PCOS DETECTION USING ULTRASOUND IMAGES

1. ResNet-50

- **Architecture**: ResNet-50 is a deep convolutional neural network (CNN) with 50 layers, designed using a **residual learning framework.** The core idea is the use of **residual connections** (also known as skip connections) that allow gradients to flow more easily through the network during backpropagation, solving the vanishing gradient problem and enabling the training of much deeper networks.
- **Key Components**:
    - **Convolutional layers**: Multiple convolutional layers are stacked to learn hierarchical features.
    - **Residual Blocks**: Introduces skip connections that bypass some layers, enabling the network to learn residual mappings instead of full mappings, making it easier to train deeper networks.
    - **Global Average Pooling**: Reduces the dimensionality of the feature maps before fully connected layers.

```
61/61 ━━━━━━━━━━━━━━━━━━━━ 11s 113ms/step - accuracy: 0.9869 - loss: 0.0277
Test Loss: 0.02654281258583069
Test Accuracy: 0.9916753172874451
61/61 ━━━━━━━━━━━━━━━━━━━━ 11s 133ms/step
```

2. CNN with Pre-Trained VGGNet16

- **Architecture**: VGG16 is a CNN architecture with **16 layers**, including 13 convolutional layers and 3 fully connected layers. It uses small **3x3 convolution filters** stacked on top of each other with max-pooling layers interleaved.
- **Key Components**:
    - **Convolutional layers**: Use a simple and uniform architecture with small filters (3x3), which helps in capturing more fine-grained features.
    - **Pre-trained Weights**: VGGNet16 can be used as a **pre-trained model**, where the model is initially trained on large datasets like ImageNet, and the learned features are transferred to other tasks, such as medical image classification.

- **Global Average Pooling**: Sometimes added on top of the convolutional layers to reduce the dimensionality of the final feature maps.

```
Validation Accuracy: 99.65398%
Validation F1 Score: 0.99639
Validation Precision: 0.99569
Validation Recall: 0.99713

Model loaded successfully
61/61 ━━━━━━━━━━━━━━━━━━━━ 10s 126ms/step
Test Accuracy: 99.17%
```

3. Stacked Ensemble with XGBoost Meta-Learner

1. Architecture and Approach:

- Stacked Ensemble Architecture: The model combines three base learners (Logistic Regression, Random Forest, SVM), and their probabilistic predictions are stacked (concatenated). These stacked predictions are then fed into a meta-learner (XGBoost) that makes the final prediction.
- Approach: Each base model learns from the data independently, and the meta-learner aggregates their predictions to improve accuracy, leveraging diverse learning techniques (linear, tree-based, and margin-based models).

2. Key Components:

- Base Learners: Logistic Regression (linear), Random Forest (non-linear, tree-based), and SVM (margin-based) are used to capture different patterns in the data.
- Meta-Learner: XGBoost (gradient-boosted decision trees) integrates the predictions of the base learners to enhance the overall predictive performance.

```
[395]   validation_0-logloss:0.39828    validation_0-error:0.00000      validation_1-logloss:0.40684    validation_1-error:0.01730
[396]   validation_0-logloss:0.39779    validation_0-error:0.00000      validation_1-logloss:0.40638    validation_1-error:0.01730
[397]   validation_0-logloss:0.39730    validation_0-error:0.00000      validation_1-logloss:0.40591    validation_1-error:0.01730
[398]   validation_0-logloss:0.39681    validation_0-error:0.00000      validation_1-logloss:0.40545    validation_1-error:0.01730
[399]   validation_0-logloss:0.39632    validation_0-error:0.00000      validation_1-logloss:0.40498    validation_1-error:0.01730
Accuracy: 0.9973985431841832
F1 Score: 0.9973066588865263
Precision: 0.9968193384223918
Recall: 0.9978089395267309
ROC AUC: 0.997808939526731
```

+ Code + Markdown

4. Multi-Model Ensemble (VGG16 + ResNet50 + RF + SVM + Logistic Regression + XGBoost)

1. Architecture and Approach:

This model implements a stacking-based ensemble for binary classification using ultrasound images for PCOS detection. **Features are extracted** from two pre-trained deep learning architectures, **VGG16 and ResNet50,** which capture high-level representations from the images. These extracted features are then combined and fed into three different base classifiers: **Logistic Regression (LR), Random Forest (RF), and Support Vector Machine (SVM).** The outputs of these base models (probabilities) are stacked together and passed to a **meta-learner (XGBoost) for final classification**, allowing for improved predictive performance.

