Deepfake Detection Methodology - April 14th

Methodology for Deepfake Detection using Deep Feature Stacking and Meta-Learning

1. Dataset Preparation and Preprocessing

1.1. Dataset Acquisition

- Utilized the CelebDF V2 dataset containing both authentic and manipulated (deepfake) face videos
- Dataset organized into three splits: training (80%), validation (10%), and testing (10%)
- Each split contains two classes: real faces and synthetically generated (fake) faces

1.2. Face Extraction and Preprocessing

- **Face Detection**: Implemented a robust CNN-based face detection pipeline using RetinaFace (InsightFace library)
    - RetinaFace selected for its superior performance in detecting faces under various conditions
    - Used "buffalo_l" model configuration for optimal accuracy-speed trade-off
    - Leveraged GPU acceleration (CUDA) when available for faster processing
- **Sampling Strategy**:
    - Extracted multiple frames (10 frames per video) at equidistant intervals
    - This approach ensures representation of temporal variations within each video
- **Face Region Processing**:
    - Extracted facial bounding boxes (x1, y1, x2, y2) from detected faces

- Applied padding to maintain facial context
- Resized facial regions to 299×299 pixels using aspect-preserving resizing
- Used bilinear interpolation for resizing to preserve facial details

- **Dataset Organization**:
  - Organized processed faces into separate folders for real and fake classes
  - Maintained train, validation, and test splits for rigorous evaluation
  - Ensured class balance to prevent bias during model training

2. Deep Feature Extraction Architecture

2.1. CNN Model Selection and Configuration

- **Dual CNN Architecture**: Employed two complementary pre-trained CNN models to extract diverse feature representations:
  - **Xception Network**: Selected for its depth-wise separable convolutions and efficient feature extraction capabilities
  - **EfficientNetV2L**: Chosen for its advanced compound scaling methodology and state-of-the-art performance
- **Model Configuration**:
  - Both models initialized with ImageNet pre-trained weights
  - Base models modified by removing classification layers (include_top=False)
  - Added custom classification heads with global average pooling
  - Implemented dropout layers (0.5, 0.3) to prevent overfitting
  - Final sigmoid activation for binary classification (real vs. fake)

2.2. Data Augmentation Pipeline

- **Image-specific Augmentation**:
  - Implemented model-specific preprocessing functions:

        - xception_preprocess for Xception
        - efficientnetv2_preprocess for EfficientNetV2L
    - Applied real-time augmentation during training:
        - Rotation (±20°)
        - Width/height shifts (±20%)
        - Shear transformations (±20%)
        - Zoom variations (±20%)
        - Horizontal flips
    - Used nearest-neighbor fill mode for maintaining border integrity

2.3. Dual-Phase Training Strategy

- **Phase 1**: Feature Extractor Fine-tuning
    - Froze base CNN layers to preserve pre-trained ImageNet weights
    - Trained only custom classification layers (1024→512→1 neurons)
    - Used binary cross-entropy loss and Adam optimizer
    - Implemented cosine annealing learning rate scheduling with warm restarts
        - Initial learning rate: 1e-3
        - Minimum learning rate: 1e-6
        - Cycle length (T_0): 10 epochs
        - Multiplication factor (T_mult): 2
- **Phase 2**: Selective Deep Fine-tuning
    - Unfroze specific sections of base models:
        - Last 30 layers for Xception
        - Last 30% of layers for EfficientNetV2L
    - Reduced learning rate (1e-4) with adjusted cosine annealing scheduling
    - Applied early stopping with 5-epoch patience

- Used model checkpointing to save best validation performance
- Retained separate models for feature extraction

3. Feature Extraction and Stacking Framework

3.1. Deep Feature Extraction

- Utilized global average pooling from the last convolutional layers to obtain compact feature vectors
- Extracted features separately for both models:
    - Xception: 2048-dimensional feature vectors
    - EfficientNetV2L: 1280-dimensional feature vectors
- Maintained identical data order across both extraction processes for proper alignment

