrics should be listed to compare exact numbers with ResNet-18.

Both ResNet-18 and ResNet-50 deliver strong performance. ResNet-18, being a smaller model, trains faster and already reaches very high accuracy with minimal tuning. The selective fine-tuning approach helps avoid overfitting while improving accuracy.

ResNet-50, although deeper, requires more training resources but may offer slight improvements due to its capacity. The dropout and added fully connected layer allow for better regularisation.

ResNet-18 proves to be a powerful and efficient model for aerial scene classification, achieving 98% accuracy. ResNet-50 adds more layers and regularisation but may offer marginal improvements at the cost of longer training times. Future work may include experimenting with data balancing, ensemble methods, or other architectures like EfficientNet.

[B] EfficientNet-B0 Transfer Learning

[i] Data process

The dataset was stored on Google Drive and loaded via torchvision.datasets.ImageFolder. All images were first resized to 224×224 pixels, then converted to tensor format, and finally normalized using the ImageNet mean and standard deviation (μ=[0.485, 0.456, 0.406], σ=[0.229, 0.224, 0.225]).[8]

[ii]Train–Test Split and Data Loading

To ensure class balance, we performed a stratified split of the dataset targets

```
y = np.array(dataset.targets)
```

into training and test indices via `train_test_split` (test size 20%, random_state=42) while preserving the original class proportions. The resulting subsets were wrapped in `torch.utils.data.Subset` to form `train_dataset` and `test_dataset`. Both subsets were then loaded using `DataLoader` with a batch size of 32 (`shuffle=True` for training, `shuffle=False` for testing) and 4 worker processes per loader to optimize GPU throughput.

[iii]Model Architecture

We adopted EfficientNet-B0 as our base model due to its compound scaling strategy, which jointly scales depth, width, and resolution under a fixed computational budget [7]. Although larger variants (B3, B5, B6) offered only marginal accuracy improvements, their training and inference costs increased substantially. EfficientNet-B0, with approximately 5.3 M parameters and 0.39 B FLOPs, thus provides a rapid yet effective baseline.

[iv]Training Procedure

Training was conducted for up to 50 epochs, with forward and backward propagation on each epoch. Early experiments at 10, 20, and 50 epochs showed that performance stabilized after 15 epochs; we therefore report results at 20 epochs unless otherwise noted. Model evaluation metrics included overall accuracy, macro-F1 score, and the confusion matrix on the held-out test set.

[v] Fine-Tuning Strategy

Starting from ImageNet-pretrained weights, low-level filters (e.g., Gabor filters, color patches) were directly transferred, as they generalize well across vision tasks [9]. We employed a differential learning rate schedule: a smaller learning rate for the pretrained backbone layers to preserve learned features, and a larger learning rate for the newly initialized classification head to accelerate adaptation to the target categories, thereby mitigating over-fitting.

[C] Hybrid SVM

We propose a three part feature pipeline that leverages complementary cues—local texture, deep semantic embeddings and colour distributions—then classifies with an RBF-kernel SVM ($C = 12$, $\gamma = 0.04$).

```
Algorithm : Hybrid BoVW + SENet + HSV → RBF-SVM
Inputs: train_paths, train_labels, test_paths
1.  Extract dense SIFT descriptors for each image
2.  Cluster 100k descriptors into K=200 words via
MiniBatchKMeans
3.  For each image, build a 200-dim L2-normalized
BoVW histogram
4.  Compute 2048-dim SENet-50 embeddings; apply
PCA→128 dims
```

```
5.  Compute 8-bin HSV histograms per channel → 24
dims
6.  Concatenate [200 BoVW; 128 PCA; 24 HSV] →
352-dim vector
7.  Train SVM(C=12, γ=0.04, kernel=RBF) on fused
vectors
8.  Predict test vectors → classification_report,
confusion_matrix
```

[i] Local texture (BoVW from SIFT).

