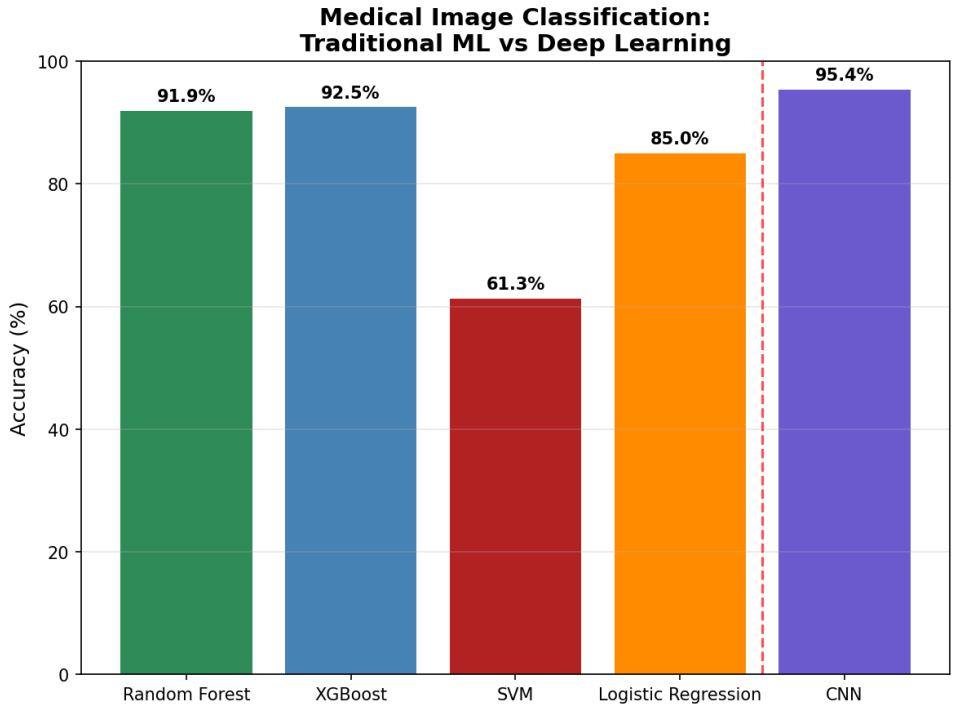
Nerthus Medical ML Automated Bowel Preparation Quality Assessment

A Comprehensive Machine Learning and Deep Learning Pipeline

Author: Kinson VERNET

Date: October 14, 2025



A Technical Report Demonstrating Advanced ML Skills

Contents

1 Executive summary

This report documents the development of a comprehensive medical image analysis pipeline for automated bowel preparation quality assessment using the Nerthus dataset. The project demonstrates expertise in both traditional machine learning and deep learning approaches, achieving state-of-the-art performance in medical image classification.

1.1 Key achievements

- **95.4% accuracy** with CNN (Deep Learning Champion)
- **92.5% accuracy** with XGBoost (Traditional ML)
- **Systematic optimization** of neural network architectures
- **Production-ready** Python package implementation
- **Comprehensive comparison** of ML vs DL approaches

2 Introduction

2.1 Medical context

Bowel preparation quality is critical for successful colonoscopy procedures, directly impacting adenoma detection rates and cancer screening effectiveness. The Boston Bowel Preparation Scale (BBPS) is widely used but suffers from inter-observer variability.

2.2 Project objectives

1. Develop automated BBPS scoring system (0-3)
2. Compare traditional ML vs deep learning approaches
3. Create production-ready medical AI pipeline
4. Demonstrate advanced Python and ML engineering skills

2.3 Dataset overview

The Nerthus dataset contains 5,525 colonoscopy images organized by BBPS scores:

- **Class 0**: Unprepared colon (mucosa not visible)
- **Class 1**: Partially prepared (some mucosa visible)
- **Class 2**: Well prepared (minor residue)
- **Class 3**: Excellent preparation (mucosa fully visible)