3.2. Feature Stacking Approach

- Concatenated Xception and EfficientNetV2L features along feature dimension
- Created 3328-dimensional stacked feature vectors (2048+1280)
- Preserved feature stacking separately for train, validation, and test sets
- Ensured synchronization of labels across all feature extraction processes

4. Feature Selection and Dimensionality Reduction

4.1. Multi-model Feature Importance Ranking

- Implemented a novel dual-model importance ranking methodology:
    - **Random Forest Feature Ranking**: Trained a Random Forest classifier (100 trees) on stacked features
    - **XGBoost Feature Ranking**: Trained an XGBoost classifier (100 estimators) on stacked features
    - Combined importance scores by computing the average between both models

4.2. Correlation-based Feature Filtering

- Computed Pearson correlation coefficients between all feature pairs
- Applied threshold-based filtering ($r > 0.9$) to identify highly correlated features
- Implemented sequential feature selection to remove redundant features while preserving unique information
- Visualized correlation matrices before and after filtering for validation

4.3. Recursive Feature Elimination with Cross-Validation (RFECV)

- Applied RFECV with 5-fold stratified cross-validation
- Used F1-score as the optimization metric
- Implemented separate RFECV processes for Random Forest and XGBoost
- Combined feature masks by selecting features chosen by at least one model
- Retained the top k% (k=30) features based on combined importance scores
- Visualized feature importance distributions for interpretability

5. Meta-Learner Architecture and Training

5.1. Meta-Learner Design

- Implemented a custom Multi-Layer Perceptron (MLP) architecture:
  - Input layer: Dimensionality matching selected features
  - Hidden layers: [256, 128, 64] neurons with ReLU activation
  - Dropout layers: [0.5, 0.5, 0.3] for regularization
  - Output layer: Single neuron with sigmoid activation for binary classification

5.2. Enhanced Training Protocol

- Optimized using Adam optimizer with initial learning rate of 1e-3
- Implemented learning rate reduction on plateau (factor=0.2, patience=3)
- Applied early stopping with restoration of best weights

- Used binary cross-entropy as the loss function
- Trained for maximum 30 epochs with batch size of 32
- Monitored validation accuracy to prevent overfitting

5.3. Ensemble Meta-Learner Implementation (for Enhanced Version)

- Developed a weighted ensemble of multiple classifiers:
  - SVM with RBF kernel and probability calibration
  - XGBoost with parameter tuning and isotonic calibration
  - MLP neural network with optimized architecture
- Assigned weights based on individual models' F1 scores on validation data
- Implemented weighted probability averaging for final predictions
- Analyzed model agreement patterns on test set predictions
- Validated calibration quality through reliability diagrams

6. Model Evaluation and Robustness Testing

6.1. Comprehensive Performance Evaluation

- Implemented multi-metric evaluation framework:
  - Accuracy: Overall classification correctness
  - Precision: Reliability of positive (fake) predictions
  - Recall: Completeness of positive (fake) class detection
  - F1-Score: Harmonic mean of precision and recall
  - AUC-ROC: Model's ability to discriminate between classes
- Generated confusion matrices for error pattern analysis
- Produced detailed classification reports with per-class metrics

6.2. Robustness Testing Methodology

- Conducted systematic robustness testing against brightness variations:
  - Tested five brightness levels: 50%, 75%, 100%, 125%, and 150%
  - Created modified test sets with controlled brightness adjustments
  - Recomputed feature extraction and selection pipeline for each level
  - Evaluated performance stability across brightness variations
  - Analyzed specific failure patterns under different conditions

6.3. Visualization and Interpretability

- Implemented visualization pipeline for model predictions:
  - Sample image visualization with prediction overlay
  - Correct vs. incorrect prediction analysis
  - Confidence score distribution analysis
- Applied GradCAM (Gradient-weighted Class Activation Mapping) to visualize regions of attention:
  - Highlighted facial regions most influential for classification
  - Compared attention maps between model types
  - Analyzed attention differences between correctly and incorrectly classified examples