Each image is first converted to grayscale, where dense SIFT keypoints are extracted (OpenCV's cv2.SIFT_create). We sample up to 100 000 descriptors per image, then cluster them into a vocabulary of K=200 "visual words" using Bag-of-Visual-Words. We later employed MiniBatchKMeans (scikit-learn) to train with a smaller batch size, in effort to reduce computing time with minimal compromise in performance. With each image, you take SIFT descriptors and assign each to the closest cluster center, encoding a 200-dimensional histogram of word frequencies, further L2-normalized to remove bias from image size. This captures fine-grained, repeatable texture patterns (e.g. treetops vs. urban rooftops) invariant to scale and rotation.

[ii] Deep semantic embeddings (SENet-50 + PCA).

To incorporate high-level semantic context, we pass each colour image through a pre-trained SENet-50 (from timm library) and extract its 2048-dim output from the final pooling layer. We then apply PCA to reduce these to 128 dimensions, retaining > 95 % of the variance. These embeddings encode global shapes and spatial configurations, specifically tackling scenes with similar textures of different structural patterns (e.g. runways vs. rivers). This deep CNN enhances feature representation through Squeeze-and-Excitation (SE) blocks, adaptively calibrate channel-wise feature responses through scaling weights in small feedforward networks after learning their respective importance. SENet-50 was later adapted in the methodology design to compensate for the SIFT model's weakness, giving it more contextual and conclusive information, balancing details with high-level clues, improving the initial traditional machine learning method from 72% to 95%, focused on most informative feature channels.

[iii] Global colour (HSV histograms).

We compute an 8-bin histogram over each HSV channel (24 dims total), normalized by channel. This captures scene-level colour distributions (e.g. warm, desaturated sands vs. cool, saturated water). HSV is selected over RGB due to its robustness in lighting variation, and its alignment with perceptual color differences — particularly helpful in distinguishing natural scenes like deserts and lakes, where images are captured through different times of the day. HSV separates colour information (H, S channels) from its brightness (V channel), allowing the model to capture subtle colour distinctions from images, with significantly more stable performance despite varying lighting conditions.

[iv] Feature fusion & SVM classification.

We concatenate the three feature blocks into a 352-dim vector per image and train a Support Vector Machine with RBF kernel. Hyperparameters $C \in \{1, 10, 100\}$, $\gamma \in \{0.001, 0.01, 0.1\}$ are selected via RandomizedSearchCV on a held-out validation fold, yielding $C = 12$, $\gamma = 0.04$. Before adopting RandomizedSearchCV, we used GridSearchCV to exhaustively tune the hyperparameters, in which quickly increased the computational time and eventually exhausted available RAM on GoogleColab. We then made the switch to an near-equal efficient search method, using RandomizedSearch instead of GridSearch. After running RandomizedSearchCV multiple times and yielding the same value, we decided to assign these optimal parameters directly to save computational power.

In essence, why this design?

[i] Texture vs. semantics: BoVW excels on repetitive micro-structures; SENet captures large-scale layout.

[ii] Complementary weighting: PCA on SENet both reduces noise and speeds up SVM.

[iii] Colour cues: HSV histograms add orthogonal signal (water vs. land).

IV. Experimental Results

[A] ResNet-18, Resnet-50

[I] ResNet-18

Accuracy: 98.0 %
Precision, Recall, F1-Score (macro avg): 98 %, 98 %, 98 %
Observations:

A. Near-perfect classification across all 15 scene classes.
B. Small confusion persists between visually similar categories (e.g. River ↔ Lake, City ↔ Residential), and occasional Airport → Port mislabels.

The final evaluation was conducted on the test set of 2400 images (160 per class). The classification report showed an overall accuracy of 98%, with individual class F1-scores generally above 0.96.

