Forest Fire Detection Using CNN: Detect Forest Fires from Images

Contents

Introduction: 2
Literature Review 2
Dataset Description: 2
Methodology 2
  Data Preparation 2
    Data Augmentation: 2
    Normalization 3
  Model Architecture: 3
  Training Process 3
  Evaluation Metrics 3
Data Sample: 3
ResNet18 vs. DenseNet121: Performance Metrics 5
  Training Metrics: 6
    ResNet18 6
    DenseNet121 6
    8
  Validation Metrics: 8
    ResNet18 8
    DenseNet121 8
  Testing Metrics: 10
    ResNet18 10
    DenseNet121 10
  Precision-Recall Curve (PRC) Analysis 11
  Confusion Matrices: 12
Visualizations: 13
  Visual Evaluation: 14
Discussion: 15
Future Scope: 15
References: 16

Introduction:

Forest fires are a significant environmental issue, causing severe damage to ecosystems, property, and even human life. Detecting forest fires at an early stage is crucial for minimizing these adverse effects. In this project, I will implement a Convolutional Neural Network (CNN) model to classify images as either containing fire or not, based on a dataset of outdoor images.

Literature Review

Recent advances in machine learning, particularly deep learning, have shown promising results in image classification tasks including natural disaster monitoring and emergency response. Convolutional Neural Networks (CNNs) have become the gold standard for these tasks due to their ability to extract high-level features from visual data. Studies have demonstrated the effectiveness of CNNs in detecting wildfires from satellite images and surveillance feeds. The use of deep architectures like ResNet and DenseNet has been particularly noted for their deep residual learning framework, which helps in training very deep networks.

Dataset Description:

The dataset utilized in this project consists of labeled images sourced from publicly available datasets on Kaggle, Roboflow and B2FIND.

The dataset at source had two categories: fire and non-fire images. I have further divided the data into training and testing sets each of which now contains two subfolders named as fire_images and non_fire_images.

fire_images: Contains images depicting outdoor fires.
non_fire_images: Contains images of natural scenes without fire.

Data Count:
Training: fire images = 2296
Training: non-fire images = 2332
Testing: fire images = 454
Testing: non-fire images = 515

The dataset is imbalanced as I have more fire images than non-fire images.
The dataset is imbalanced, with more fire images than non-fire images. Therefore, I will process the data to ensure a balanced validation set during training.

Methodology

Data Preparation

Loading and Splitting the Data: Data are from the respective directories and assigned label 0 and 1 respectively for non-fire and fire images to perform supervised learning.

Data Augmentation: To enhance the model's robustness and generalizability, the following transformations are applied:

1. Resizing to 224x224 pixels
2. Random horizontal flipping
3. Random rotation of 15 degrees

Normalization: Images are normalized using mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225] values typical for models pre-trained on ImageNet.

Model Architecture:

For first model I have used ResNet18 model which will be pretrained on the ImageNet dataset as the base model. I the final iteration I am planning to include DenseNet121 model and compare both models performance. The ResNet18 architecture is well-suited for image classification tasks due to its deep residual learning framework and will be beneficial for this type of dataset.

- Pre-trained on ImageNet: Utilizes learned features from a vast and diverse dataset, providing a strong starting point for feature extraction.
- Modification: The final fully connected layer is adjusted to output two classes, corresponding to fire and non-fire.

Training Process

- Optimizer: Adam optimizer, known for its adaptive learning rate capabilities, is used.
- Loss Function: CrossEntropyLoss, which is suitable for binary classification tasks.
- Epochs and Batches: The model is trained over multiple epochs in batches of 32 images, allowing for efficient gradient updates.

Evaluation Metrics

Performance was evaluated using the following metrics:

Precision: Indicates the ratio of correctly predicted positive observations to the total predicted positives. Higher precision relates to a lower false positive rate.
Recall (TPR): Indicates the ratio of correctly predicted positive observations to all actual positives. It shows the model's ability to find all the relevant cases (fires).
F1 Score: A weighted average of Precision and Recall. This score takes both false positives and false negatives into account.
Accuracy: Indicates the overall correctness of the model.
FPR: The proportion of non-fire instances that are incorrectly classified as fire.
Area Under Precision-Recall Curve (AUPRC): A comprehensive measure of model performance, especially useful in imbalanced datasets.

Data Sample:

Fig 1: Training data sample (fire)

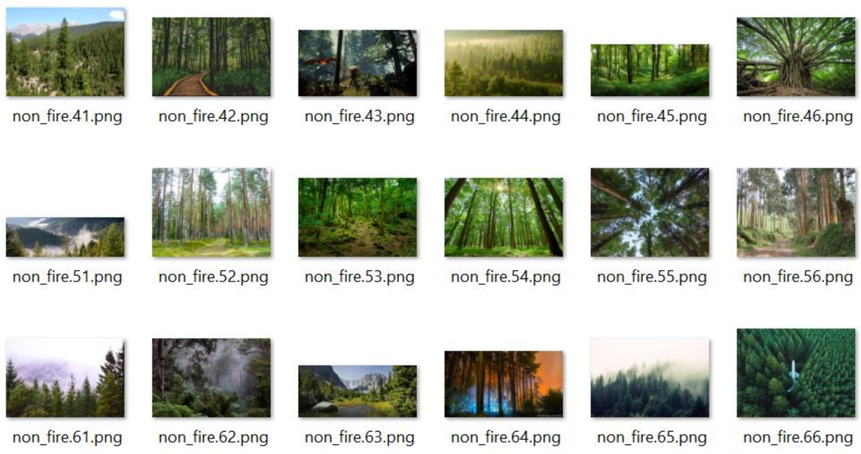


Fig 2: Training data sample (non-fire)

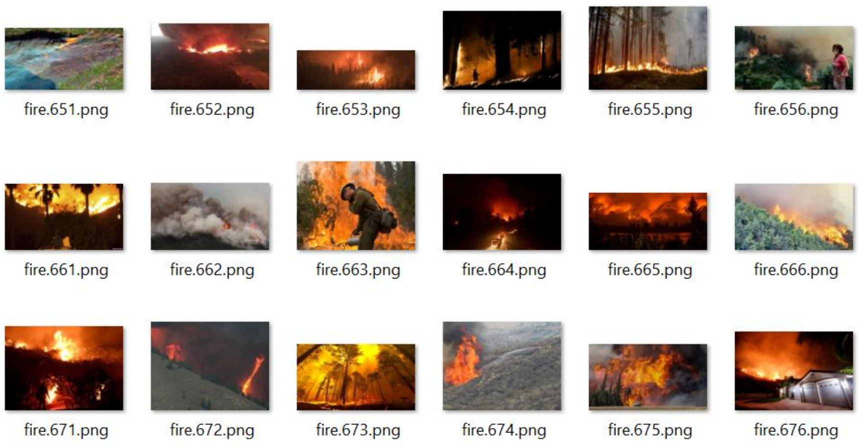


Fig 3: Testing data sample (fire)