7. Deployment and Inference Framework

7.1. Model Persistence

- Saved trained meta-learner model in H5 format
- Preserved feature selection indices for consistent inference
- Stored performance metrics for reference and comparison
- Created comprehensive model documentation for reproducibility

7.2. Real-time Inference Pipeline

- Implemented single-image and video inference capabilities:

    - Consistent preprocessing with training phase
    - Feature extraction using both CNN models
    - Feature stacking and selection using preserved indices
    - Efficient meta-learner prediction with calibrated confidence scores
- Integrated timing measurements for performance benchmarking
- Developed visualization tools for inference results:
    - Confidence score display
    - Frame-by-frame analysis for videos
    - Temporal consistency verification

7.3. Additional Datasets and Extensibility

- Identified compatible datasets for methodology extension:
    - FaceForensics++ (FF++): 1,000 original videos and 4,000 manipulated videos
    - DeepFake Detection Challenge (DFDC): 124K videos with varied deepfake techniques
    - DeeperForensics-1.0: 60K manipulated videos with 17.6M frames
    - WildDeepfake: 7,314 real-world deepfake sequences
    - UADFV: 49 real and 49 deepfake videos (suitable for rapid testing)
- Ensured preprocessing compatibility across datasets
- Established transfer learning methodology for cross-dataset adaptation

This comprehensive methodology details our end-to-end approach for deepfake detection, from data preprocessing to model deployment, with emphasis on feature extraction, selection, and meta-learning techniques that achieve state-of-the-art performance.

```
# Additional Deepfake Detection Datasets

Here are additional datasets that would be suitable for our workflow, considering

## 1. FaceForensics++ (FF++)
- **Description**: Contains 1,000 original videos and 4,000 manipulated videos u
- **Advantages**: Multiple manipulation techniques (Deepfakes, Face2Face, Fac
- **Compatibility**: Videos that can use our existing face extraction pipeline
- **Website**: [https://github.com/ondyari/FaceForensics](https://github.com/or

## 2. DeepFake Detection Challenge (DFDC)
- **Description**: Released by Facebook, contains 124K videos with various dee
- **Advantages**: Large-scale, diverse demographic representation
- **Compatibility**: Video format similar to CelebDF, compatible with our pipeline
- **Website**: [https://ai.facebook.com/datasets/dfdc/](https://ai.facebook.com/

## 3. DeeperForensics-1.0
- **Description**: 60K manipulated videos with 17.6M frames created using Deep
- **Advantages**: Includes various real-world perturbations (compression, blurr
- **Compatibility**: Consistent with our processing pipeline
- **Website**: [https://github.com/EndlessSora/DeeperForensics-1.0](https://gith

## 4. WildDeepfake
- **Description**: 7,314 face sequences extracted from various social platforms
- **Advantages**: 'In-the-wild' deepfakes with real-world degradations and com
- **Compatibility**: Already provides cropped faces, minimal preprocessing nee
- **Website**: [https://github.com/deepfakeinthewild/deepfake-in-the-wild](http

## 5. UADFV (University at Albany DeepFake Videos)
- **Description**: 49 real videos and 49 deepfake videos with 32,752 frames in t
- **Advantages**: Smaller dataset good for quick experimentation
- **Compatibility**: Fairly clean videos that work well with our face extraction
- **Website**: Part of [https://github.com/danmohaha/WIFS2018_In_Ictu_Oculi](h

## Preprocessing Considerations
```

```
For adapting these datasets to our workflow:

1. **Face Detection**: Our current RetinaFace-based pipeline works well with all
2. **Image Resizing**: The current 299×299 preprocessing for Xception is comp
3. **Augmentation**: Current data augmentation techniques apply well to these d
4. **Visual Artifacts**: Some datasets (especially WildDeepfake) contain compre

Using multiple datasets would help our model generalize better across different d
```