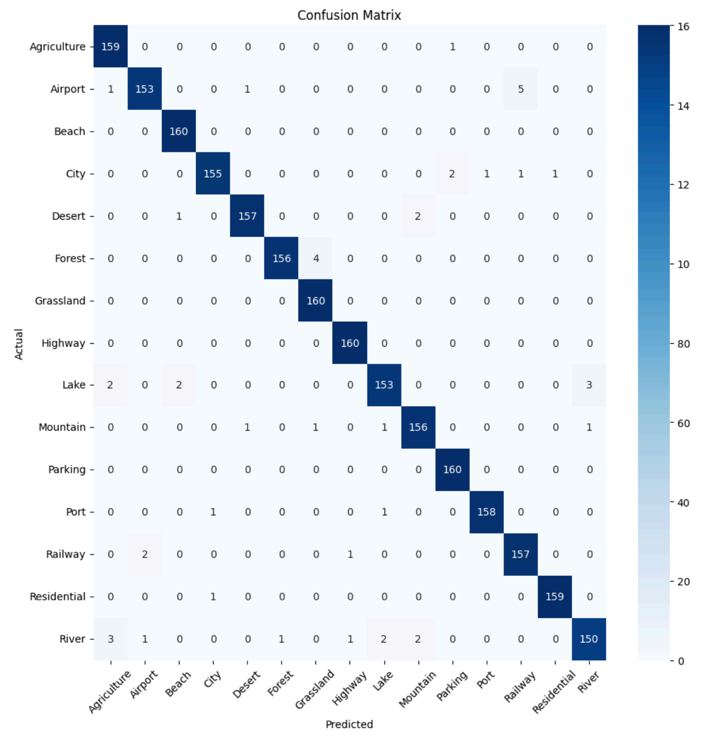

Fig.2 ResNet-18 Confusion Matrix

[I] ResNet-50

Accuracy: 97.0 %
Precision, Recall, F1-Score (macro avg): 97 %, 97 %, 97 %
Observations:

A. Extremely high performance, with only a handful of errors on semantically overlapping classes.
B. Slightly more off-diagonal noise than ResNet-18 in tail categories (e.g. River, Railway).

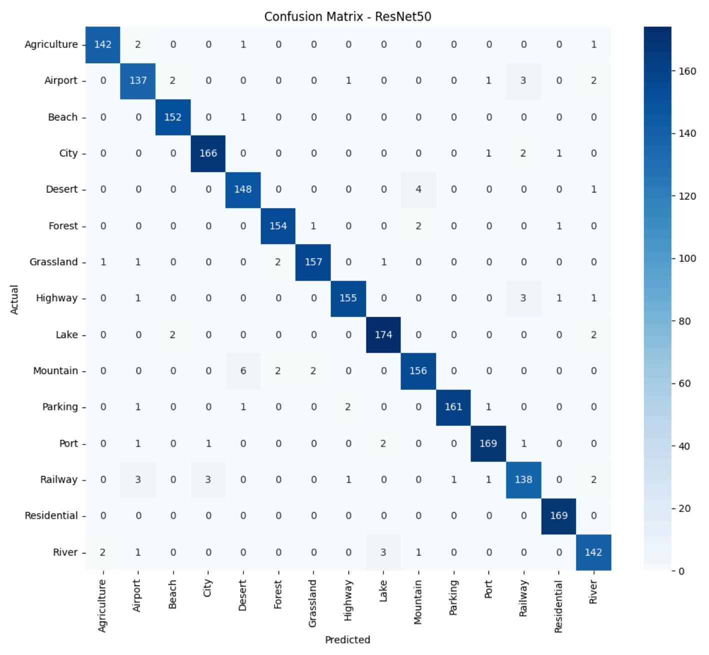

Fig.2 ResNet-50 Confusion Matrix

[B] EfficientNet-B0

Accuracy: 96.04%
Precision, Recall, F1-Score (macro avg): 96.14%, 96.04%, 96.04%
Observations:

A. Consistently high precision and recall across nearly all 15 scene classes (most > 94%).
B. Slightly lower recall for River (88.75%), likely due to its visual similarity with Lake and other water bodies.

