Comparison between Image Classification Algorithms

Qi Fan[1], Changrun Jia[1], Junwei Tang[1], Borong Xu[1, 2], and Jiaao Yu[1]

[1]School of Computer Science and Engineering, University of New South Wales, Australia
[2]Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong SAR

***Abstract*—Image classification is a fundamental task in computer vision, and deep neural networks have become the mainstream solution. Meanwhile, traditional machine learning algorithms with feature extraction can also have good performance compared to their modern counterparts. In this experiment, we test both traditional machine learning with feature extraction and deep learning methods on RESISC45, an aerial landscape dataset. We compare the performance of traditional feature extraction methods (SIFT, LBP) with modern deep learning approaches (ResNet, KAN, Vision Transformer), providing comprehensive analysis and insights for method selection in aerial image classification tasks.**

***Index Terms*—image classification, feature extraction, deep learning, SIFT, LBP, ResNet, Vision Transformer, aerial imagery**

I. Introduction & Literature Review

Image classification is a fundamental task in computer vision, and deep neural networks have become the mainstream solution. Meanwhile, traditional machine learning algorithms with feature extraction can also have good performance compared to their modern counterparts. In this experiment, we test both traditional machine learning with feature extraction and deep learning methods on RESISC45, an aerial landscape dataset.

In small-sample image classification, traditional feature extraction methods such as SIFT and LBP, combined with classifiers like SVM and kNN, still have advantages. They have simple models, clear decision boundaries, and low dependence on training data. Therefor they are especially suitable for images with clear textures and distinct edges. These methods are also easier to understand and have lower training costs compared with deep learning.

ResNet [1], the winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2015, is considered the state-of-the-art model of image classification. The model benefits from the large amount of data in ImageNet. While in our experiment, ResNet will be pre-trained with RESISC45, to prove its performance on fine-tuning in a smaller dataset.

While MLP has been the majority model of Deep Learning, there are new concepts of models that have been discussed in recent years. KAN [2], which is a special neural network that contains trainable activation functions. In our experience, ResNet will be used as a feature extraction tool for KAN. The output of ResNet will be input as high-dimensional features to KAN. Then, the classification will be done by KAN.

Transformers [3] have been popular recently with the rise of General Pretrained Transformer (GPT). It has been proven as the state-of-the-art model in the field of Natural Language Processing. The success of the transformer relies on its special attention mechanism, which can extract information through the whole sequence. There is research conducted on transforming the success of the transformer to the field of image classification. In this experiment, we will implement a Vision Transformer (ViT) and compare its performance to traditional machine learning methods and ResNet.

Starting from the algorithm principle, this report will analyze the advantages and disadvantages of each method in detail, and compare the performance of different combination models through experimental data, providing a method selection reference and practical guidance for subsequent related tasks.

II. Methods

A. Dataset

The datasets are divided into training and test data with an 8:2 ratio. Both datasets contain evenly distributed data with 15 balanced classes. The test set served as an evaluation between algorithms. Each algorithm will be tested with the test set only once, maintaining the integrity of the test and detecting overfitting of the algorithm. Furthermore, the training dataset can be divided into a validation set and a training set method if needed, when training each algorithm.

B. Feature Extraction with Machine Learning

1) Feature Extraction: We used the Scale-Invariant Feature Transform (SIFT) to find important keypoints in each image. Then describe them as 128-dimensional vectors. SIFT is good at capturing local patterns that don't change much with scale, rotation, or lighting [4].

However, different images may have different numbers of SIFT features. To solve this, we used the Bag-of-Visual-Words (BoVW) method to turn them into fixed-length feature vectors. We collected all the SIFT features from the training images and used MiniBatchKMeans to group them into 100 clusters (called "visual words"). Afterwards, each image was represented by counting how many features fell into each cluster.

This gave us a consistent-length histogram that describes the image [5] [6].

We also tested Local Binary Pattern (LBP) features to capture texture information. LBP compares each pixel with its neighbors and encodes the result as a binary number [7]. We used 8 neighbors and a radius of 1. The result was converted into a histogram and normalized to form the LBP feature vector.

