UNIVERSITI SAINS ISLAM MALAYSIA

جامعة العلوم الإسلامية الماليزية

ISLAMIC SCIENCE UNIVERSITY OF MALAYSIA

Faculty of Science and Technology (FST)

Implementation Progress Report

(TriDCCS-SVM: A Hybrid Model for Satellite Image Classification)

Presented by:
YOUSEF H S ALSAFADI

Supervised by:
Professor. Rosalina Abdul Salam

2025

Introduction & Background

This work presents the implementation of TriDCCS-SVM, a hybrid model designed to enhance the classification accuracy of high-resolution satellite images. The model combines three deep convolutional neural networks (ResNet50, DenseNet169, and EfficientNetB0) for feature extraction and integrates a Support Vector Machine (SVM) for classification.

<u>The implementation focuses on:</u>

1. Data preprocessing: Cleaning datasets, resizing images, and handling invalid entries.
2. Feature Extraction: Extracting deep features using three pretrained CNN architectures.
3. Feature Fusion: Combining features from all three networks into a high-dimensional vector.
4. Classification: Training SVM on the fused features to perform the final classification.
5. Evaluation: Achieving high accuracy and outperforming previous studies.

<u>Key results</u>: Achieved high classification accuracy on benchmark datasets:

- UC Merced: Achieved 97.86% accuracy.
- SIRI-WHU: Achieved 96.25% accuracy.

USIM

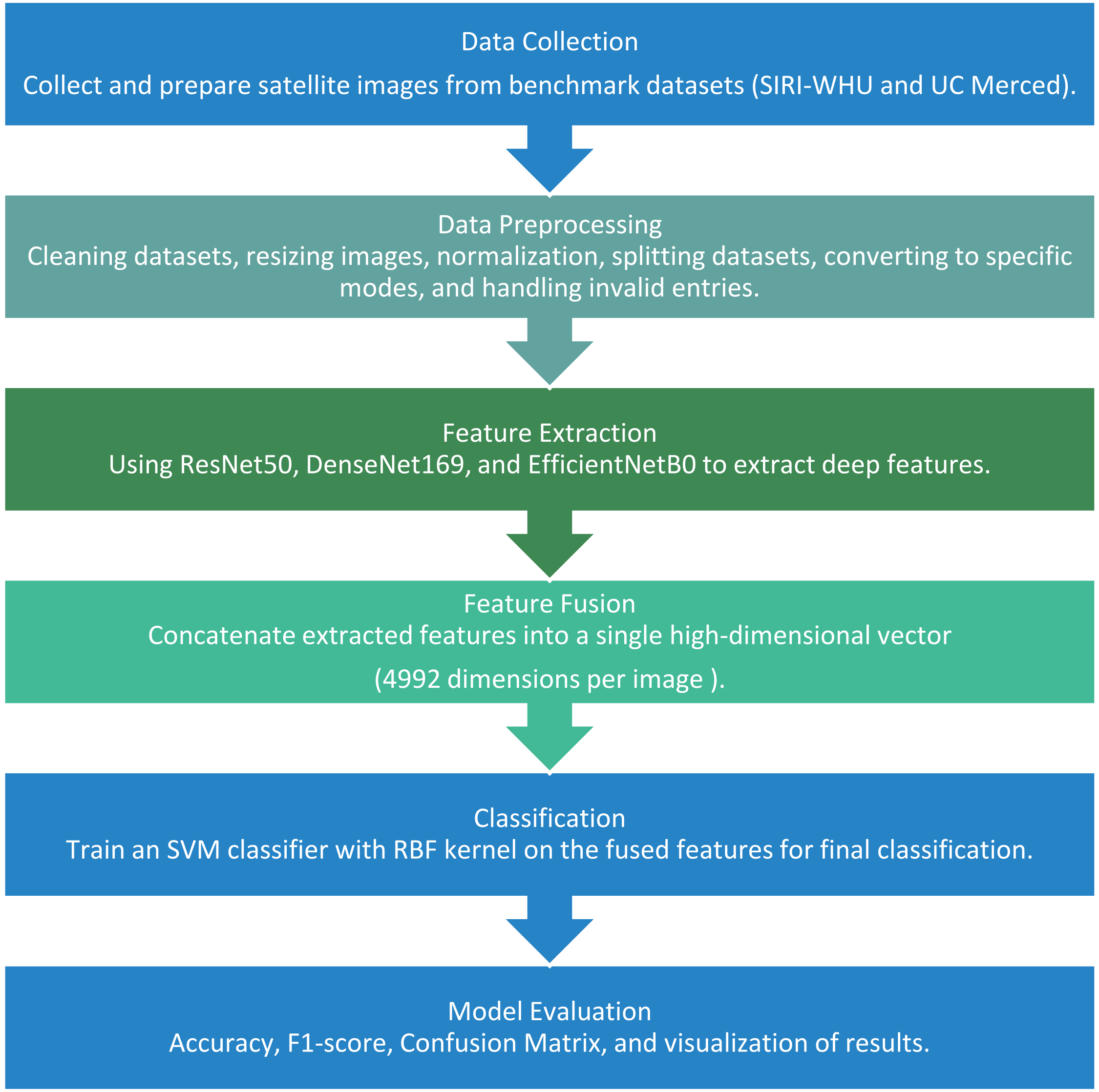


Figure 1: Framework Flow

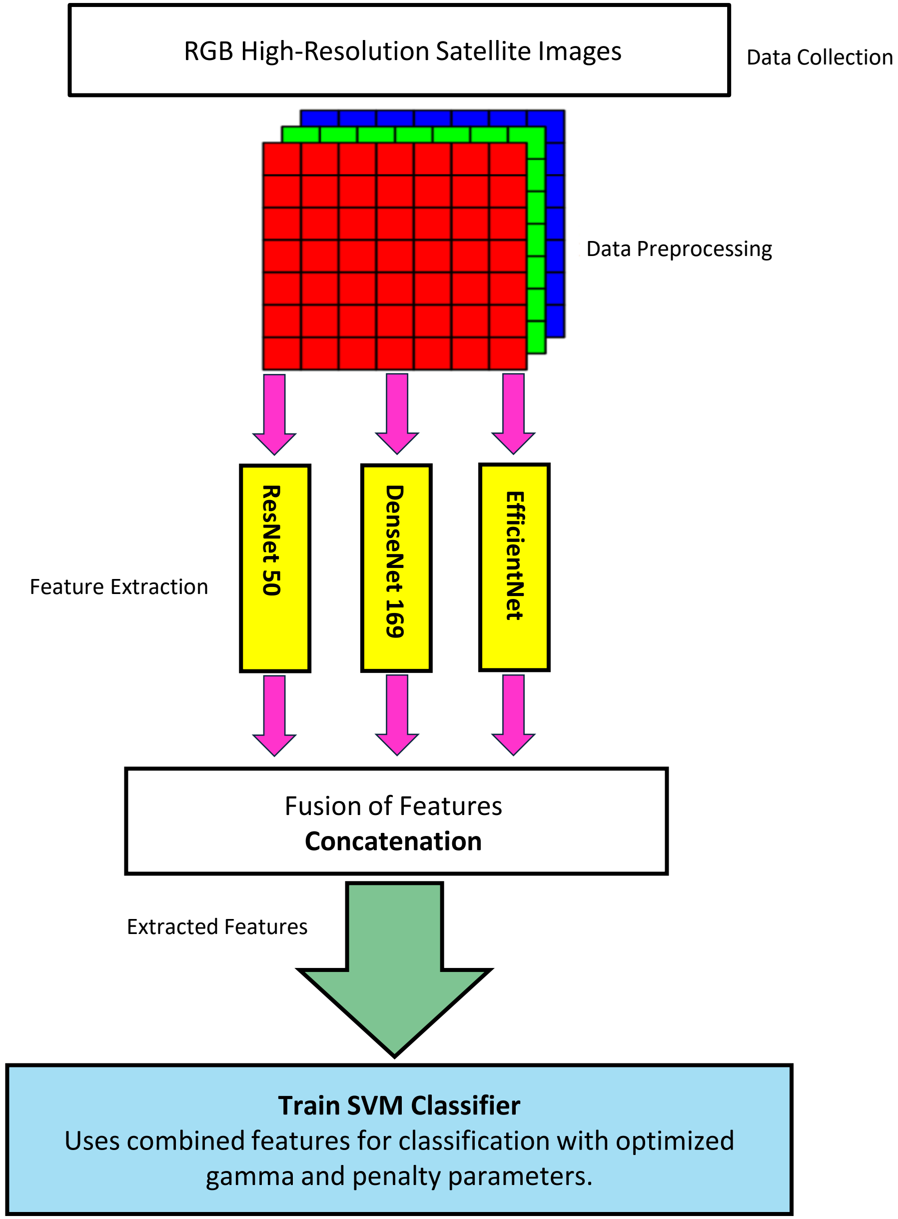


Figure 2: Architecture of TriDCCS-SVM

Structure

```
implementation-code-tridccs-svm/
|
|-- datasets/                # Input datasets (SIRI-WHU, UC Merced)
|    |-- siri_whu/
|    |    |-- agriculture/
|    |    |-- commercial/
|    |    +-- ...
|    +-- uc_merced/
|         |-- airplane/
|         |-- buildings/
|         +-- ...
|
|-- notebooks/               # Jupyter notebook with implementation
|    +-- tridccs_svm_yousef.ipynb
|   -- code-pdf /
|      +-- tridccs-svm-yousef-alsafadi.pdf
|   -- code-html /
|      +-- tridccs-svm-yousef-alsafadi.html
|
|-- outputs/
|    |-- models/             # Trained SVM models
|    |-- processed/          # Extracted and fused features
|    |-- results/            # Evaluation metrics and CSV reports
|    |-- visualizations
|
|| -- Implementation Progress Report/
|    +-- Presentation.pptx
|
|-- requirements.txt         # Python dependencies
+-- README.md                # Project documentation
```

Requirements

This implementation was developed and tested with:

- Python 3.12.7
- TensorFlow 2.19.0
- Keras
- scikit-learn
- OpenCV
- NumPy, Pandas
- Matplotlib, Seaborn
- tqdm
- joblib

System Requirements

This implementation can run on a standard desktop or laptop. For better performance during feature extraction and model training, the following specifications are recommended:

- Processor (CPU): Quad-core Intel i5 / AMD Ryzen 5 or higher
- RAM: Minimum 8 GB (Recommended: 16 GB or more)
- GPU (Optional): NVIDIA GPU with CUDA support (Recommended: 4 GB VRAM or higher, e.g., GTX 1650, RTX series)
- Disk Space: Minimum 5 GB free for datasets and outputs
- Operating System: Windows 10/11, Ubuntu 20.04+, or macOS 11+

> Note: GPU acceleration is optional but strongly recommended for faster CNN feature extraction. Without GPU, the process may take longer.

Datasets Overview

The approach was tested on two benchmark datasets:

1. SIRI-WHU Dataset

- Includes 2,400 images
- 12 land-cover classes (200 images per class)
- Each image size: 200×200 pixels.
- Example classes: agriculture, commercial, harbor, meadow, residential, water.

2. UC Merced Land Use Dataset

- Includes 2,100 images
- 21 land-use classes (100 images per class)
- Each image size: 256×256 pixels.
- Example classes: airplane, beach, forest, parking lot, golfcourse, freeway.

> Both datasets are split into 80% training and 20% testing, maintaining class balance.

SIRI-WHU Dataset – Sample Images

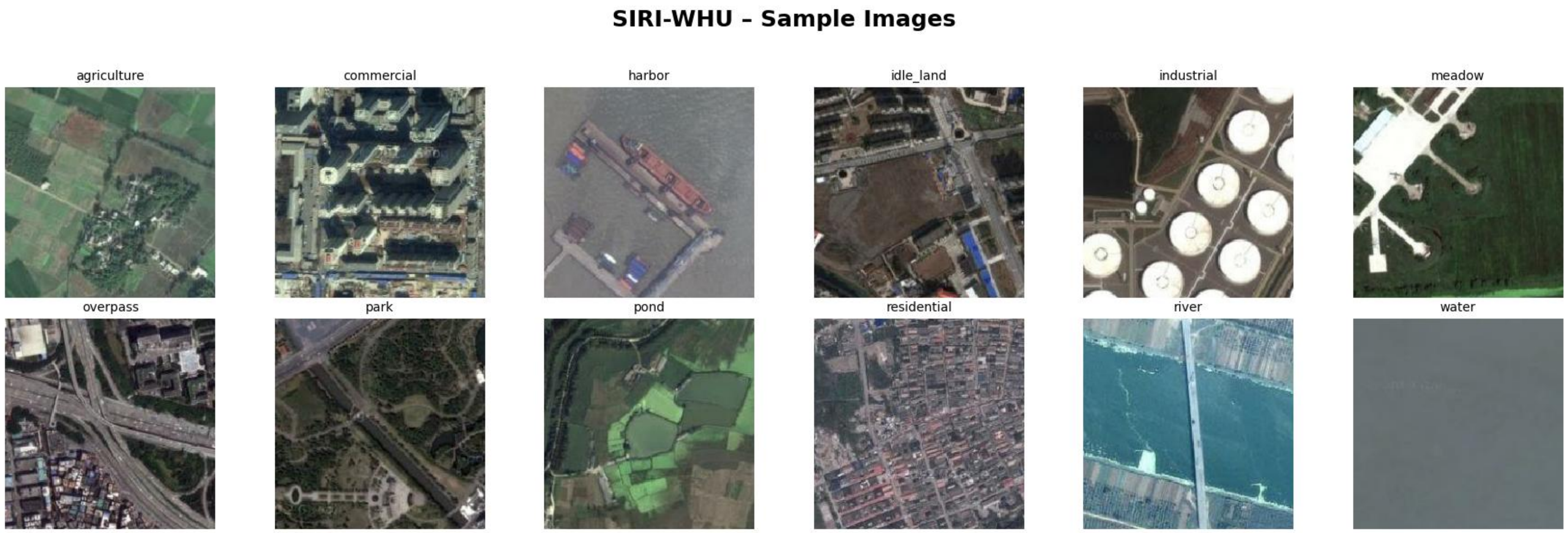
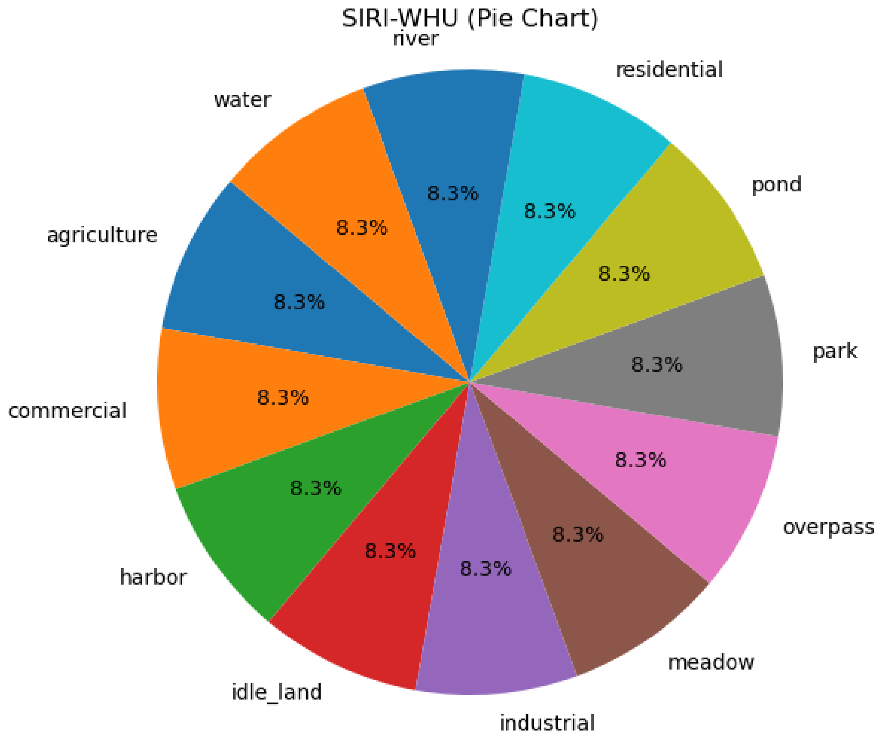


Figure 3: Sample images from the SIRI-WHU Dataset

SIRI-WHU Dataset- Pie Chart & Bar Chart



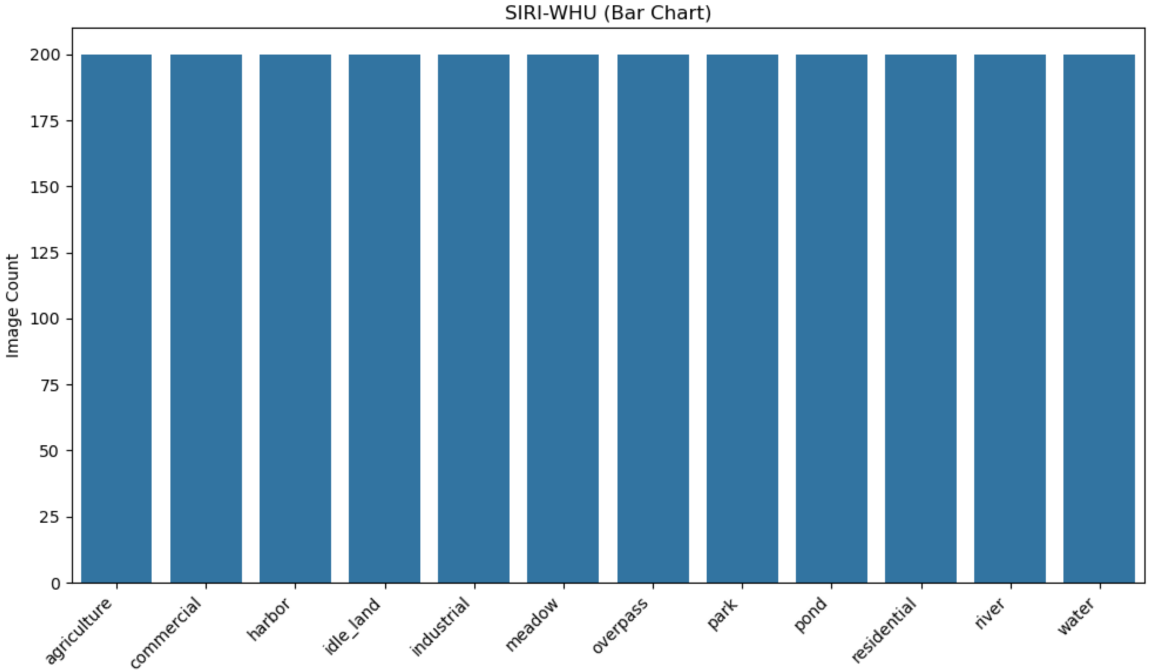
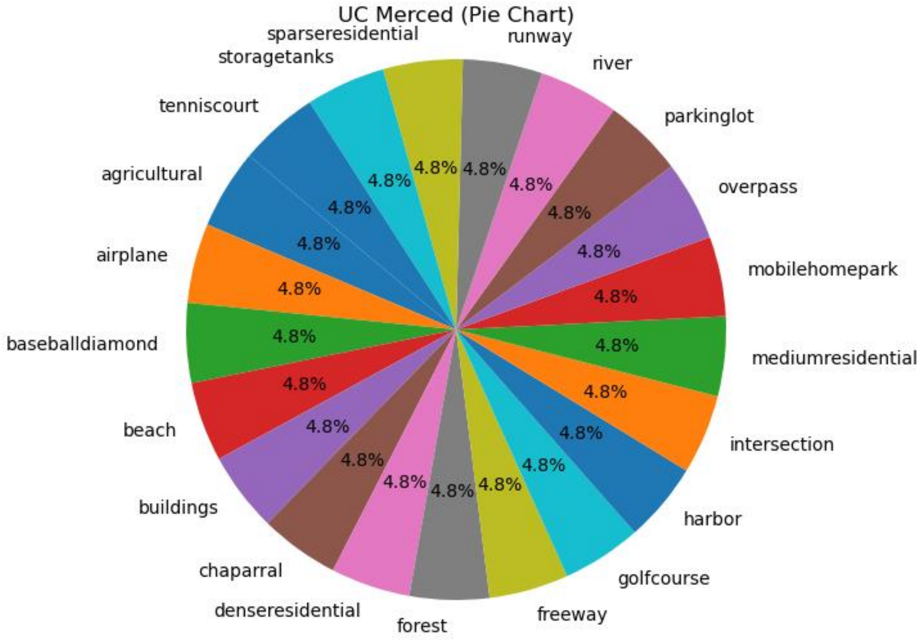