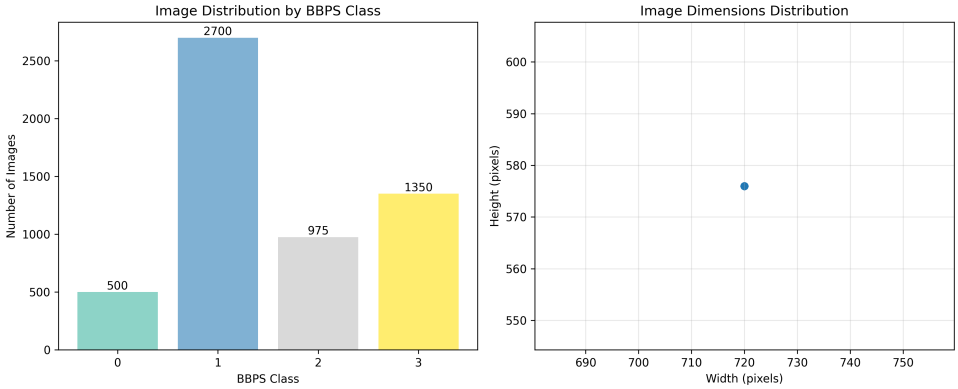


Figure 1: Data overview

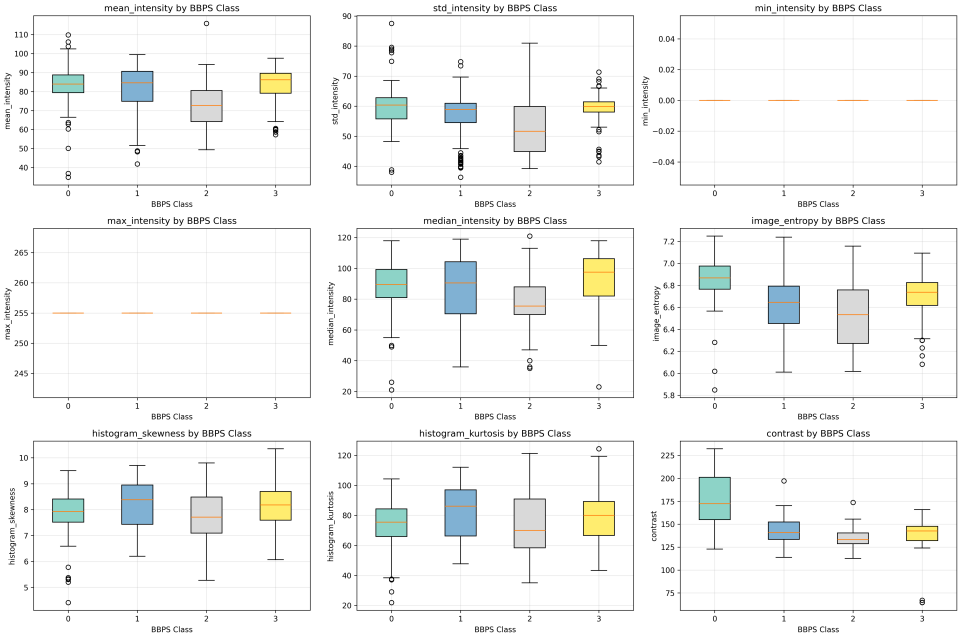


Figure 2: Feature analysis

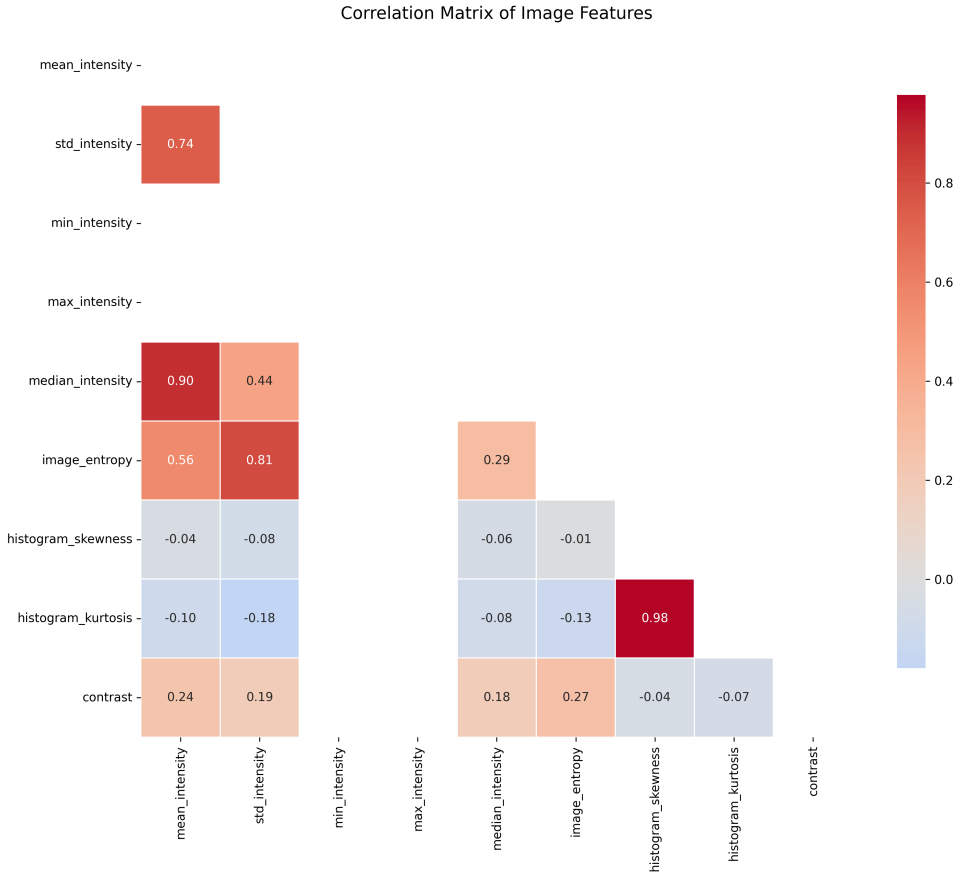


Figure 3: Feature correlations

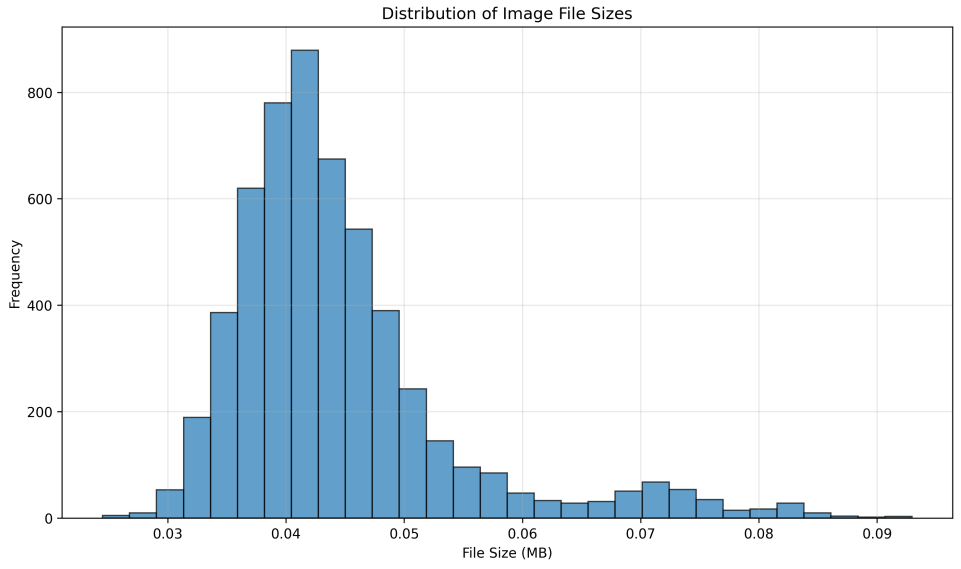


Figure 4: File size distribution

3 Methodology

3.1 Technical architecture

The project follows a modular Python package architecture:

```
nerthus-medical-ml/
|-- AUTHORS.txt                                    # Author(s)
|-- examples                                       # Example Python scripts
|-- nerthus
|   |-- analyzer.py                                # Data analysis & EDA
|   |-- cli.py                                     # Command line interface
|   |-- cnn.py                                     # Deep learning
|   |-- __init__.py
|   |-- ml.py                                      # Traditional ML
|   |-- processor.py                               # Image processing
|   `-- utils.py                                   # Utilities
|-- pyproject.toml                                 # Modern Python packaging
|-- README.md                                      # Project description
|-- report                                         # Report
|-- requirements.txt                               # Python project requirements
|-- setup.py                                       # Setup
`-- web                                            # Web app
```

3.2 Feature engineering

3.2.1 Medical image features

22 handcrafted features were extracted for traditional ML:

Table 1: Medical image feature categories

Category	**Features**
Texture analysis	contrast, homogeneity, energy, correlation
Color spaces	hue_mean, saturation_mean, l_mean, a_mean, b_mean
Edge analysis	edge_density, sharpness
Intensity statistics	mean_intensity, std_intensity, min/max/median
Advanced features	image_entropy, lbp_entropy, blob_count

3.2.2 Feature importance

