ICSI 435/535 - ARTIFICIAL INTELLIGENCE

Lung Cancer Detection using CNN

MEET THE TEAM 13

Jyotsana Parkhedkar

Harshini Narra

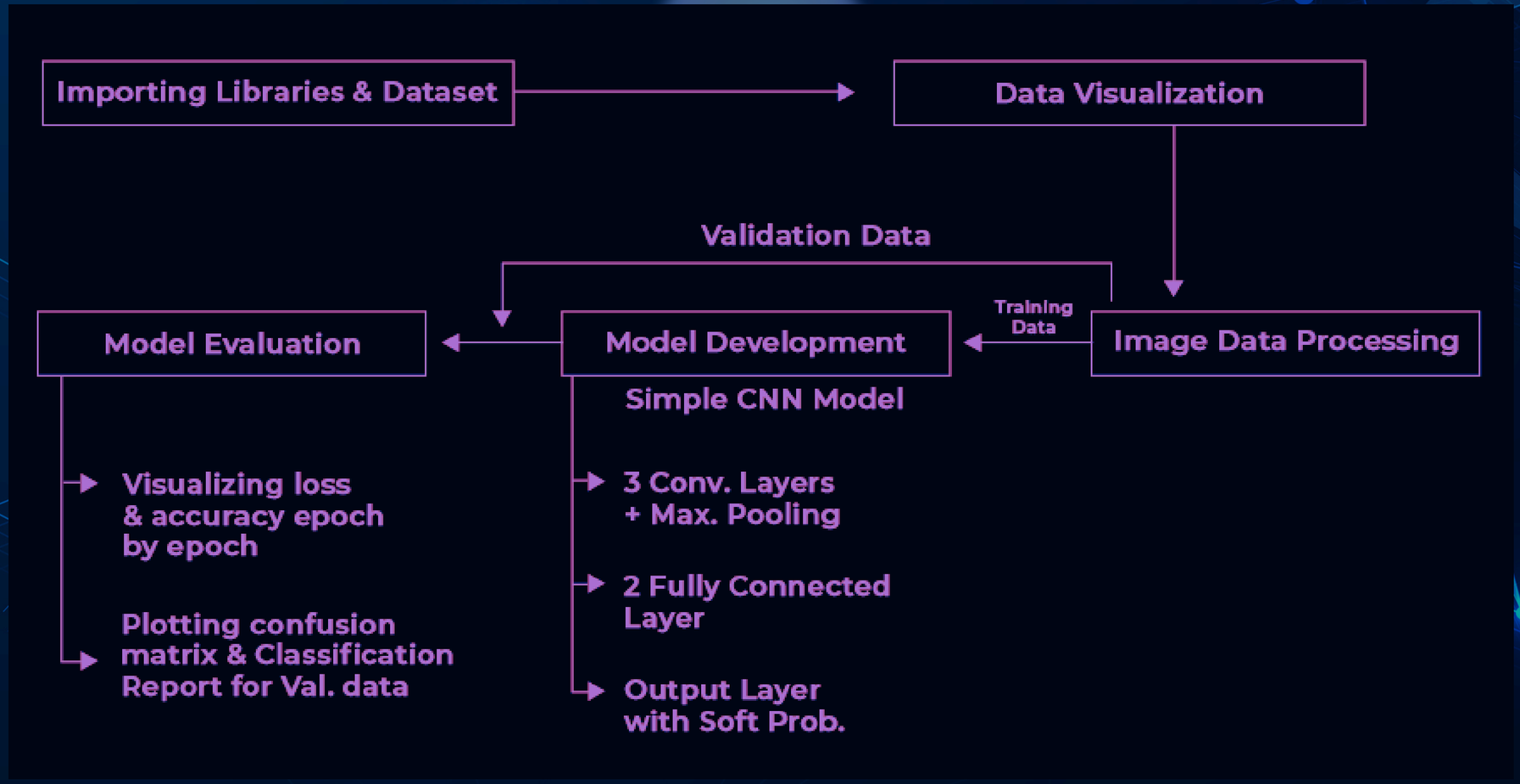
Abhishek Santhakumar

Prathmesh Kale

Rishikesh Sirisilla

PLAN AND MILESTONE

PLAN AND MILESTONE



WORK BREAKDOWN STRUCTURE

01 Data scouting team

Searching for datasets with accurate, consistent and high quality data. Cleaning and prepping the data before training the model.