Figure 4: Class distribution of the SIRI-WHU dataset visualized using pie and bar charts

UC Merced Land Use Dataset - Sample Images

Figure 5: Sample images from the UC Merced Land Use Dataset

UC Merced Land Use Dataset - Pie Chart & Bar Chart



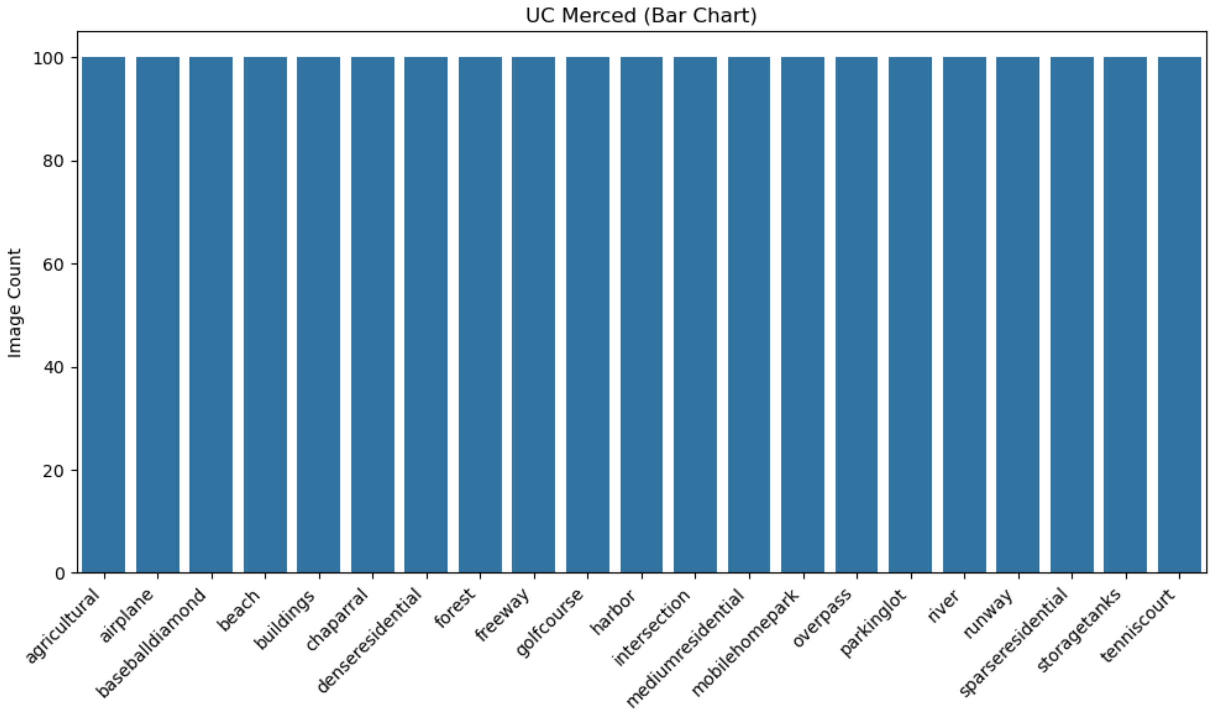


Figure 6: Class distribution of UC Merced Land Use Dataset visualized using pie and bar charts

Importing Required Libraries

- After Installing Python Packages, all necessary Python libraries are imported to support data handling, image processing, deep learning, machine learning, and visualization tasks.

Key Libraries Imported:

- Standard Libraries: os, sys, platform, subprocess, cpuinfo.
- Data Processing: numpy, pandas, joblib, psutil.
- Image Processing: cv2, PIL.
- Deep Learning: tensorflow, keras.
- Machine Learning: scikit-learn, xgboost.
- Visualization: matplotlib, seaborn.
- Utilities: tqdm, IPython.display.

> These libraries provide all the tools required for the implementation workflow.

USIM

Displaying System Specifications

```
System Specifications
----------------------------------------
[CPU Information]
Processor Name      : 12th Gen Intel(R) Core(TM) i7-1280P
Physical Cores      : 14
Logical Cores       : 20
Max Frequency (MHz) : 2000.00
Current Frequency   : 2000.00

[Memory (RAM)]
Total RAM (GB)      : 15.71
Available RAM (GB)  : 5.67
RAM Usage (%)       : 63.9%

[Operating System]
System              : Windows 11
Machine Architecture: AMD64

[GPU Information]
No NVIDIA GPU detected or 'nvidia-smi' not available.
```

> This output confirms that the system has adequate CPU and RAM resources for running the implementation.

Directory Setup

```python
# 1.6 – Define and Create Core Project Directories
# Set up the folder structure for datasets, outputs, models, etc.

# Root directory
PROJECT_ROOT = Path("C:/Users/User/Desktop/implementation/implementation-code-tridccs-svm")

# Define main subdirectories
DATASETS_DIR    = PROJECT_ROOT / "datasets"
OUTPUTS_DIR     = PROJECT_ROOT / "outputs"
MODELS_DIR      = OUTPUTS_DIR / "models"
RESULTS_DIR     = OUTPUTS_DIR / "results"
VISUALS_DIR     = OUTPUTS_DIR / "visualizations"
PROCESSED_DIR   = OUTPUTS_DIR / "processed"  # For preprocessed features

# Create directories if missing
for path in [DATASETS_DIR, OUTPUTS_DIR, MODELS_DIR, RESULTS_DIR, VISUALS_DIR, PROCESSED_DIR]:
    path.mkdir(parents=True, exist_ok=True)

print("Project folders created and ready.")
```

```
Project folders created and ready.
```

USIM

Dataset Loading

```python
# 2.1 - Load and resize images from dataset folders (one folder per class)

def load_dataset_images(dataset_path, target_size=(224, 224)):
    images = []
    labels = []
    class_names = sorted([d for d in os.listdir(dataset_path) if (dataset_path / d).is_dir()])
    label_map = {cls: idx for idx, cls in enumerate(class_names)}

    for cls in class_names:
        class_dir = dataset_path / cls
        for img_name in os.listdir(class_dir):
            img_path = class_dir / img_name
            try:
                img = load_img(img_path, target_size=target_size)
                img_array = img_to_array(img)
                images.append(img_array)
                labels.append(label_map[cls])
            except Exception as e:
                print(f"Warning: Failed to load {img_path} ({e})")

    return np.array(images), np.array(labels), class_names

# Load SIRI-WHU dataset
SIRI_DIR = DATASETS_DIR / "siri_whu"
X_siri, y_siri, class_names_siri = load_dataset_images(SIRI_DIR, target_size=(224, 224))

# Load UC Merced dataset
UC_DIR = DATASETS_DIR / "uc_merced"
X_uc, y_uc, class_names_uc = load_dataset_images(UC_DIR, target_size=(224, 224))

print("Datasets loaded successfully:")
```

Output:

Datasets loaded successfully:

SIRI-WHU dataset loaded: 2400 images, 12 classes   UC Merced dataset loaded: 2100 images, 21 classes

Checking for Corrupt and Blank Images

- Before training, datasets were validated for corrupt or blank images to ensure data quality.
- A custom function was implemented using OpenCV to scan each image:
  - Detects unreadable (corrupt) images.
  - Flags near-black images with negligible pixel intensity.
- Findings:
  - SIRI-WHU Dataset: No corrupt or blank images detected.
  - UC Merced Dataset: No corrupt or blank images detected.
  - Both datasets are clean and ready for training without data loss.

Splitting Datasets into Train/Test Sets

- Datasets were split into 80% training and 20% testing, maintaining class balance (stratified split).
- Random seeds ensured reproducibility:
  - SIRI-WHU → random_state=20
  - UC Merced → random_state=42
- Results: SIRI-WHU Split:
  - Training: 1,920 images (12 classes).
  - Testing: 480 images (12 classes).
- Results: UC Merced Split:
  - Training: 1,680 images (21 classes).
  - Testing: 420 images (21 classes).

> *Class Distribution Example (SIRI-WHU – Train):*

agriculture : 160 samples
commercial : 160 samples
...
water : 160 samples

Loading Pretrained CNN Models

- Three pretrained CNN architectures were loaded for feature extraction:
  - ResNet50: 2048-dimensional features.
  - DenseNet169: 1664-dimensional features.
  - EfficientNetB0: 1280-dimensional features.
- All models were loaded without their top layers and configured with Global Average Pooling to produce compact feature vectors.

> Output Shapes (after pooling):

ResNet50 → (None, 2048)
DenseNet169 → (None, 1664)
EfficientNetB0 → (None, 1280)

✓ *This dimensionality was later fused for classification*

USIM

CNN Feature Summary and Fusion

- Feature Dimensions Table:

Model	Output Dimension	Cumulative Dimension
ResNet50	2048	2048
DenseNet169	1664	3712
EfficientNetB0	1280	4992

- Features from all three networks are concatenated to form a single representation of size 4992.

USIM

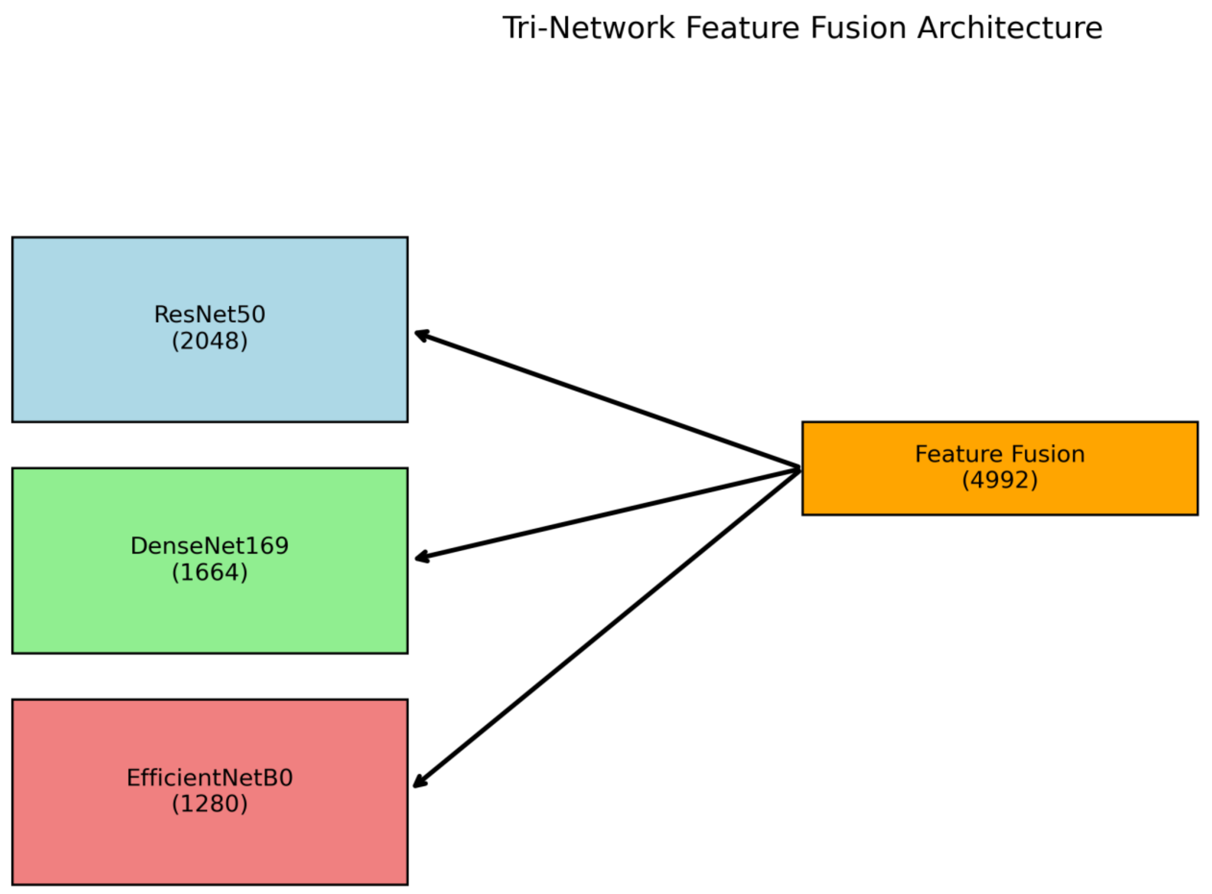


Figure 7: A fusion diagram illustrates how features from ResNet, DenseNet, and EfficientNet are combined into a unified vector.

Preprocessing & Feature Extraction Overview

- Preprocessing Notes:
  - ResNet50: Converts RGB to BGR and subtracts mean pixel values.
  - DenseNet169: Normalizes pixel values to [0, 1] range.
  - EfficientNetB0: Scales pixel values to [-1, 1] range.
  - These steps are applied automatically during feature extraction inside extract_features().

- Key Point: The preprocessing ensures each CNN receives data in the optimal format for extracting meaningful deep features.

USIM

Feature Extraction and Scaling for SIRI-WHU & UC Merced

- Features were extracted from each pretrained CNN (ResNet50, DenseNet169, EfficientNetB0):

- SIRI-WHU:
  - Training Set: 1,920 samples
  - Testing Set: 480 samples
- UC Merced:
  - Training Set: 1,680 samples
  - Testing Set: 420 samples
- Feature Dimensions:
  - ResNet50 → 2048
  - DenseNet169 → 1664
  - EfficientNetB0 → 1280

- Scaling Applied: StandardScaler (ensures zero-mean and unit-variance features) used to normalize features for improved SVM classification performance.

> Result: All CNN outputs were successfully extracted and scaled for both datasets.

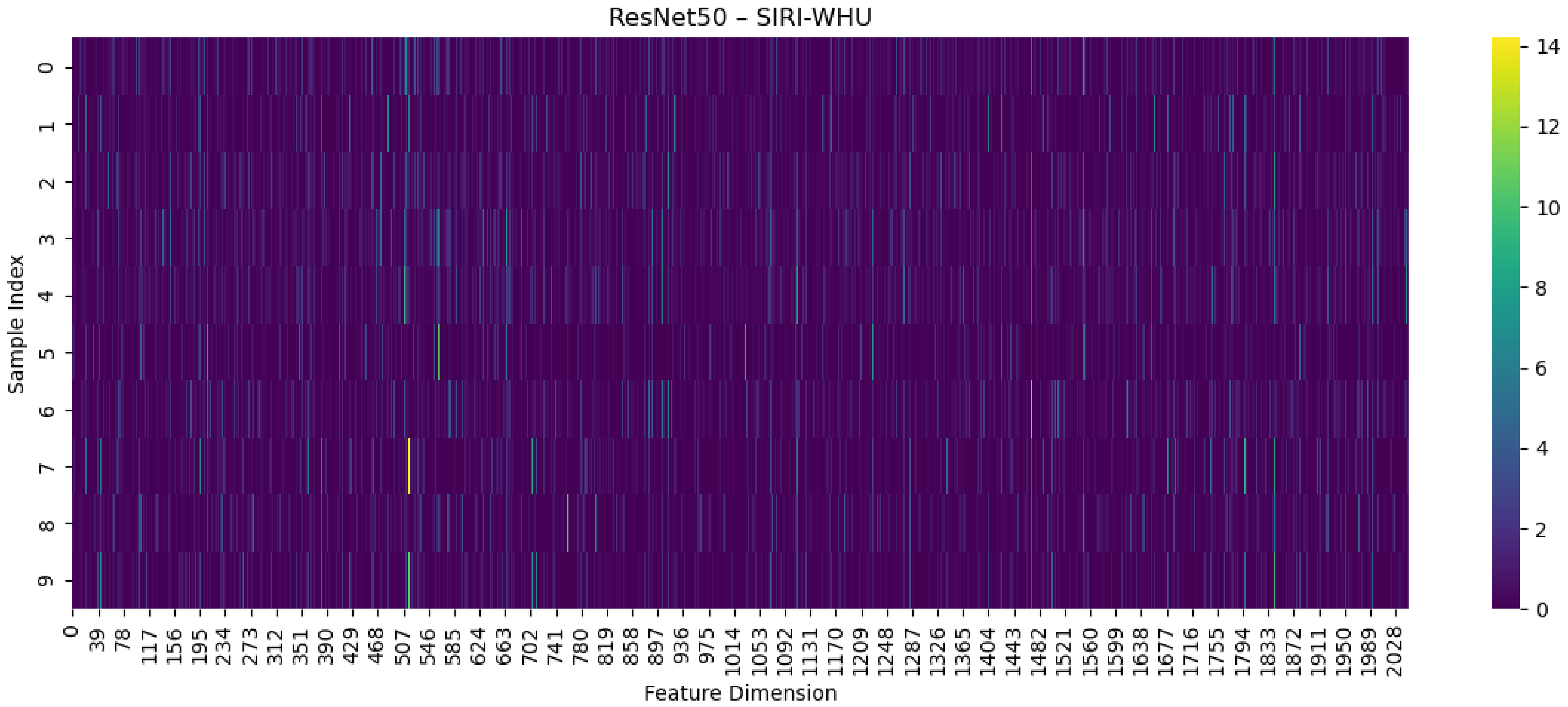
Correlation Between CNN Feature Sets

- Purpose of Analysis: Evaluate the degree of similarity between feature representations extracted from different CNNs.
- Lower correlation → more complementary features → better fusion results.
- Correlation Results (SIRI-WHU):
  - ResNet50 vs DenseNet169: 0.0132
  - ResNet50 vs EfficientNetB0: 0.0291
  - DenseNet169 vs EfficientNetB0: 0.0122
- Correlation Results (UC Merced):
  - ResNet50 vs DenseNet169: 0.0237
  - ResNet50 vs EfficientNetB0: 0.0392
  - DenseNet169 vs EfficientNetB0: 0.0215

✓ Key Insight: low correlation values confirm that each CNN learns unique and complementary features.