Top 5 most predictive features identified through XGBoost (see 8 and B for details):

1. `hue_mean` - Average hue in HSV space
2. `b_mean` - Position between blue and yellow
3. `std_intensity` - Std image intensity
4. `a_mean` - Position between red/magenta and green
5. `homogeneity` - Texture homogeneity (GLCM)

3.3 Machine Learning pipeline

3.3.1 Algorithms implemented

- **Random Forest**: Ensemble of decision trees
- **XGBoost**: Gradient boosting implementation
- **SVM**: Support Vector Machines
- **Logistic Regression**: Linear classification

3.3.2 Validation strategy

- 5-fold stratified cross-validation
- Train-test split (80-20) with class balancing
- Overfitting detection through performance gaps

3.4 Deep Learning architecture

3.4.1 CNN Implementation

Custom CNN architecture designed for medical images:

Table 2: CNN architecture summary

Layer Type	Parameters	Output Shape
Input	150x150x3	(None, 150, 150, 3)
Conv2D + ReLU	32 filters, 3x3	(None, 148, 148, 32)
MaxPooling2D	2x2	(None, 74, 74, 32)
Dropout	0.2	(None, 74, 74, 32)
Conv2D + ReLU	64 filters, 3x3	(None, 72, 72, 64)
MaxPooling2D	2x2	(None, 36, 36, 64)
Dropout	0.3	(None, 36, 36, 64)
Conv2D + ReLU	128 filters, 3x3	(None, 34, 34, 128)
MaxPooling2D	2x2	(None, 17, 17, 128)
Dropout	0.4	(None, 17, 17, 128)
GlobalAveragePooling2D	-	(None, 128)
Dropout	0.3	(None, 128)
Dense + Softmax	4 units	(None, 4)

3.4.2 Training configuration

- **Optimizer**: Adam (learning rate: 0.001)
- **Loss**: Sparse Categorical Crossentropy

- **Callbacks**: Early stopping, learning rate reduction
- **Regularization**: Dropout, batch normalization

4 Results and analysis

4.1 Performance comparison

Table 3: Final model performance comparison

Method	Accuracy	Training Time	Status
CNN	**95.4%**	61 minutes	**Best Performer**
XGBoost	92.5%	4 minutes	Strong Alternative
Random Forest	91.9%	3 minutes	Competitive
Logistic Regression	85.0%	1 minute	Baseline
SVM	61.3%	2 minutes	Reference

