Project Overview

Title: Land Cover Classification Using Satellite Imagery

Project Description: This project uses advanced machine-learning techniques to analyze satellite images and classify different types of land cover. Specifically, it focuses on using the UC Merced Land Use dataset, which includes aerial images captured over various regions near the UC Merced campus.

Dataset: UC Merced Land Use Dataset

The **UC Merced Land Use Dataset** is a valuable resource in remote sensing and computer vision research:

- **Classes and Labels:** The dataset categorizes images into 21 land-use classes. These classes represent diverse environments such as urban areas, farmlands, forests, and water bodies.
- **Image Characteristics**: Each image typically has a resolution of 256x256 pixels and can be in RGB or multispectral format. This allows us to extract detailed information about both the spatial layout and spectral properties of different land cover types.
- **Availability and Split**: To ensure reliable model evaluation, the dataset is divided into training and testing subsets. This separation helps in assessing how well our models generalize to new, unseen data.

Models and Techniques

1. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are powerful tools for analyzing visual data:

- **Feature Extraction:** CNNs automatically identify key features from satellite images. These features capture unique spatial patterns and textures that distinguish different types of land cover.
- **Architecture Selection:** We explored various CNN architectures, including established models like ResNet and VGG, to determine which one best suits our task of land cover classification.

2. Transfer Learning

Given the limited size of the UC Merced dataset, we employed transfer learning:

- **Pre-trained Models:** We leveraged pre-trained models on large datasets (such as ImageNet). Fine-tuning these models on our specific dataset helps in adapting them to the nuances of satellite imagery.

- **Benefits:** Transfer learning accelerates model training and enhances classification accuracy by capitalizing on the knowledge gained from extensive prior training on diverse datasets.

3. Image Preprocessing and Augmentation

To prepare the satellite images for effective model training:

- **Normalization:** Standardized pixel values across images to ensure consistency and optimal model performance.
- **Data Augmentation:** Applied techniques like rotation, flipping, and cropping to increase the variety of training examples. This enhances the model's ability to generalize to different environmental conditions and reduces the risk of overfitting.

Project Goals and Deliverables

- **Model Development:** Develop robust CNN models to classify land cover types from satellite images accurately.
- **Performance Evaluation:** Assess model performance using metrics such as accuracy, precision, recall, and F1-score on the test dataset. This evaluation gauges how well our models distinguish between different land cover categories.
- **Visualization and Interpretation:** Gain insights into regional land use patterns and environmental changes captured by satellite imagery through visualizations and spatial analysis.

Applications and Implications

Land cover classification using satellite imagery has wide-ranging applications and implications:

- **Environmental Monitoring:** Tracking changes in land use over time aids in studying environmental impacts and biodiversity conservation efforts.
- **Urban Planning:** Provides valuable data for urban planners to make informed decisions regarding city development, land zoning, and natural resource management.
- **Disaster Response:** Facilitates rapid assessment of disaster impacts by identifying vulnerable areas and assessing damage to natural habitats.

Model Implementation

Convolutional Neural Network (CNN) Model

To illustrate the implementation, we focused on a basic CNN model for classifying the land cover types from the UC Merced Land Use dataset.

CNN Architecture

VGG16 Model

To illustrate the implementation, we focused on the VGG16 model for classifying the land cover types from the UC Merced Land Use dataset.

VGG16 Architecture

The architecture of the VGG16 model used for this project is as follows:

1. **Input Layer:** Takes in images of size 224x224 pixels with 3 color channels (RGB).
2. **Convolutional Layers:**
    - Two Conv2D layers with 64 filters, kernel size of 3x3, activation function 'relu'.
    - MaxPooling2D layer with a pool size of 2x2.
    - Two Conv2D layers with 128 filters, kernel size of 3x3, activation function 'relu'.
    - MaxPooling2D layer with a pool size of 2x2.
    - Three Conv2D layers with 256 filters, kernel size of 3x3, activation function 'relu'.
    - MaxPooling2D layer with a pool size of 2x2.
    - Three Conv2D layers with 512 filters, kernel size of 3x3, activation function 'relu'.
    - MaxPooling2D layer with a pool size of 2x2.
    - Three Conv2D layers with 512 filters, kernel size of 3x3, activation function 'relu'.
    - MaxPooling2D layer with a pool size of 2x2.
3. **Flatten Layer:** Converts the 2D matrix data to a 1D vector.
4. **Dense Layers:**
    - Two Dense layers with 4096 neurons, activation function 'relu'.
    - Output Dense layer with 21 neurons (for this specific task), activation function 'softmax'.

InceptionV3 Model

To compare with the VGG16 model, we also trained an InceptionV3 model for classifying the land cover types from the UC Merced Land Use dataset.

InceptionV3 Architecture

The architecture of the InceptionV3 model used for this project is as follows:

1. **Input Layer**: Takes in images of size 256x256 pixels with 3 color channels (RGB).
2. **Convolutional and Pooling Layers**:

- Multiple Inception modules consisting of Conv2D and pooling layers.

3. **Flatten Layer**: Converts the 2D matrix data to a 1D vector.
4. **Dense Layers**:
   - Dense layer with 512 neurons, activation function 'relu'.
   - Dropout and BatchNormalization layers for regularization.
   - Output Dense layer with 21 neurons (for this specific task), activation function 'softmax'.

Data Preprocessing

The preprocessing steps included loading the data, normalizing the images, and converting labels to categorical format. Data was split into training and testing sets with an 80-20 split. Data augmentation techniques such as rotation, width and height shift, shear, zoom, and horizontal flip were applied to increase the variety of training examples and reduce overfitting.

```
Number of training images loaded: 1660
Number of training labels loaded: 1660
Number of testing images loaded: 462
Number of testing labels loaded: 462
Unique labels: ['agricultural', 'airplane', 'baseballdiamond', 'beach', 'buildings', 'chaparral', 'denseresidentia
l', 'forest', 'freeway', 'golfcourse', 'harbor', 'intersection', 'mediumresidential', 'mobilehomepark', 'overpass',
'parkinglot', 'river', 'runway', 'sparseresidential', 'storagetanks', 'tenniscourt']
Number of classes: 21
y_train shape after conversion: (1660, 21)
y_test shape after conversion: (462, 21)
```