USIM

Heatmaps of CNN Feature Representations

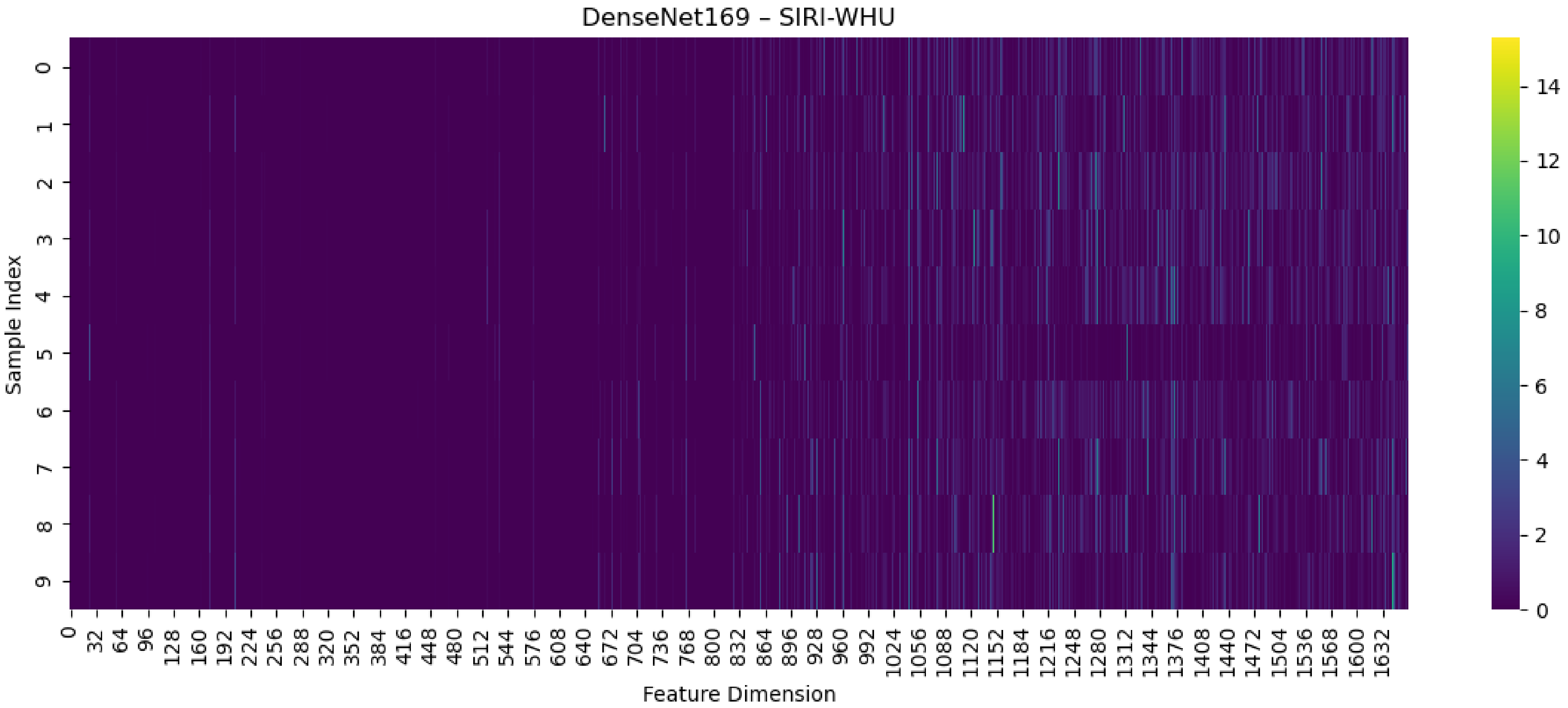


Key Observation:

- Heatmaps provide a visual summary of feature activations across all samples and dimensions.
- Individual CNN heatmaps show distinct feature activations.
- Fused feature heatmaps combine these activations into a comprehensive representation.

> Outputs: Heatmaps saved under : outputs/visualizations/

Heatmaps of CNN Feature Representations

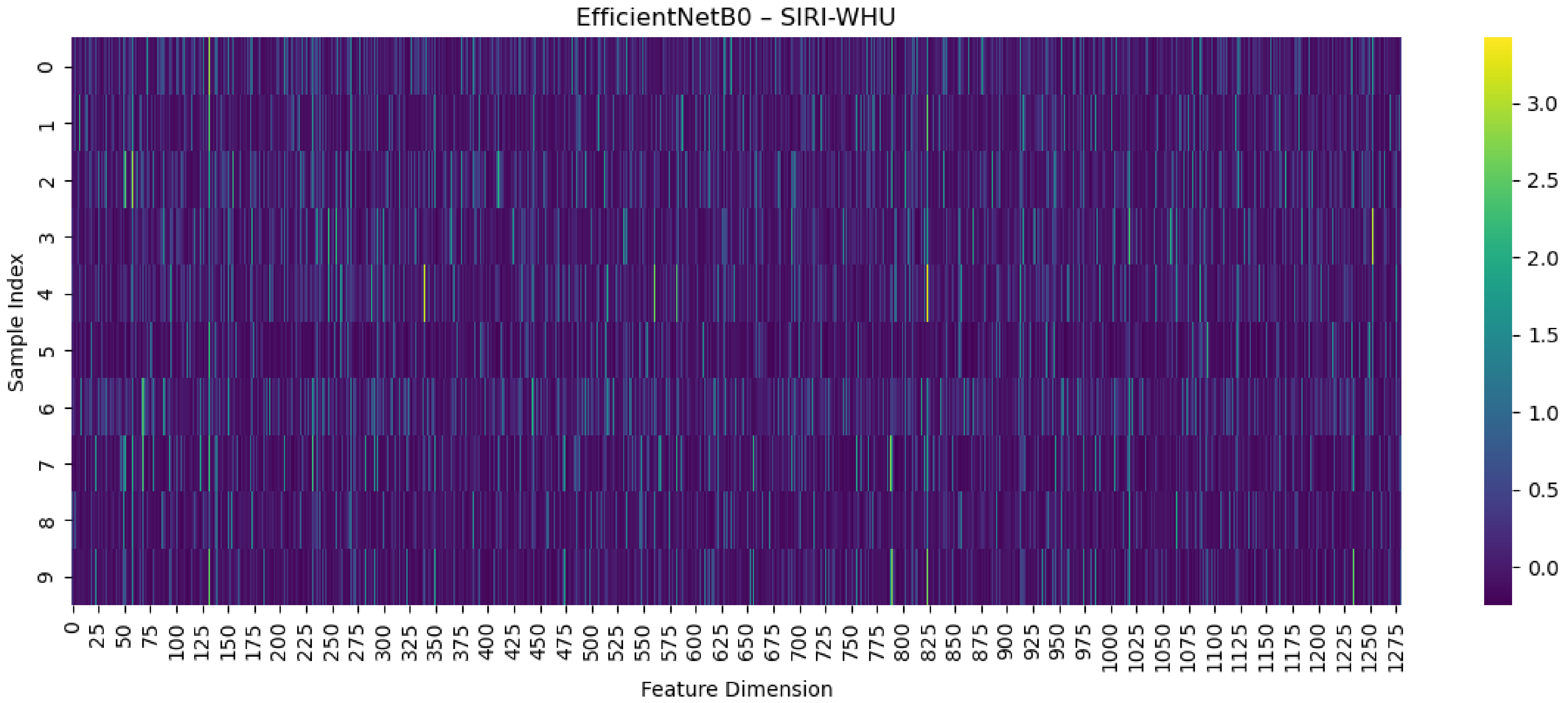


Key Observation:

- Individual CNN heatmaps show distinct feature activations.
- Fused feature heatmaps combine these activations into a comprehensive representation.

> Outputs: Heatmaps saved under : outputs/visualizations/

Heatmaps of CNN Feature Representations

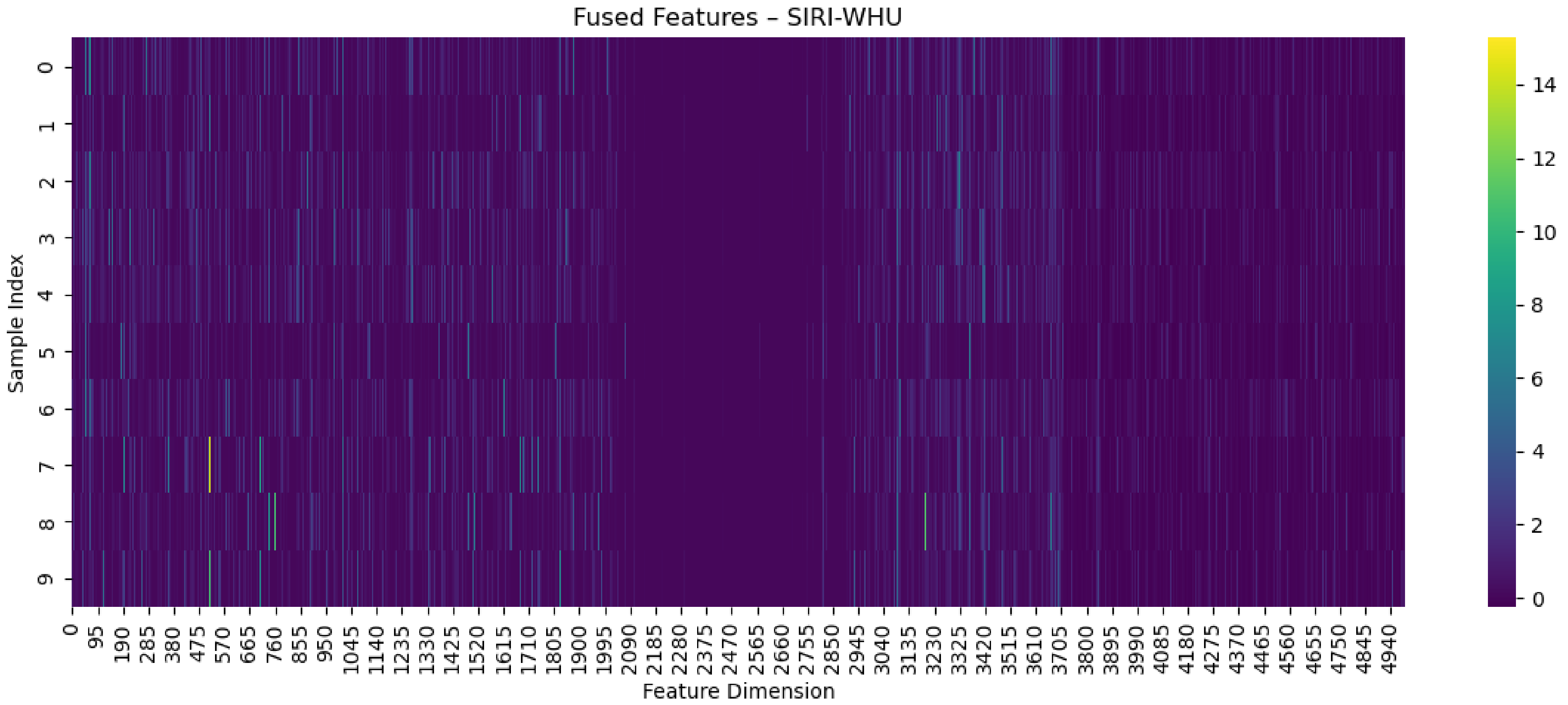


Key Observation:

- Individual CNN heatmaps show distinct feature activations.
- Fused feature heatmaps combine these activations into a comprehensive representation.

> Outputs: Heatmaps saved under : outputs/visualizations/

Heatmaps of CNN Feature Representations



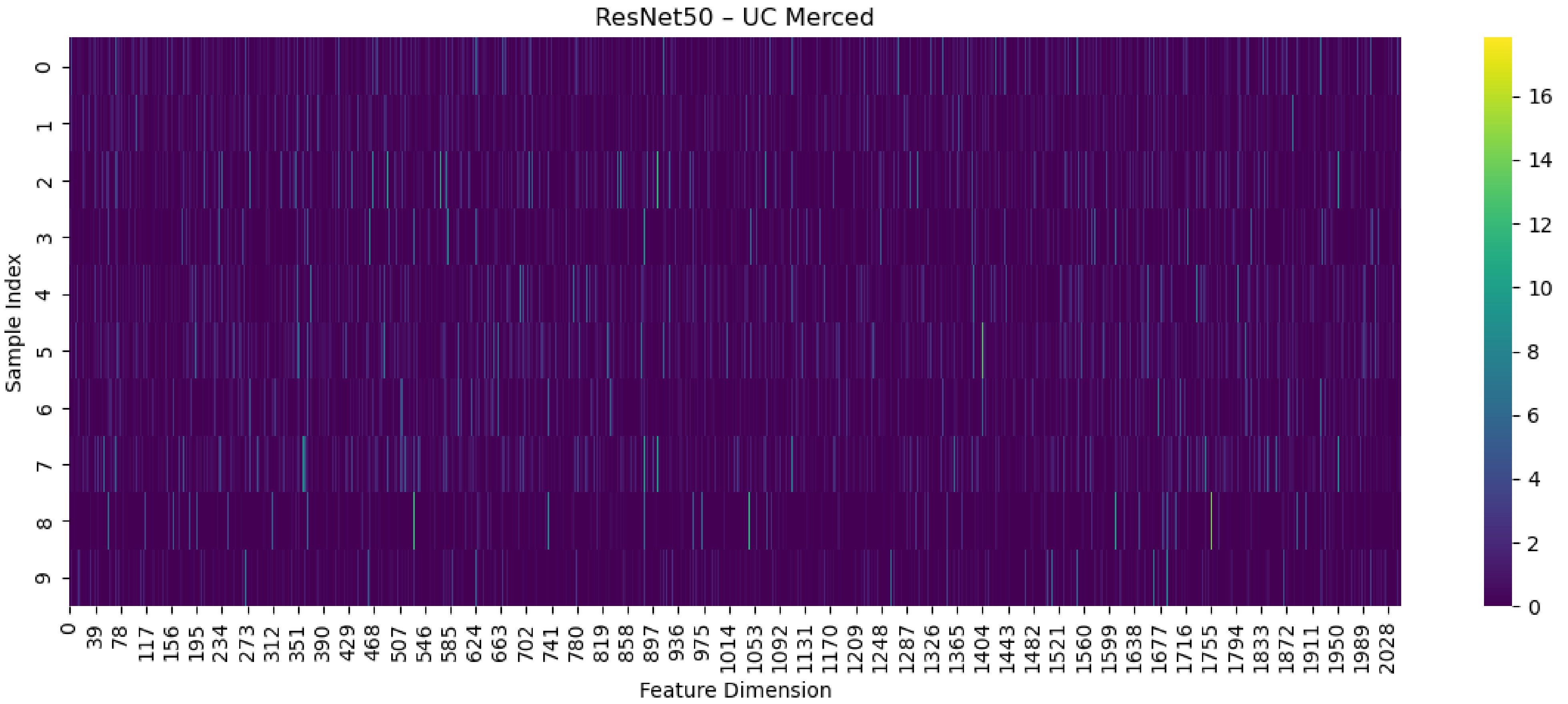
Key Observation:

- Individual CNN heatmaps show distinct feature activations.
- Fused feature heatmaps combine these activations into a comprehensive representation.

> Outputs: Heatmaps saved under : outputs/visualizations/

USIM

Heatmaps of CNN Feature Representations

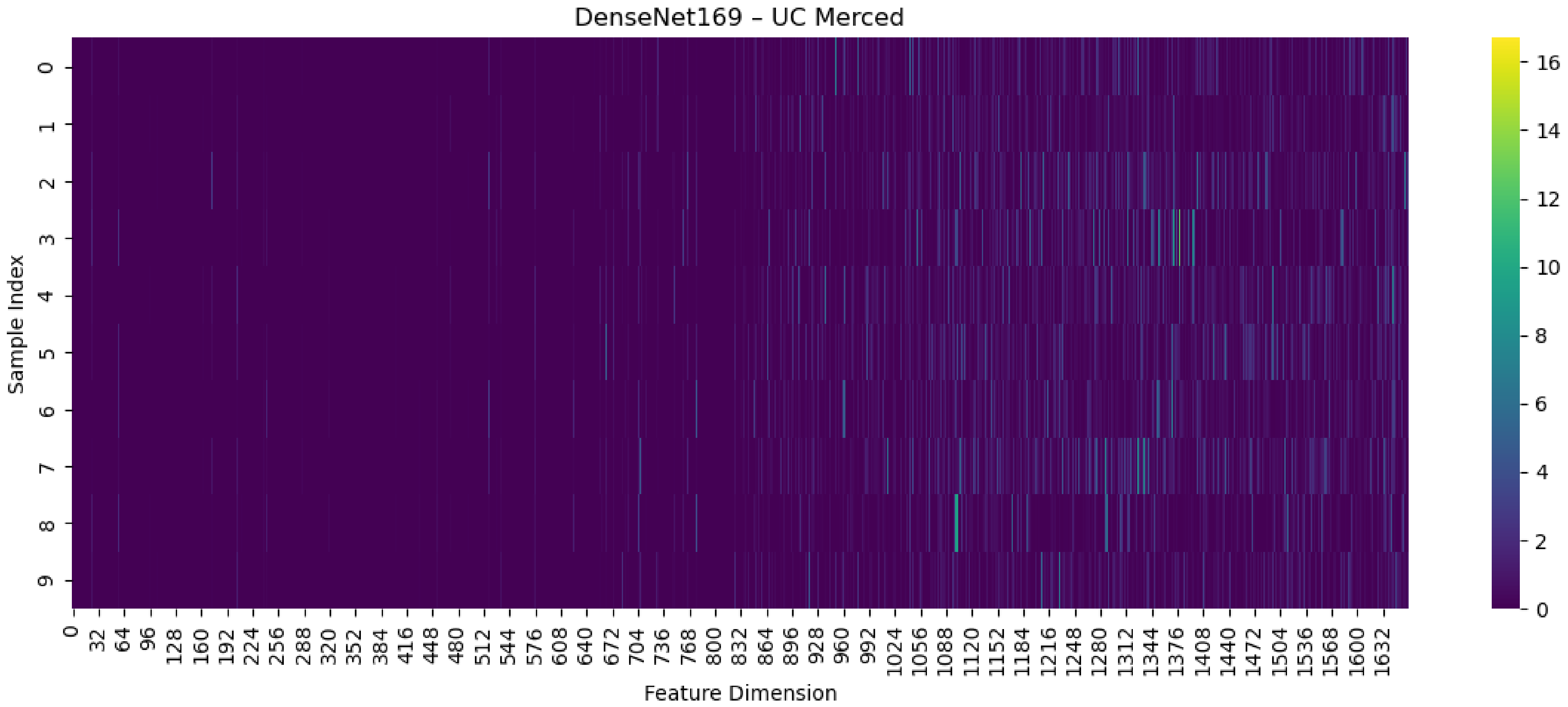


Key Observation:

- Individual CNN heatmaps show distinct feature activations.
- Fused feature heatmaps combine these activations into a comprehensive representation.