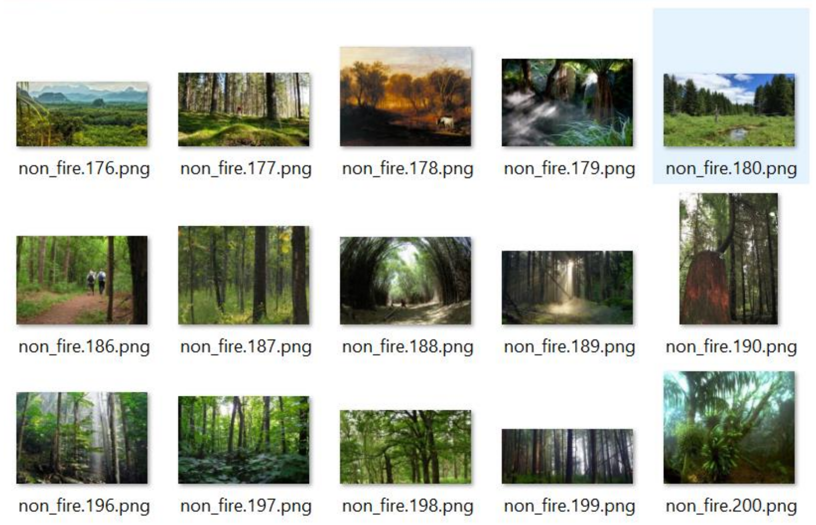
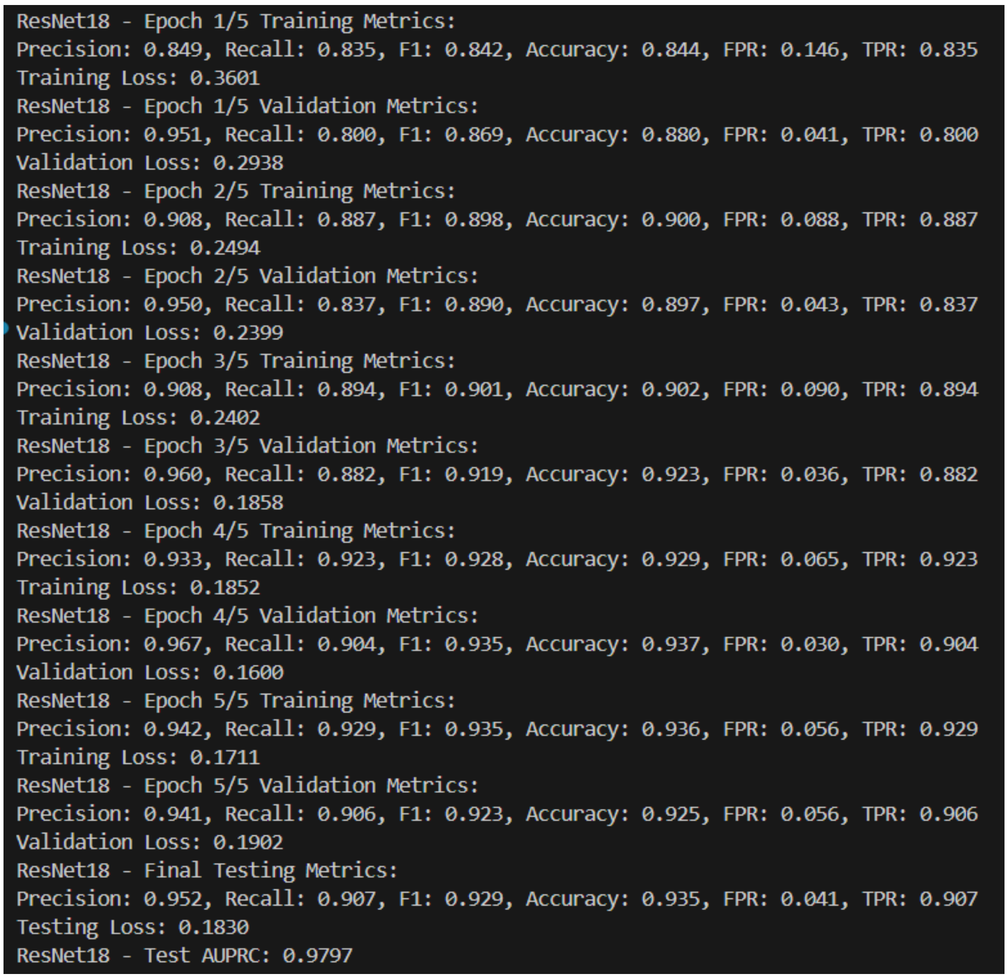


Fig 4: Testing data sample (non-fire)

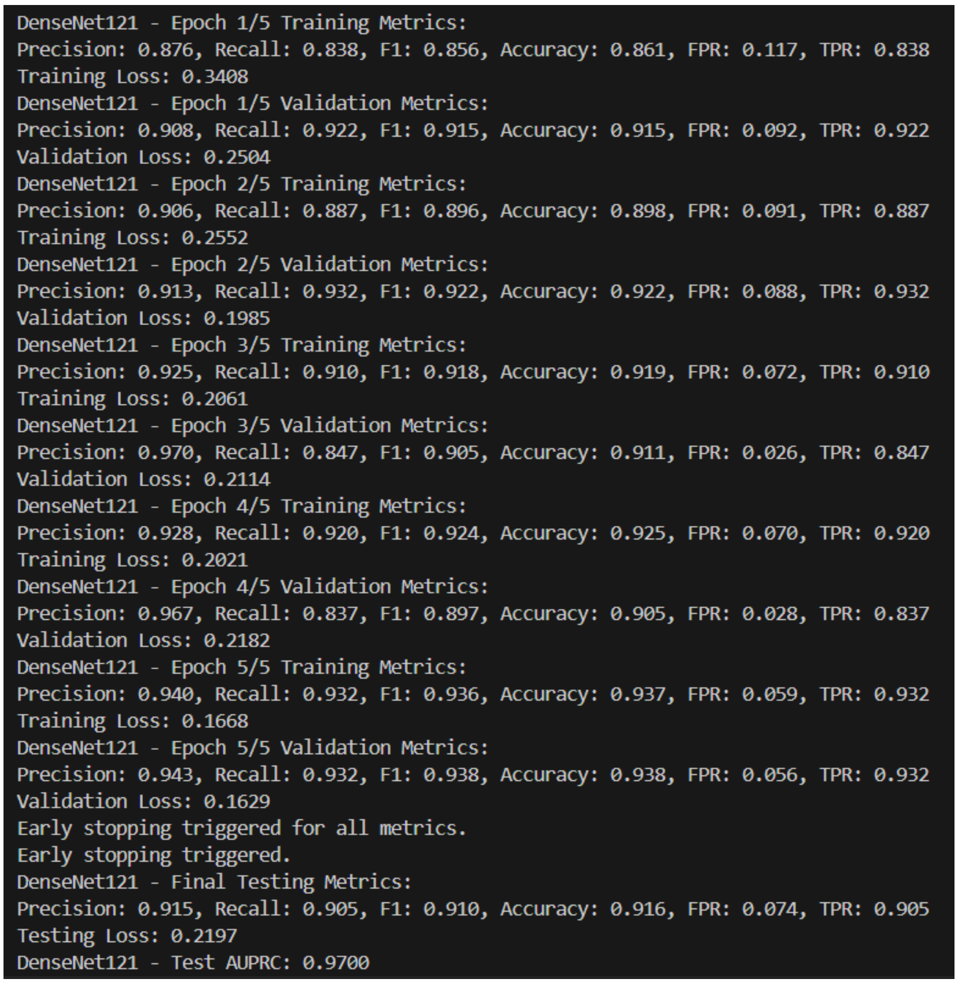
ResNet18 vs. DenseNet121: Performance Metrics

