Trinity V5.1

ML & Deep Learning Audio Pipeline

Technical Documentation

⚠️ **ALPHA VERSION - PLEASE TAKE NOTE**

Glass Stone LLC
Glass Stone Research

Built by Gabriel B. Rodriguez, CEO

Version 5.1.0a (Alpha)

January 11, 2026

Runs in Cloud | macOS | Windows

Trinity V5.1

ML & Deep Learning Audio Pipeline

Technical Documentation

Developed by Glass Stone LLC | Glass Stone Research
Built by: Gabriel B. Rodriguez, CEO
Last Updated: January 11, 2026
Version: 5.1.0a (ALPHA)
Product: Trinity Vocal Enhancement Pipeline

Deployment Options: Cloud | macOS | Windows
Integration: Voxis Cloud-Based Audio Enhancement Solution

⚠️ IMPORTANT NOTICE - ALPHA SOFTWARE

This software is currently in ALPHA status. Please take note:

Trinity V5.1.0a is under active development and testing. While functional and performant, users should be aware: - Features and APIs may change without notice - Production deployments should include thorough testing - Performance characteristics may vary across platforms - Bug reports and feedback are actively encouraged - Not all features are fully optimized for all platforms

For production-critical applications, please contact Glass Stone Research for enterprise support and stability guarantees.

Table of Contents

1. Executive Summary
2. System Overview
3. Architecture
4. Core Components
5. Machine Learning Models
6. Processing Pipeline
7. Technical Specifications
8. Voxis Integration
9. Performance Metrics
10. API Reference
11. Deployment Guide
12. Troubleshooting
13. Future Roadmap
14. References

---

1. Executive Summary

Trinity V5.1 represents Glass Stone LLC's flagship ML-powered audio processing pipeline, designed to deliver professional-grade vocal enhancement through advanced deep learning techniques. As the core engine powering Voxis, Trinity processes audio through a multi-stage pipeline optimized for real-world scenarios including church acoustics, podcast production, conference recordings, and multimedia content creation.

Key Capabilities: - Adaptive noise reduction using custom-trained ML models - Intelligent vocal isolation across diverse acoustic environments - Audio restoration for damaged or degraded recordings - Real-time spectrum analysis and adaptive EQ - Support for both male and female vocal profiles - GPU-accelerated processing for large-scale workloads

---

2. System Overview

2.1 Purpose and Scope

Trinity V5.1 serves as the backend audio processing engine for Voxis, a cloud-deployed audio enhancement solution. The pipeline addresses the fundamental challenge of extracting clean, intelligible vocal content from recordings captured in sub-optimal acoustic environments.

Deployment Flexibility: Trinity V5.1 is designed to run across multiple platforms: - **Cloud Deployment**: AWS, Google Cloud, Azure with GPU acceleration - **macOS**: Native support for Apple Silicon (M1/M2/M3) and Intel-based Macs - **Windows**: Windows 10/11 with CUDA GPU support or CPU-only mode

This cross-platform capability ensures Trinity can be deployed in enterprise data centers, edge computing environments, or on individual workstations depending on workflow requirements.

2.2 Design Philosophy

The Trinity architecture follows three core principles:

1. **Separation of Concerns**: Spectrum analysis operates independently from ML denoising, allowing modular optimization
2. **Model Efficiency**: Neural architectures optimized for inference speed without sacrificing quality
3. **Scalability**: Designed for both edge deployment and cloud-scale processing

2.3 Version History

- **v5.0.x**: Initial production release with DeepFilterNet integration
- **v5.1.0a**: Current version featuring enhanced spectrum analyzer and expanded training dataset

---

3. Architecture

3.1 High-Level Architecture

```
┌──────────────────────────────────────────────────────────┐
│                    TRINITY V5.1 PIPELINE                  │
└──────────────────────────────────────────────────────────┘
                             │
                             ▼
┌──────────────────────────────────────────────────────────┐
│              STAGE 1: SPECTRUM ANALYSIS                   │
│  ┌────────────────────────────────────────────────────┐  │
│  │  Custom Spectrum Analyzer                          │  │
│  │  • FFT-based frequency decomposition               │  │
│  │  • Dynamic range analysis                          │  │
│  │  • Vocal formant detection                         │  │
│  │  • Noise floor estimation                          │  │
│  └────────────────────────────────────────────────────┘  │
└───────────────────────────┬──────────────────────────────┘
                            │
                            ▼
┌──────────────────────────────────────────────────────────┐
│              STAGE 2: ML DENOISING                        │
│  ┌────────────────────────────────────────────────────┐  │
│  │  Custom-Trained Denoiser                           │  │
│  │  • U-Net based architecture                        │  │
│  │  • Trained on 200+ hours of vocal data            │  │
│  │  • Multi-scenario conditioning                     │  │
│  │  • Gender-adaptive processing                      │  │
│  └────────────────────────────────────────────────────┘  │
└───────────────────────────┬──────────────────────────────┘
                            │
                            ▼
┌──────────────────────────────────────────────────────────┐
│            STAGE 3: VOCAL ISOLATION                       │
│  ┌────────────────────────────────────────────────────┐  │
│  │  Source Separation Engine                          │  │
│  │  • Deep learning-based stem extraction            │  │
│  │  • Adaptive spectral gating                        │  │
│  │  • Harmonic-percussive separation                  │  │
│  └────────────────────────────────────────────────────┘  │
└───────────────────────────┬──────────────────────────────┘
                            │
                            ▼
┌──────────────────────────────────────────────────────────┐
│              STAGE 4: RESTORATION                         │
│  ┌────────────────────────────────────────────────────┐  │
│  │  Audio Enhancement Suite                           │  │
│  │  • Super-resolution upsampling (AudioSR)        │  │
│  │  • Artifact reduction                           │  │
│  │  • Dynamic range optimization                   │  │
│  │  • Spectral balancing                           │  │
│  └─────────────────────────────────────────────────┘  │
└──────────────────────────┬────────────────────────────┘
                           │
                           ▼
┌───────────────────────────────────────────────────────┐
│              STAGE 5: OUTPUT GENERATION               │
│  ┌─────────────────────────────────────────────────┐  │
│  │ Format Conversion & Export                      │  │
│  │  • Multi-format support (WAV, MP3, FLAC)        │  │
│  │  • Bitrate optimization                         │  │
│  │  • Metadata preservation                        │  │
│  └─────────────────────────────────────────────────┘  │
└───────────────────────────────────────────────────────┘
```