> Outputs: Heatmaps saved under : outputs/visualizations/

Heatmaps of CNN Feature Representations

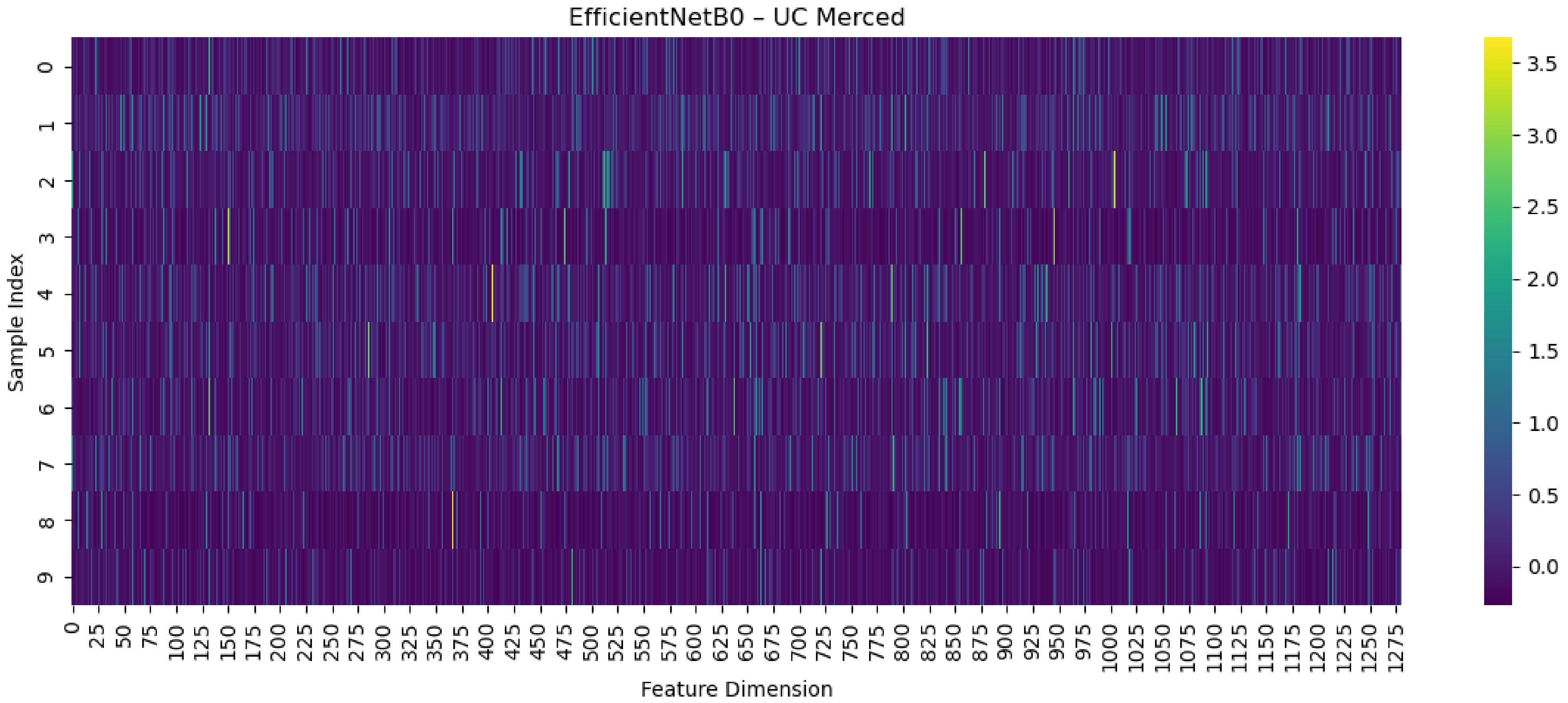


Key Observation:

- Individual CNN heatmaps show distinct feature activations.
- Fused feature heatmaps combine these activations into a comprehensive representation.

> Outputs: Heatmaps saved under : outputs/visualizations/

Heatmaps of CNN Feature Representations

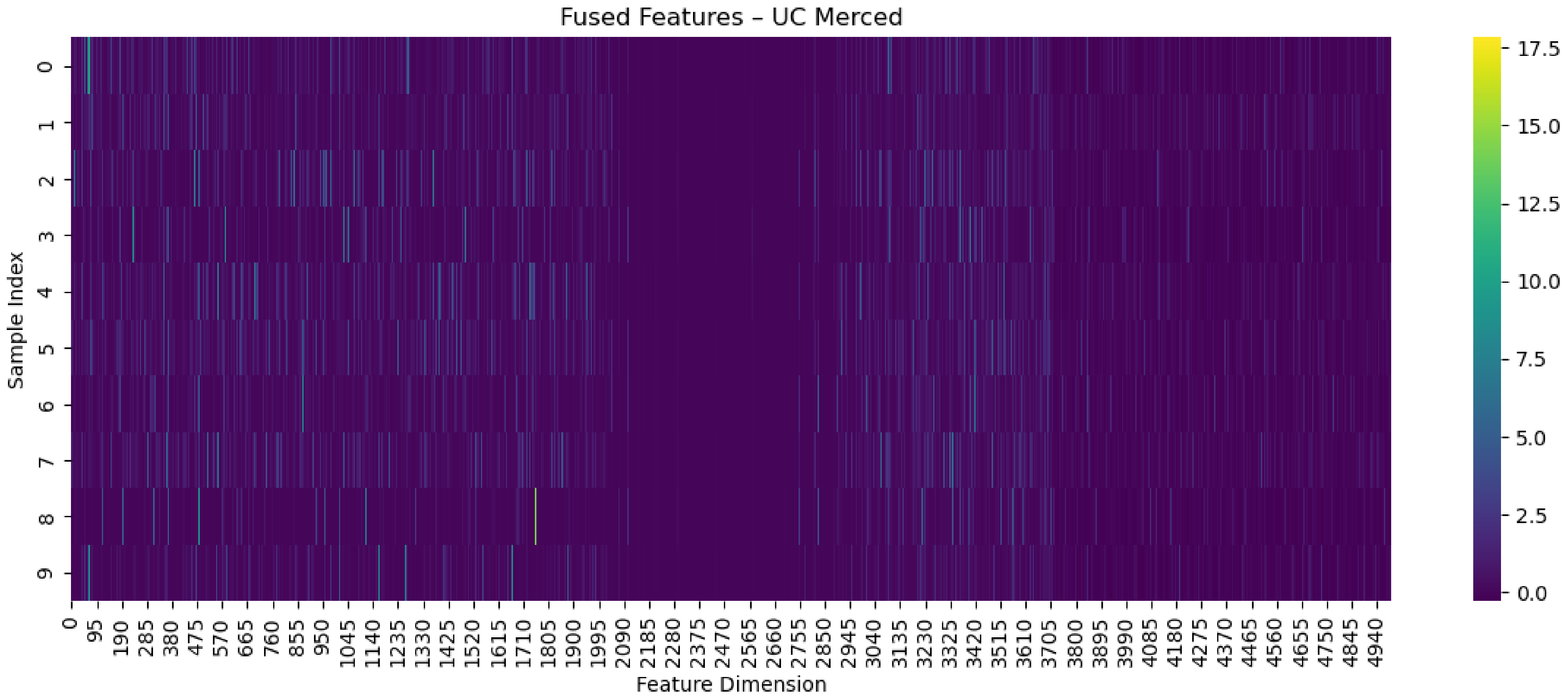


Key Observation:

- Individual CNN heatmaps show distinct feature activations.
- Fused feature heatmaps combine these activations into a comprehensive representation.

> Outputs: Heatmaps saved under : outputs/visualizations/

Heatmaps of CNN Feature Representations

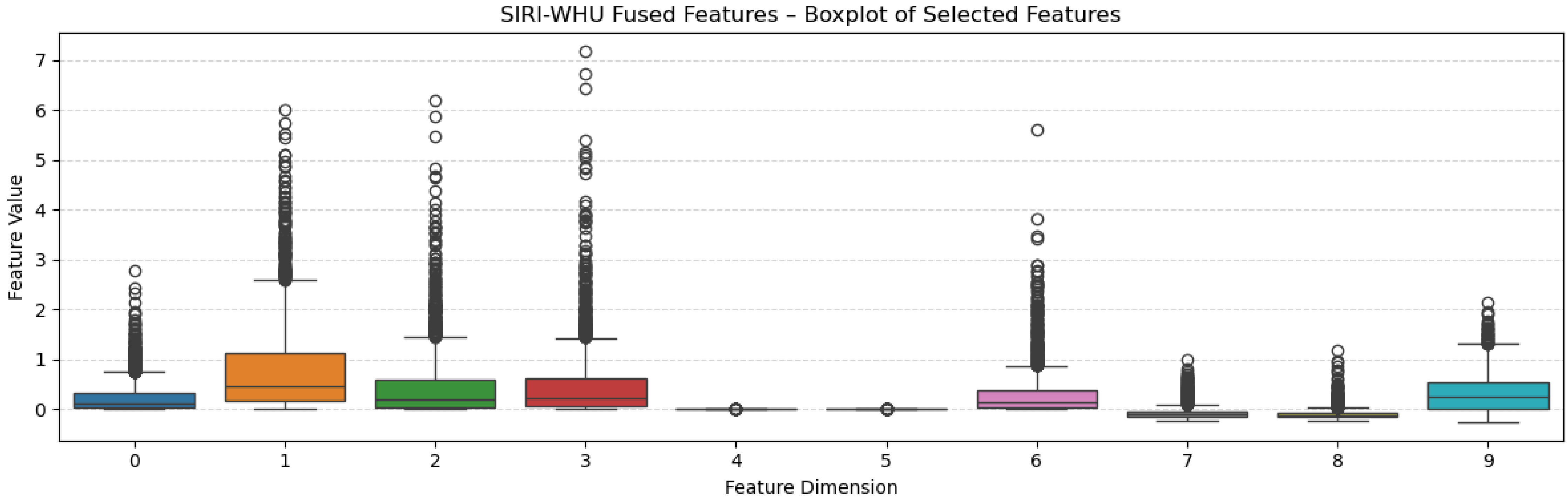


Key Observation:

- Individual CNN heatmaps show distinct feature activations.
- Fused feature heatmaps combine these activations into a comprehensive representation.

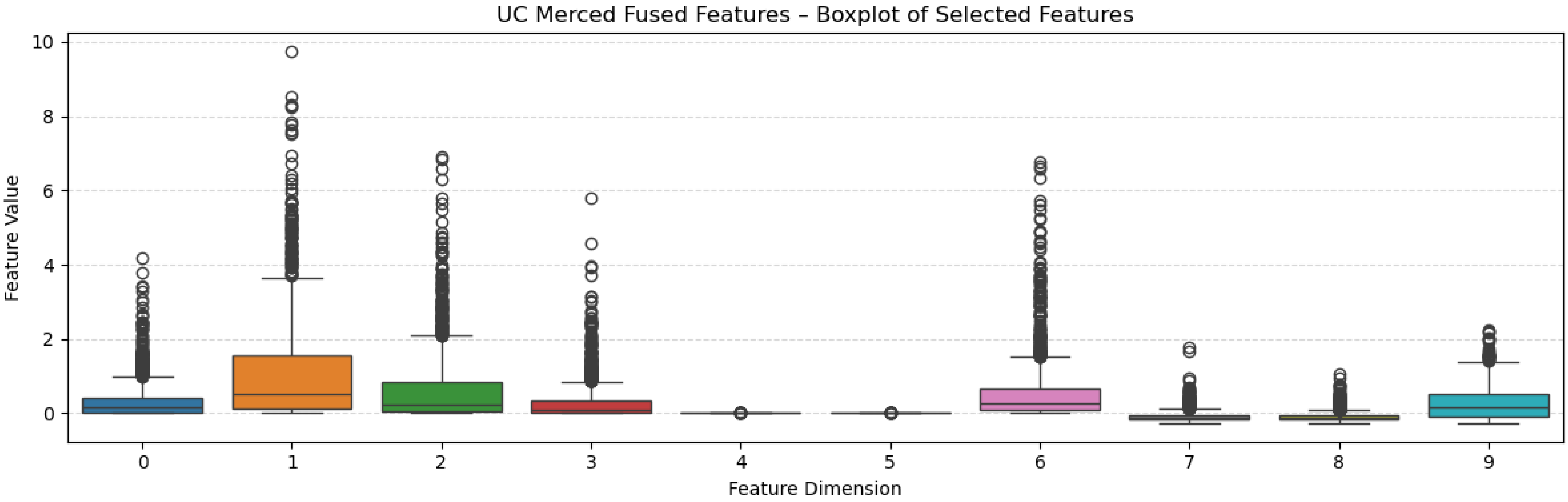
> Outputs: Heatmaps saved under : outputs/visualizations/

Fused Feature Distribution – Boxplots



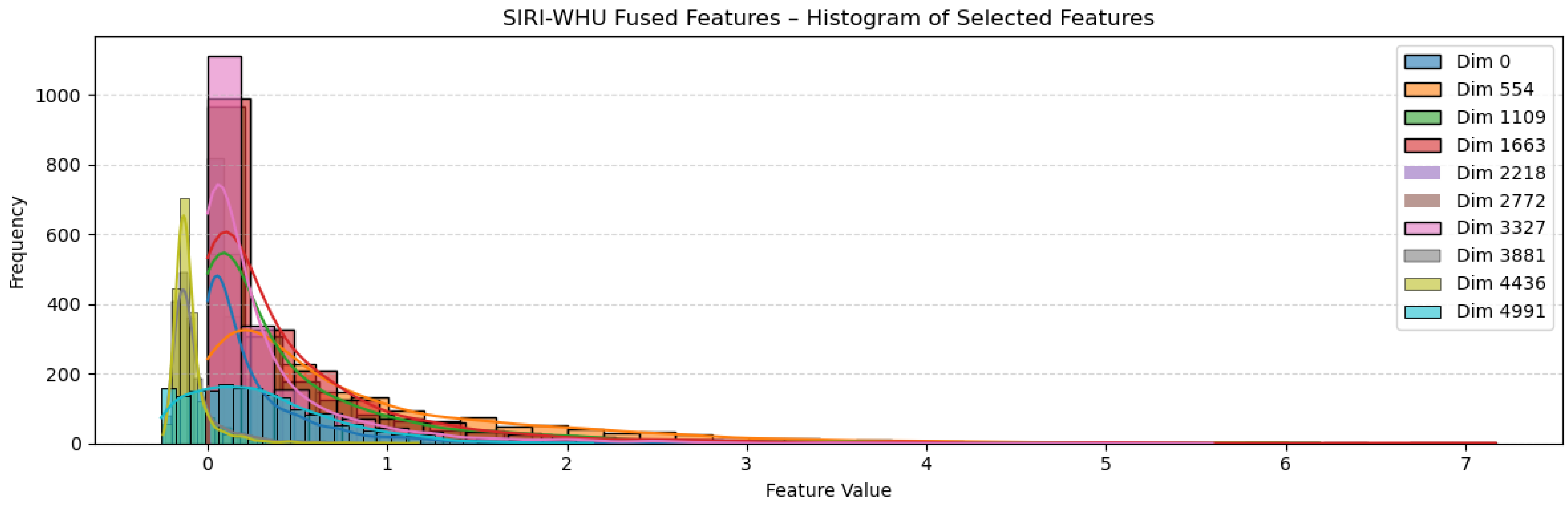
- Visualize how fused features are distributed across selected dimensions to check for outliers and variability.
- Boxplots show variations in feature values and help identify dimensions with potential outliers.
- Boxplot saved under: outputs/visualizations/

Fused Feature Distribution – Boxplots



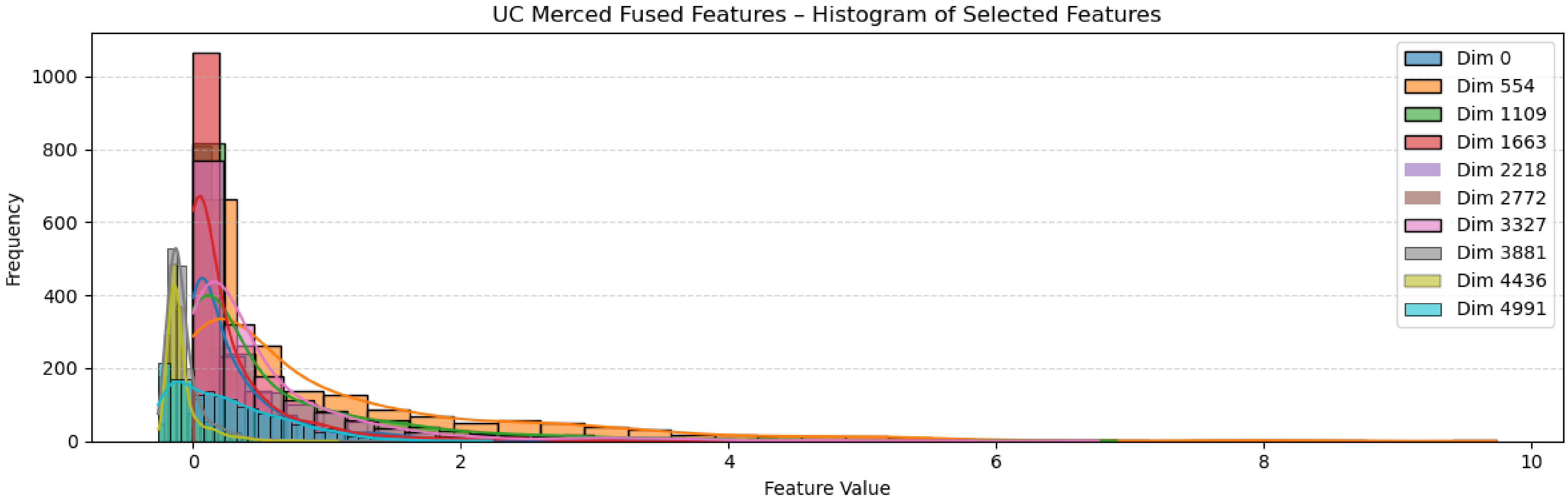
- Visualize how fused features are distributed across selected dimensions to check for outliers and variability.
- Boxplots show variations in feature values and help identify dimensions with potential outliers.
- Boxplot saved under: outputs/visualizations/

Fused Feature Distribution – Histograms



- ❖ Provide frequency distribution of selected feature dimensions to understand data spread and density.
- ❖ Histograms display normalized feature values and highlight dominant value ranges.
- ❖ Boxplot saved under: outputs/visualizations/

Fused Feature Distribution – Histograms



- ❖ Provide frequency distribution of selected feature dimensions to understand data spread and density.
- ❖ Histograms display normalized feature values and highlight dominant value ranges.
- ❖ Boxplot saved under: outputs/visualizations/

Saving Extracted Features for Reuse

- Store extracted features from CNNs and their corresponding labels to avoid recomputation.
- Saved ResNet50, DenseNet169, and EfficientNetB0 features for both training and testing sets.
- Saved separately for SIRI-WHU and UC Merced datasets.
- Format: .pkl files using joblib.