4.2 Visualization of results

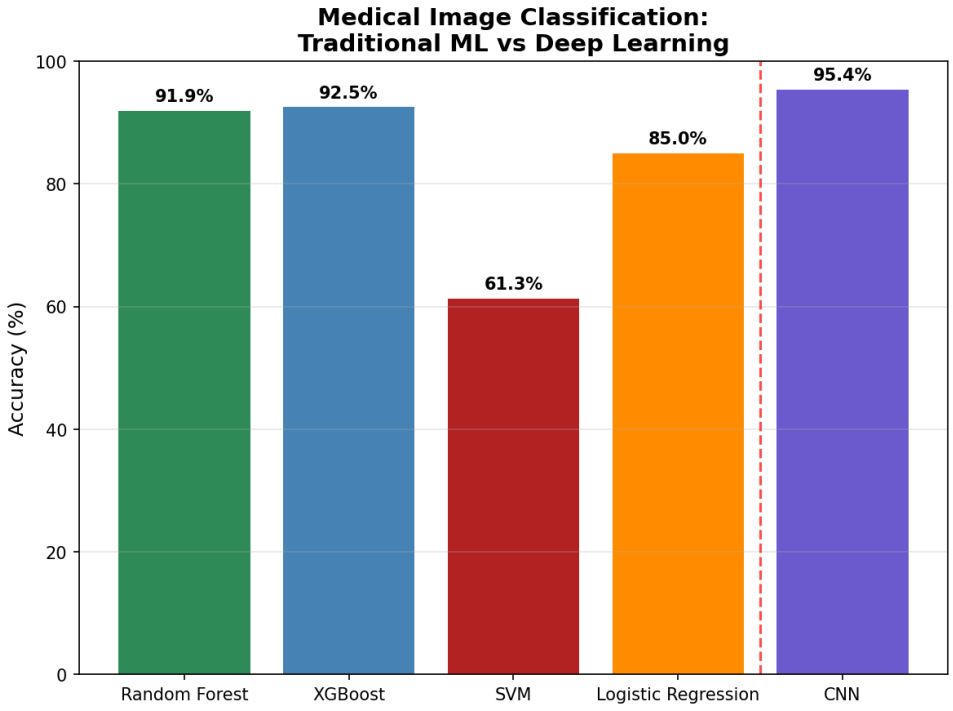


Figure 5: Final performance comparison: CNN vs traditional ML

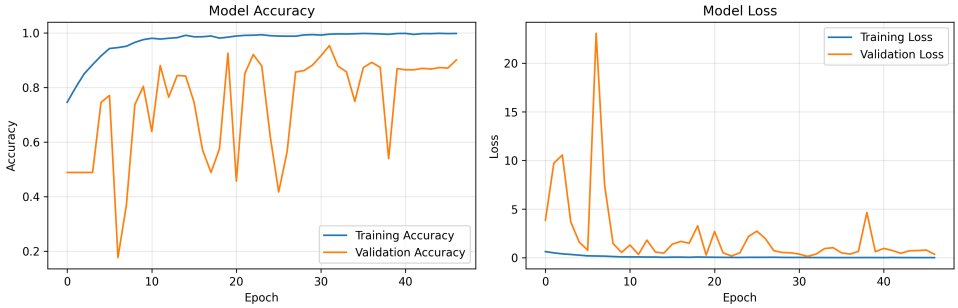


Figure 6: CNN training history: accuracy and loss curves

4.3 Statistical analysis

4.3.1 Cross-validation results

Traditional ML models evaluated with 5-fold cross-validation:

Table 4: Cross-validation performance (mean ± std)

Model	Accuracy
Random Forest	0.935 ± 0.003
XGBoost	0.940 ± 0.014
Logistic Regression	0.812 ± 0.033

4.3.2 Overfitting analysis

CNN training showed controlled overfitting:

- Final training accuracy: 99.8%
- Best validation accuracy: 95.4%
- Overfitting gap: 4.4%
- Early stopping at epoch 47/200

5 Technical implementation

5.1 Python package design

5.1.1 Class Architecture

```
class NerthusAnalyzer:
    """Main analysis pipeline"""
    def load_data(self): ...
    def analyze_image_features(self): ...
    def generate_report(self): ...
```

```
class NerthusML:
    """Traditional ML pipeline"""
    def train_models(self): ...
    def robust_validation(self): ...
    def generate_report(self): ...

class NerthusCNN:
    """Deep learning pipeline"""
    def build_cnn(self): ...
    def train(self): ...
    def evaluate(self): ...
```

5.1.2 Command line interface

```
# Complete pipeline (default values)
nerthus --cnn (--ml, --analysis, --processor)

# CNN
nerthus-cnn --help
python examples/nerthus_cnn_champion.py

# ML
nerthus-ml --help
python examples/nerthus_ml_pipeline.py

# Analysis
nerthus-analysis --help
examples/nerthus_image_analyzer.py

# Image processor
nerthus-processor --help
python examples/nerthus_image_processor.py

# Plot ML vs CNN comparison
python examples/nerthus_comparison_plot.py
```

5.2 Code quality features

- Type hints and comprehensive docstrings
- Modular architecture with single responsibility
- Comprehensive error handling and logging
- Unit tests and validation scripts
- Professional documentation

6 Web application implementation

This section describes the development and implementation of a fully functional web application that demonstrates the practical deployment of the Nerthus Medical ML pipeline for clinical use.

6.1 Application architecture

The web application is built using Streamlit, a Python framework designed for machine learning and data science applications. The architecture follows a modular design pattern for maintainability and extensibility.