To combine both local structure and texture, we concatenated the BoVW feature vector (from SIFT) and the normalized LBP histogram to form a new feature vector. This combined feature is expected to improve classification performance by capturing more information from each image.

2) Classification: We used a linear Support Vector Machine (sklearn.svm.SVC) with class_weight='balanced' to handle class imbalance. SVM is well-suited for high-dimensional data and performs well in separating complex boundaries [8].

We also tested kNN (sklearn.neighbors.KNeighborsClassifier) with k=5. It is a simple and effective method that classifies images based on the labels of their nearest neighbors in feature space [9].

3) Normalization: Before feeding features into classifiers, we normalized them using StandardScaler, which removes the mean and scales to unit variance. This step helps improve performance, especially for distance-based models like kNN and margin-based models like SVM.

C. ResNet 18 + Kolmogorov–Arnold Network (KAN)

1) ResNet18: ResNet is one of the most widely adopted convolutional architectures, known for its residual connections that ease the training of deep networks [10]. In this project, we adopt the ResNet18 model provided by PyTorch, retaining its pretrained weights on the ImageNet dataset [11] to ensure strong generalization and feature quality.

2) KAN: On top of the extracted features, we applied the Kolmogorov–Arnold Network (KAN), a recently proposed architecture introduced by Zhang and Xu (2024) [12]. KAN replaces traditional fully connected layers with spline-based functional approximators, using piecewise cubic polynomials to model activation functions. This functional design grants KAN greater flexibility and interpretability while keeping the model lightweight. It is especially effective in capturing complex nonlinear patterns from lower-dimensional inputs, making it well-suited for classification tasks following convolutional feature extraction. By combining the powerful ResNet backbone with the adaptive capabilities of KAN, this hybrid architecture provides a robust and efficient solution for image classification, balancing pretrained generalization with flexible decision boundaries.

3) Data Preprocessing: All input images are resized to 224×224 pixels to match ResNet18 input size and normalized using ImageNet's mean and standard deviation values: mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225].

4) Feature Extraction:

- **ResNet18 Backbone:** A pretrained ResNet18 model from PyTorch is used as a feature extractor. The final classification layer is removed, and the output from the penultimate layer (512-dimensional vector) is used as deep features.
- **Frozen Backbone:** The ResNet model is used in `eval()` mode and not fine-tuned; all gradients are detached, making it a frozen feature extractor.
- **Feature Collection:** Features from train, validation, and test sets are extracted in batches using `torch.no_grad()` to avoid memory overhead and stored as NumPy arrays for downstream classification.

5) Model Architecture:

- **Network Structure:** The KAN architecture used is $[512, 128, 64, 15]$, where 512 is the input feature size from ResNet, followed by two hidden layers with spline activations, and a final softmax output for 15-class classification.
- **Spline Configuration:** Each neuron splits its input into 5 intervals and fits a 3rd-degree polynomial (cubic spline), enabling localized smooth modeling of features.

6) Training and Early Stopping:

- **Loss Function and Optimizer:** CrossEntropyLoss is used as the objective, and the Adam optimizer with $\text{lr} = 0.001$ is used to update model parameters.
- **EarlyStopping Strategy:** An early stopping mechanism is implemented based on validation F1-score. If the F1 score does not improve for 5 consecutive epochs, training is halted to prevent overfitting.
- **Epochs and Logging:** The model is trained for up to 42 epochs with training loss and validation F1 recorded each epoch. The best model is restored before final testing.

D. Resnet 34 & Resnet 50

1) Data Preprocessing:

- **Training Data Augmentation:** Applied random horizontal flips and rotations to enhance model generalization.
- **Image Resizing and Normalization:** All images are resized to 224×224 pixels and normalized using ImageNet's mean and standard deviation, aligning with ResNet's pretraining parameters.

2) Model Configuration:

- **Pretrained Model:** Loaded a pretrained ResNet34 and ResNet50 model.
- **Layer Freezing:** Froze all layers except for layer4 and the fully connected (FC) layer to reduce training time.
- **Output Layer Modification:** Replaced the FC layer to output predictions for 15 classes.
- **Optimizer and Scheduler:** Used the Adam optimizer with an initial learning rate of 10^{-4}, reducing the learning rate by a factor of 10 every 3 epochs.