We will go through the performance matrices for each model and analyze the performances and compare them. We will also show relevant visualizations for better understanding.



```
ResNet18 - Epoch 1/5 Training Metrics:
Precision: 0.849, Recall: 0.835, F1: 0.842, Accuracy: 0.844, FPR: 0.146, TPR: 0.835
Training Loss: 0.3601
ResNet18 - Epoch 1/5 Validation Metrics:
Precision: 0.951, Recall: 0.800, F1: 0.869, Accuracy: 0.880, FPR: 0.041, TPR: 0.800
Validation Loss: 0.2938
ResNet18 - Epoch 2/5 Training Metrics:
Precision: 0.908, Recall: 0.887, F1: 0.898, Accuracy: 0.900, FPR: 0.088, TPR: 0.887
Training Loss: 0.2494
ResNet18 - Epoch 2/5 Validation Metrics:
Precision: 0.950, Recall: 0.837, F1: 0.890, Accuracy: 0.897, FPR: 0.043, TPR: 0.837
Validation Loss: 0.2399
ResNet18 - Epoch 3/5 Training Metrics:
Precision: 0.908, Recall: 0.894, F1: 0.901, Accuracy: 0.902, FPR: 0.090, TPR: 0.894
Training Loss: 0.2402
ResNet18 - Epoch 3/5 Validation Metrics:
Precision: 0.960, Recall: 0.882, F1: 0.919, Accuracy: 0.923, FPR: 0.036, TPR: 0.882
Validation Loss: 0.1858
ResNet18 - Epoch 4/5 Training Metrics:
Precision: 0.933, Recall: 0.923, F1: 0.928, Accuracy: 0.929, FPR: 0.065, TPR: 0.923
Training Loss: 0.1852
ResNet18 - Epoch 4/5 Validation Metrics:
Precision: 0.967, Recall: 0.904, F1: 0.935, Accuracy: 0.937, FPR: 0.030, TPR: 0.904
Validation Loss: 0.1600
ResNet18 - Epoch 5/5 Training Metrics:
Precision: 0.942, Recall: 0.929, F1: 0.935, Accuracy: 0.936, FPR: 0.056, TPR: 0.929
Training Loss: 0.1711
ResNet18 - Epoch 5/5 Validation Metrics:
Precision: 0.941, Recall: 0.906, F1: 0.923, Accuracy: 0.925, FPR: 0.056, TPR: 0.906
Validation Loss: 0.1902
ResNet18 - Final Testing Metrics:
Precision: 0.952, Recall: 0.907, F1: 0.929, Accuracy: 0.935, FPR: 0.041, TPR: 0.907
Testing Loss: 0.1830
ResNet18 - Test AUPRC: 0.9797
```

Fig 5: ResNet18 model’s performance over each epoch



```
DenseNet121 - Epoch 1/5 Training Metrics:
Precision: 0.876, Recall: 0.838, F1: 0.856, Accuracy: 0.861, FPR: 0.117, TPR: 0.838
Training Loss: 0.3408
DenseNet121 - Epoch 1/5 Validation Metrics:
Precision: 0.908, Recall: 0.922, F1: 0.915, Accuracy: 0.915, FPR: 0.092, TPR: 0.922
Validation Loss: 0.2504
DenseNet121 - Epoch 2/5 Training Metrics:
Precision: 0.906, Recall: 0.887, F1: 0.896, Accuracy: 0.898, FPR: 0.091, TPR: 0.887
Training Loss: 0.2552
DenseNet121 - Epoch 2/5 Validation Metrics:
Precision: 0.913, Recall: 0.932, F1: 0.922, Accuracy: 0.922, FPR: 0.088, TPR: 0.932
Validation Loss: 0.1985
DenseNet121 - Epoch 3/5 Training Metrics:
Precision: 0.925, Recall: 0.910, F1: 0.918, Accuracy: 0.919, FPR: 0.072, TPR: 0.910
Training Loss: 0.2061
DenseNet121 - Epoch 3/5 Validation Metrics:
Precision: 0.970, Recall: 0.847, F1: 0.905, Accuracy: 0.911, FPR: 0.026, TPR: 0.847
Validation Loss: 0.2114
DenseNet121 - Epoch 4/5 Training Metrics:
Precision: 0.928, Recall: 0.920, F1: 0.924, Accuracy: 0.925, FPR: 0.070, TPR: 0.920
Training Loss: 0.2021
DenseNet121 - Epoch 4/5 Validation Metrics:
Precision: 0.967, Recall: 0.837, F1: 0.897, Accuracy: 0.905, FPR: 0.028, TPR: 0.837
Validation Loss: 0.2182
DenseNet121 - Epoch 5/5 Training Metrics:
Precision: 0.940, Recall: 0.932, F1: 0.936, Accuracy: 0.937, FPR: 0.059, TPR: 0.932
Training Loss: 0.1668
DenseNet121 - Epoch 5/5 Validation Metrics:
Precision: 0.943, Recall: 0.932, F1: 0.938, Accuracy: 0.938, FPR: 0.056, TPR: 0.932
Validation Loss: 0.1629
Early stopping triggered for all metrics.
Early stopping triggered.
DenseNet121 - Final Testing Metrics:
Precision: 0.915, Recall: 0.905, F1: 0.910, Accuracy: 0.916, FPR: 0.074, TPR: 0.905
Testing Loss: 0.2197
DenseNet121 - Test AUPRC: 0.9700
```

Fig 6: DenseNet121 model's performance over each epoch

Training Metrics:

ResNet18:

- Final Epoch Precision: 0.942
- Final Epoch Recall (TPR): 0.929
- Final Epoch F1-Score: 0.935
- Final Epoch Accuracy: 0.936
- Final Epoch FPR: 0.056
- Final Epoch Training Loss: 0.1711

DenseNet121:

- Final Epoch Precision: 0.940
- Final Epoch Recall (TPR): 0.932
- Final Epoch F1-Score: 0.936
- Final Epoch Accuracy: 0.937
- Final Epoch FPR: 0.059
- Final Epoch Training Loss: 0.1668

Observation: Both models exhibit similar precision, recall, F1-score, and accuracy on the training set, indicating they both learned the training data well.