Example Saved Files:

resnet_train.pkl → shape: (1920, 2048)
densenet_test.pkl → shape: (480, 1664)
efficientnet_train.pkl → shape: (1920, 1280)

USIM

Training and Evaluating SVM Classifier on SIRI-WHU Dataset

- Training Support Vector Machine (SVM) on fused CNN features to classify SIRI-WHU dataset.

- ✓ Training Details:
  1. Kernel: RBF
  2. Hyperparameters: C=10, gamma="scale"
- ✓ Performance:
  1. Overall Accuracy: 96.25%
  2. Weighted F1-Score: 96.25%
- ✓ Outputs:
  1. Trained SVM Model
  2. Predictions on Test Set
  3. Evaluation Metrics

SIRI-WHU – Confusion Matrix

- Visual heatmap of predicted vs actual classes.

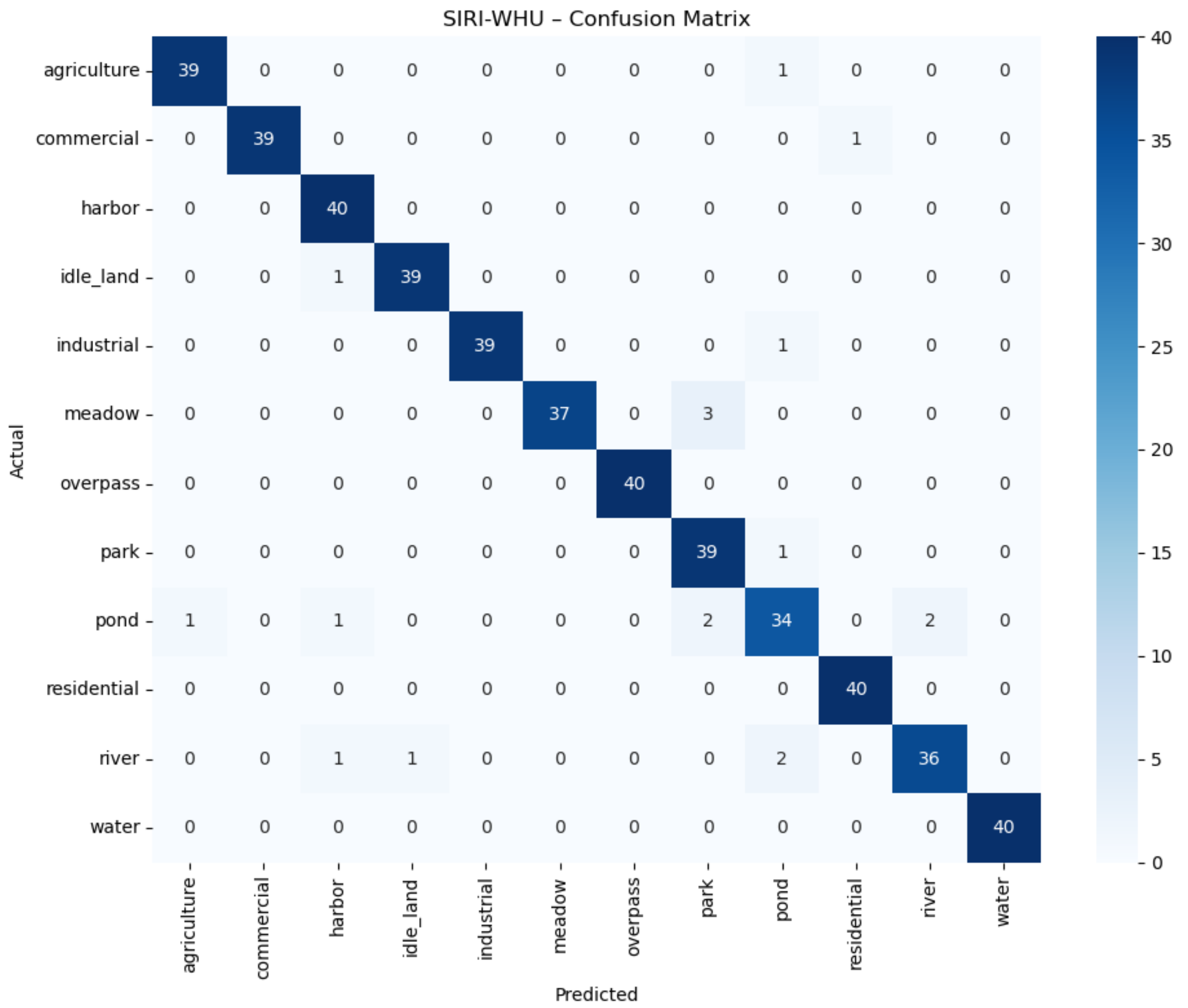


Figure : SIRI-WHU Confusion Matrix (12 Classes)

Training and Evaluating SVM Classifier on UC Merced Dataset

- Train Support Vector Machine (SVM) on fused CNN features for UC Merced dataset.

- ✓ Training Details:
  1. Kernel: RBF
  2. Hyperparameters: C=10, gamma="scale"
- ✓ Performance:
  1. Overall Accuracy: 97.86%
  2. Weighted F1-Score: 97.85%
- ✓ Outputs:
  1. Trained SVM Model
  2. Predictions on Test Set
  3. Evaluation Metrics

UC Merced – Confusion Matrix

- Visual heatmap of predicted vs actual classes.

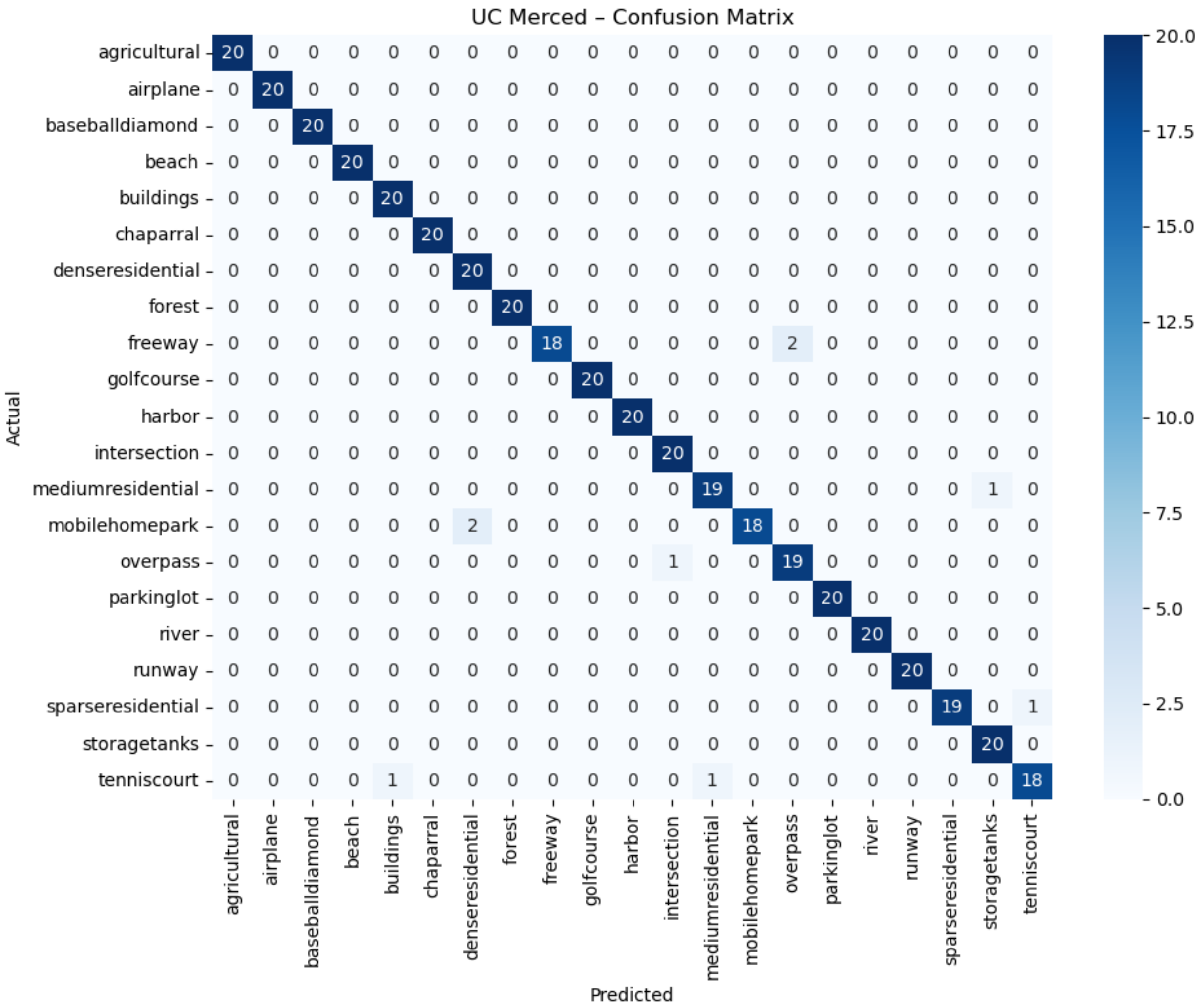


Figure : UC Merced Confusion Matrix (21 Classes)

Saving Trained Models for Future Use

- ✓ Saved Models:
  1. SVM Model (SIRI-WHU) → svm_model_siri_whu.pkl
  2. SVM Model (UC Merced) → svm_model_uc_merced.pkl
- ✓ Purpose of Saving:
  1. Allow reuse without retraining
  2. Support deployment or additional analysis

> All models saved under: outputs/models/svm/

USIM

Final Test Accuracy: SIRI-WHU vs UC Merced

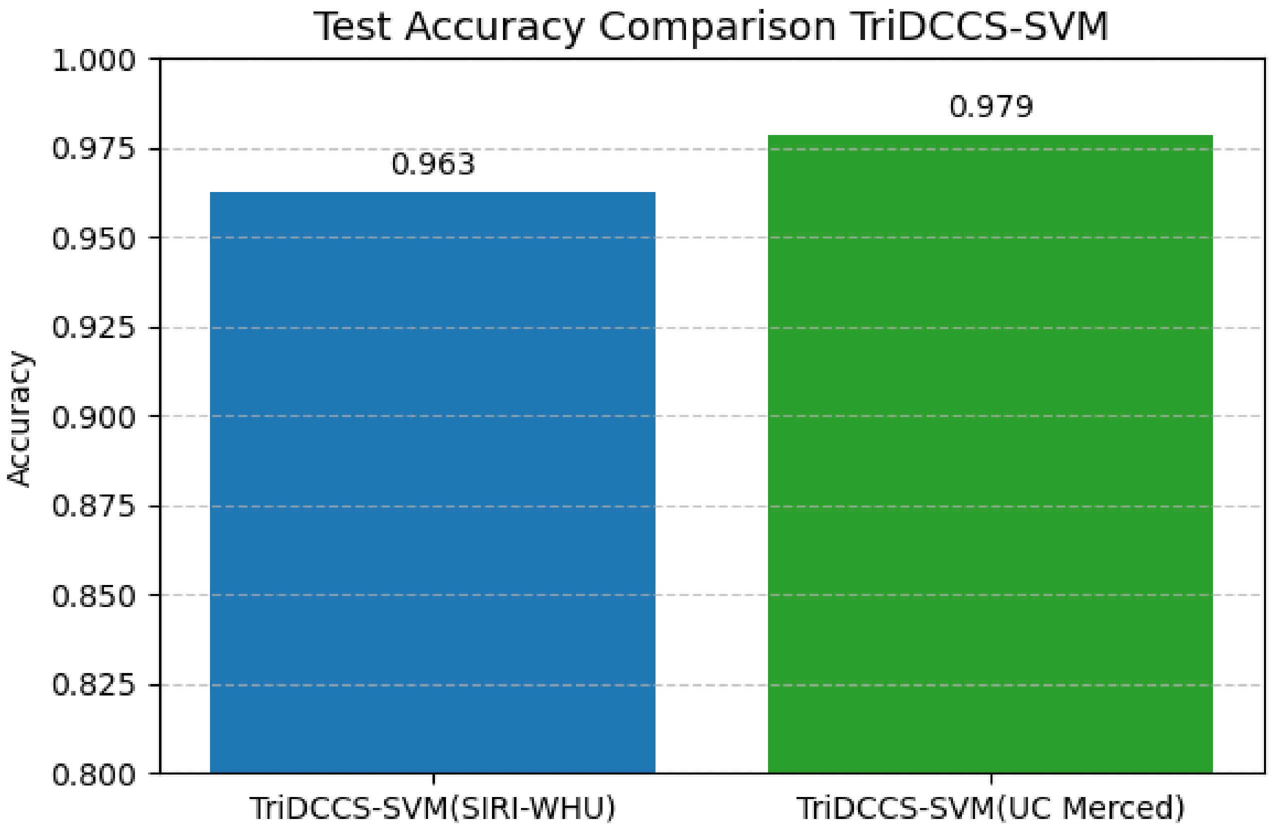


Figure : Test Accuracy: SIRI-WHU vs UC Merced

Detailed Per-Class Evaluation: SIRI-WHU

- ✓ Metrics Computed for Each Class: Accuracy, Precision, Recall, F1-Score
- ✓ Key Insights:
  1. Most classes achieved ≥ 95% accuracy.
  2. Minor variations in Pond and River classes due to intra-class variability.

	Class	Accuracy	Precision	Recall	F1-Score
0	agriculture	0.975	0.9750	0.975	0.9750
1	commercial	0.975	1.0000	0.975	0.9873
2	harbor	1.000	0.9302	1.000	0.9639
3	idle_land	0.975	0.9750	0.975	0.9750
4	industrial	0.975	1.0000	0.975	0.9873
5	meadow	0.925	1.0000	0.925	0.9610
6	overpass	1.000	1.0000	1.000	1.0000
7	park	0.975	0.8864	0.975	0.9286
8	pond	0.850	0.8718	0.850	0.8608
9	residential	1.000	0.9756	1.000	0.9877
10	river	0.900	0.9474	0.900	0.9231
11	water	1.000	1.0000	1.000	1.0000

Table : Detailed Per-Class Evaluation

Detailed Per-Class Evaluation: SIRI-WHU

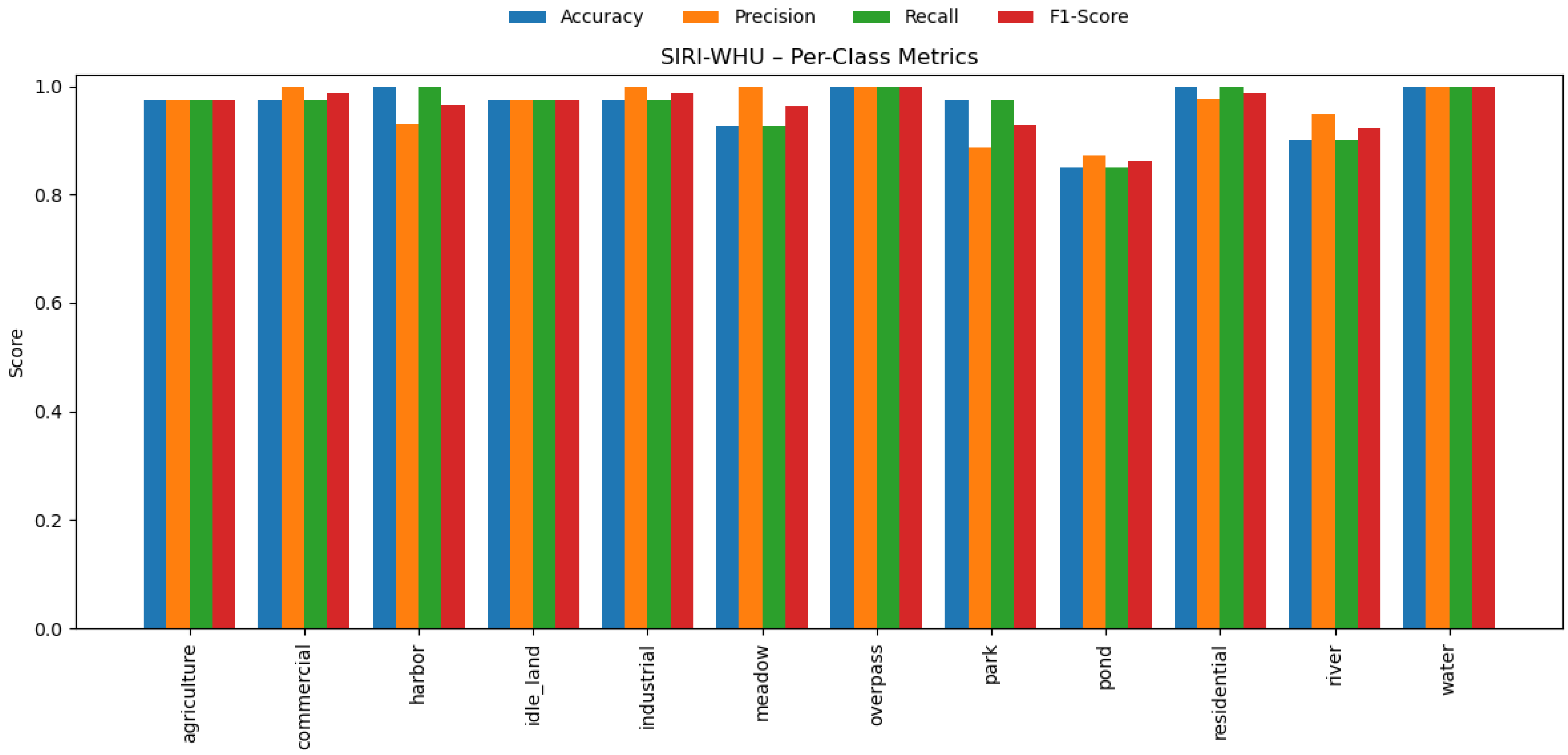


Figure : SIRI-WHU – Per-Class Metrics

Detailed Per-Class Evaluation: UC Merced

- ✓ Metrics Computed for Each Class: Accuracy, Precision, Recall, F1-Score
- ✓ Slight drops in freeway, tenniscourt, and mobilehomepark.