3) Training and Evaluation:

- **Training:** Conducted training over 20 epochs.
- **Evaluation Metrics:** After each epoch, evaluated the model on the test set using accuracy, precision, recall, and F1 score.

- **Model Saving:** Saved the trained model weights to `resnet34_skyview.pth` or `resnet50_skyview.pth`.
- **Visualization:** Plotted training loss and test accuracy over epochs, saving the figure as `resnet34_training_metrics.png` or `resnet50_training_metrics.png`.

E. Vision Transformer (ViT)

1) Transformer: Transformers and Attention mechanisms have been proven in the Natural Language Processing(NLP) field. With the encoder-decoder architecture, Transformers can process sequential data [3]. ViT is based on treating image data as sequential data by dividing the image into patches and feeding it to the encoder [13].

2) ViT: In ViT, each image is divided into patches, analogous to tokens in NLP. A linear transformation with trainable parameters embeds each image patch into a higher-dimensional vector. Positional embeddings are then added to encode the spatial location of each patch within the image.

The transformer block processes these embeddings using a Multi-Head Self-Attention mechanism. Each patch vector is split into parts and transformed into queries, keys, and values for each attention head [13]. After self-attention computation, the outputs from multiple heads are concatenated and passed through another linear transformation layer. Layer normalization and residual connections are incorporated throughout the transformer architecture.

The transformer blocks' outputs are concatenated and fed into a final linear transformation layer with softmax activation. The model is trained using the Adam optimizer with cross-entropy loss.

The key advantage of ViT over CNNs lies in its ability to model long-range dependencies between different image regions. While CNNs use max pooling layers to compress spatial dimensions and feed the resulting features into an MLP, this approach primarily captures local relationships between neighboring regions. In contrast, ViT's self-attention mechanism enables direct modeling of relationships between any pair of patches, regardless of their spatial distance. However, ViT's reliance on linear transformations for patch embedding means it may not capture local pixel-level patterns as effectively as CNNs [3].

The parameters used for the ViT model are shown in Table I.

TABLE I
VIT PARAMETERS

Parameter	Value
Number of patches	16
Hidden dimension	64
MLP ratio	16
Batch size	512
Number of Attention head	8
Number of transformer block	4
Number of epoch	200
Learning rate	1e-3

III. EXPERIMENTAL RESULTS AND DISCUSSION

Table II shows the comparison of different methods on the test dataset.

TABLE II
METHOD COMPARISON RESULTS

Method	Acc.	Prec.	Rec.	F1
SIFT+BoVW+SVM	61.46	61.18	61.46	61.23
LBP+SVM	49.58	49.75	49.58	48.90
SIFT+LBP+SVM	71.67	71.78	71.67	71.57
SIFT+BoW+kNN	49.67	54.03	49.67	48.71
LBP+kNN	49.00	51.09	49.00	48.99
SIFT+LBP+kNN	58.63	61.05	58.63	58.17
ResNet18	88.83	88.85	88.83	88.80
ResNet34	97.21	97.23	97.21	97.21
ResNet50	97.67	97.68	97.67	97.67
ViT	64.25	56.77	64.25	59.93

All values are in percentages (%)

A. Traditional Machine Learning

1) SIFT + BoVW + SVM: BoVW effectively quantifies SIFT features, but in the absence of texture supplementation, the model is difficult to fully describe the image, and the classification effect is limited. The strong generalization ability of SVM partially makes up for this deficiency.

2) LBP + SVM: Although LBP is computationally efficient, it only captures texture information and lacks structural expression capabilities. Although SVM has advantages, its performance cannot be fully utilized when feature information is insufficient.

3) SIFT + LBP + SVM: The structural and texture information complement each other and enrich the image expression. Coupled with the strong discrimination ability of SVM, this model performs best in multiple categories of remote sensing images and has strong promotional potential.

4) SIFT + BoW + kNN: kNN itself lacks discrimination ability, especially when the feature space is high-dimensional. This combination is suitable for scenarios with few training samples and rapid deployment, but the overall performance is limited.