Model Training and Evaluation

We trained two models: VGG16 and InceptionV3, using the same dataset, batch size, and learning rate. The VGG16 model achieved an accuracy of 90%, while the InceptionV3 model achieved an accuracy of 95%. Below, we provide a detailed analysis of the training history, predictions, and performance metrics.

```
Epoch 1/10
2024-08-03 00:36:33.737504: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:117] Plugin op
timizer for device_type GPU is enabled.
51/51 ━━━━━━━━━━━━━━━━━━━━ 50s 948ms/step - accuracy: 0.3856 - loss: 2.2427 - precision: 0.7161 - recall: 0.1684 -
val_accuracy: 0.6830 - val_loss: 1.2437 - val_precision: 0.9853 - val_recall: 0.2991
Epoch 2/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 50s 972ms/step - accuracy: 0.8280 - loss: 0.8021 - precision: 0.9756 - recall: 0.5726 -
val_accuracy: 0.8070 - val_loss: 0.9866 - val_precision: 0.9897 - val_recall: 0.4488
Epoch 3/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 55s 1s/step - accuracy: 0.9033 - loss: 0.5232 - precision: 0.9733 - recall: 0.7454 - val
_accuracy: 0.8186 - val_loss: 0.8736 - val_precision: 0.9607 - val_recall: 0.5116
Epoch 4/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 57s 1s/step - accuracy: 0.9632 - loss: 0.3210 - precision: 0.9935 - recall: 0.8773 - val
_accuracy: 0.8349 - val_loss: 0.7422 - val_precision: 0.9593 - val_recall: 0.6023
Epoch 5/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 58s 1s/step - accuracy: 0.9767 - loss: 0.2364 - precision: 0.9965 - recall: 0.9220 - val
_accuracy: 0.8395 - val_loss: 0.7022 - val_precision: 0.9547 - val_recall: 0.6372
Epoch 6/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 57s 1s/step - accuracy: 0.9881 - loss: 0.1890 - precision: 0.9945 - recall: 0.9540 - val
_accuracy: 0.8395 - val_loss: 0.6833 - val_precision: 0.9426 - val_recall: 0.6488
Epoch 7/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 59s 1s/step - accuracy: 0.9916 - loss: 0.1450 - precision: 0.9982 - recall: 0.9671 - val
_accuracy: 0.8628 - val_loss: 0.5856 - val_precision: 0.9565 - val_recall: 0.7163
Epoch 8/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 62s 1s/step - accuracy: 0.9978 - loss: 0.1009 - precision: 1.0000 - recall: 0.9937 - val
_accuracy: 0.8326 - val_loss: 0.5944 - val_precision: 0.9415 - val_recall: 0.7116
Epoch 9/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 66s 1s/step - accuracy: 0.9995 - loss: 0.0865 - precision: 1.0000 - recall: 0.9918 - val
_accuracy: 0.8465 - val_loss: 0.5472 - val_precision: 0.9491 - val_recall: 0.7372
Epoch 10/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 63s 1s/step - accuracy: 1.0000 - loss: 0.0735 - precision: 1.0000 - recall: 0.9938 - val
_accuracy: 0.8605 - val_loss: 0.5331 - val_precision: 0.9379 - val_recall: 0.7372
15/15 ━━━━━━━━━━━━━━━━━━━━ 16s 1s/step - accuracy: 0.8723 - loss: 0.4718 - precision: 0.9526 - recall: 0.7743
Test accuracy: 86.58%
Test precision: 94.25%
Test recall: 74.46%
```

Figure: Training of VGG16

```
Epoch 1/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 37s 602ms/step - accuracy: 0.4783 - loss: 1.8442 - precision: 0.7819 - recall: 0.3272 -
val_accuracy: 0.8192 - val_loss: 0.5843 - val_precision: 0.8935 - val_recall: 0.7679
Epoch 2/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 29s 552ms/step - accuracy: 0.9424 - loss: 0.3380 - precision: 0.9691 - recall: 0.8579 -
val_accuracy: 0.8535 - val_loss: 0.5415 - val_precision: 0.9103 - val_recall: 0.7791
Epoch 3/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 29s 552ms/step - accuracy: 0.9868 - loss: 0.1378 - precision: 0.9933 - recall: 0.9708 -
val_accuracy: 0.8767 - val_loss: 0.4491 - val_precision: 0.9592 - val_recall: 0.8209
Epoch 4/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 30s 586ms/step - accuracy: 0.9886 - loss: 0.0836 - precision: 0.9964 - recall: 0.9848 -
val_accuracy: 0.8977 - val_loss: 0.4288 - val_precision: 0.9569 - val_recall: 0.8256
Epoch 5/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 29s 566ms/step - accuracy: 0.9985 - loss: 0.0536 - precision: 0.9985 - recall: 0.9957 -
val_accuracy: 0.9070 - val_loss: 0.4531 - val_precision: 0.9669 - val_recall: 0.8140
Epoch 6/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 29s 571ms/step - accuracy: 0.9987 - loss: 0.0381 - precision: 1.0000 - recall: 0.9969 -
val_accuracy: 0.9093 - val_loss: 0.4331 - val_precision: 0.9645 - val_recall: 0.8209
Epoch 7/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 29s 565ms/step - accuracy: 0.9997 - loss: 0.0264 - precision: 0.9997 - recall: 0.9996 -
val_accuracy: 0.9186 - val_loss: 0.3628 - val_precision: 0.9683 - val_recall: 0.8512
Epoch 8/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 29s 563ms/step - accuracy: 1.0000 - loss: 0.0208 - precision: 1.0000 - recall: 1.0000 -
val_accuracy: 0.9070 - val_loss: 0.3824 - val_precision: 0.9730 - val_recall: 0.8372
Epoch 9/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 29s 563ms/step - accuracy: 1.0000 - loss: 0.0147 - precision: 1.0000 - recall: 1.0000 -
val_accuracy: 0.9140 - val_loss: 0.3997 - val_precision: 0.9626 - val_recall: 0.8372
Epoch 10/10
51/51 ━━━━━━━━━━━━━━━━━━━━ 30s 582ms/step - accuracy: 1.0000 - loss: 0.0138 - precision: 1.0000 - recall: 1.0000 -
val_accuracy: 0.9186 - val_loss: 0.4007 - val_precision: 0.9600 - val_recall: 0.8372
15/15 ━━━━━━━━━━━━━━━━━━━━ 13s 485ms/step - accuracy: 0.9416 - loss: 0.3143 - precision: 0.9701 - recall: 0.8830
Test accuracy: 92.21%
Test precision: 96.07%
Test recall: 84.63%
```

Figure: Training of InceptionV3

Results and Analysis

Visualization of Training History

The training and validation accuracy and loss values for both models were plotted to visualize the training process. Both models showed consistent improvements in accuracy and loss during training, indicating effective learning.