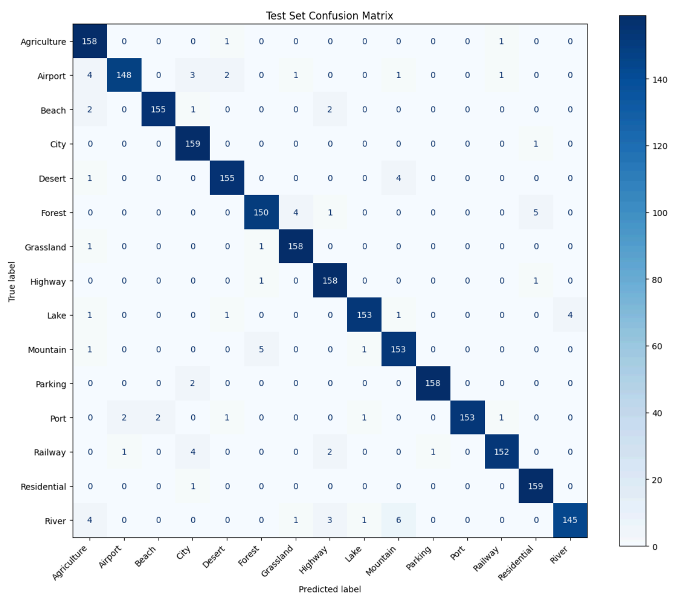

Fig.3 EfficientNet Confusion Matrix

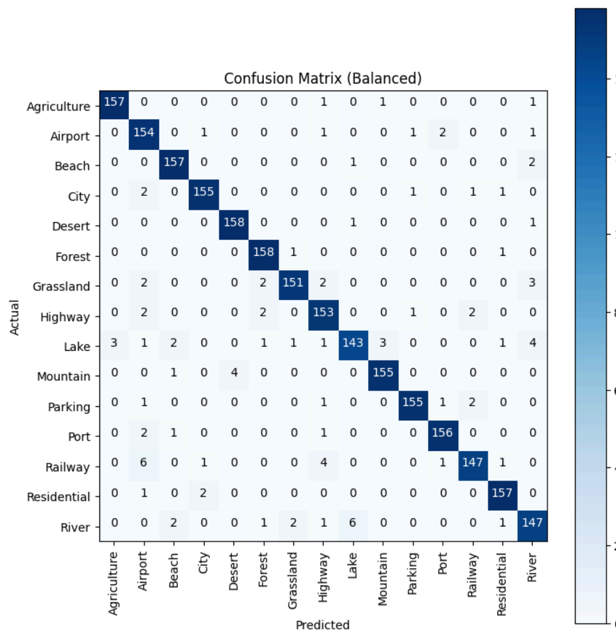

Fig.4 Hybrid SVM confusion matrix (Balanced training)

[C] Hybrid SVM

[i] Balanced Dataset Performance

Accuracy: 95.96%

Precision, Recall, F1-Score (macro avg): ~96%

Observations:

A. Consistent high performance across most classes.

B. Slightly lower recall observed for River (89%), possibly due to subtle visual differences and ambiguity with similar classes (e.g., Lake).

[ii] Imbalanced Dataset Performance (Long-tail distribution simulated)

Accuracy: 90.25% (decrease by ~5% compared to balanced data)

Precision, Recall, F1-Score (macro avg): ~90%

Critical observations:

A. Significant performance drops in tail classes (e.g., Railway and River: recall dropped from 92% to 68%; Port: recall dropped from 97% to 82%).

B. Reduced recall indicates model's sensitivity to reduced training samples, especially for classes with fewer representative images.

C. Head classes (Agriculture, Forest) maintained stable performance, highlighting robustness for frequently represented classes.