2. Key Components:

- Feature Extractors: VGG16 and ResNet50 are used to extract features from images, leveraging their deep learning architectures pre-trained on ImageNet.
- Base Classifiers: Logistic Regression, Random Forest, and Support Vector Machine are used as the initial classifiers in the stacking ensemble, providing multiple perspectives on the input data.
- Meta-Learner: XGBoost, a powerful gradient boosting algorithm, serves as the final classifier in the ensemble, combining the predictions from the base models to make more accurate final predictions.

Output 1:
lr = LogisticRegression(max_iter=500)
rf = RandomForestClassifier(n_estimators=100)
svm = SVC(probability=True)
meta_learner = xgb.XGBClassifier(n_estimators=200, learning_rate=0.01, max_depth=4, eval_metric=['logloss', 'error'])

```
[195]   validation_0-logloss:0.07168    validation_0-error:0.00000      validation_1-logloss:0.13811    validation_1-error:0.02768
[196]   validation_0-logloss:0.07095    validation_0-error:0.00000      validation_1-logloss:0.13768    validation_1-error:0.02768
[197]   validation_0-logloss:0.07023    validation_0-error:0.00000      validation_1-logloss:0.13725    validation_1-error:0.02768
[198]   validation_0-logloss:0.06952    validation_0-error:0.00000      validation_1-logloss:0.13683    validation_1-error:0.02768
[199]   validation_0-logloss:0.06882    validation_0-error:0.00000      validation_1-logloss:0.13642    validation_1-error:0.02768
Test Accuracy: 99.58377%
Test F1 Score: 0.99569
Test Precision: 0.99493
Test Recall: 0.99649
Test ROC AUC: 0.99649
```

+ Code + Markdown

Output 2:
meta_learner = xgb.XGBClassifier(n_estimators=200, learning_rate=0.01, max_depth=6, eval_metric=['logloss', 'error'])

```
[195]   validation_0-logloss:0.07168    validation_0-error:0.00000      validation_1-logloss:0.15479    validation_1-error:0.03460
[196]   validation_0-logloss:0.07095    validation_0-error:0.00000      validation_1-logloss:0.15442    validation_1-error:0.03460
[197]   validation_0-logloss:0.07023    validation_0-error:0.00000      validation_1-logloss:0.15407    validation_1-error:0.03460
[198]   validation_0-logloss:0.06952    validation_0-error:0.00000      validation_1-logloss:0.15372    validation_1-error:0.03460
[199]   validation_0-logloss:0.06882    validation_0-error:0.00000      validation_1-logloss:0.15338    validation_1-error:0.03460
Test Accuracy: 99.47971%
Test F1 Score: 0.99462
Test Precision: 0.99368
Test Recall: 0.99562
Test ROC AUC: 0.99562
```

+ Code + Markdown

Output 3:
meta_learner = xgb.XGBClassifier(n_estimators=500, learning_rate=0.01, max_depth=4, eval_metric=['logloss', 'error'])

```
[495]   validation_0-logloss:0.00447    validation_0-error:0.00000      validation_1-logloss:0.13055    validation_1-error:0.02422
[496]   validation_0-logloss:0.00443    validation_0-error:0.00000      validation_1-logloss:0.13070    validation_1-error:0.02422
[497]   validation_0-logloss:0.00440    validation_0-error:0.00000      validation_1-logloss:0.13086    validation_1-error:0.02422
[498]   validation_0-logloss:0.00436    validation_0-error:0.00000      validation_1-logloss:0.13102    validation_1-error:0.02422
[499]   validation_0-logloss:0.00433    validation_0-error:0.00000      validation_1-logloss:0.13117    validation_1-error:0.02422
Test Accuracy: 99.63580%
Test F1 Score: 0.99623
Test Precision: 0.99556
Test Recall: 0.99693
Test ROC AUC: 0.99693
```