Class	Accuracy	Precision	Recall	F1-Score	
0	agricultural	1.00	1.0000	1.00	1.0000
1	airplane	1.00	1.0000	1.00	1.0000
2	baseballdiamond	1.00	1.0000	1.00	1.0000
3	beach	1.00	1.0000	1.00	1.0000
4	buildings	1.00	0.9524	1.00	0.9756
5	chaparral	1.00	1.0000	1.00	1.0000
6	denseresidential	1.00	0.9091	1.00	0.9524
7	forest	1.00	1.0000	1.00	1.0000
8	freeway	0.90	1.0000	0.90	0.9474
9	golfcourse	1.00	1.0000	1.00	1.0000
10	harbor	1.00	1.0000	1.00	1.0000
11	intersection	1.00	0.9524	1.00	0.9756
12	mediumresidential	0.95	0.9500	0.95	0.9500
13	mobilehomepark	0.90	1.0000	0.90	0.9474
14	overpass	0.95	0.9048	0.95	0.9268
15	parkinglot	1.00	1.0000	1.00	1.0000
16	river	1.00	1.0000	1.00	1.0000
17	runway	1.00	1.0000	1.00	1.0000
18	sparseresidential	0.95	1.0000	0.95	0.9744
19	storagetanks	1.00	0.9524	1.00	0.9756
20	tenniscourt	0.90	0.9474	0.90	0.9231

Table : Detailed Per-Class Evaluation

Detailed Per-Class Evaluation: UC Merced

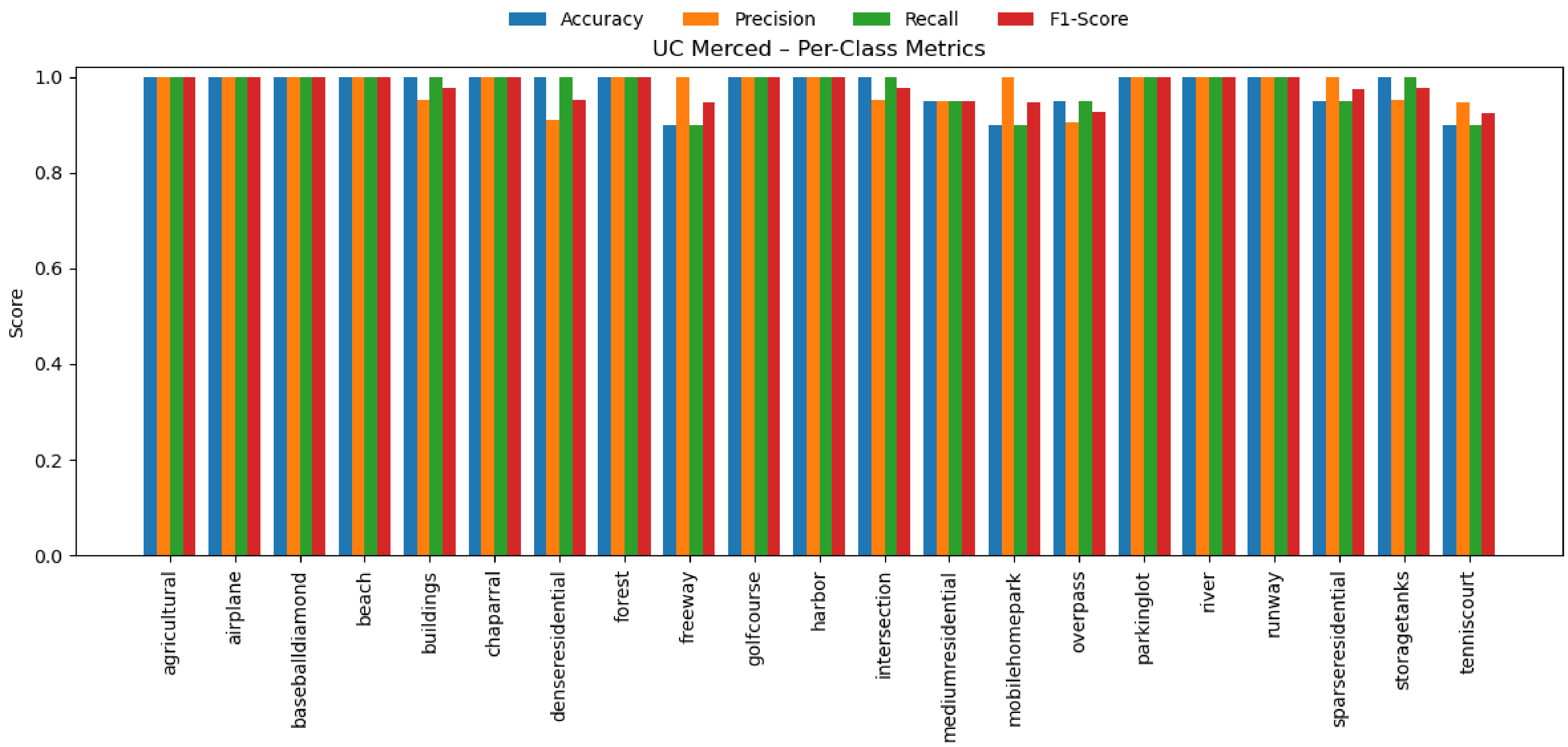


Figure : UC Merced – Per-Class Metrics

Comparison of TriDCCS-SVM with Previous Studies

TriDCCS-SVM vs. VGG16-SVM (Tun et al., 2021)

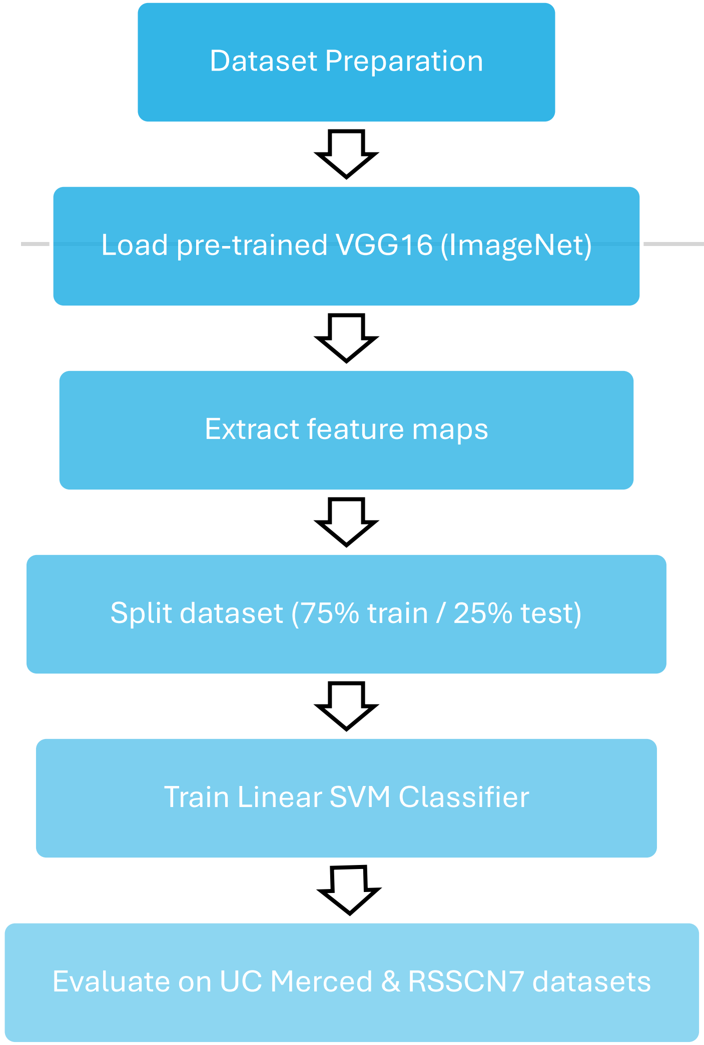


Figure : VGG-SVM classifier algorithm - Combining VGG16 with SVM

Paper Title: Remote Sensing Data Classification Using a Hybrid Pre-Trained VGG16 CNN-SVM Classifier

(ElConRus 2021 | Nyan Linn Tun et al.)

- Methodology:
  1. Pre-trained VGG16 for feature extraction.
  2. SVM classifier for final prediction (Linear Kernel).
- Hardware Specs:
  - Ryzen 5, GPU GTX 10703, 32GB RAM.
- Datasets Used:
  1. UC Merced Land Dataset
  2. RSSCN7 Dataset
- Reported Accuracy:
  1. UC Merced → 87.71%
  2. RSSCN7 → 95.24%

> TriDCCS-SVM:

- ✓ Used 3 modern CNNs for richer and complementary features.
- ✓ Feature Fusion → Captured diverse hierarchical representations.
- ✓ Advanced SVM Kernel: Used RBF Kernel for better handling of non-linear separations.
- ✓ Achieved Accuracy: UC Merced → 97.86% (+10.15% improvement) → SIRI-WHU → 96.25%

Comparison of Classification Accuracy: TriDCCS-SVM vs. VGG16-SVM (Tun et al., 2021)

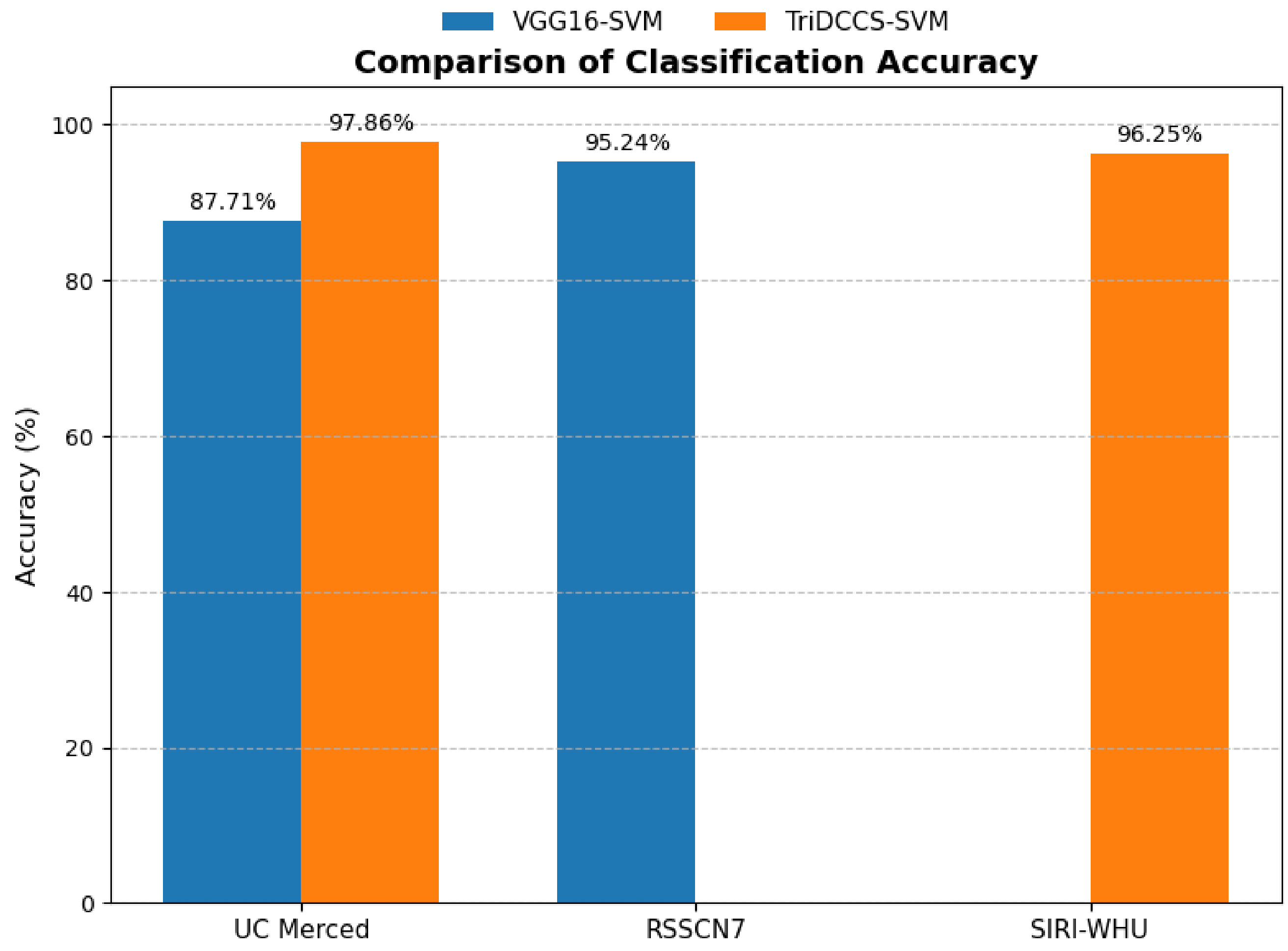


Figure : Comparison of Classification Accuracy: TriDCCS-SVM vs. VGG16-SVM

Dataset	VGG16-SVM (Tun et al., 2021)	TriDCCS-SVM
UC Merced	87.71%	**97.86% (+10%)**
RSSCN7	95.24%	-
SIRI-WHU	-	**96.25%**

Table : Comparison of Classification Accuracy Between VGG16-SVM Models and TriDCCS-SVM

TriDCCS-SVM vs. Single CNN (Ramasamy et al., 2023)

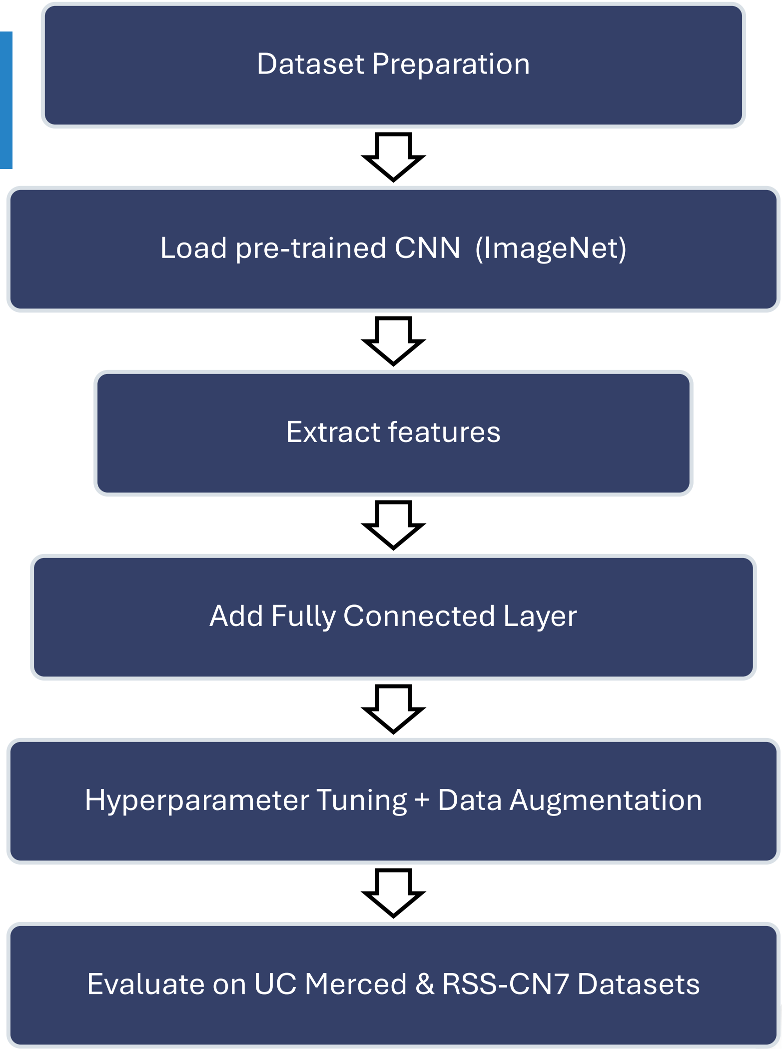


Figure : Single CNN Models Workflow

Paper Title: Investment of Classic Deep CNNs and SVM for Classifying Remote Sensing Images

(Advances in Electrical and Computer Engineering)

- Methodology:
  1. Transfer Learning approach using three pre-trained models (VGG16, ResNet50, DenseNet121).
  2. Fully Connected Layer for classification.
  3. Data Augmentation techniques applied.
  4. Hyperparameter tuning with Random Search.
- Datasets Used:
  1. UC Merced Land Dataset
  2. RSS-CN7 Dataset
- Reported Accuracy:

Model	UC Merced	RSS-CN7
ResNet50	91.72%	88.06%
DenseNet121	93.19%	89.82%
VGG16	95.88%	91.26%

> TriDCCS-SVM:

✓ Used three modern CNNs and performed feature fusion.
✓ Applied advanced SVM (RBF kernel).
✓ Achieved Accuracy:
- UC Merced → 97.86% (+1.98%, +4.67%, +6.14% over VGG16, DenseNet121, ResNet50).
- SIRI-WHU → 96.25% (+2.05%, +3.65%, +0.45% over DenseNet121, VGG16, ResNet50).