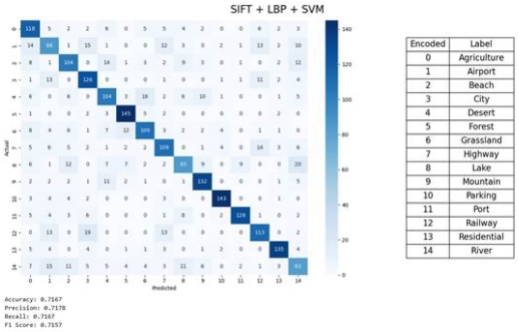
5) LBP + kNN: The combination of LBP and kNN is suitable for running in resource-constrained environments, but the classification accuracy is limited and is only suitable for applications with low performance requirements.

6) SIFT + LBP + kNN: Although kNN has limited performance, it effectively improves the model's discriminability by fusing structure and texture information, proving that feature fusion also has a positive effect on weak classifiers.

The best performance came from the combination of SIFT + LBP + SVM, with an accuracy of 71.67%.

The confusion matrix for this best method is shown in Fig. 1.

The results show that feature combination significantly improves classification performance. No matter which classifier is used, combining SIFT and LBP always has better



Encoded	Label
0	Agriculture
1	Airport
2	Beach
3	City
4	Desert
5	Forest
6	Grassland
7	Highway
8	Lake
9	Mountain
10	Parking
11	Port
12	Railway
13	Residential
14	River

Fig. 1. Confusion Matrix – SIFT+LBP+SVM

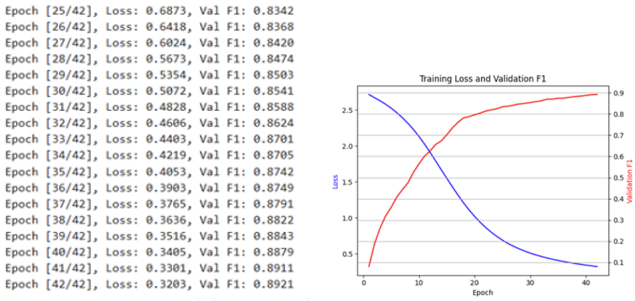


Fig. 2. The training of ResNet-18 KAN. Left: Loss and validation F1 over epochs. Right: Training curves showing consistent improvement.

performance. The improvement is especially obvious when using SVM, reaching an F1-score above 0.71.

In general, SVM performs better than kNN. This is because SVM can handle high-dimensional feature spaces and find clear decision boundaries. On the other hand, kNN is sensitive to local noise and lacks stability across categories.

LBP alone performs the worst in most cases. This shows that LBP is good at capturing texture, but struggles with global or structured features like buildings or roads.

We further analyzed performance across three types of categories (see Table III).

TABLE III
CATEGORY-LEVEL F1-SCORE COMPARISON

Category	SIFT+LBP+SVM F1	SIFT+BoVW F1	LBP+SVM F1
Parking	**0.90**	0.84	0.47
Forest	**0.81**	0.70	0.76
River	0.53	0.37	0.35

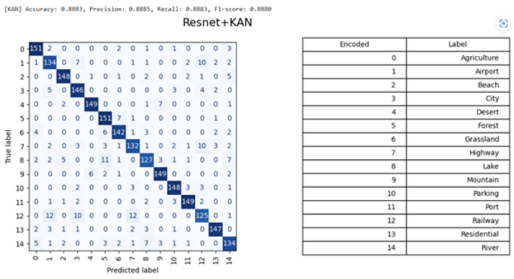
SIFT works better for categories like Parking and Port because it can detect clear edges and structures. On the other hand, LBP is more effective for images like 'Forest' and 'Grassland' that have a lot of texture. For harder categories like River and Lake, all methods find it difficult due to unclear textures and shapes, but using both SIFT and LBP together helps improve the results.

In general, combining these two methods makes the model more reliable because each method focuses on different features. This gives a strong starting point for comparing with deep learning methods later.

B. ResNet-18 + KAN

The left part of Figure 2 shows the value of loss and validation F1 in each epoch. In the figure, we can see that the early stopping mechanism we set was never triggered (it was configured to activate if the validation F1 score showed no significant improvement for 5 consecutive epochs). In practice, the F1 score kept improving steadily, even if the rate of improvement was relatively slow.