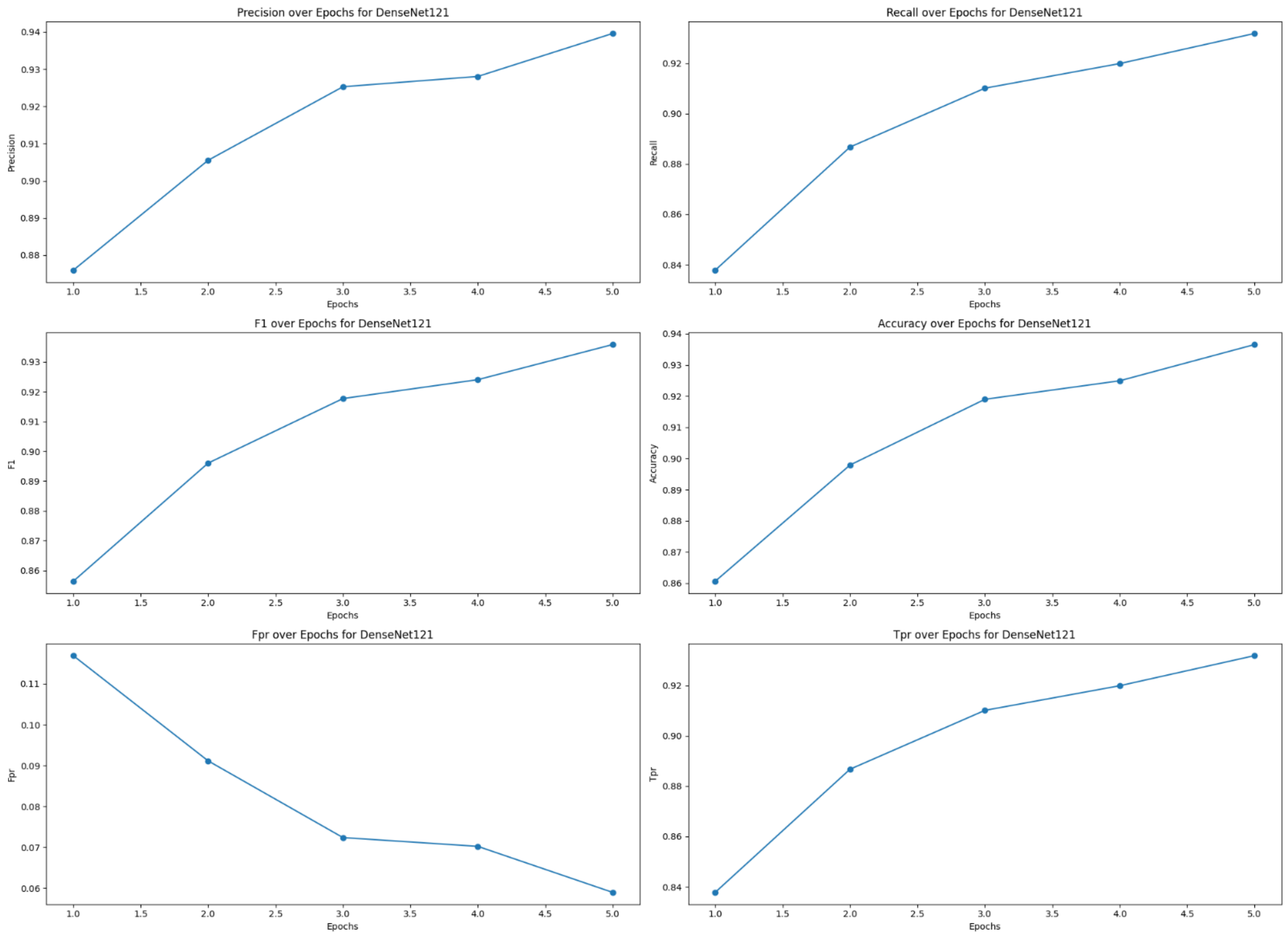


Fig 7: DenseNet121 model's performance over training dataset

Figure 7 shows the performance of each performance metrics of DenseNet121 model over training dataset for each epochs and their progression.

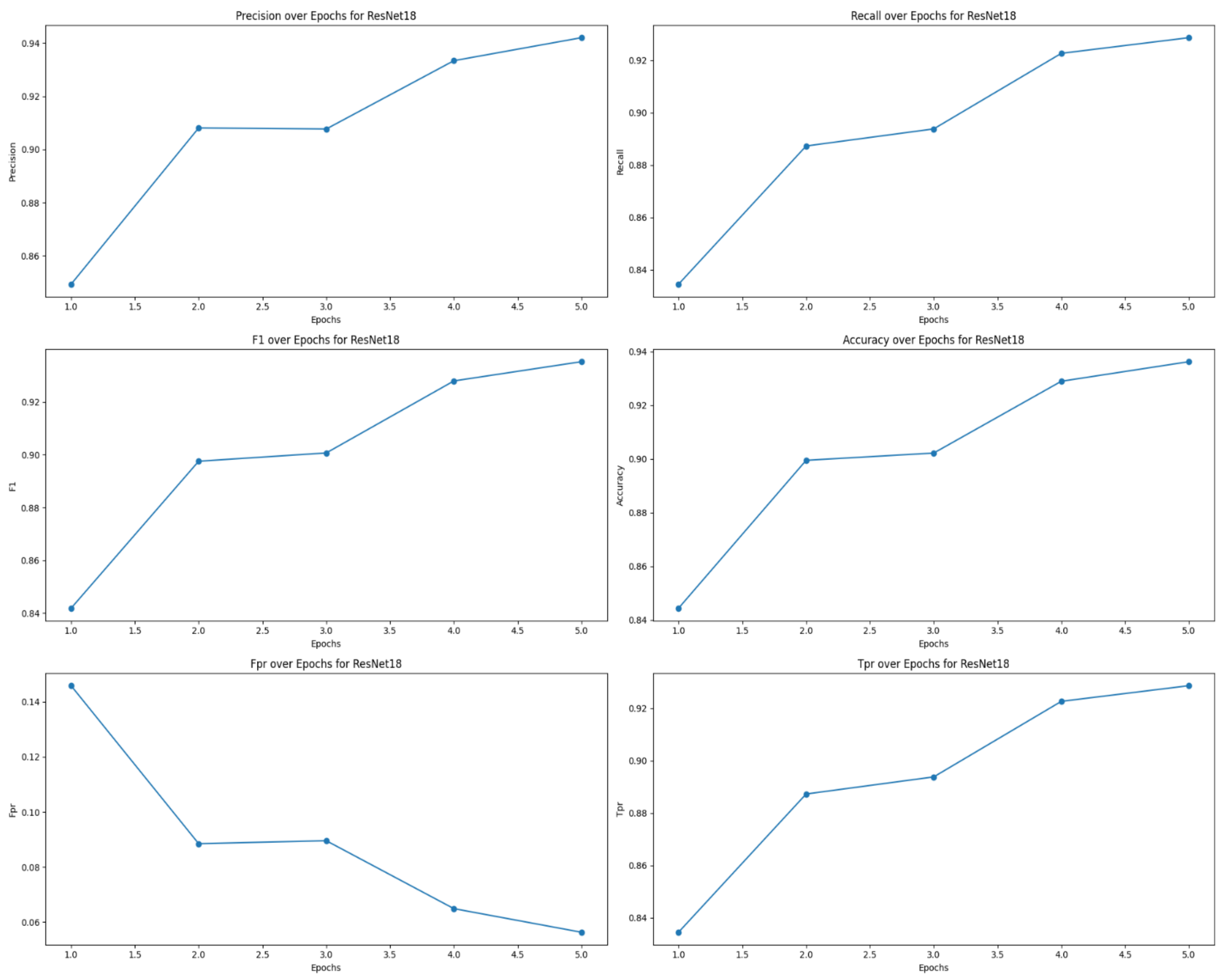


Fig 8: ResNet18 model's performance over training dataset

Figure 8 shows the performance of each performance metrics of ResNet18 model over training dataset for each epochs and their progression.