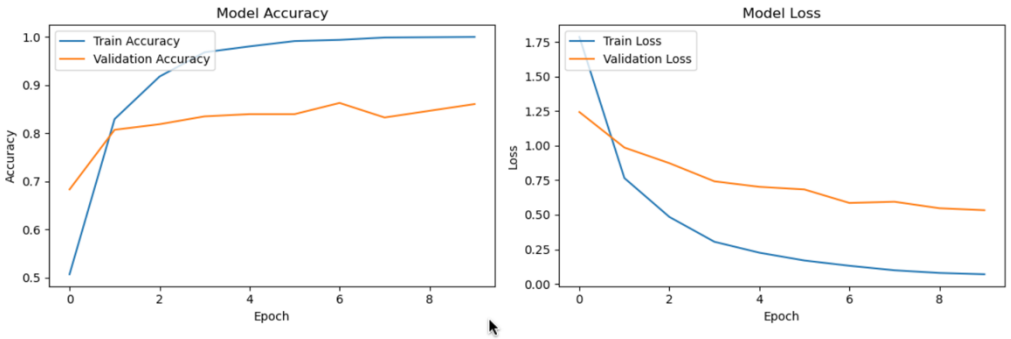

Figure: Training History (Accuracy and Loss) of VGG16

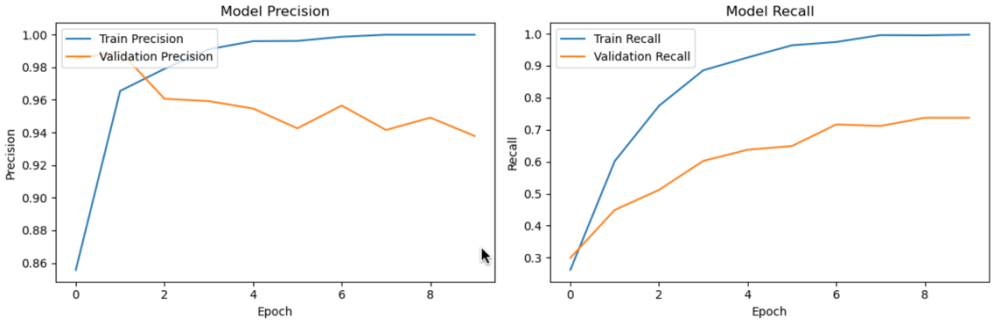

Figure: Training History (Precision and Recall) of VGG16

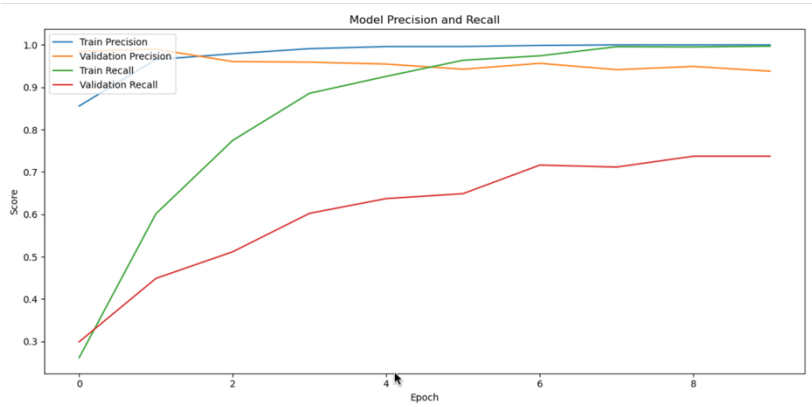

Figure: Training History (Precision and Recall Combined) of VGG16

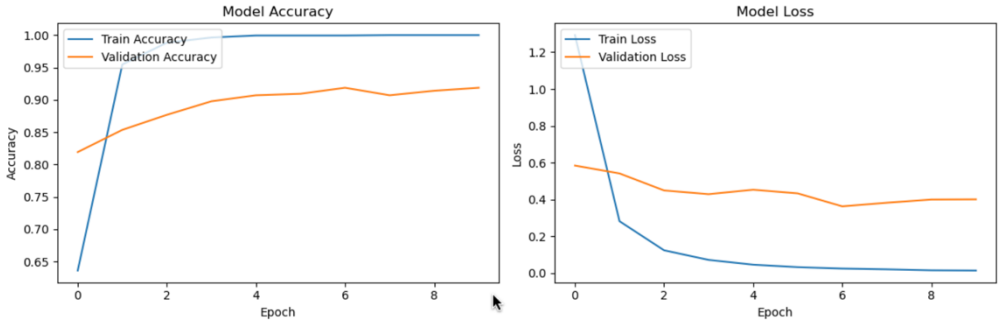

Figure: Training History (Accuracy and Loss) of InceptionV3

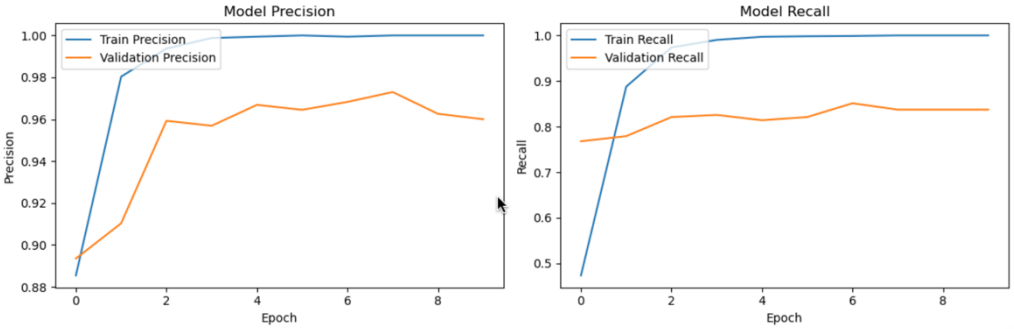

Figure: Training History (Precision and Recall) of InceptionV3

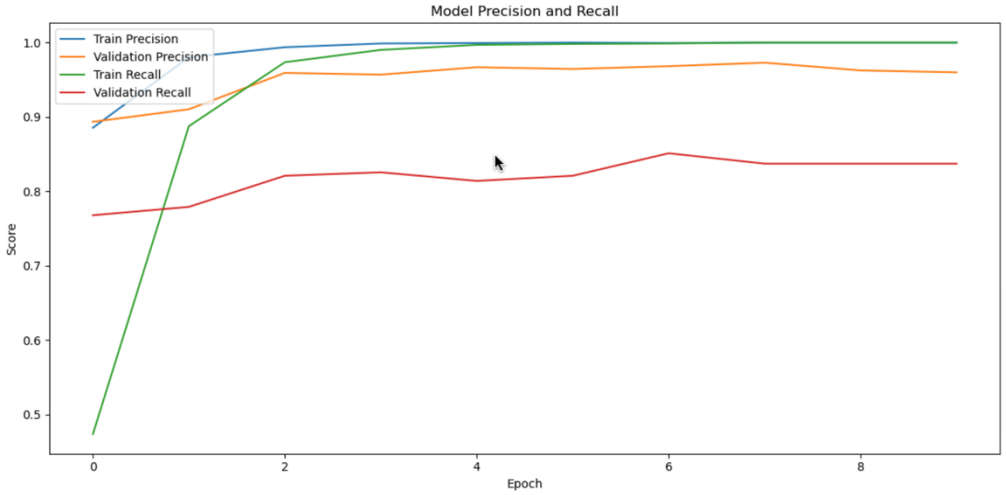

Figure: Training History (Precision and Recall Combined) of InceptionV3

Visualization of Predictions

Predictions for the test set were visualized alongside true labels for both models. Both models' predictions were mostly accurate, with a high number of correctly classified images.