About the right figure, we can observe that both curves exhibit a consistent trend: as the training loss steadily decreases, the validation F1 score continuously increases. There is no indication of the F1 score declining and then rebounding, which suggests that the model is not overfitting. Given this pattern, we can reasonably conclude that the model is still improving. Although further increasing the maximum number of epochs might lead to even better performance, the experiment was stopped at this point due to hardware limitations.



Encoded	Label
0	Agriculture
1	Airport
2	Beach
3	City
4	Desert
5	Forest
6	Grassland
7	Highway
8	Lake
9	Mountain
10	Parking
11	Port
12	Railway
13	Residential
14	River

Fig. 3. Confusion matrix for ResNet-18 + KAN model

Fig. 4. Examples of challenging cases. From left to right: railway, highway, and airport, showing similar features like direct lines and blocks.

The confusion matrix (Figure 3) is generated based on the model's prediction results on the test dataset. It's good in general. There are some predictions of categories that are not that good, as shown in Figure 4.

From left to right are railway, highway, and airport. As we can see, they do have a lot of similar features, like direct lines and blocks, which may confuse the model.

Even if we can increase the KANs stack or increase ResNet to raise the accuracy of the model, I still think the best solution is to increase the size of the training dataset.

In addition, by comparing the final prediction performance of KAN and MLP (Figure 5), we can see that KAN slightly outperforms MLP, further supporting the validity of the KAN framework.

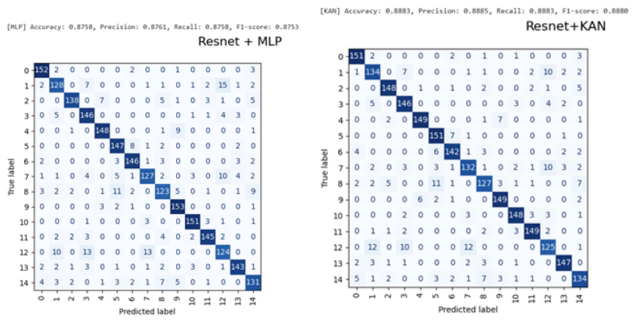


Fig. 5. Comparison of prediction performance between KAN and MLP

C. ResNet 34 & 50

1) Training Dynamics: Following is the training dynamics of ResNet34 and ResNet50.

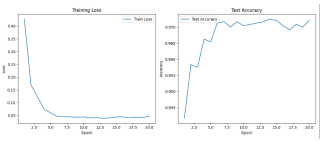


Fig. 6. Training dynamics of ResNet34

ResNet34

- Initial Phase
  - Epoch 1: Loss 0.4276, Acc 94.17%
  - Epoch 2–3: Rapid drop to Loss 0.1227, Acc 95.75%
- Mid/Late Phase
  - By Epoch 5: Loss 0.0597, Acc 96.54%
  - Plateau around Loss 0.04–0.05, Acc 97.1–97.2% between Epoch 6–20
  - Minor fluctuations (e.g. small dip at Epoch 13)

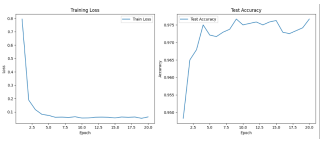


Fig. 7. Confusion matrix for ResNet50

ResNet50

- Initial Phase
  - Epoch 1: Loss 0.7952, Acc 94.83%
  - Epoch 2–3: Quick drop to Loss 0.1169, Acc 96.79%
- Mid/Late Phase
  - By Epoch 5: Loss 0.0736, Acc 97.21%
  - Plateau around Loss 0.05–0.06, Acc 97.2–97.7% between Epoch 6–20
  - Very stable test accuracy with minimal oscillation

2) Evaluation: The performance metrics for ResNet34 and ResNet50 are shown in Table IV.

- ResNet50 shows slight gains in challenging classes (e.g. Railway, Mountain, Airport), reducing misclassifications in the confusion matrix.