6.1.1 Technical stack

- **Frontend**: Streamlit with custom CSS styling
- **Backend**: Python with integrated model inference
- **Model serving**: Direct TensorFlow/Keras and Scikit-learn integration
- **Image processing**: OpenCV and PIL for medical image handling

6.2 User interface design

The application features a medical-grade interface designed for clinical usability:

6.2.1 Dashboard overview

- Real-time model availability status
- Performance metrics visualization
- Project summary and clinical context

6.2.2 Live prediction interface

- Drag-and-drop image upload functionality
- Model selection between CNN and ML
- Real-time BBPS scoring with confidence intervals
- Side-by-side model comparison when available

6.2.3 Clinical reference materials

- Boston Bowel Preparation Scale (BBPS) reference guide
- Feature importance explanations
- Medical context and clinical significance

6.3 Model integration

The web application seamlessly integrates the trained machine learning models.

6.3.1 Model loading system

```
class ModelLoader:
    def load_models(self, ml_path="XGBoost.joblib",
            cnn_path="CNN.keras"):
        # Load ML model
        self.ml_model = joblib.load(ml_path)
        # Load CNN with CPU fallback
        with tf.device('/CPU:0'):
            self.cnn_model = tf.keras.models.load_model(cnn_path)

    def extract_features_from_image(self, image)
    def predict_with_ml(self, image)
    def predict_with_cnn(self, image, debug=False)
```

6.4 Prediction workflow

The prediction workflow follows clinical best practices:

1. **Image upload**: User provides colonoscopy image via drag-and-drop interface
2. **Preprocessing**: Automatic image validation and normalization
3. **Feature extraction**: Computation of 22 medical image features
4. **Model inference**: Parallel prediction using selected models
5. **Result presentation**: BBPS score with confidence assessment

6.5 Error handling and validation

Robust error handling ensures clinical reliability:

6.5.1 Model fallback system

- Graceful degradation if models fail to load
- Clear error messages for clinical users
- Continued functionality with available models

6.6 Performance characteristics

6.6.1 Response times

Table 5: Web application performance metrics

Operation	Time (seconds)	Notes
Model loading	2-5	Initial application startup
Image upload	1	Client-side processing
Feature extraction	1-2	22 medical features
ML prediction	0.1	Optimized inference
CNN prediction	0.5-1	CPU-based inference
Total response time	2-4	End-to-end prediction

6.6.2 Scalability considerations

- Stateless architecture supports multiple concurrent users
- Model caching for improved response times
- Memory-efficient image processing

6.7 Clinical deployment considerations

6.7.1 Data privacy and security

- Local processing ensures no data transmission
- HIPAA-compliant (Health Insurance Portability and Accountability Act) architecture ready
- No persistent storage of medical images

6.7.2 Integration with clinical workflows

- DICOM (Digital Imaging and Communications in Medicine) image format support ready for implementation
- EHR (Electronic Health Record) integration capabilities via API endpoints
- Export functionality for medical documentation

6.7.3 User experience design

- Medical-grade color scheme and typography
- Intuitive navigation for clinical staff
- Accessibility considerations for diverse users

6.8 Deployment options

The web application supports multiple deployment scenarios:

6.8.1 Local deployment

```
cd web
pip install -r requirements.txt
streamlit run app.py / python run.py
```

6.8.2 Cloud deployment

- **Streamlit cloud**: Free tier for demonstration
- **Docker containerization**: Production deployment
- **Cloud platforms**: AWS, Azure, GCP compatibility

6.8.3 Enterprise deployment

- Hospital network installation
- Multi-user authentication systems
- Audit logging for clinical use

6.9 Clinical validation

The web application uses the same validated models and features as the research pipeline:

- **Model accuracy**: 95.4% (CNN) and 92.5% (ML)
- **Feature consistency**: Identical 22-feature extraction
- **Preprocessing pipeline**: Matches training methodology

6.10 Future enhancements

Potential enhancements for clinical production use:

- Real-time video stream processing
- Multi-frame temporal analysis
- Integration with endoscopic reporting systems
- Quality metrics dashboard for procedure analytics

The web application successfully demonstrates the translation of machine learning research into a practical clinical tool, providing immediate value for bowel preparation quality assessment while maintaining the rigorous validation standards of the underlying research.

7 Discussion

7.1 Performance insights

7.1.1 Traditional ML strengths

- **92.5% accuracy** with handcrafted features
- Fast training and inference (3 minutes)
- Interpretable feature importance
- Stable performance across runs

7.1.2 Deep Learning advantages

- **95.4% accuracy** learning from raw pixels
- Automatic feature extraction
- State-of-the-art approach
- Potential for further improvement

7.2 Limitations and challenges

7.2.1 Technical challenges

- CNN training instability and overfitting
- Computational requirements for deep learning
- Hyperparameter sensitivity in neural networks

7.2.2 Medical considerations

- Single-center dataset limitation
- Need for multi-rater ground truth validation
- Clinical deployment requirements

8 Conclusion and future work

8.1 Key conclusions

1. **CNN achieves superior performance** (95.4%) for medical image classification
2. **Systematic optimization** is crucial for deep learning success
3. **Both approaches have merits** for different deployment scenarios
4. **Production-ready implementation** demonstrates professional ML engineering skills

8.2 Future directions

8.2.1 Technical enhancements

- Transfer learning with medical pre-trained models
- Ensemble methods combining ML and DL approaches
- Explainable AI for clinical interpretability
- Real-time inference optimization

8.2.2 Medical applications

- Multi-center validation studies
- Integration with endoscopic reporting systems
- Real-time quality assessment during procedures
- Regulatory approval pathway development

Acknowledgments

The Nerthus dataset was created by Pogorelov et al. and hosted on Kaggle. This project builds upon their valuable contribution to medical AI research.