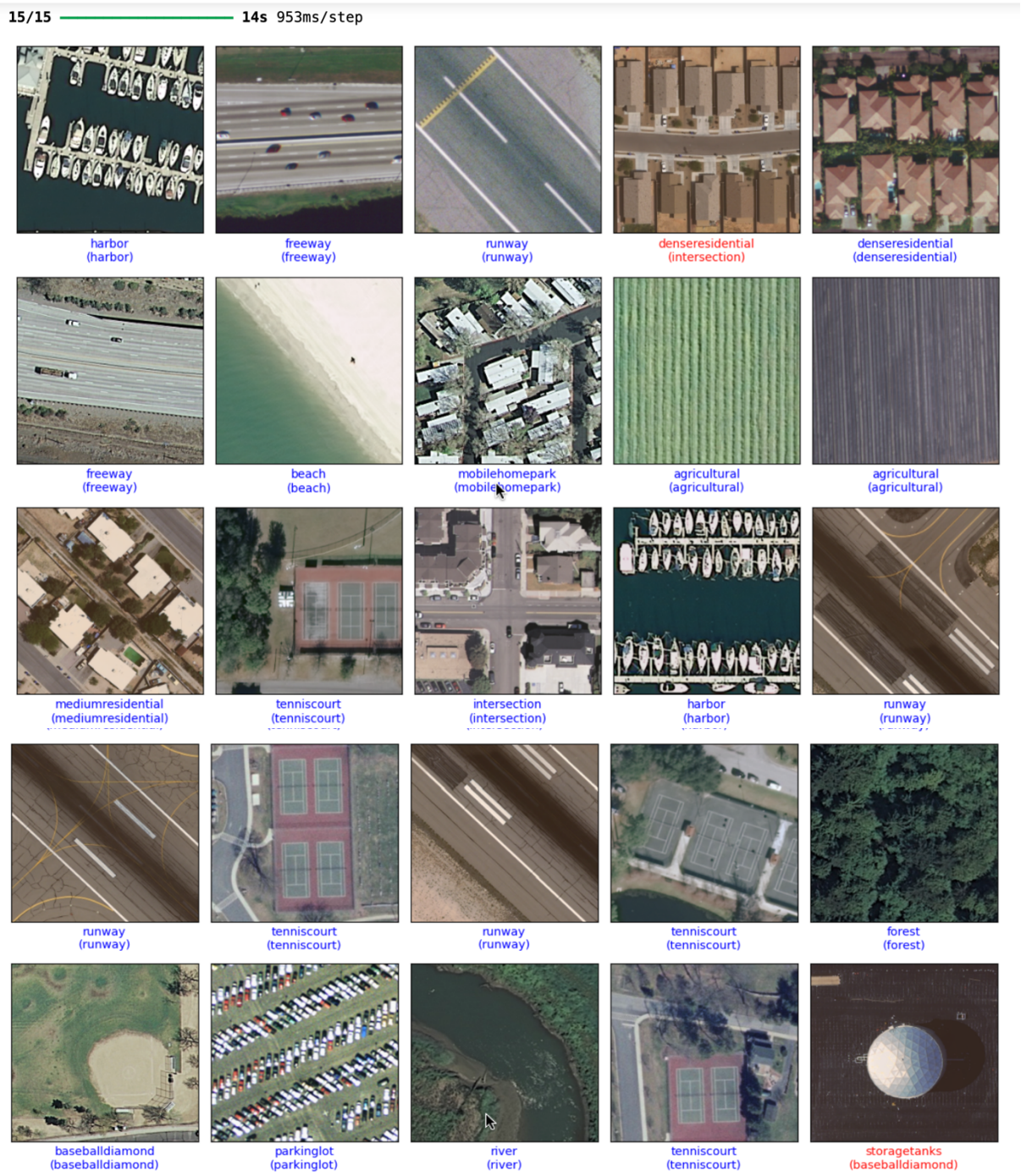

Figure: Prediction of VGG16

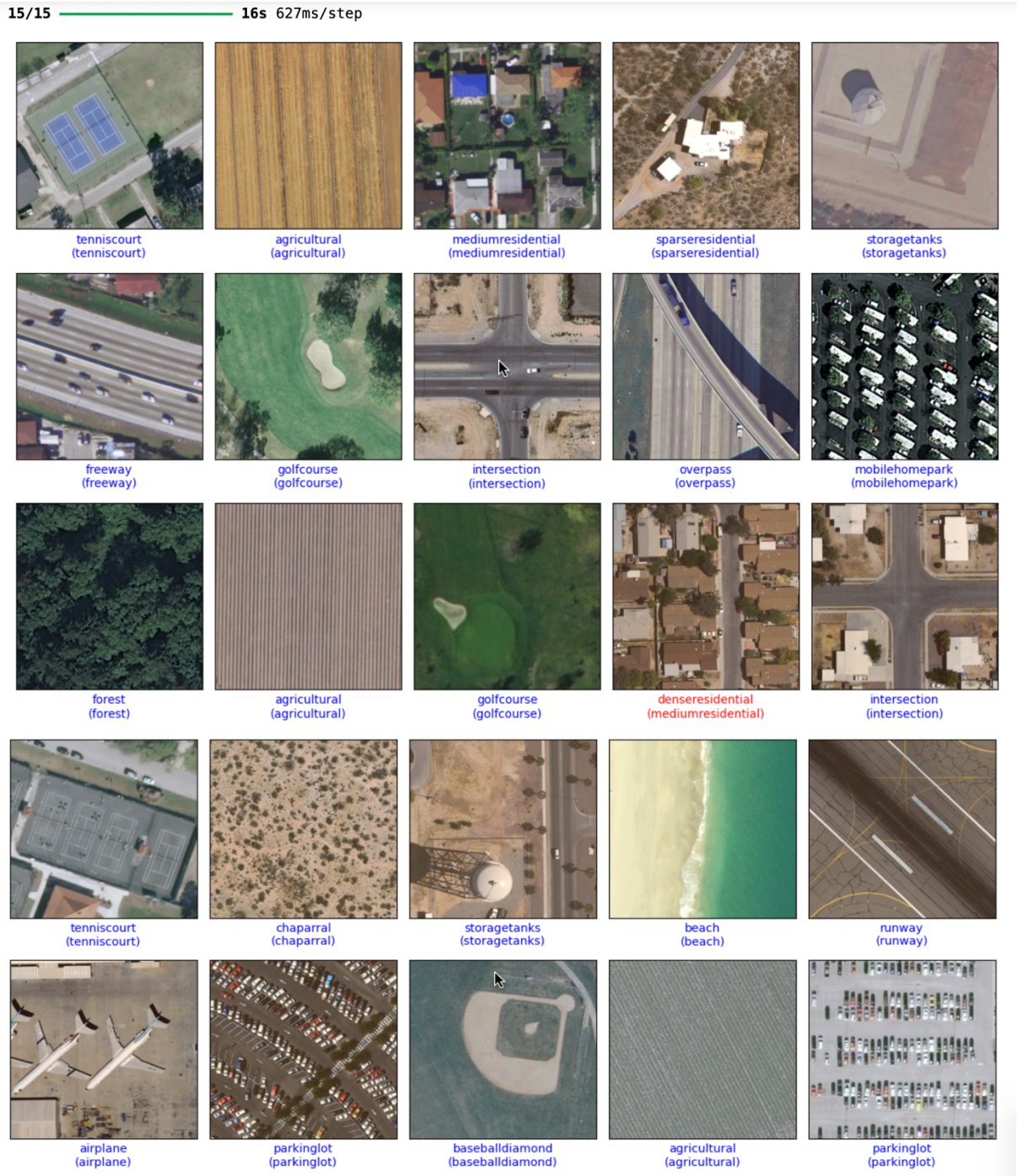

Figure: Prediction of InceptionV3

Detailed Analysis and Interpretation

Model Performance Comparison

Accuracy

- VGG16
    - Test accuracy: 86.58%
    - Test precision: 94.25%
    - Test recall: 74.46%

- InceptionV3
    - Test accuracy: 92.21%
    - Test precision: 96.07%
    - Test recall: 84.63%

Precision, Recall, and F1-Score

Precision, Recall, and F1-score for each class were calculated and plotted for both models. Both models demonstrated high precision and recall across most classes, with the InceptionV3 model slightly outperforming VGG16.