Comparison of Classification Accuracy: TriDCCS-SVM vs. Single CNN (Ramasamy et al., 2023)

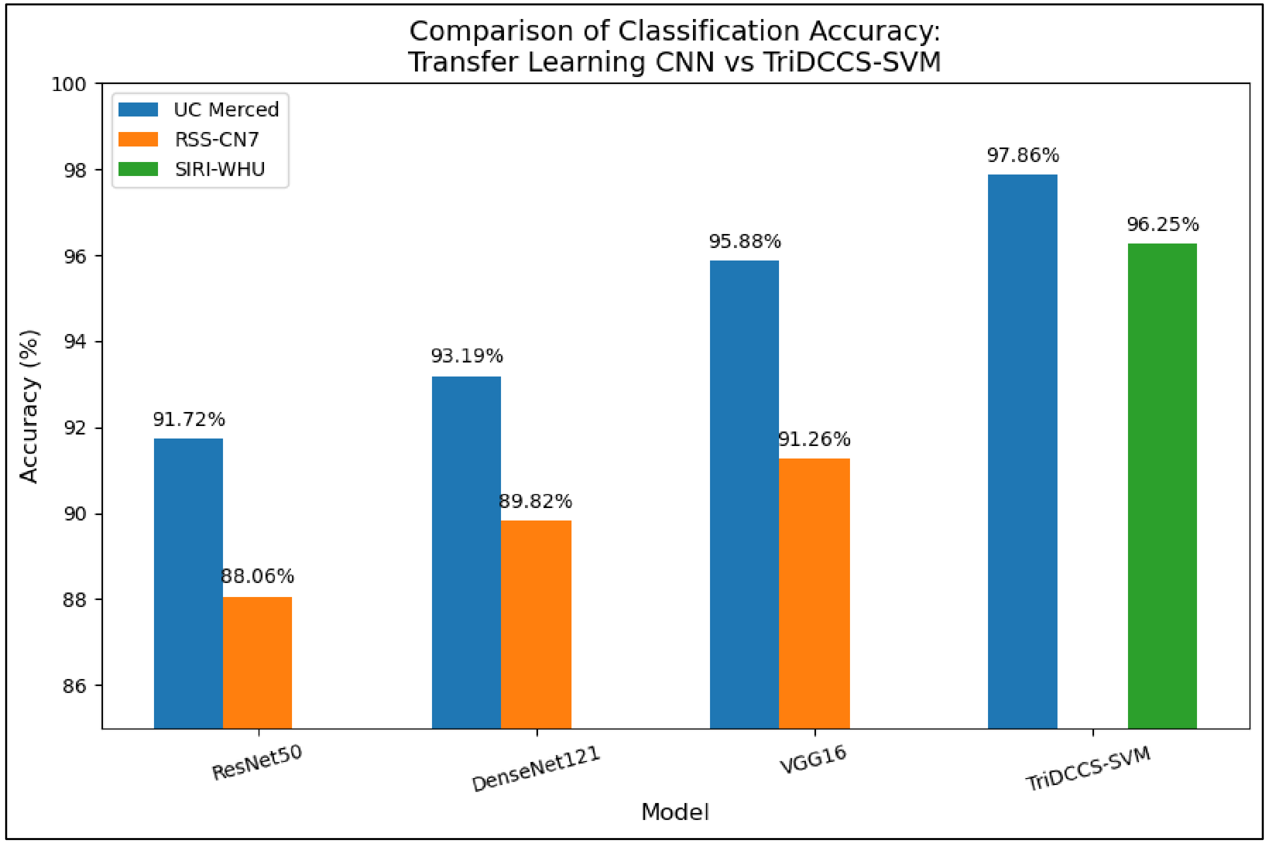


Figure : Comparison of Classification Accuracy: TriDCCS-SVM vs. Single CNN (Ramasamy et al., 2023)

Model	UC_Merced (%)	RSS-CN7 (%)	SIRI-WHU (%)
ResNet50	91.72	88.06	-
DenseNet121	93.19	89.82	-
VGG16	95.88	91.26	-
TriDCCS-SVM	97.86	-	96.25

Table : Comparison of Classification Accuracy Between CNN Models and TriDCCS-SVM

USIM

TriDCCS-SVM vs. Single CNN-SVM (AlAfandy et al., 2020)

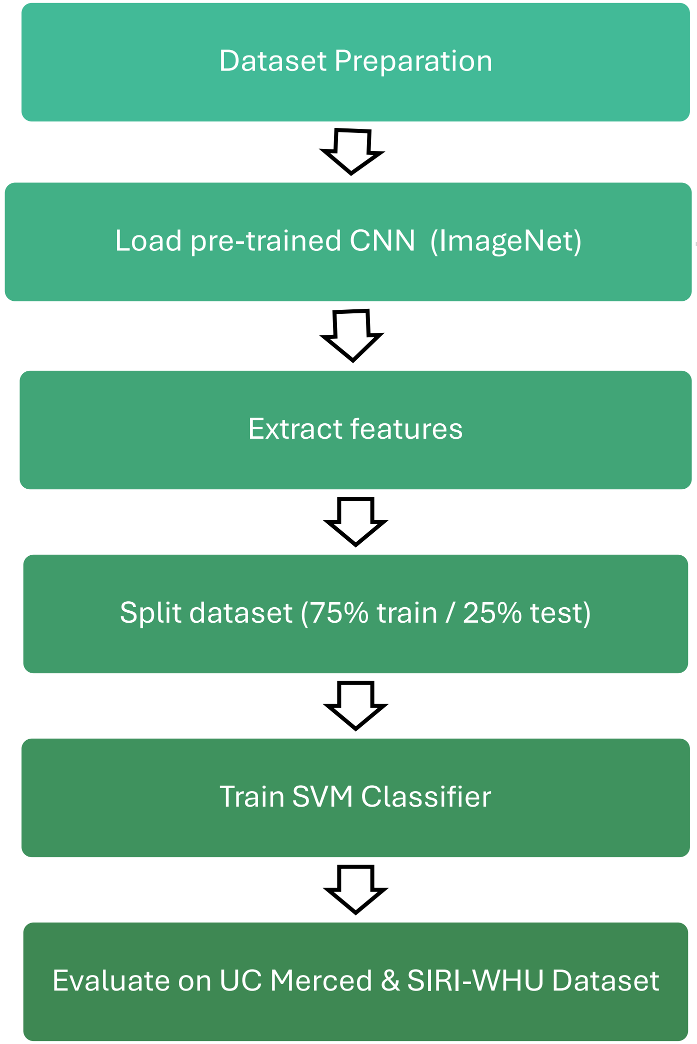


Figure : Single CNN-SVM model Workflow

Paper Title: Investment of Classic Deep CNNs and SVM for Classifying Remote Sensing Images

(Advances in Science, Technology and Engineering Systems Journal)

- Methodology:
  1. Pre-trained CNN for feature extraction
  2. SVM classifier for final prediction.
- Hardware Specs:
  - Google Colab: 2-core Xeon CPU, Tesla K80 GPU (12GB), 13GB RAM.
- Datasets Used:
  1. UC Merced Land Dataset
  2. SIRI-WHU Dataset.
- Reported Accuracy:

Model	UC Merced	SIRI-WHU
DenseNet-SVM	90.20%	94.20%
VGG16-SVM	88.10%	92.60%
ResNet50-SVM	**90.40%**	**95.80%**

> TriDCCS-SVM:

✓ Used three modern CNNs and performed feature fusion.
✓ Applied advanced SVM (RBF kernel).
✓ Achieved Accuracy:
- UC Merced → 97.86% (+7.66%, +9.76%, +7.46% over DenseNet-SVM, VGG16-SVM, ResNet50-SVM).
- SIRI-WHU → 96.25% (+2.05%, +3.65%, +0.45% over DenseNet-SVM, VGG16-SVM, ResNet50-SVM).

Comparison of Classification Accuracy: TriDCCS-SVM vs. Single CNN-SVM

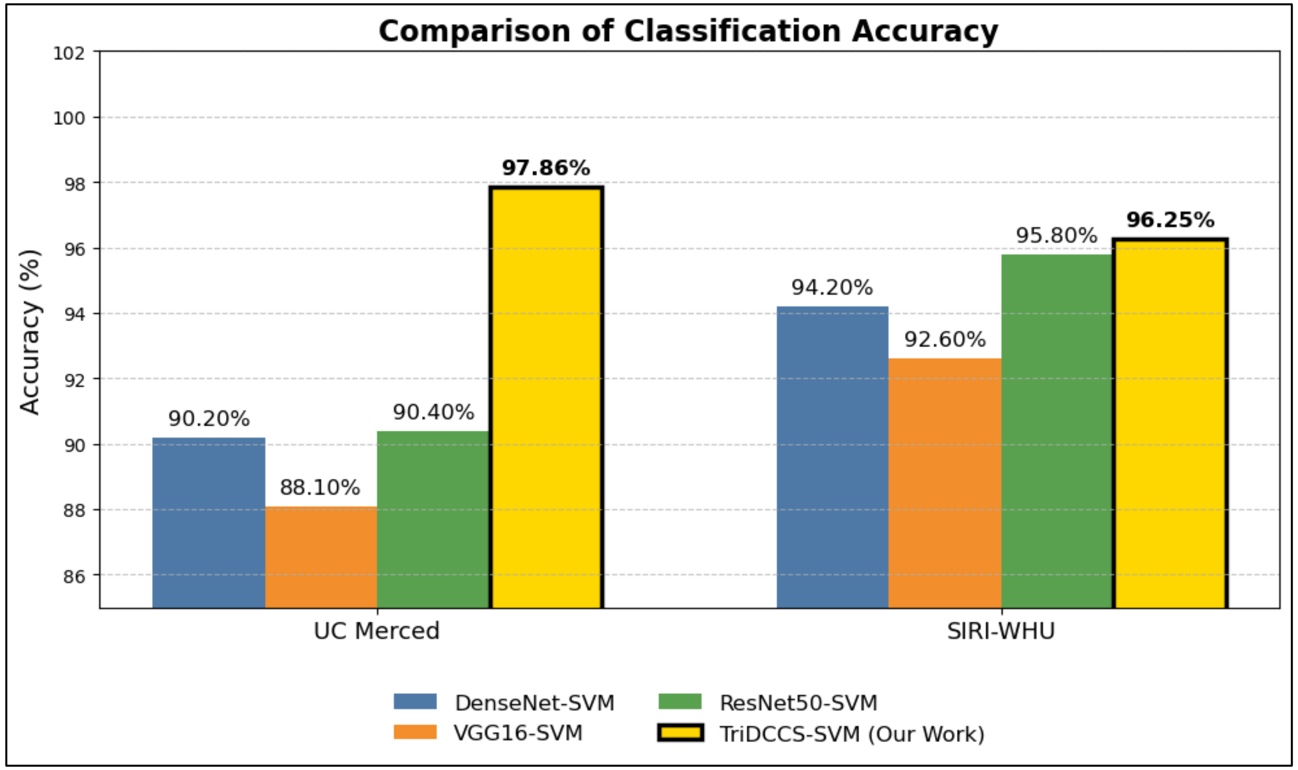


Figure : Comparison of Classification Accuracy

Dataset	DenseNet-SVM	VGG16-SVM	ResNet50-SVM	TriDCCS-SVM
UC Merced	90.20%	88.10%	**90.40%**	**97.86%**
SIRI-WHU	94.20%	92.60%	**95.80%**	**96.25%**

Table : Comparison of Classification Accuracy Between Single CNN-SVM Models and TriDCCS-SVM

TriDCCS-SVM vs. AlexNet CNN (Shafaey et al., 2019)

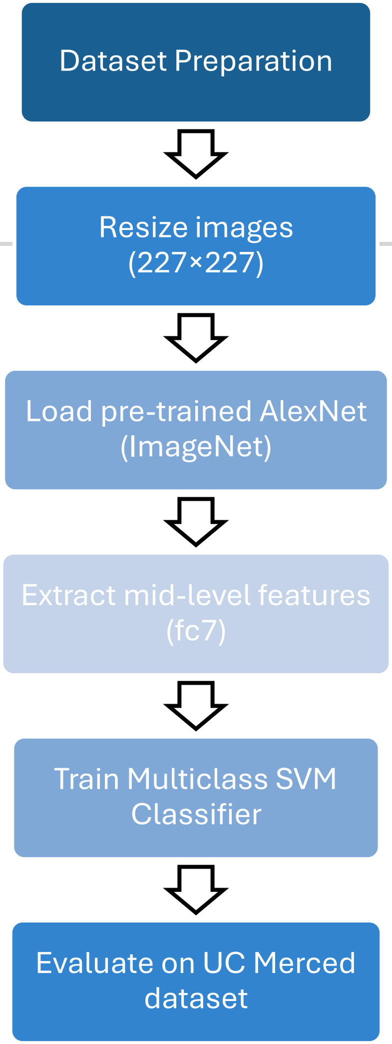


Figure : AlexNet-CNN classifier model Combining CNN with SVM

Paper Title: Deep Learning for Satellite Image Classification

(AISI 2018 | Mayar A. Shafaey et al.)

- Methodology:
  1. Pre-trained AlexNet CNN for feature extraction.
  2. Multiclass SVM classifier for final prediction.
- Hardware Specs:
  - Machine 1: Intel i7-2670QM @ 2.20GHz, 8GB RAM.
  - Machine 2: Intel i7-7700HQ @ 2.20GHz, GPU GTX 1050, 16GB RAM.
- Datasets Used: UC Merced Land Dataset
- Reported Accuracy: AlexNet CNN: **94%**

> TriDCCS-SVM:

- ✓ Used three modern CNNs for richer and complementary features.
- ✓ Feature Fusion → Captured diverse hierarchical representations.
- ✓ Advanced SVM Kernel: Used RBF Kernel for better handling of non-linear separations.
- ✓ Achieved Accuracy: UC Merced → 97.86% (+3.86% improvement), SIRI-WHU →96.25%

Comparison of Classification Accuracy: TriDCCS-SVM vs. AlexNet-CNN (Shafaey et al., 2019)

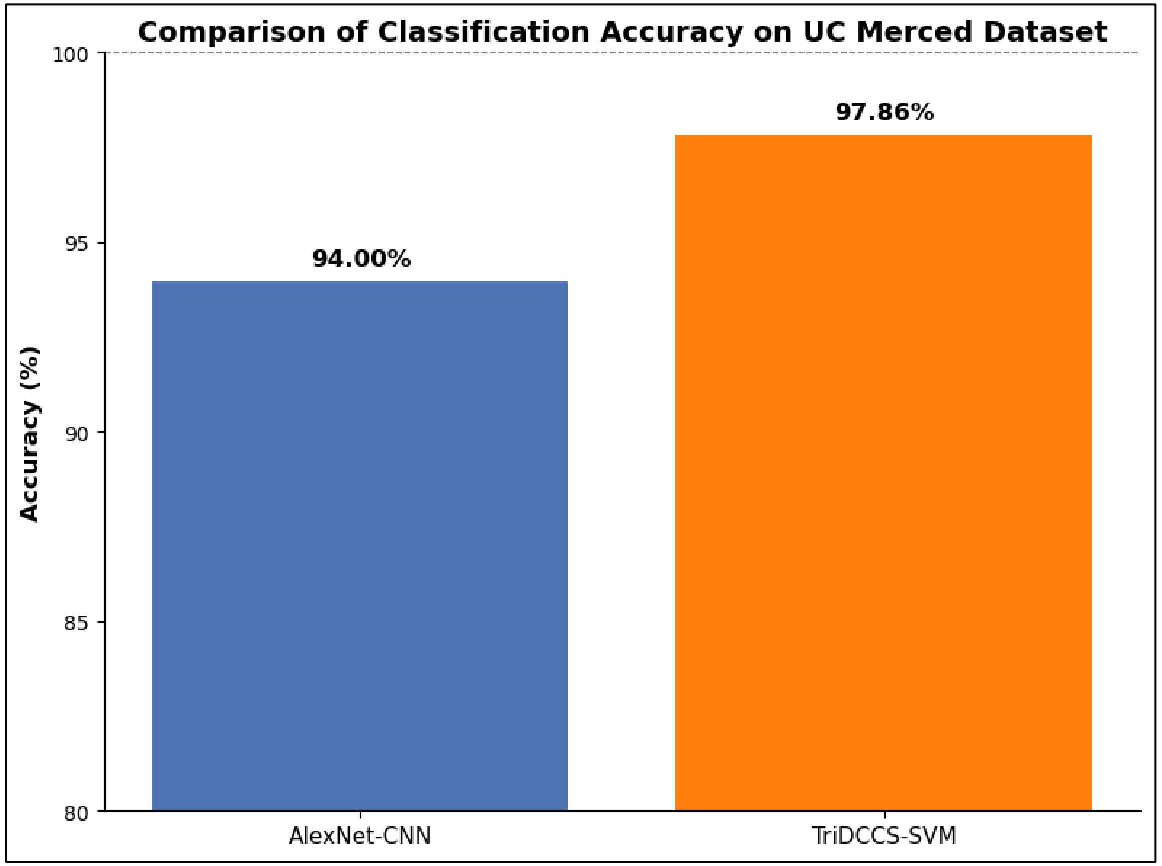


Figure : Comparison of Classification Accuracy between AlexNet-CNN and TriDCCS-SVM

Dataset	AlexNet-CNN)	TriDCCS-SVM
UC Merced	94.0%	**97.86% (+3.86%)**

Table : Classification Accuracy on UC Merced Dataset

USIM

TriDCCS-SVM vs. Lightweight CNN (Dwivedi et al., 2022)

Dataset Preparation
⇩
Image Preprocessing (Normalization)
⇩
Configure Lightweight CNN Architecture
⇩
Train Lightweight CNN Model
⇩
Evaluate on UC Merced Dataset
⇩
Compare with State-of-the-Art Models

Figure : Lightweight CNN classifier workflow

Paper Title: Lightweight Convolutional Neural Network for Land Use Image Classification

(JAGST 2022 | Dwijendra N. Dwivedi & Ganesh Patil)

- Methodology:
  1. Developed a lightweight CNN for land use classification.
  2. Used dropout, improved normalization, and optimized convolution/max-pooling layers.
- Hardware Specs:
  - Intel Core i7 CPU, GPU unspecified, RAM: 16GB.
- Datasets Used: UC Merced Land Dataset
- Reported Accuracy: UC Merced: 88.29%

> TriDCCS-SVM:

✓ Used three modern CNNs for richer and complementary features.
✓ Feature Fusion → Captured diverse hierarchical representations.
✓ Advanced SVM Kernel: Used RBF Kernel for better handling of non-linear separations.
✓ Achieved Accuracy: UC Merced :97.86% (+9.57% improvement).