Validation Metrics:

ResNet18:

- Final Epoch Precision: 0.941
- Final Epoch Recall (TPR): 0.906
- Final Epoch F1-Score: 0.923
- Final Epoch Accuracy: 0.925
- Final Epoch FPR: 0.056
- Final Epoch Validation Loss: 0.1902

DenseNet121:

- Final Epoch Precision: 0.943
- Final Epoch Recall (TPR): 0.932
- Final Epoch F1-Score: 0.938
- Final Epoch Accuracy: 0.938

- Final Epoch FPR: 0.056
- Final Epoch Validation Loss: 0.1629

Observation: DenseNet121 slightly outperforms ResNet18 in terms of precision, recall, F1-score, accuracy, and validation loss, suggesting better generalization on the validation set.

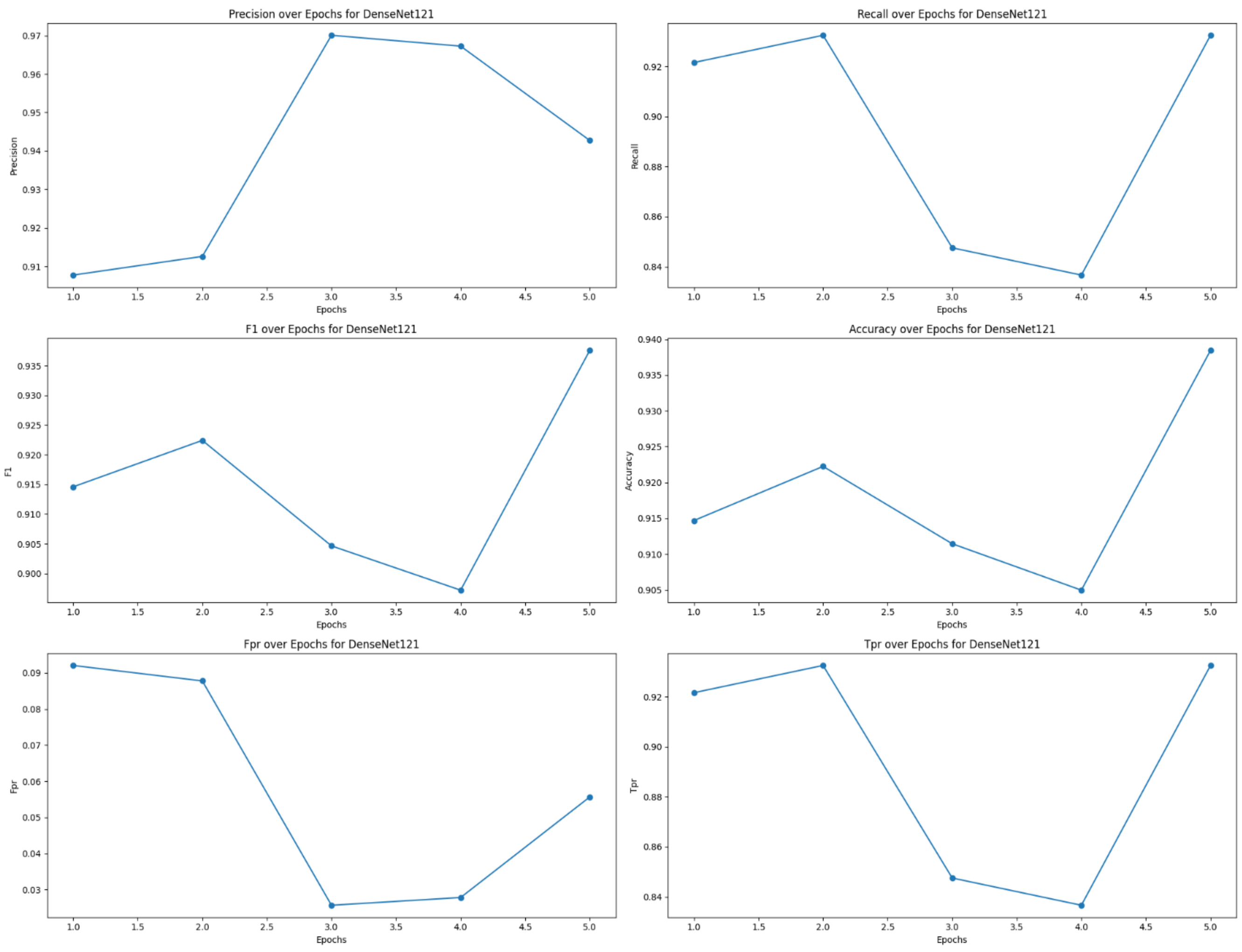


Fig 9: DenseNet121 model's performance over validation dataset

Figure 9 shows the performance of each performance metrics of DenseNet121 model over validation dataset for each epochs and their progression.