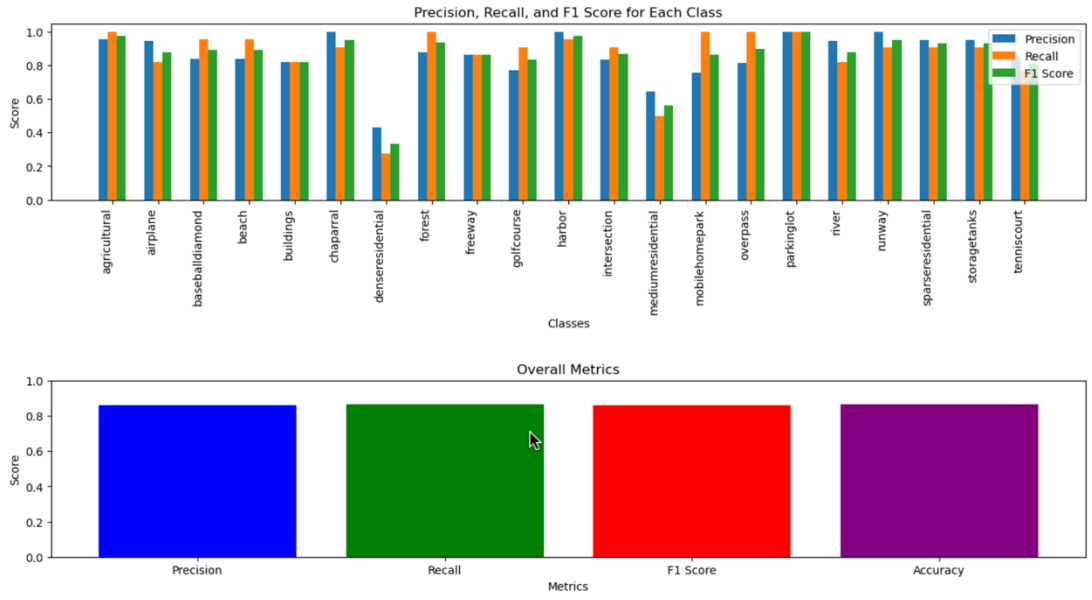

Figure: Precision, Recall and F1-Score of VGG16

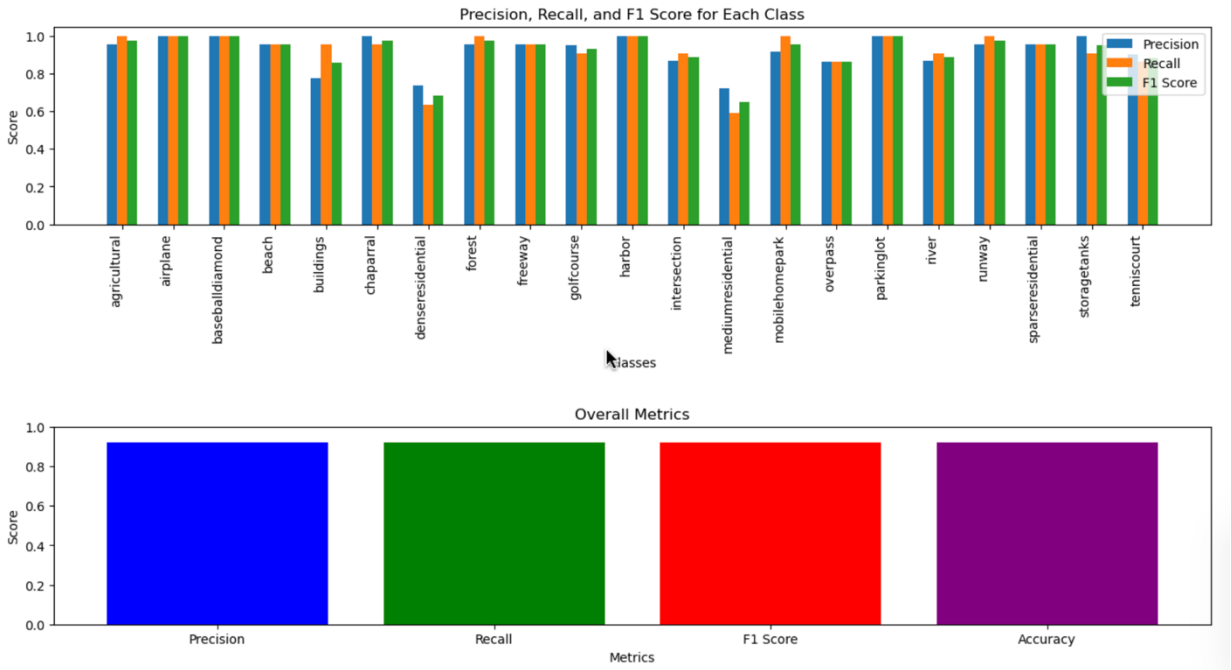

Figure: Precision, Recall and F1-Score of InceptionV3

Classification Report

The classification report provides detailed insights into the model's performance on a per-class basis, including precision, recall, and F1-score for each class.

```
Classification Report:
                    precision    recall  f1-score   support

      agricultural       0.96      1.00      0.98        22
          airplane       0.95      0.82      0.88        22
  baseballdiamond       0.84      0.95      0.89        22
             beach       0.84      0.95      0.89        22
         buildings       0.82      0.82      0.82        22
        chaparral        1.00      0.91      0.95        22
 denseresidential       0.43      0.27      0.33        22
           forest       0.88      1.00      0.94        22
          freeway       0.86      0.86      0.86        22
       golfcourse       0.77      0.91      0.83        22
           harbor       1.00      0.95      0.98        22
     intersection       0.83      0.91      0.87        22
mediumresidential       0.65      0.50      0.56        22
   mobilehomepark       0.76      1.00      0.86        22
         overpass       0.81      1.00      0.90        22
       parkinglot       1.00      1.00      1.00        22
            river       0.95      0.82      0.88        22
           runway       1.00      0.91      0.95        22
sparseresidential       0.95      0.91      0.93        22
      storagetanks       0.95      0.91      0.93        22
       tenniscourt       0.85      0.77      0.81        22

         accuracy                           0.87       462
        macro avg       0.86      0.87      0.86       462
     weighted avg       0.86      0.87      0.86       462
```

Fifure: Classification Report of VGG16

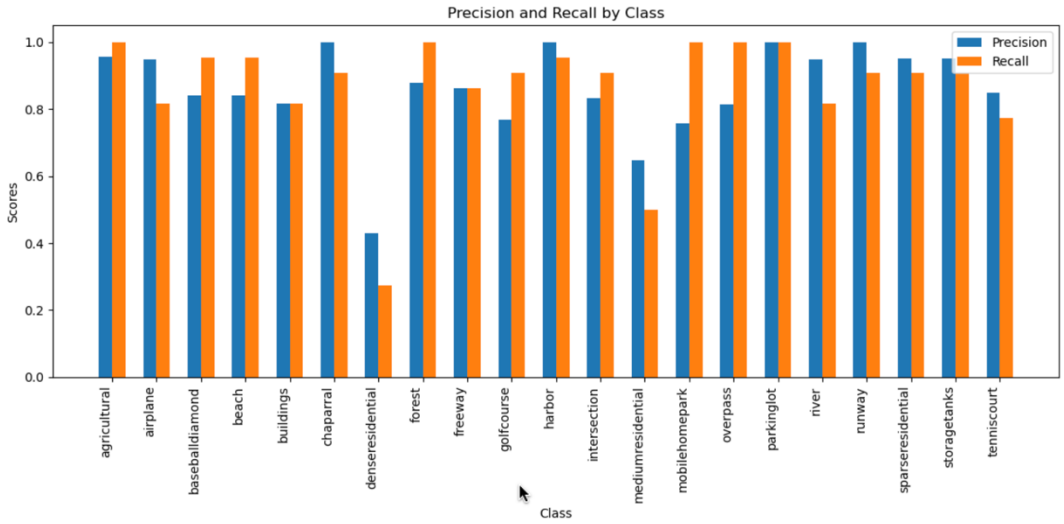

Figure: Precision and Recall (by class) of VGG16

Classification Report:

	precision	recall	f1-score	support
agricultural	0.96	1.00	0.98	22
airplane	1.00	1.00	1.00	22
baseballdiamond	1.00	1.00	1.00	22
beach	0.95	0.95	0.95	22
buildings	0.78	0.95	0.86	22
chaparral	1.00	0.95	0.98	22
denseresidential	0.74	0.64	0.68	22
forest	0.96	1.00	0.98	22
freeway	0.95	0.95	0.95	22
golfcourse	0.95	0.91	0.93	22
harbor	1.00	1.00	1.00	22
intersection	0.87	0.91	0.89	22
mediumresidential	0.72	0.59	0.65	22
mobilehomepark	0.92	1.00	0.96	22
overpass	0.86	0.86	0.86	22
parkinglot	1.00	1.00	1.00	22
river	0.87	0.91	0.89	22
runway	0.96	1.00	0.98	22
sparseresidential	0.95	0.95	0.95	22
storagetanks	1.00	0.91	0.95	22
tenniscourt	0.90	0.86	0.88	22
accuracy			0.92	462
macro avg	0.92	0.92	0.92	462
weighted avg	0.92	0.92	0.92	462