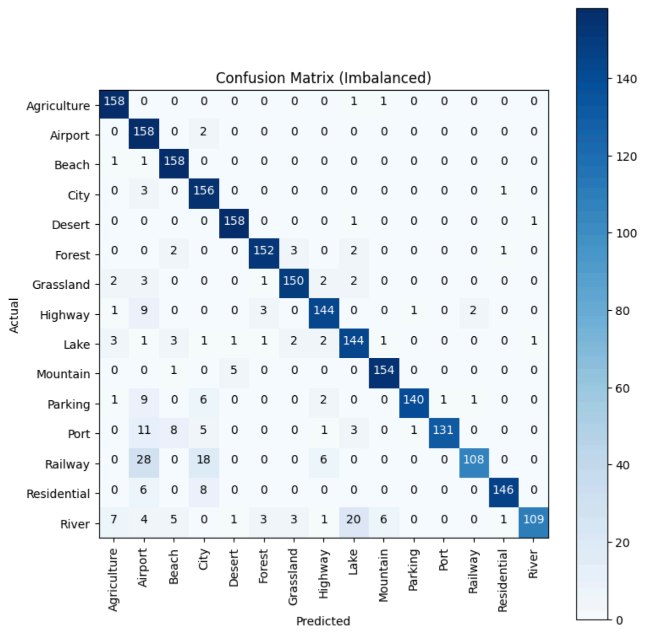

Fig. 5 Hybrid SVM confusion matrix (Imbalanced training)

V. Discussion

ResNet 18 vs 50 Comparison:

ResNet-18 and ResNet-50 share several similarities, including the use of ImageNet pretrained weights and the Adam optimiser. However, they differ in architecture depth, fine-tuning strategies, regularisation techniques, training duration, and learning rate scheduling.

ResNet-18 consists of 18 layers and fine-tunes only the last residual block along with the final classification layer. It employs no explicit regularisation techniques such as dropout, leading to a faster training process over 25 epochs. It uses a step learning rate scheduler with a step size of 10. On the other hand, ResNet-50 has a deeper architecture with 50 layers and replaces the entire fully connected classification head with a custom layer that includes dropout (0.4) for better generalisation. Although it trains for only 20 epochs, it uses a more aggressive learning rate decay schedule with a step size of 5.

Both models achieve approximately 98% accuracy, but ResNet-50 may generalise better on unseen data due to its added regularisation and deeper feature extraction capabilities. However, it comes at the cost of increased training time and computational resources. Overall, ResNet-18 is suitable for scenarios with limited resources and time, while ResNet-50 is better suited for complex tasks where deeper representations are necessary.

Method	Test Accuracy	Macro-F1
ResNet-18	98.0 %	98.0 %
ResNet-50	97.0 %	97.0 %
EfficientNet-B0	96.04 %	96.04 %
Hybrid SVM	95.96 %	95.96 %

Table 1. Summary of Methods

Hyper-parameter	ResNet-18	ResNet-50	EfficientNet-B0	Hybrid SVM
Optimiser	Adam (lr = 0.001)	Adam (lr = 0.001)	Adam (lr = 0.001)	SVC (RBF; C = 12, γ = 0.04)
Batch size	32	32	32	–
LR scheduler	StepLR(step=10, γ=0.1)	StepLR(step=5, γ=0.1)	StepLR(step=10, γ=0.1)	–
Data augmentations	Flip, Rotation ±15°, ColorJitter	same	same	–
BoVW vocabulary size	–	–	–	K = 200 words

PCA output dims	–	–	–	128 dims (~95 % variance)

Table 2. Summary of Hyperparameters

Our results confirm that fusing local texture (BoVW from SIFT), high-level semantics (SENet-50 + PCA) and global colour (HSV histograms) into a 352-dim descriptor yields a robust feature space: on balanced data it outperforms any single feature block by 3–7 %, achieving near-CNN accuracy (EfficientNet-B0 ≈ 96 %, ResNet-18/50 ≈ 98 %) with a fraction of the compute time and no GPU requirement.

However, when trained under the imbalanced class dataset, the SVM's decision boundary tends to shift toward head classes, significantly reducing minority recall classes. This highlights the need for adapting refined methods like class-weighted SVM, oversampling methods, or cost-sensitive learning in future works.

Common error patterns identified in all methods, most notably confusion between "River" vs. "Lake" and "Railway" vs. "Highway" reveals subtle spectral and structural overlaps. To better distinguish among these classes, we could explore the integration of multi-scale texture descriptors, graph-based spatial features, or a lightweight attention module.