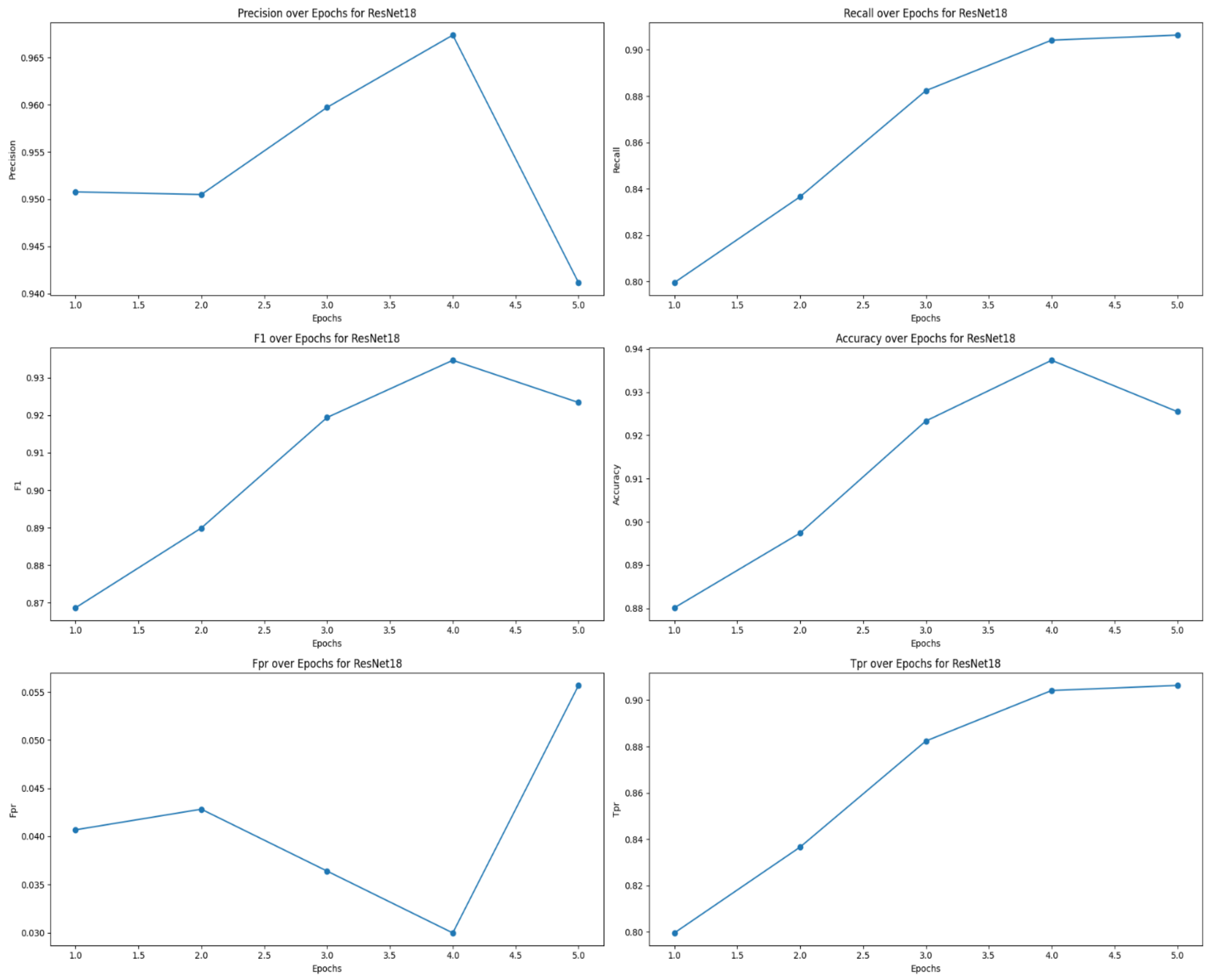


Fig 10: ResNet18 model's performance over validation dataset

Figure 10 shows the performance of each performance metrics of ResNet18 model over validation dataset for each epochs and their progression.

Testing Metrics:

ResNet18:

- Precision: 0.952
- Recall (TPR): 0.907
- F1-Score: 0.929
- Accuracy: 0.935
- FPR: 0.041
- Testing Loss: 0.1830
- AUPRC: 0.9797

DenseNet121:

- Precision: 0.915
- Recall (TPR): 0.905
- F1-Score: 0.910

- Accuracy: 0.916
- FPR: 0.074
- Testing Loss: 0.2197
- AUPRC: 0.9700

Observation: On the test set, ResNet18 achieved higher precision, F1-score, accuracy, and a slightly better AUPRC compared to DenseNet121. However, DenseNet121 had slightly better recall. The AUPRC values suggest that both models are highly effective, but ResNet18 shows a marginally better overall balance between precision and recall.

Precision-Recall Curve (PRC) Analysis

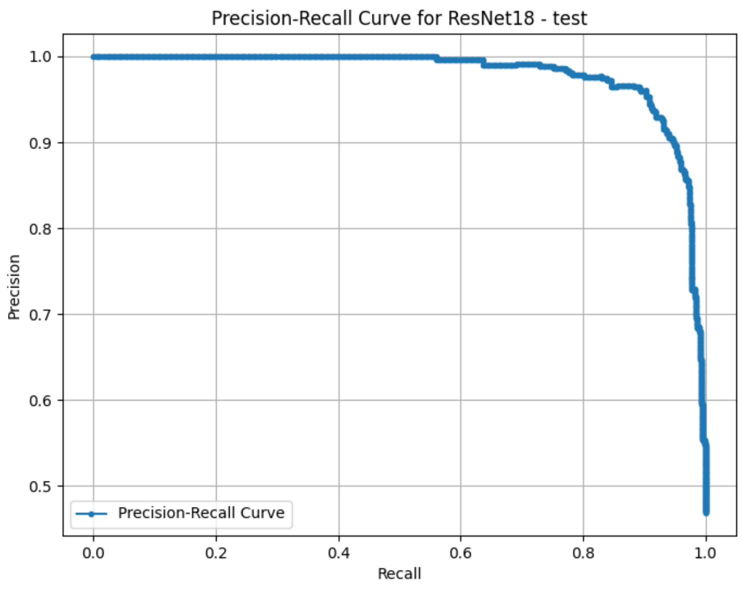


Fig 11: Precision-Recall Curve for Resnet18 model

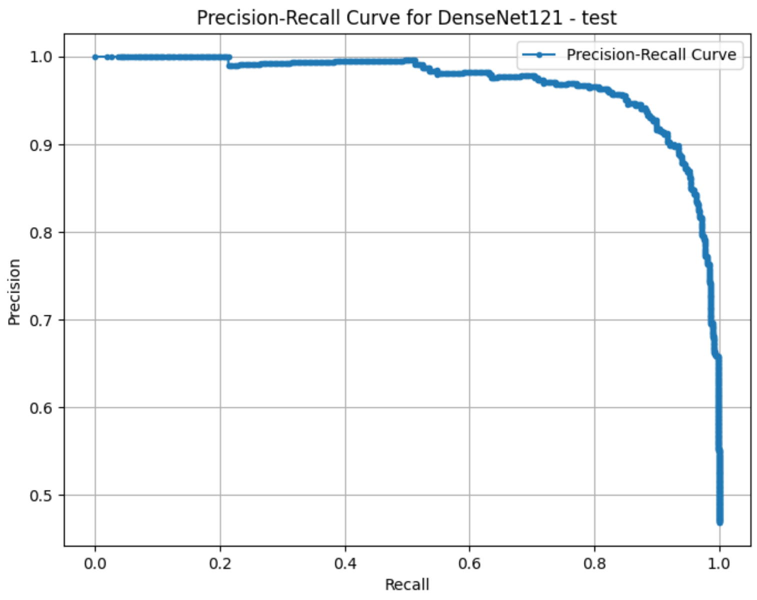


Fig 12: Precision-Recall Curve for Densenet121 model

The AUPRC (Area Under Precision-Recall Curve) is a useful measure for evaluating the trade-offs between precision and recall for different threshold settings, especially in imbalanced datasets.

- **ResNet18** has an AUPRC of 0.9797.
- **DenseNet121** has an AUPRC of 0.9700.

The higher AUPRC for ResNet18 indicates that it performs slightly better in distinguishing between the positive and negative classes across different thresholds. This suggests that ResNet18 may be more reliable in scenarios where maintaining a balance between precision and recall is critical.