Figure: Classification report of InceptionV3

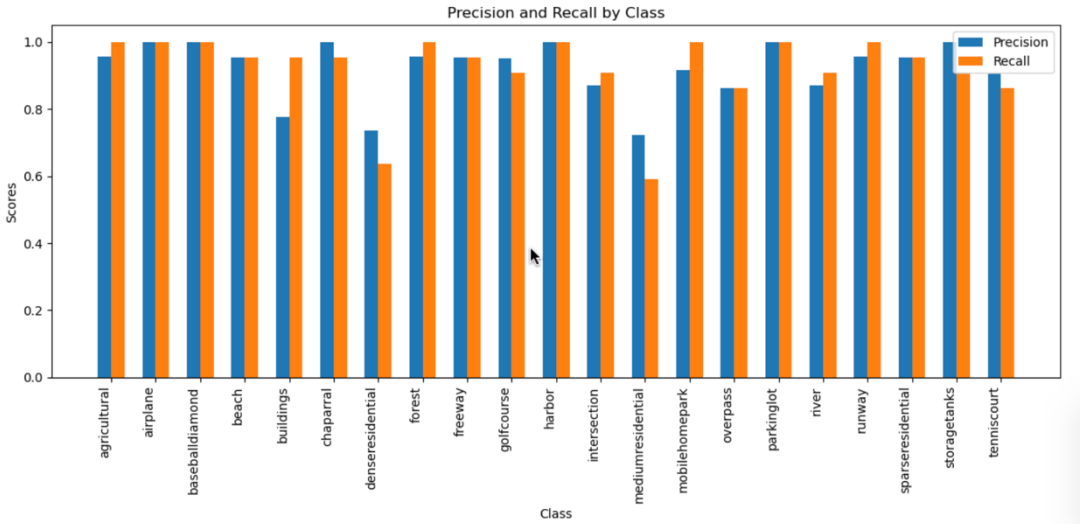

Figure: Precision and Recall (by class) of InceptionV3

True Positive Rate (TPR) and False Positive Rate (FPR)

True Positive Rate (TPR) and False Positive Rate (FPR) were calculated and plotted for each class. Both models showed high TPRs and low FPRs, with InceptionV3 performing slightly better.