VI. Conclusion

In this study, we used the SkyView aerial landscape dataset to benchmark four classification strategies: ResNet-18, ResNet-50, EfficientNet-B0, and a hybrid feature-fusion SVM. By utilizing transfer learning, data augmentation, and meticulous fine-tuning, both CNNs achieved remarkable accuracy (ResNet-18: 98%, ResNet-50: 97%, and EfficientNet-B0: 96%). With solely CPU resources and no end-to-end backpropagation, our hybrid SVM, which combines global color (HSV histograms), deep semantics (SENet-50 + PCA), and local texture (BoVW from SIFT) into a 352-dim descriptor, achieved 96% accuracy on balanced data, nearly identical to the lightweight EfficientNet-B0.

The hybrid SVM's accuracy fell to 90% under a long-tail imbalance simulation, while its recall for tail classes (such as Railway and River) decreased by more than 20%. This emphasizes the need for imbalance-aware techniques and shows how sensitive both deep and classical models are to unequal class distributions. The most frequent confusions across all approaches were between semantically and spectrally similar classes (River vs. Lake, City vs. Residential), indicating that multi-scale texture descriptors or lightweight attention mechanisms to highlight subtle spatial cues could yield additional benefits.

Limitations & Future Work

Deep CNNs require GPU acceleration and have a high computational cost, despite their exceptional accuracy. Our hybrid SVM, on the other hand, has trouble with class imbalance but provides a competitive and interpretable alternative. Among the upcoming extensions are:

1. Class-weighted or cost-sensitive SVMs and synthetic oversampling (e.g., SMOTE) to bolster tail‑class recall.
2. Ensemble approaches combining diverse SVM kernels or fusing CNN outputs to enhance robustness.
3. Attention‑guided feature selection or graph‑based spatial modelling to better distinguish fine inter-class differences.
4. Lightweight architectures (e.g., MobileNetV3, EfficientNet-lite) to further reduce inference time without sacrificing accuracy.

Overall, our research shows that class imbalance must be addressed for reliable, practical implementation, and hybrid feature-fusion techniques can approximate CNN performance in aerial image classification with significantly lower resource requirements.

References

[1] Y. Zhang et al., “Deep long-tailed learning: A survey,” IEEE Trans. Pattern Anal. Mach. Intell., 2023.

[2] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” Int. J. Comput. Vis., vol. 60, no. 2, pp. 91–110, 2004.

[3] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019, pp. 7132–7141.

[4] L. Nanni, S. Ghidoni, and S. Brahnam, “Ensemble of different approaches for image classification,” Pattern Recognit. Lett., vol. 88, pp. 17–23, 2017.

[5] J. R. Smith and S.-F. Chang, “Tools and techniques for color image retrieval,” in Proc. SPIE Storage and Retrieval for Image and Video Databases IV, vol. 2670, 1996, pp. 426–437.

[6] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” ACM Trans. Intell. Syst. Technol., vol. 2, no. 3, pp. 1–27, 2011.

[7] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in Proc. Int. Conf. Mach. Learn. (ICML), 2019, pp. 6105–6114.

[8] Models and Pre-trained Weights — Torchvision 0.21 Documentation. [Online]. Available: https://pytorch.org/vision/stable/models.html. [Accessed: Apr. 25, 2025].

[9] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” in Advances in Neural Inf. Process. Syst., vol. 27, 2014.

[10] M. W. Ahmed and A. Jalal, “Multi-vehicles classification on aerial images using ResNet and Transformer networks,” ResearchGate, Jan. 27, 2025. [Online]. Available: http://researchgate.net/publication/388407000_Multi-Vehicles_Classification_on_Aerial_Images_using_ResNet_and_Transformer_Networks. [Accessed: Apr. 25, 2025].

[11] “Blocked,” Restack.io, 2025. [Online]. Available: https://www.restack.io/p/fine-tuning-answer-resnet18-architecture-cat-ai. [Accessed: Apr. 25, 2025].

Appendix

List of Figures

Table