Harshini
Abhishek
Jyotsana

02 Model training team

Training the CNN model using the data by coding the basic structure with libraries like Keras, Tensorflow and openCV.

Prathmesh
Rishi
Harshini

03 Model optimization team

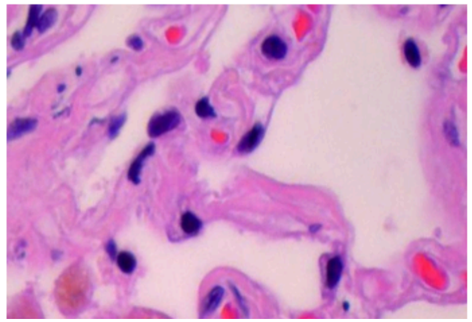
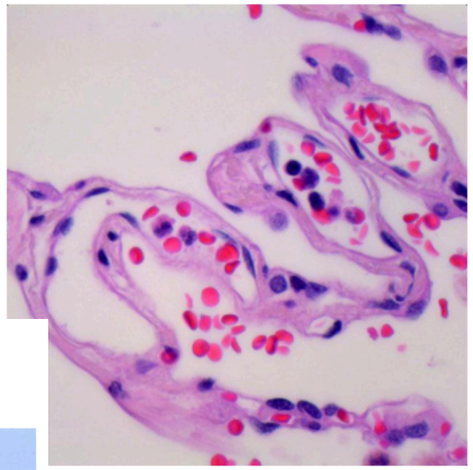
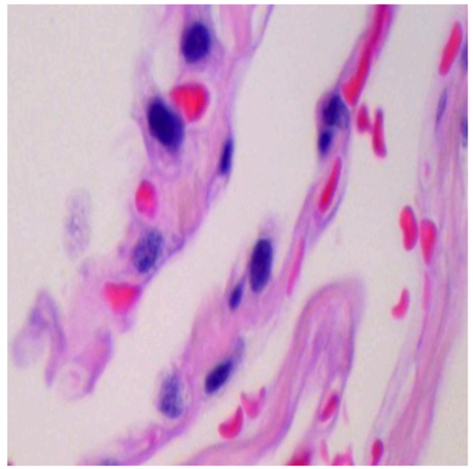
Optimizing the model to ensure its consistency, avoiding bias and overfitting and fine tuning it to produce higher accuracies.

Abhishek
Jyotsana
Rishi

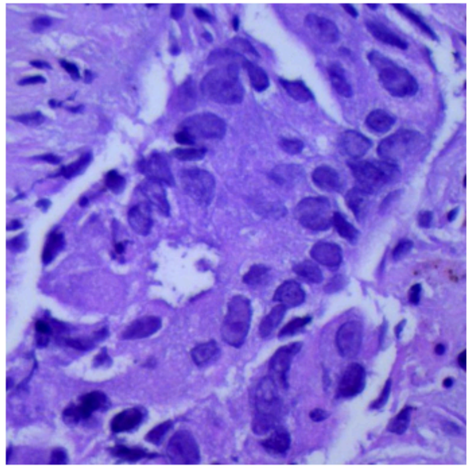
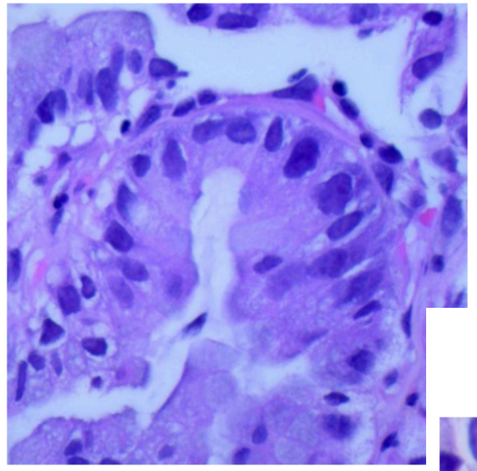
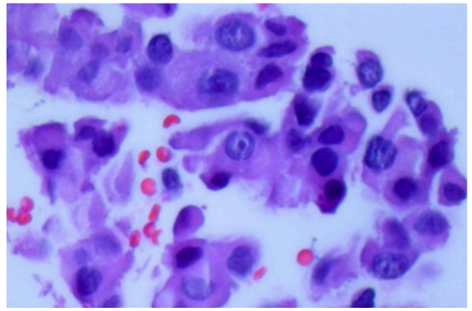
DATASET COLLECTION