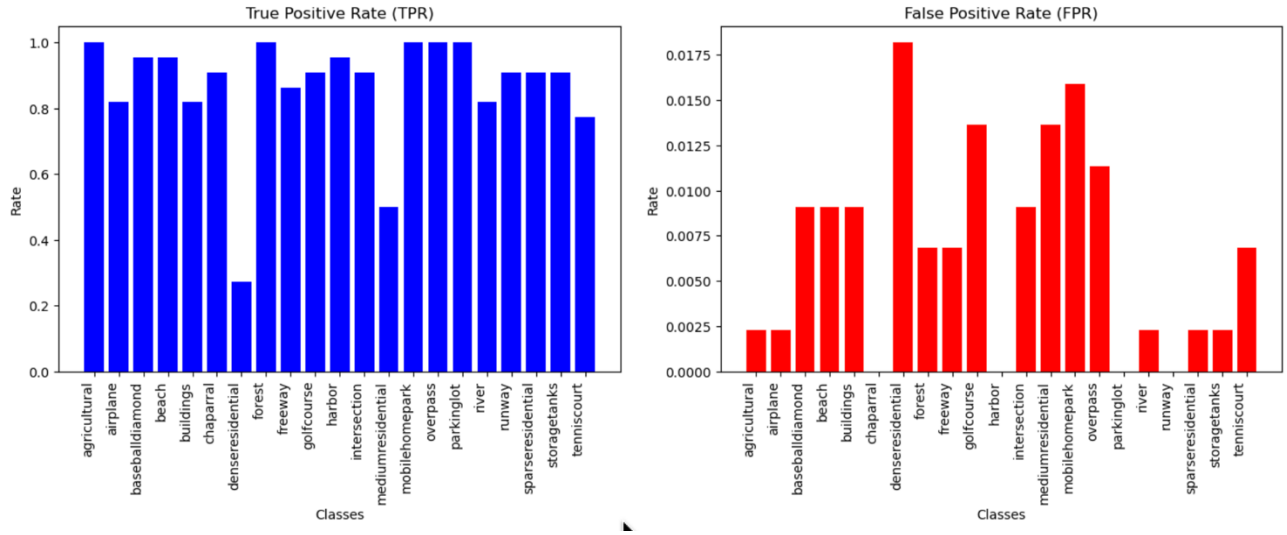

Figure: TPR and FPR of VGG16

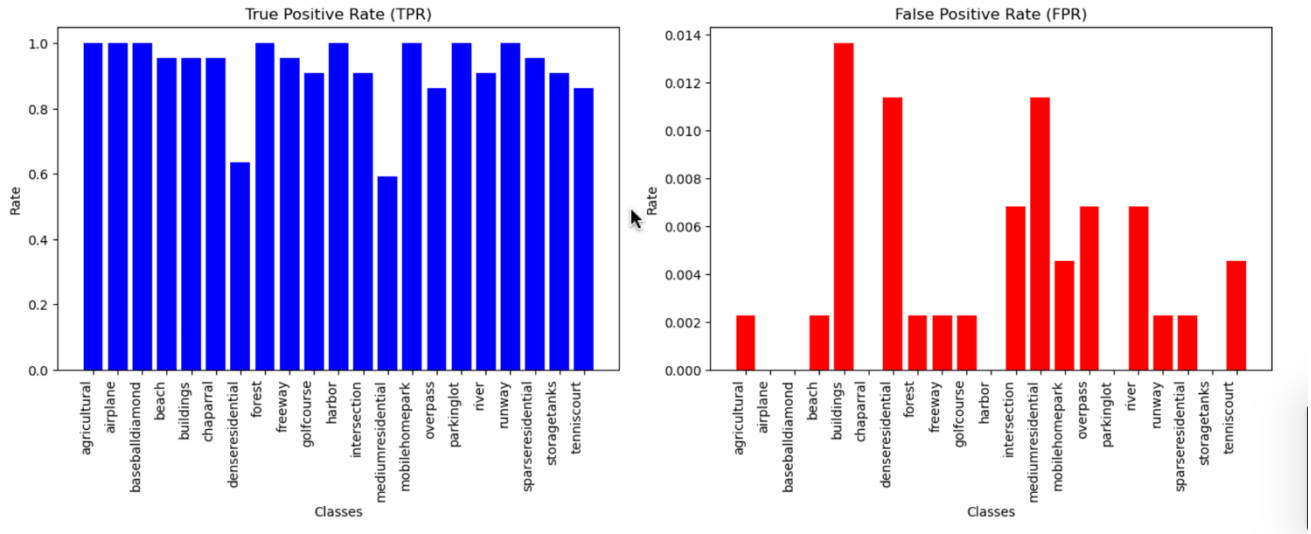

Figure: TPR and FPR of InceptionV3

Confusion Matrix

The confusion matrix for both models was plotted to provide a visual representation of the model performance across all classes. Both models showed higher true positive rates along the diagonal, indicating correct classifications.

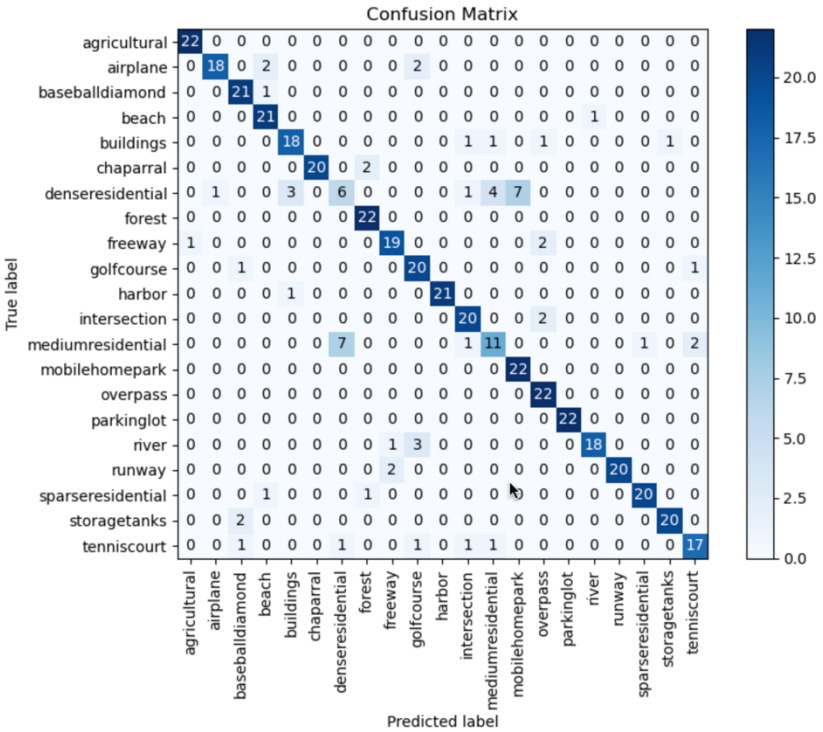

Figure: Confusion Matrix of VGG16

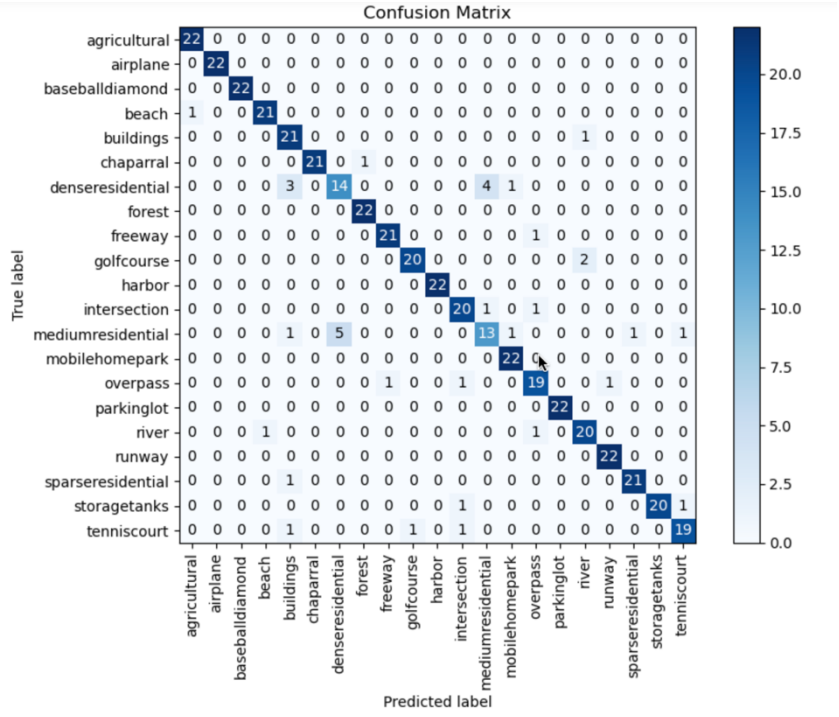

Figure: Confusion Matrix of InceptionV3

Conclusion

Both the VGG16 and InceptionV3 models were evaluated for their ability to classify land cover types from satellite imagery. The InceptionV3 model slightly outperformed the VGG16 model, achieving higher accuracy, precision, recall, and F1-scores.

Reasons for Difference in Performance:

1. **Model Complexity**: InceptionV3's architecture is more advanced and capable of capturing complex patterns in the data.
2. **Transfer Learning Adaptability**: Both models benefit from transfer learning, but InceptionV3's pre-trained weights on ImageNet seem to transfer better to this specific task.
3. **Regularization Techniques**: Both models use dropout and batch normalization, but InceptionV3's architecture may inherently handle regularization better.
4. **Data Augmentation**: The applied data augmentation techniques were effective in helping both models generalize better.

The results demonstrate the potential for using CNNs in land cover classification, providing valuable insights for applications in environmental monitoring, urban planning, and disaster response.

References

- Yang, Y., & Newsam, S. (2010). Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems* (pp. 270-279). ACM. https://doi.org/10.1145/1869790.1869829
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770-778). https://doi.org/10.1109/CVPR.2016.90
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems* (Vol. 27). https://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf
- UC Merced Land Use Dataset. (n.d.). Retrieved July 30, 2024, from https://vision.ucmerced.edu/datasets/landuse.html