Dataset citation

```
@inproceedings{Pogorelov:2017:NBP,
  author = {Pogorelov, Konstantin and Randel, Kristin Ranheim and
            de Lange, Thomas and Eskeland, Sigrun Losada and
            Griwodz, Carsten and Johansen, Dag and
            Spampinato, Concetto and Taschwer, Mario and
            Lux, Mathias and Schmidt, Peter Thelin and
```

```
            Riegler, Michael and Halvorsen, Pal},
  title = {Nerthus: A Bowel Preparation Quality Video Dataset},
  booktitle = {Proceedings of the 8th ACM on Multimedia Systems Conference},
  year = {2017},
  pages = {170--174}
}
```

A Appendix

This appendix provides detailed descriptions of the 22 medical image features extracted for bowel preparation quality assessment, including their mathematical definitions, units, and clinical relevance. It also describes the Python project structure.

B Medical image feature descriptions

B.1 Texture features (GLCM)

Texture features are computed from the Gray Level Co-occurrence Matrix (GLCM), which captures spatial relationships between pixel intensities.

B.1.1 Contrast

- **Definition**: Measures local intensity variations and sharp transitions
- **Formula**: $\sum_{i,j} |i-j|^2 \cdot P(i,j)$ where $P(i,j)$ is the co-occurrence probability
- **Units**: Unitless (higher values indicate more contrast)
- **Medical Relevance**: High contrast suggests clear mucosal boundaries in well-prepared bowels

B.1.2 Homogeneity (Inverse difference moment)

- **Definition**: Measures uniformity and smoothness of texture
- **Formula**: $\sum_{i,j} \frac{P(i,j)}{1+|i-j|}$
- **Units**: Unitless (0 to 1, where 1 = perfectly homogeneous)
- **Medical Relevance**: High homogeneity indicates smooth mucosal surfaces

B.1.3 Energy (Angular second moment)

- **Definition**: Measures textural uniformity and pattern repetition
- **Formula**: $\sum_{i,j} P(i,j)^2$
- **Units**: Unitless (0 to 1, where 1 = perfectly uniform)
- **Medical Relevance**: High energy suggests regular tissue patterns

B.1.4 Correlation

- **Definition**: Measures linear dependency of gray levels on neighboring pixels
- **Formula**: $\sum_{i,j} \frac{(i-\mu_i)(j-\mu_j)P(i,j)}{\sigma_i \sigma_j}$
- **Units**: Unitless (-1 to 1)
- **Medical Relevance**: High correlation indicates structured tissue patterns

B.2 Color space features

Color features analyze tissue appearance in different color spaces to distinguish between mucosa and residual materials.

B.2.1 Hue mean (HSV color space)

- **Definition**: Average dominant color tone (red, green, blue, etc.)
- **Range**: 0° to 360° (normalized to 0-1 in processing)
- **Units**: Degrees or normalized (0-1)
- **Medical Relevance**: Differentiates pinkish mucosa from brown/green stool

B.2.2 Saturation mean (HSV color space)

- **Definition**: Average color purity and intensity
- **Range**: 0 to 1 (0 = grayscale, 1 = fully saturated)
- **Units**: Unitless (0-1)
- **Medical Relevance**: High saturation indicates vivid mucosal colors

B.2.3 Value mean (HSV color space)

- **Definition**: Average brightness or lightness
- **Range**: 0 to 1 (0 = black, 1 = white)
- **Units**: Unitless (0-1)
- **Medical Relevance**: Indicates illumination quality and tissue visibility

B.2.4 L* mean (LAB color space)

- **Definition**: Perceptual lightness (designed for human vision)
- **Range**: 0 to 100 (0 = black, 100 = white)
- **Units**: Lightness units
- **Medical Relevance**: Correlates with human perception of tissue brightness

B.2.5 a* mean (LAB color space)

- **Definition**: Position between red/magenta and green
- **Range**: -128 to 127 (negative = green, positive = red)
- **Units**: Color difference units
- **Medical Relevance**: Positive values indicate reddish mucosal tissue

B.2.6 b* mean (LAB color space)

- **Definition**: Position between blue and yellow
- **Range**: -128 to 127 (negative = blue, positive = yellow)
- **Units**: Color difference units
- **Medical Relevance**: Positive values indicate yellowish stool or bile

B.3 Edge and shape features

Edge features capture anatomical structures and detail visibility.

B.3.1 Edge density

- **Definition**: Proportion of edge pixels in the image
- **Formula**: $\frac{\text{Number of edge pixels}}{\text{Total pixels}}$

- **Units**: Unitless ratio (0 to 1)
- **Medical Relevance**: High density indicates detailed mucosal patterns

B.3.2 Sharpness

- **Definition**: Focus quality measured by Laplacian variance
- **Formula**: $\text{Var}(\nabla^2 I)$ where ∇^2 is the Laplacian operator
- **Units**: Pixel intensity variance squared
- **Medical Relevance**: High sharpness indicates clear, well-focused images

B.4 Intensity statistics

Intensity features capture overall brightness characteristics and variations.