- Dataset Source:
  - Utilized the "Lung and Colon Cancer Histopathological Images" dataset from Kaggle.
  - Contains labeled images of lung tissues categorized into:
    - Normal (non-cancerous)
    - Benign (early-stage abnormalities)
    - Malignant (cancerous)

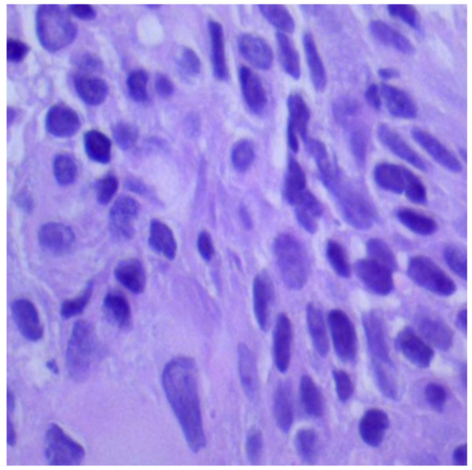
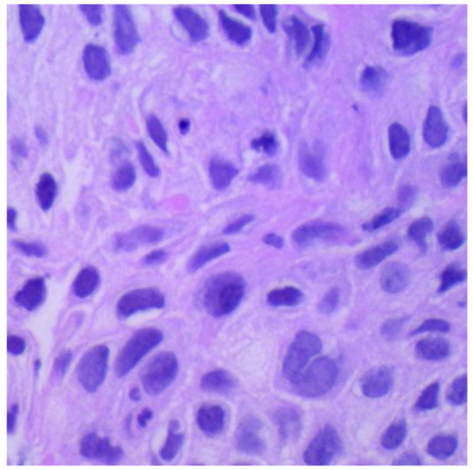
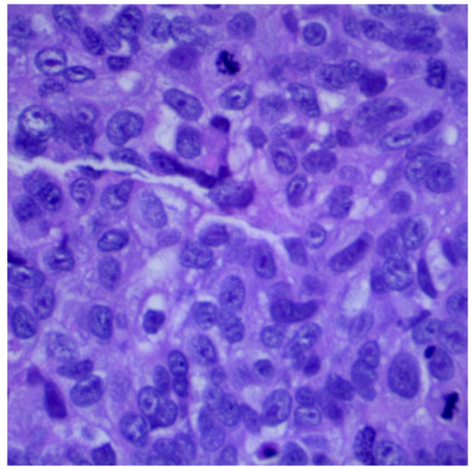
Images for lung_n category . . . .

Images for lung_aca category . . . .

Images for lung_scc category . . . .

DATASET PRE-PROCESSING

01 Image Loading and Resizing

- Images are loaded using OpenCV and resized to a uniform dimension (256x256 pixels) to ensure consistency.

02 Normalization

- Pixel values are scaled to the range [0, 1] by dividing by 255 to improve model convergence.

03 Train-Validation Split

- Data is split into training (80%) and validation (20%) sets using train_test_split from Scikit-learn.

04 One-Hot Encoding

- Class labels are converted into one-hot encoded vectors for multi-class classification (e.g., [1, 0, 0] for "Normal").

05 Data Augmentation

- Techniques like rotation, flipping, and brightness adjustment can be applied to increase dataset diversity and reduce overfitting.

Model Design and Architecture

The model is a Convolutional Neural Network (CNN) built using TensorFlow/Keras. Here's a breakdown of its architecture:

01 Input Layer

- Takes RGB lung histopathology images resized to 256×256 pixels.
- Input shape: (256, 256, 3)

02 Convolutional & Pooling Layers

- 1st Conv Layer: 32 filters, 5×5 kernel, ReLU activation → MaxPooling
- 2nd Conv Layer: 64 filters, 3×3 kernel, ReLU activation → MaxPooling
- 3rd Conv Layer: 128 filters, 3×3 kernel, ReLU activation → MaxPooling

03 Flatten + Fully Connected Layers

- Flatten layer converts 3D features to 1D.
- Dense(256) with ReLU → BatchNormalization
- Dense(128) with ReLU → Dropout (0.3) → BatchNormalization

04 Output Layer

- Dense(3) with Softmax activation (for 3-class classification: adenocarcinoma, squamous cell carcinoma, benign).

These layers extract increasing levels of abstract features from the input images — from edges and colors to complex tumor-like shapes.