Comparison of Classification Accuracy: TriDCCS-SVM vs. Lightweight CNN (Dwivedi et al., 2022)

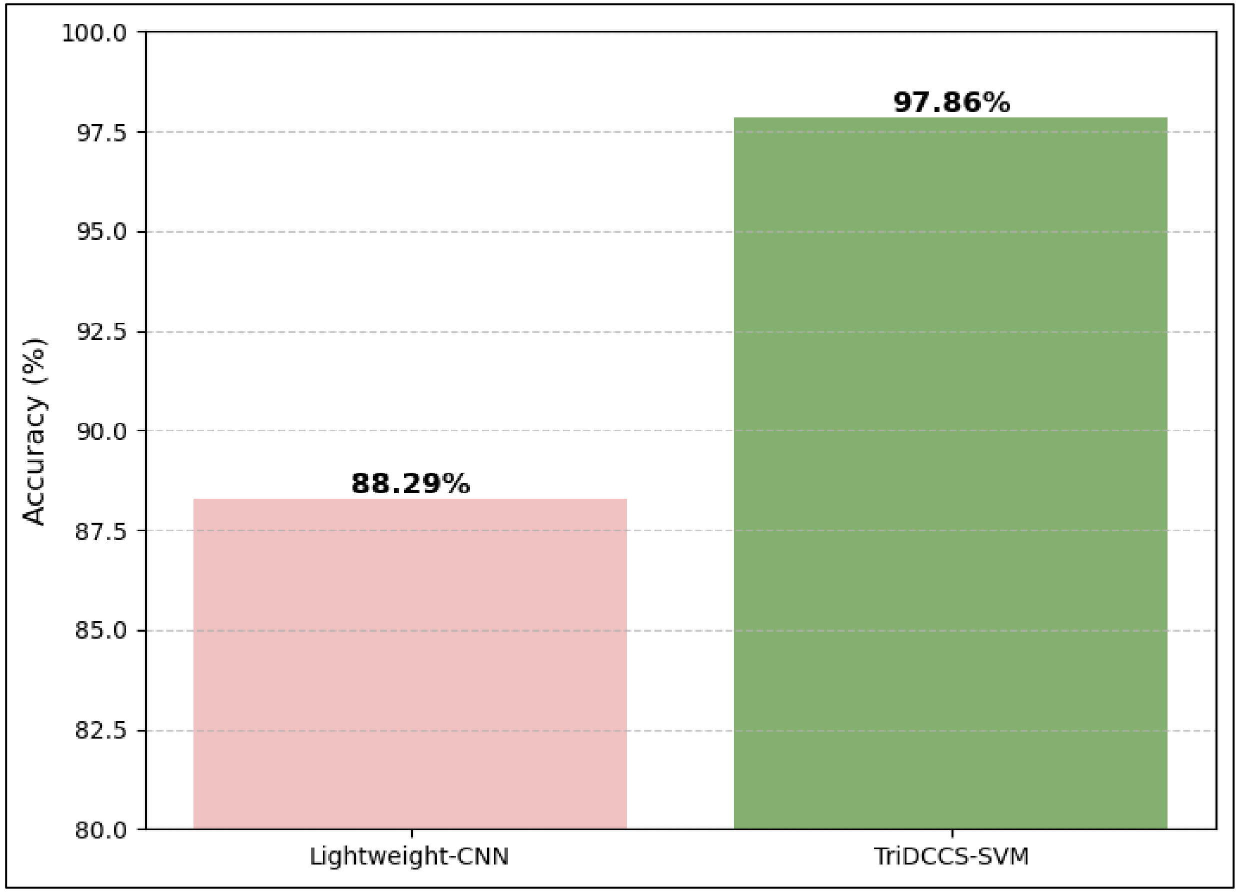


Figure : Comparison of Classification Accuracy

Dataset	Lightweight CNN	TriDCCS-SVM
UC Merced	88.29%	**97.86% (+9.57%)**

Table: Classification Accuracy Comparison

TriDCCS-SVM vs. CNN-FE, TL, Fine-Tuning (Alem et al., 2022)

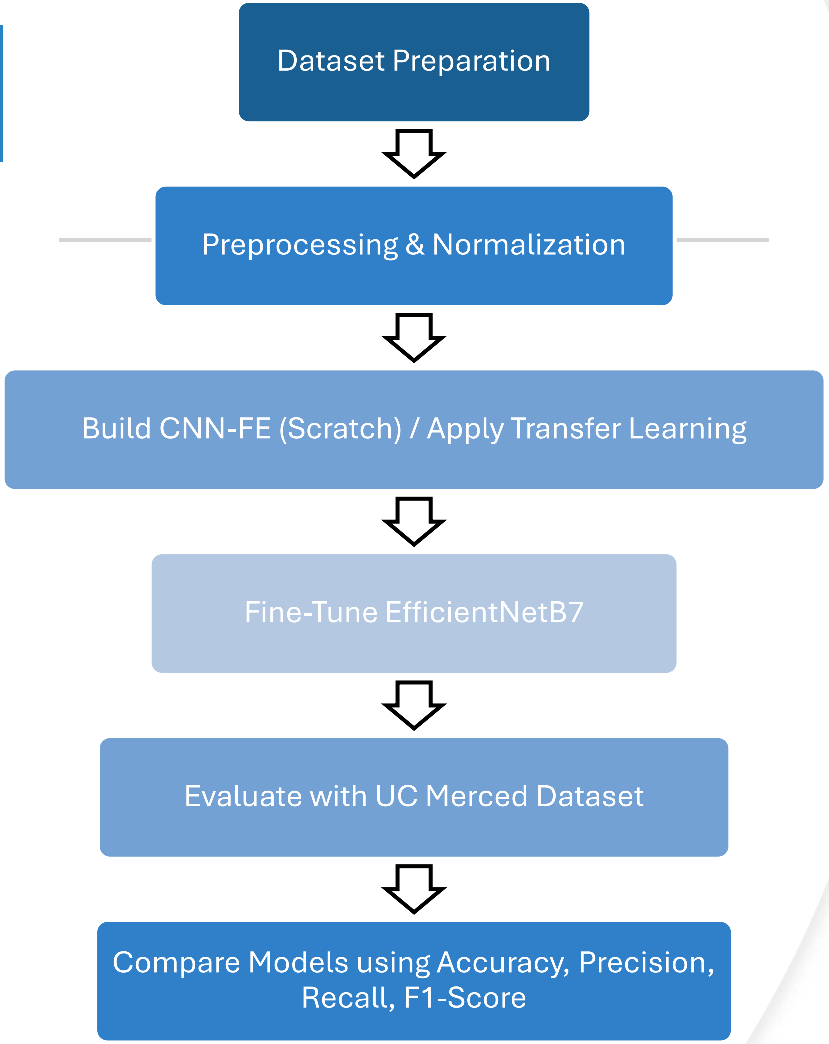


Figure : Deep Learning Model workflow

Paper Title: Deep Learning Models Performance Evaluations for Remote Sensed Image Classification

(IEEE Access 2022 | Abebaw Alem & Shailender Kumar)

- Methodology:
  1. Developed three deep learning models for classification:
     - CNN-FE: Built from scratch.
     - TL: Transfer Learning using EfficientNetB7.
     - Fine-Tuning: Pre-trained EfficientNetB7 with all layers trainable.
  2. Evaluated models using Accuracy, Precision, Recall, F1-Score, and Confusion Matrix.
- Hardware Specs:
  - Intel Core i3-4000M @ 2.40 GHz, 4GB RAM (Google Colab Tesla K80 GPU).
- Datasets Used: UC Merced Land Dataset
- Reported Accuracy:
  - CNN-FE: 84.76%
  - Transfer Learning (TL): 87.90%
  - Fine-Tuning: 88.00%

> TriDCCS-SVM:

✓ Used three modern CNNs for richer and complementary features.
✓ Feature Fusion → Captured diverse hierarchical representations.
✓ Advanced SVM Kernel: Used RBF Kernel for better handling of non-linear separations.
✓ Achieved Accuracy: UC Merced → 97.86% (+9.86% improvement).

Comparison of Classification Accuracy: TriDCCS-SVM vs. CNN-FE, TL, Fine-Tuning (Alem et al., 2022)

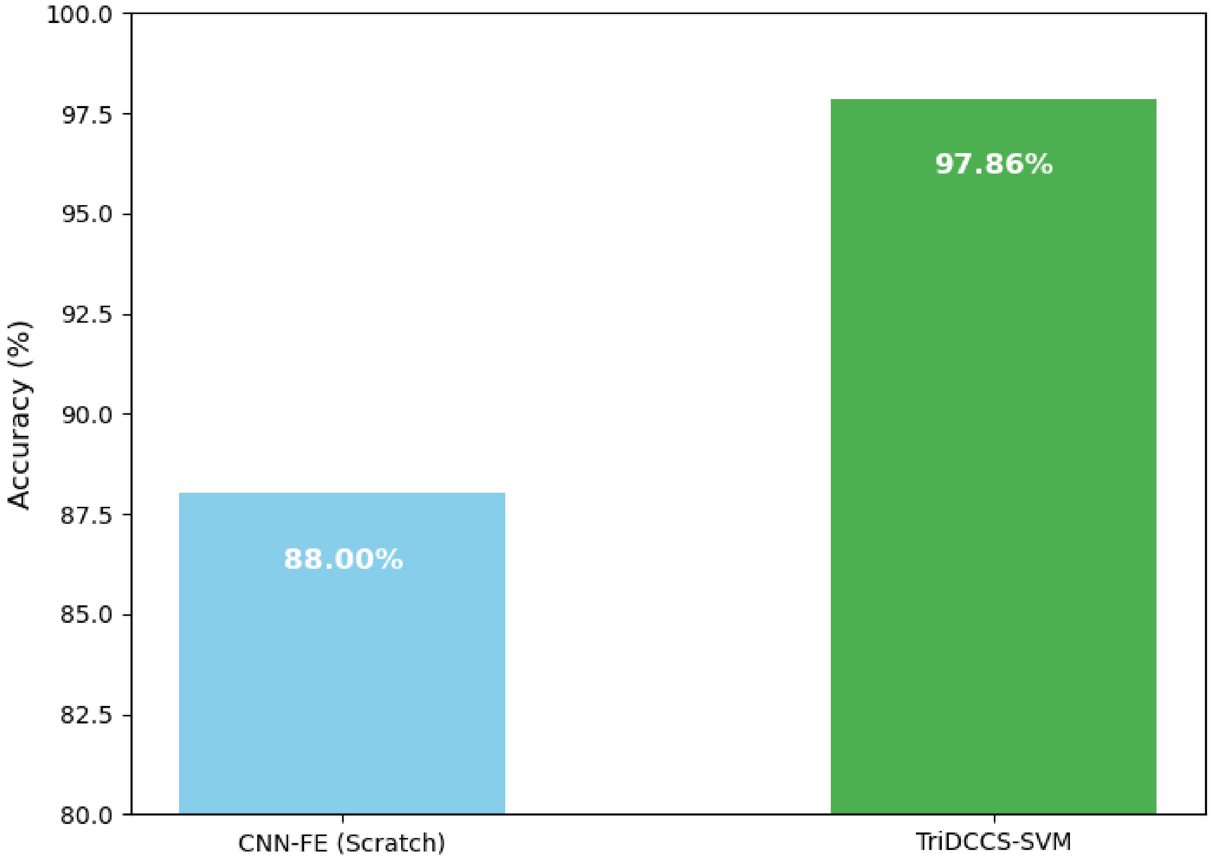


Figure : Comparison of Classification

Dataset	CNN-FE	Transfer Learning (TL)	Fine-Tuning	TriDCCS-SVM
UC Merced	84.76%	87.90%	88.00%	**97.86% (+9.86%)**

Table: Comparison of CNN-FE, TL, Fine-Tuning, and TriDCCS-SVM on UC Merced Dataset.

TriDCCS-SVM vs. AlexNet, VGG16, VGG19 (Thirumaladevi et al., 2023)

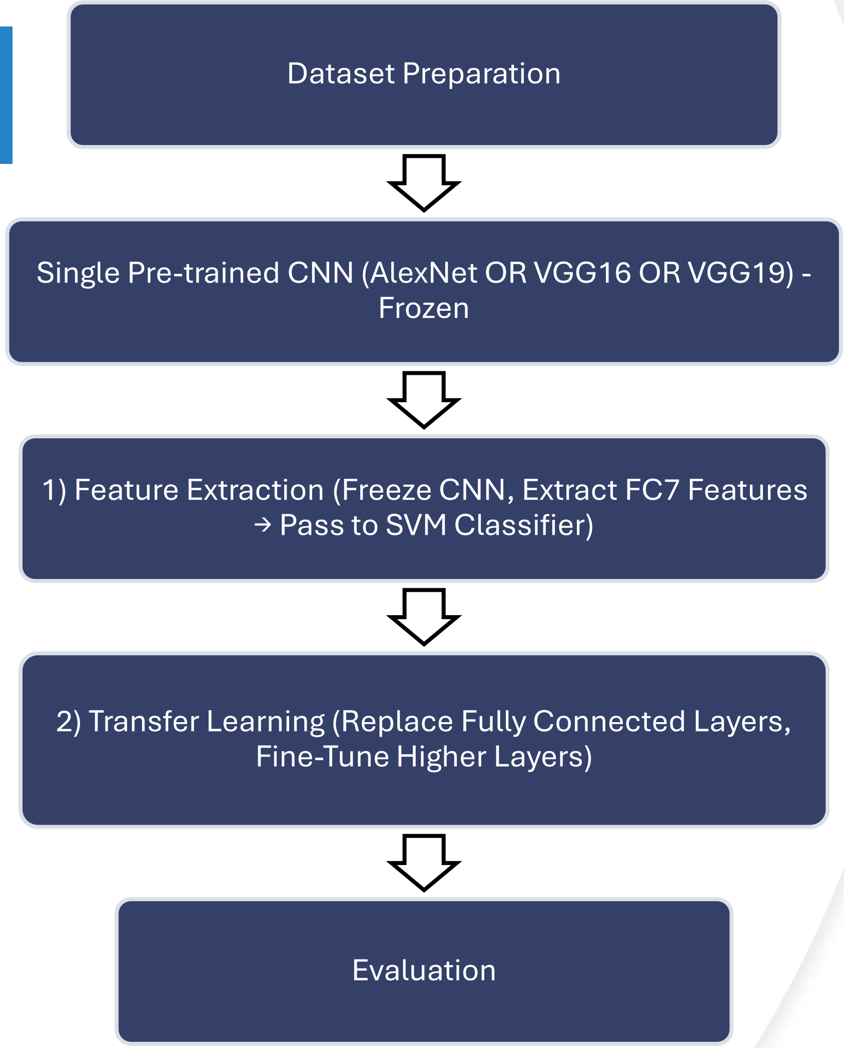


Figure : Deep Learning Model workflow

Paper Title: Remote sensing image scene classification by transfer learning to augment the accuracy

(Measurement: Sensors, 2023 | S. Thirumaladevi et al.)

- Methodology:
  1. Extracted features independently from the fc7 layer of AlexNet, VGG16, and VGG19, and classified them separately using SVM.
  2. Developed Transfer Learning models by replacing fully connected layers for classification.
- Hardware Specs:
  - Workstation with GPU (Details not specified).
- Datasets Used:
  1. UC Merced Land Dataset.
  2. SIRI-WHU Dataset.
- Reported Accuracy

Dataset	Network	Single pre-trained CNN + SVM (%)	Transfer Learning (%)	TriDCCS-SVM
UC Merced	AlexNet	79.76	93.57	
	VGG19	81.19	94.08	
	VGG16	83.81	**95.00%**	**97.86% (+2.86%)**
SIRI-WHU	AlexNet	86.52	91.34	
	VGG19	87.60	92.78	
	VGG16	88.04	**93.40%**	**96.20% (+2.80%)**

Table: Classification Accuracy Comparison

Comparison of Classification Accuracy: TriDCCS-SVM vs. VGG16, VGG19 (Thirumaladevi et al., 2023)

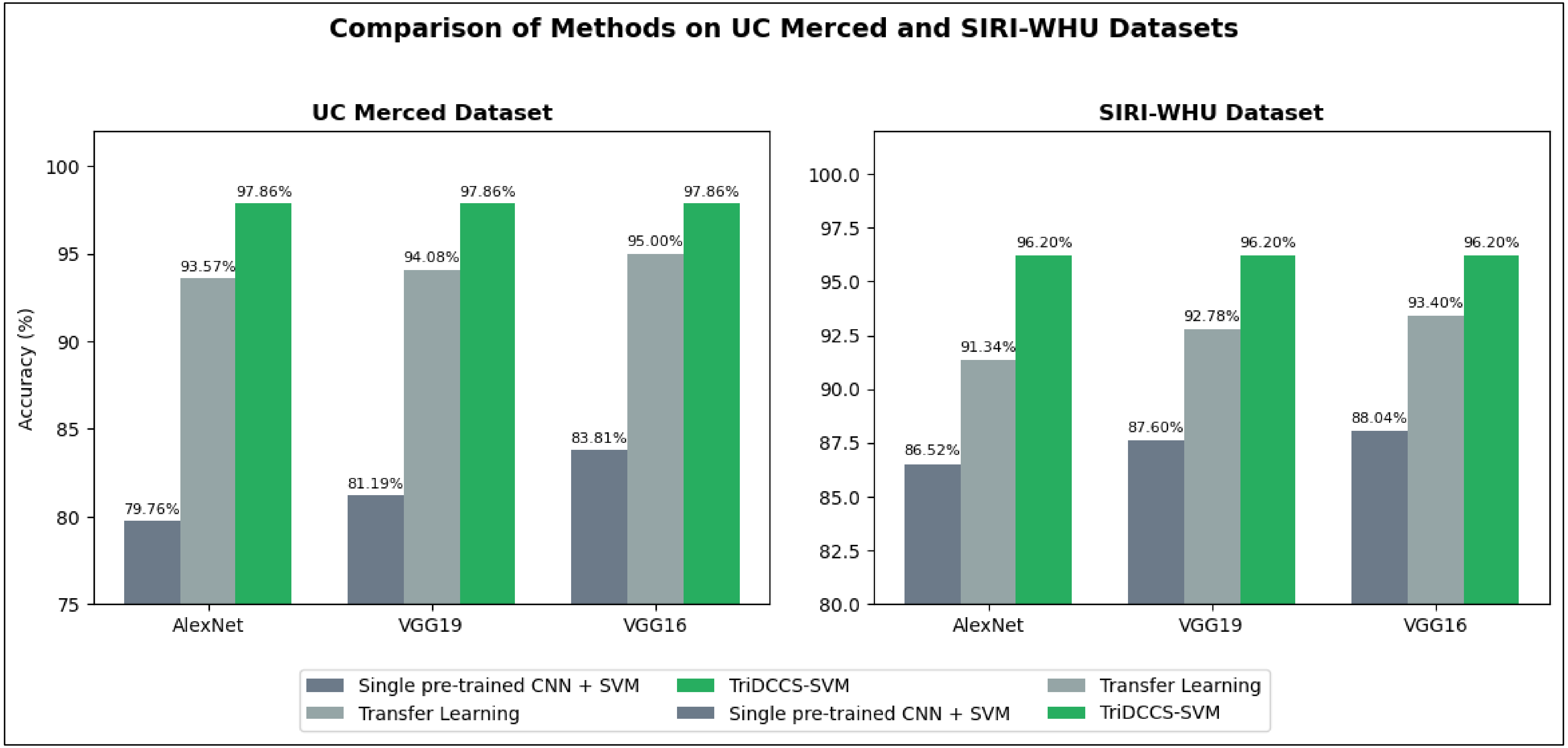


Figure : Comparison of classification accuracy between AlexNet, VGG16, VGG19 (Transfer Learning) and TriDCCS-SVM

UNIVERSITI SAINS ISLAM MALAYSIA

جامعة العلوم الإسلامية الماليزية

ISLAMIC SCIENCE UNIVERSITY OF MALAYSIA

Faculty of Science and Technology (FST)

2025