Confusion Matrices:

DenseNet121 Confusion Matrix

- True Positives (TP - Fire correctly predicted as Fire): 411
- True Negatives (TN - Non-Fire correctly predicted as Non-Fire): 477
- False Positives (FP - Non-Fire incorrectly predicted as Fire): 38
- False Negatives (FN - Fire incorrectly predicted as Non-Fire): 43

ResNet18 Confusion Matrix

- True Positives (TP - Fire correctly predicted as Fire): 412
- True Negatives (TN - Non-Fire correctly predicted as Non-Fire): 494
- False Positives (FP - Non-Fire incorrectly predicted as Fire): 21
- False Negatives (FN - Fire incorrectly predicted as Non-Fire): 42

ResNet18 demonstrates a superior ability to correctly classify non-fire images and minimize false positives, which is a critical advantage in practical applications where false alarms can have significant negative consequences.

DenseNet121, while still effective, shows a slightly higher tendency to misclassify non-fire images as fire (higher false positives), which can be less desirable in scenarios requiring high precision.

Overall, the confusion matrices reinforce the previous metrics and analysis, confirming that ResNet18 offers a more balanced and reliable performance for the task of fire detection in this dataset.

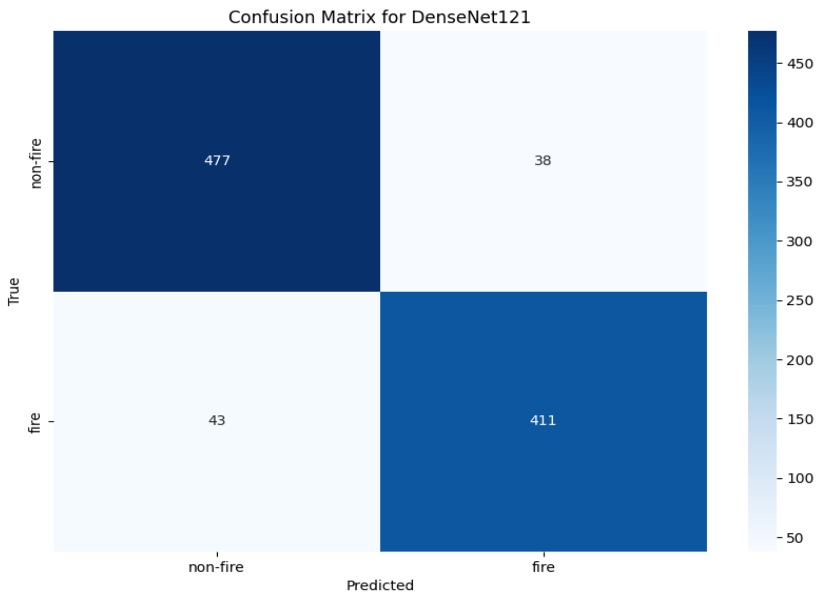


Fig 13: Confusion matrix for DenseNet121

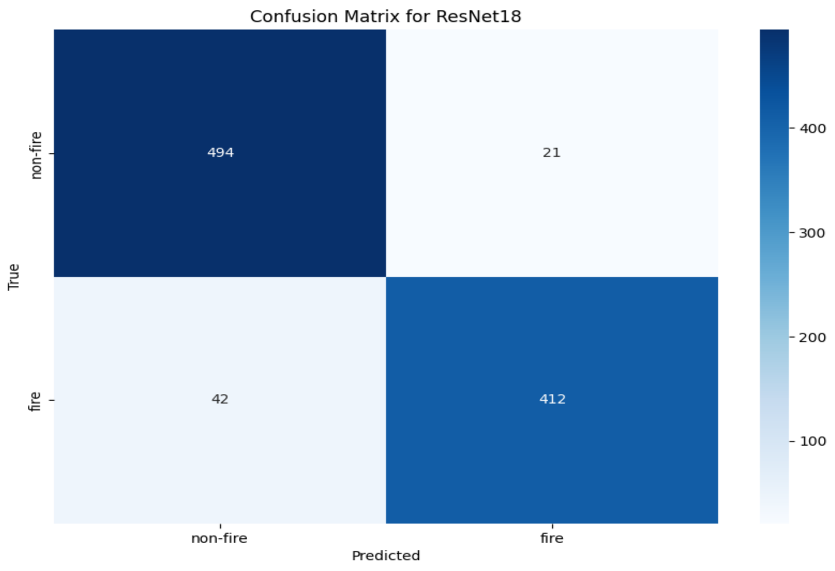


Fig 14: Confusion matrix for ResNet121

Visualizations:

We will check the empirical performance of the models through correctness of actual labeling of the pictures for both models. To do that, we have chosen 20 sample images for each model and labeled them with the predicted and actual outcome.

We will also check the confusion metrics and measure performance of the models.

Visual Evaluation:

Despite DenseNet121's superior performance in quantitative metrics, visual assessment of the classification results revealed some discrepancies. DenseNet121 sometimes misclassified 'Non-Fire' images as 'Fire', possibly due to its sensitivity to patterns resembling fire.

ResNet18 showed a better qualitative performance in some cases, identifying fire images with high accuracy and maintaining lower false positives in practical scenarios. This could be indicative of better generalization on diverse image sets outside of the test metrics.