Training Parameters and Optimization

Hyperparameters

- Image size: 256 × 256
- Batch size: 64
- Epochs: 10
- Split: 80% training / 20% validation

Optimizer

Adam optimizer: adaptive learning rate for fast and stable convergence.

Loss Function

Categorical Crossentropy: Suitable for multi-class classification with one-hot encoded labels.

Training Parameters and Optimization

Regularization and Optimization Techniques

- Dropout (30%): Prevents overfitting by randomly deactivating neurons.
- Batch Normalization: Stabilizes and speeds up training.
- EarlyStopping and ReduceLROnPlateau (callbacks):
  - Automatically stops training when no improvement is observed.
  - Reduces learning rate if validation accuracy plateaus.

MODEL TRAINING

```
#Model Training

history = model.fit(X_train, Y_train,
                    validation_data = (X_val, Y_val),
                    batch_size = BATCH_SIZE,
                    epochs = EPOCHS,
                    verbose = 1,
                    callbacks = [es, lr, myCallback()])
```

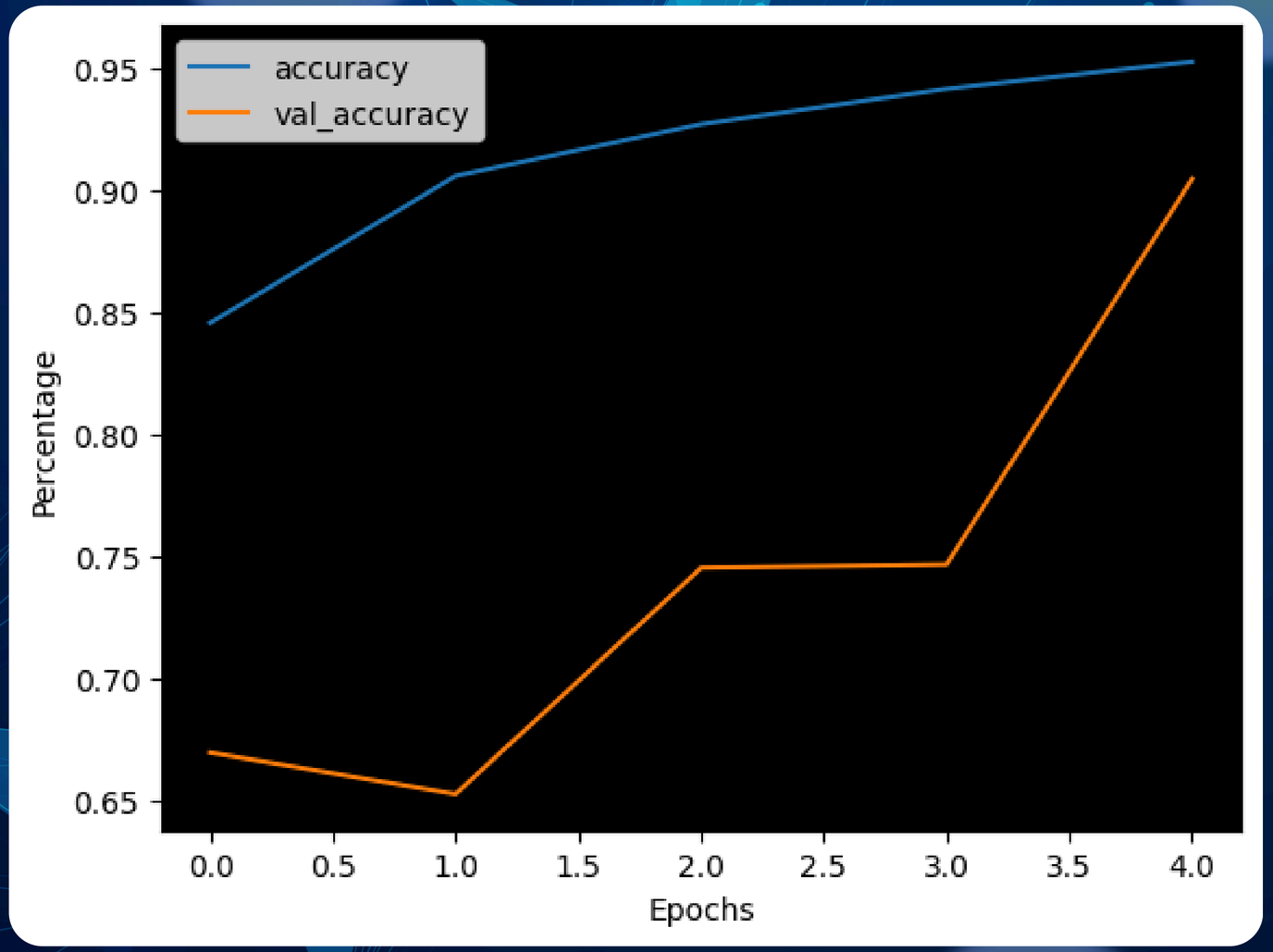
[10] Python

```
Epoch 1/10
188/188 ━━━━━━━━━━━━━━━━━━━━ 589s 3s/step - accuracy: 0.7979 - loss: 0.4692 - val_accuracy: 0.6697 - val_loss: 2.5110 - learning_rate: 0.0010
Epoch 2/10
188/188 ━━━━━━━━━━━━━━━━━━━━ 656s 3s/step - accuracy: 0.9002 - loss: 0.2406 - val_accuracy: 0.6527 - val_loss: 1.3363 - learning_rate: 0.0010
Epoch 3/10
188/188 ━━━━━━━━━━━━━━━━━━━━ 453s 2s/step - accuracy: 0.9245 - loss: 0.1897 - val_accuracy: 0.7453 - val_loss: 0.7364 - learning_rate: 0.0010
Epoch 4/10
188/188 ━━━━━━━━━━━━━━━━━━━━ 448s 2s/step - accuracy: 0.9449 - loss: 0.1407 - val_accuracy: 0.7467 - val_loss: 1.0761 - learning_rate: 0.0010
Epoch 5/10
188/188 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.9516 - loss: 0.1249
 Validation accuracy has reached upto                    90% so, stopping further training.
188/188 ━━━━━━━━━━━━━━━━━━━━ 442s 2s/step - accuracy: 0.9516 - loss: 0.1249 - val_accuracy: 0.9047 - val_loss: 0.3223 - learning_rate: 0.0010
```