+ Code + Markdown

Output 4:
meta_learner = xgb.XGBClassifier(n_estimators=200, learning_rate=0.05, max_depth=4, eval_metric=['logloss', 'error'])

```
[195]   validation_0-logloss:0.00122    validation_0-error:0.00000      validation_1-logloss:0.22616    validation_1-error:0.03460
[196]   validation_0-logloss:0.00122    validation_0-error:0.00000      validation_1-logloss:0.22616    validation_1-error:0.03460
[197]   validation_0-logloss:0.00122    validation_0-error:0.00000      validation_1-logloss:0.22616    validation_1-error:0.03460
[198]   validation_0-logloss:0.00122    validation_0-error:0.00000      validation_1-logloss:0.22616    validation_1-error:0.03460
[199]   validation_0-logloss:0.00122    validation_0-error:0.00000      validation_1-logloss:0.22616    validation_1-error:0.03460
Test Accuracy: 99.47971%
Test F1 Score: 0.99462
Test Precision: 0.99368
Test Recall: 0.99562
Test ROC AUC: 0.99562
```

+ Code + Markdown

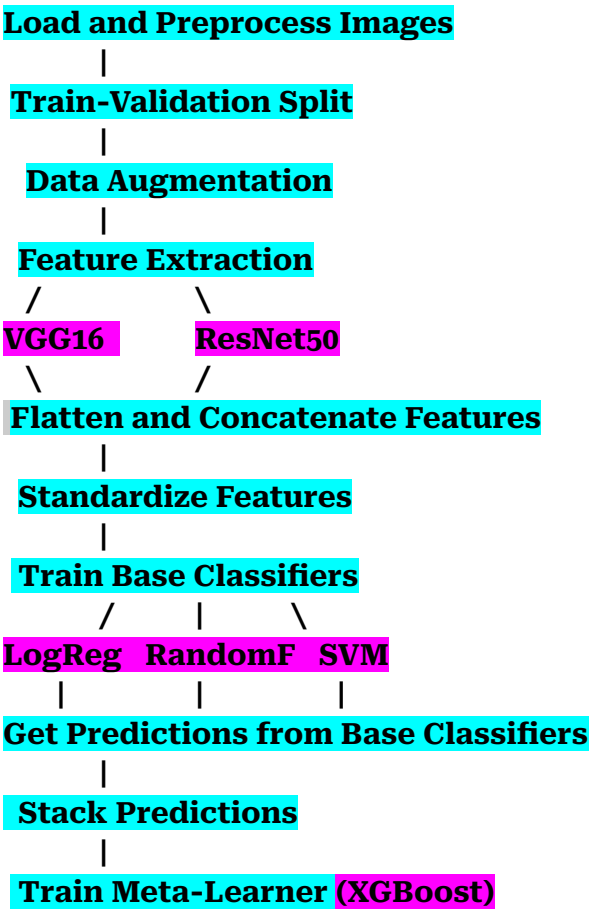
Output 5:
meta_learner = xgb.XGBClassifier(n_estimators=200, learning_rate=0.001, max_depth=4, eval_metric=['logloss', 'error'])

```
[196]   validation_0-logloss:0.51373    validation_0-error:0.00000      validation_1-logloss:0.51670    validation_1-error:0.01730
[197]   validation_0-logloss:0.51305    validation_0-error:0.00000      validation_1-logloss:0.51605    validation_1-error:0.01730
[198]   validation_0-logloss:0.51237    validation_0-error:0.00000      validation_1-logloss:0.51540    validation_1-error:0.01730
[199]   validation_0-logloss:0.51169    validation_0-error:0.00000      validation_1-logloss:0.51475    validation_1-error:0.01730
Test Accuracy: 99.73985%
Test F1 Score: 0.99731
Test Precision: 0.99682
Test Recall: 0.99781
Test ROC AUC: 0.99781
```

+ Code + Markdown

Output 5:
meta_learner = xgb.XGBClassifier(n_estimators=500, learning_rate=0.001, max_depth=4, eval_metric=['logloss', 'error'])

```
[498]   validation_0-logloss:0.35124    validation_0-error:0.00000      validation_1-logloss:0.35482    validation_1-error:0.
[499]   validation_0-logloss:0.35082    validation_0-error:0.00000      validation_1-logloss:0.35441    validation_1-error:0.
Test Accuracy: 99.89594%
Test F1 Score: 0.99892
Test Precision: 0.99872
Test Recall: 0.99912
Test ROC AUC: 0.99912
```



|

Evaluate Model