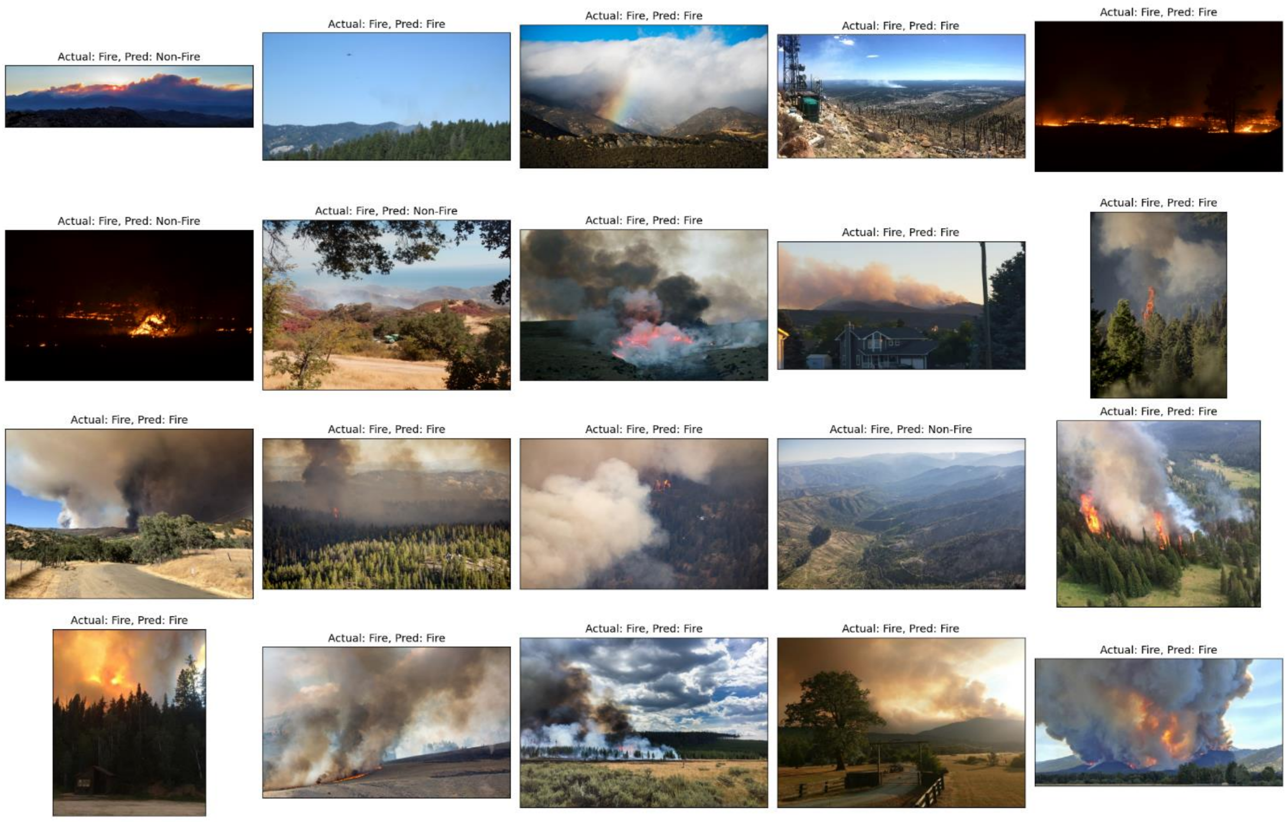


Fig 15: Classification of images using DenseNet121

Form the visualization, we can observe that from 20 samples, DenseNet121 model has correctly identified 16 of them indicating 80% empirical accuracy despite having better performance in quantitative performance analysis. The wrongly classified images indicate that there is a critical learning opportunity for the model at future stages.

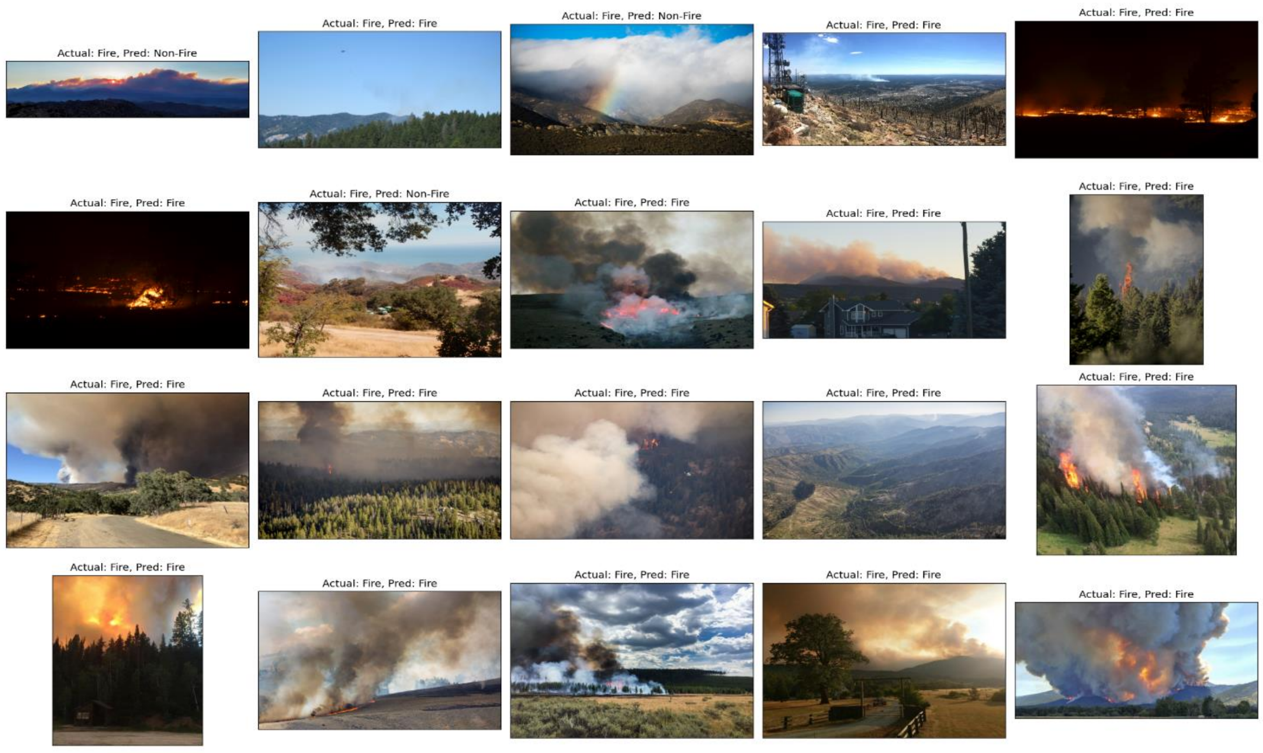


Fig 16: Classification of images using ResNet18

ResNet18 model has shown better performance in terms of labeling the images correctly in visualization. From 20 sample images, ResNet18 model has correctly identified 17 images demonstrating 85% empirical accuracy which is desired in a sensitive case like forest fire detection.

Discussion:

Both ResNet18 and DenseNet121 are capable deep learning models for fire detection, demonstrating high accuracy and reliability. However, ResNet18 slightly outperforms DenseNet121 in key areas, including precision, accuracy, and AUPRC. The lower false positive rate and higher precision make ResNet18 a preferable choice for applications where the cost of false alarms is high.

ResNet18's balanced performance, with fewer fluctuations in metrics and a stronger ability to generalize, indicates its suitability for deployment in real-world fire detection systems. On the other hand, while DenseNet121 shows good potential, it may require further tuning or modifications to achieve the same level of reliability and precision as ResNet18.

Future Scope:

In future, we can focus on further tuning the models. We can do so by expanding the dataset and adding more computational power which will enable us to perform cross validation and hyperparameter tuning. We can check different values of hyperparameter and see which values provides with better generalization. We can explore ensemble methods as well which will combine the strengths of both architectures to improve both sensitivity and specificity.

References:

1. https://www.kaggle.com/datasets/phylake1337/fire-dataset
2. https://www.kaggle.com/code/holdmykaggle/fire-detection-in-images
3. https://www.kaggle.com/code/delllectron/fire-detection-computer-vision
4. https://universe.roboflow.com/fire-detection-3acsn/fire-detection-6xj8n/dataset/4
5. https://b2find.dkrz.de/dataset/d07be6f0-d1c4-5f91-ba06-f225c87c82f8
6. https://medium.com/analytics-vidhya/resnet-understand-and-implement-from-scratch-d0eb9725e0db
7. https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a
8. https://medium.com/deepkapha-notes/implementing-densenet-121-in-pytorch-a-step-by-step-guide-c0c2625c2a60
9. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10125611&casa_token=koXK5x80AiMAAAAA:zDW1EV8scHkqsVC4QGIxul4EPf2q9xvyNujrmHMkJeGEW50NOA-QftcDm7hYX1gH-cQgxVMpDYo&tag=1