TRAINING ACCURACY

Key Takeaways -

- Training accuracy steadily improves, suggesting the model is learning well.
- Validation accuracy lagged initially, but shows a strong improvement by epoch 4.
- The gap between training and validation accuracy narrows, which is a good sign — suggests improved generalization and reduced overfitting by the final epoch.



EARLY STOPPING

Early stopping is a regularization technique where training is stopped once the model's performance on a validation set starts to degrade (even though it may still be improving on the training set).

- Prevent Overfitting
- As training continues, the model may start to memorize the training data, including noise, rather than learning general patterns.
- This leads to poor performance on new, unseen data.
- Early stopping halts training before this happens.
- Improve Generalization
- The goal of a model is to perform well on unseen (test or real-world) data, not just the training set.
- By monitoring validation loss, early stopping ensures the model remains generalized.
- Save Computational Resources
- It avoids unnecessary epochs that do not improve validation performance.
- This can significantly cut down training time and energy usage.

MODEL EVALUATION

Three classes -

- lung adenocarcinoma(987 samples)
- lung benign tissue(977 samples)
- lung squamous cell carcinoma(1036 samples)

Total Samples = 3000

Performance Summary-

- lung_aca - Solid but slightly lower; may benefit from better representation or features
- lung_n - Extremely high precision, but some false negatives.
- lung_scc - Strong and balanced. High recall indicates few missed cases.

Key Takeaways -

- The CNN model is performing very well overall, with 90% accuracy and strong precision/recall across classes.
- Class lung_n has perfect precision — no false positives, but still has some false negatives.
- Performance is fairly balanced across classes, and there's no strong class imbalance (as support is similar across all three).

```
···  94/94 ━━━━━━━━━━━━━━━━━━━━ 27s 272ms/step
                precision    recall  f1-score   support

      lung_aca       0.85      0.86      0.86       987
        lung_n       1.00      0.92      0.96       977
      lung_scc       0.88      0.93      0.90      1036

      accuracy                           0.90      3000
     macro avg       0.91      0.90      0.91      3000
  weighted avg       0.91      0.90      0.91      3000

                precision    recall  f1-score   support

      lung_aca       0.85      0.86      0.86       987
        lung_n       1.00      0.92      0.96       977
      lung_scc       0.88      0.93      0.90      1036

      accuracy                           0.90      3000
     macro avg       0.91      0.90      0.91      3000
  weighted avg       0.91      0.90      0.91      3000
```


THANKYOU