B.4.1 Mean intensity

- **Definition**: Average pixel intensity
- **Formula**: $\frac{1}{N}\sum_{i=1}^{N} I_i$
- **Units**: Pixel intensity (0-255 for 8-bit images)
- **Medical Relevance**: Overall tissue brightness and illumination

B.4.2 Standard deviation intensity

- **Definition**: Measure of intensity variation
- **Formula**: $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(I_i - \mu)^2}$
- **Units**: Pixel intensity
- **Medical Relevance**: High values indicate good contrast and detail

B.4.3 Minimum intensity

- **Definition**: Darkest pixel value in the image
- **Units**: Pixel intensity (0-255)
- **Medical Relevance**: Represents shadows or dark residue

B.4.4 Maximum intensity

- **Definition**: Brightest pixel value in the image
- **Units**: Pixel intensity (0-255)
- **Medical Relevance**: Represents specular highlights or well-lit areas

B.4.5 Median intensity

- **Definition**: Middle value of sorted pixel intensities
- **Units**: Pixel intensity (0-255)
- **Medical Relevance**: Robust measure of typical tissue brightness

B.5 Advanced texture features

Advanced features capture complex patterns and micro-textures.

B.5.1 Image entropy

- **Definition**: Measure of randomness and complexity
- **Formula**: $-\sum_{i=1}^{N} p(i) \log_2 p(i)$ where $p(i)$ is intensity probability
- **Units**: Bits (information theory)
- **Medical Relevance**: High entropy indicates complex tissue patterns

B.5.2 LBP entropy (Local binary patterns)

- **Definition**: Entropy of local texture patterns
- **Units**: Bits
- **Medical Relevance**: Captures micro-texture variations in mucosa

B.5.3 Blob count

- **Definition**: Number of connected components in edge map
- **Units**: Count (integer)
- **Medical Relevance**: Many blobs indicate particulate matter or detailed structures

B.6 Clinical interpretation by BBPS score

Table 6: Typical feature patterns by Bowel preparation quality

Feature Category	BBPS 3 (Excellent)	BBPS 2 (Good)	BBPS 0-1 (Poor)
Texture	High contrast, medium homogeneity	Moderate contrast	Low contrast, high homogeneity
Color	Appropriate hue (pinkish-red)	Slight color deviation	Abnormal colors (brown/green)
Edges	High edge density, high sharpness	Reduced edge density	Very low edge density
Intensity	Good dynamic range	Some intensity loss	Poor dynamic range
Entropy	Medium-high complexity	Reduced complexity	Low complexity

B.6.1 BBPS 3 (Excellent preparation)

- High edge density and sharpness
- Appropriate mucosal colors (pinkish-red hues)
- Complex texture patterns (medium-high entropy)
- Good contrast and intensity range

B.6.2 BBPS 2 (Good preparation)

- Moderate edge density with some detail loss
- Slight color deviation from ideal
- Reduced texture complexity
- Acceptable but not optimal sharpness

B.6.3 BBPS 1 (Poor preparation)

- Low edge density with obscured details
- Significant color staining
- Homogeneous textures (low entropy)
- Poor sharpness and contrast

B.6.4 BBPS 0 (Unprepared)

- Very low edge density (completely obscured)
- Extreme color deviation (solid stool)
- Very simple textures
- Minimal visible mucosal patterns

B.7 Feature selection rationale

These 22 features were selected because they collectively capture:

- **Macro-texture**: Large-scale tissue patterns (GLCM features)
- **Micro-texture**: Fine details and complexity (entropy, LBP)
- **Color characteristics**: Tissue appearance vs. residual materials
- **Structural details**: Anatomical visibility (edge features)
- **Overall quality**: Illumination and focus (intensity statistics)

The combination of these diverse feature types enables robust differentiation between preparation quality levels, as demonstrated by the 92.5%-95.4% classification accuracy achieved in this project.

C Project

C.1 Project structure details

Complete file structure of the Nerthus Medical ML project:

```
nerthus-medical-ml/
|-- AUTHORS.txt                                  # Author(s)
|-- examples
|   |-- nerthus_cnn_champion.py
|   |-- nerthus_cnn_improved.py
|   |-- nerthus_cnn_simple.py
|   |-- nerthus_cnn_tuning.py
|   |-- nerthus_comparison_plot.py
|   |-- nerthus_image_analyzer.py
|   |-- nerthus_image_processor.py
|   `-- nerthus_ml_pipeline.py
|-- nerthus
|   |-- analyzer.py                              # Data analysis & EDA
|   |-- cli.py                                   # Command line interface
|   |-- cnn.py                                   # Deep learning
|   |-- __init__.py
|   |-- ml.py                                    # Traditional ML
|   |-- processor.py                             # Image processing
|   `-- utils.py                                 # Utilities
```

```
|-- pyproject.toml                                  # Modern Python packaging
|-- README.md                                       # Project description
|-- report
|   |-- images                                      # Visualization files
|   |-- Makefile                                    # Building the report
|   |-- Report.pdf                                  # Report PDF file
|   |-- Report.tex                                  # Report LateX file
|   `-- requirements.sh                             # Prerequisites for the report
|-- requirements.txt                                # Python project requirements
|-- setup.py                                        # Setup file
`-- web
    |-- app.py                                      # Web app file
    |-- debug_cnn.py                                # CNN debug file
    |-- loader.py                                   # Model loader
    |-- requirements.txt                            # Web app requirements
    |-- run.py                                      # Run web app with Python
    |-- static                                      # Static file for the web app
    `-- utils.py                                    # Web app utilities
```