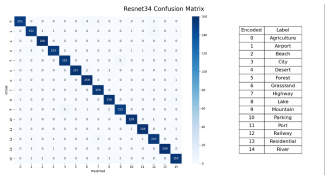


Fig. 8. Confusion matrix for ResNet34

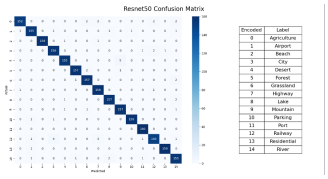


Fig. 9. Confusion matrix for ResNet50

- Overall, both models achieve excellent coverage, but ResNet50 edges out in precision/recall for several categories.

3) Summary Recommendations:

- Rapid Prototyping: For quick validation of model effectiveness, ResNet34 is a suitable choice.
- High-Precision Applications: For applications requiring higher model performance and generalization, ResNet50 is recommended.

Selecting the appropriate model architecture based on specific project requirements and resource constraints can enhance the effectiveness of image classification tasks.

D. Vision Transformer (ViT)

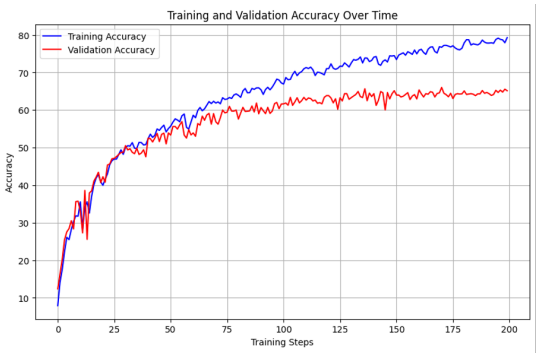
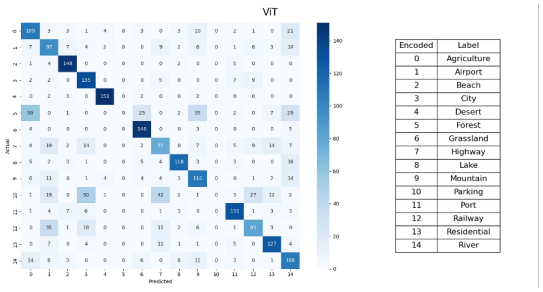


Fig. 10. Training and validation accuracy curves for ViT, showing overfitting

The performance of ViT is relatively poor compared to other deep learning methods, there are several reasons behind this.

TABLE IV
RESNET PERFORMANCE METRICS

Metric	ResNet34	ResNet50
Test Accuracy	97.21%	97.67%
Macro F1-Score	0.97	0.97
Weighted F1	0.97	0.97



Encoded	Label
0	Agriculture
1	Airport
2	Beach
3	City
4	Desert
5	Forest
6	Grassland
7	Highway
8	Lake
9	Mountain
10	Parking
11	Port
12	Railway
13	Residential
14	River

Fig. 11. Confusion matrix for ViT model

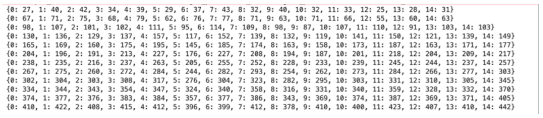


Fig. 12. Screenshot of the training progress

With the training and validation diagram (Figure 10), we can see that the validation accuracy is below the training accuracy. This implies the model suffers from overfitting. This suggests that the model is overcomplicated for this task, and the amount of data does not support the model.

For the false classification, a weird result shows that all the 'Forest' and 'Parking' images are classified as others. Although the training dataset contains these two classes, the number of outputs of the softmax layer is checked to be correct. And there is a classification of these two classes in the training progress, suggesting that the training progress includes these classes(Figure 12).

The 'Railway' images can easily be mixed up with airports. These faults are intuitive since some of the images look similar.

Fig. 13. Examples of Railway-Airport confusion in ViT predictions

IV. Conclusion

This project systematically evaluates various image classification models, ranging from feature extraction methods (such as SIFT, LBP) combined with traditional classifiers (SVM, kNN) to deep neural networks (ResNet18/34/50, KAN, and Vision Transformer) for remote sensing image classification tasks. Through the experimental process of balanced data sets and strict control of variables, the following key conclusions were drawn:

A. Findings

1) Traditional feature fusion strategies perform well in small sample scenarios: Combining the structural features of SIFT with the texture information of LBP and using SVM classification can achieve an accuracy of 71.67% and an F1 score of 71.57% without deep learning, showing the feasibility of traditional methods in resource-limited environments.

2) Deep convolutional networks are still the main force in image classification performance: Fine-tuned ResNet34 and ResNet50 achieved a classification accuracy of more than 97% in this experiment. Among them, ResNet50 has stronger generalization ability in multi-class classification tasks with deeper residual units, and is suitable for deployment in scenarios with high accuracy requirements.

3) KAN provides high-level expression capabilities for lightweight architectures: Inputting the features output by ResNet18 into KAN for classification shows an ideal compromise performance (F1-score of 88.80%) in situations where hardware resources are limited but accuracy is still required, proving the adaptability of this new activation method in low-parameter scenarios.

4) Vision Transformer has certain challenges on small dataset: Due to the high model complexity and high dependence on training samples, ViT has overfitting and fails to recognize specific categories on RESISC45. This model is more suitable for training and tuning on large-scale datasets and still has development potential in the future.

B. Current limitations

1) Data scale limitation: Although RESISC45 is a standard remote sensing dataset, it cannot cover more diverse and complex landforms and cross-scale interference.
2) Insufficient training conditions for Transformer: Lack of sufficient training samples and pre-trained parameter support limits the actual potential of the ViT model.
3) The fusion model is shallow: Currently, SIFT, LBP, and deep features have not yet achieved true deep fusion or joint optimization.

In summary, this project demonstrates the advantages and limitations of traditional methods and deep models in remote sensing image classification tasks through comparative experiments on multiple feature combinations and model structures. In small sample and low resource environments, manual features such as SIFT and LBP combined with traditional classifiers such as SVM are still competitive and suitable for scenarios with high demands for model interpretability and

rapid deployment. In the deep learning part, ResNet50 shows excellent performance and is suitable for task scenarios that pursue high precision and stability; KAN provides a new trade-off between lightweight and nonlinear expression. Although ViT has not fully exerted its advantages under current data conditions, its global modeling capabilities have long-term research value in remote sensing image understanding.

This study provides guidelines for model selection and fusion strategies in remote sensing image classification, and also provides a reference for the design of cross-generation architectures that combine traditional methods with deep models.

V. Future Work

1) **Hybrid Fusion Modeling**: We can try to concatenate the texture and edge features extracted by SIFT/LBP with the deep semantic features of ResNet or ViT to design a multimodal fusion classifier, such as CNN+SIFT/LBP+MLP.
2) **Enhanced Data-driven Capabilities**: Introduce transfer learning (such as using large-scale remote sensing data, such as ImageNet, DOTA, NWPU, to pre-train weights), and combine data enhancement (rotation, affine, light perturbation, etc.) to improve the generalization ability of ViT.
3) **Model Lightweighting and Deployment Optimization**: Research models such as MobileNetV3, EfficientViT, KAN-Lite, etc., reduce parameter scale and inference latency, and make the model more suitable for practical scenarios such as drones and edge computation.
4) **Expanded Exploration for Multi-source Data**: Further expand the input source, integrate multi-spectral and time-series remote sensing data, develop multi-channel input models, and expand from two-dimensional images to higher-dimensional remote sensing understanding.
5) **Improved Transformer Architecture**: Try lightweight versions of ViT (such as DeiT, Swin Transformer), adapt to the feature distribution of "strong local details and weak context" in remote sensing images, and improve the model's responsiveness to texture/boundary details.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.

[2] Z. Liu et al., "KAN: Kolmogorov-Arnold Networks," *arXiv preprint arXiv:2404.19756*, Apr. 2024.

[3] A. Vaswani et al., "Attention is all you need," *arXiv.org*, Jun. 2017.

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

[5] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. ICCV*, 2003.

[6] Y. Yang and T. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proc. ACM SIGSPATIAL*, pp. 270–279, 2010.

[7] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, 2002.

[8] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[9] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv.org*, Dec. 2015.

[11] J. Deng et al., "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[12] X. Zhang and K. Xu, "Kolmogorov–Arnold Networks," *arXiv preprint arXiv:2404.19756*, Apr. 2024.

[13] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv.org*, Oct. 2020.