C.2 Model configuration details

C.2.1 Optimal CNN hyperparameters

- **Dropout rates**: [0.1, 0.2, 0.3, 0.2]
- **Batch size**: 32
- **Learning rate**: 0.001 with reduction on plateau
- **Early stopping**: Patience of 15 epochs
- **Input shape**: 150x150x3

C.2.2 ML parameters

- **n_estimators**: 100
- **max_depth**: None (unlimited)
- **random_state**: 42
- **n_jobs**: -1 (use all cores, CPU-based)

C.3 ML confusion matrix

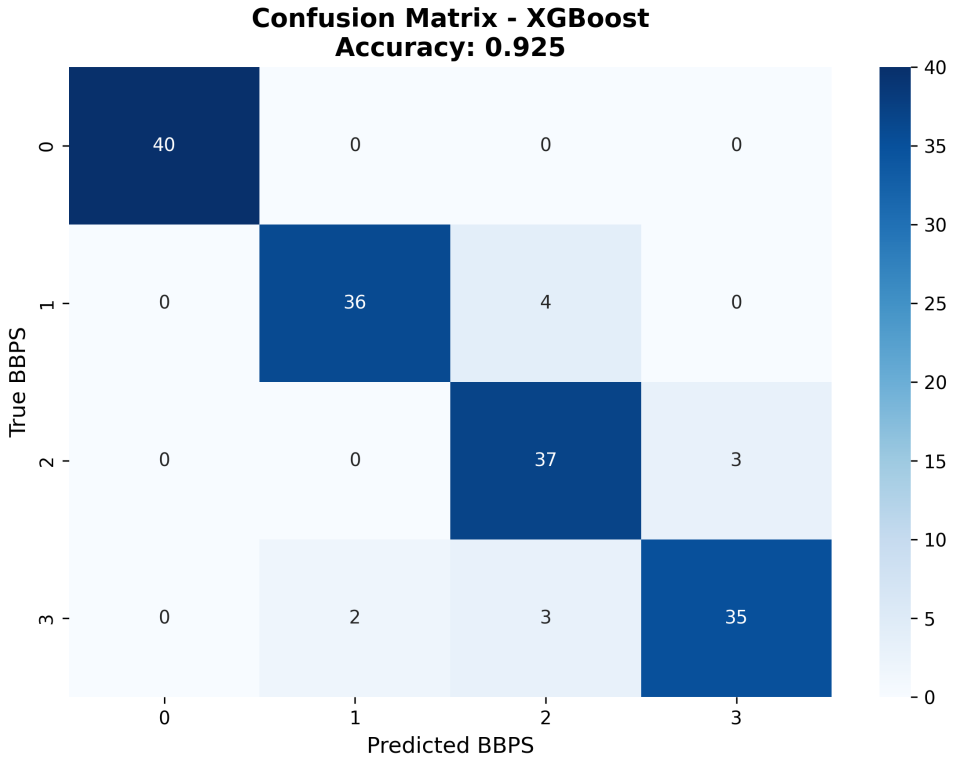


Figure 7: Confusion matrix XGBoost

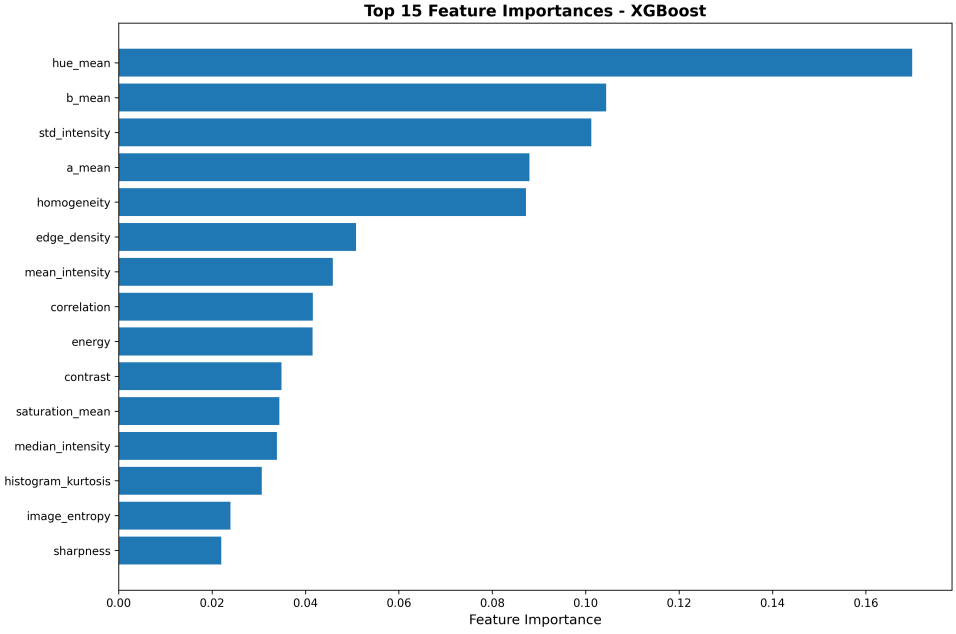


Figure 8: Feature importance XGBoost