3.2 Component Interaction

The Trinity pipeline operates as a sequential processing chain where each stage receives the output of the previous stage. However, the spectrum analyzer operates independently and provides metadata that conditions downstream ML models rather than directly modifying the audio stream.

---

4. Core Components

4.1 Spectrum Analyzer Module

Purpose: Provides detailed frequency-domain analysis to inform downstream processing decisions.

Technical Implementation: - **Algorithm**: Short-Time Fourier Transform (STFT) with Hann windowing - **Window Size**: Configurable, default 2048 samples - **Hop Length**: 512 samples (75% overlap) - **Frequency Resolution**: 21.5 Hz at 44.1kHz sample rate - **Output**: JSON metadata containing spectral features

Key Features: - Real-time fundamental frequency (F0) tracking - Formant frequency extraction for vocal characterization - Noise floor estimation using percentile-based statistical methods - Spectral centroid and bandwidth computation - Harmonic-to-noise ratio (HNR) calculation

Separation Rationale: The spectrum analyzer is architecturally separated from the Trinity core pipeline to enable: 1. Independent optimization and updating 2. Reusability across different audio processing contexts 3. Modular testing and validation 4. Potential real-time analysis during live recording

Code Reference: `voxis/backend/utils/spectrum_analyzer.py`

4.2 ML Denoiser

Architecture: Modified U-Net with attention mechanisms

Model Specifications: - **Framework**: PyTorch 2.0+ - **Input**: Magnitude spectrogram (STFT) - **Output**: Denoised magnitude spectrogram - **Parameters**: ~12.4M trainable parameters - **Inference Time**: 1.2x real-time on CPU, 8.5x on NVIDIA T4 GPU

Training Dataset: The denoiser was trained on a comprehensive dataset encompassing: - **Total Duration**: 200+ hours of labeled audio - **Recording Environments**: - Studio recordings (clean baseline) - Conference rooms with HVAC noise - Church sanctuaries with reverb and ambient noise - Outdoor environments (wind, traffic) - Home offices with background activity - Classroom and lecture hall acoustics - **Noise Conditions**: - Background speech (babble noise) - Mechanical noise (fans, HVAC) - Electronic hum (50/60Hz power line interference) - Reverberation and room reflections - Transient noises (door slams, coughs, clicks)

Gender Representation: - Male voices: 52% of training data - Female voices: 48% of training data - Age range: 18-75 years - Accent diversity: North American, British, Australian, Indian English variants

Training Methodology: - **Loss Function**: Combination of L1 magnitude loss and perceptual loss - **Augmentation**: Dynamic mixing of clean speech with noise at SNR ratios from -5dB to 20dB - **Epochs**: 150 with early stopping on validation set - **Validation**: 15% holdout set with stratified sampling by gender and environment

Performance Metrics: - **PESQ**: 3.85 (Mean Opinion Score equivalent) - **STOI**: 0.94 (Short-Time Objective Intelligibility) - **SI-SDR**: 18.2 dB improvement over noisy input

Model Weights: Hosted on Hugging Face Hub
Integration: Loaded via `torch.jit` for optimized inference

4.3 Vocal Isolation Engine

Technology Stack: **- Primary Model**: Demucs v4 (Hybrid Transformer-CNN architecture) **- Fallback**: Spleeter 5-stem model for compatibility **- Custom Enhancements**: Spectral gating post-processing

Stem Separation: Trinity extracts the following stems: 1. Vocals (primary target) 2. Instrumental (drums + bass + other) 3. Residual (artifacts and ambiguous content)

Processing Strategy: - Vocals stem is retained and passed to restoration - Instrumental stem is analyzed for bleed detection - Residual stem is inspected for leakage artifacts

Isolation Quality Metrics: **- SDR (Source-to-Distortion Ratio)**: 12.8 dB average **- SIR (Source-to-Interference Ratio)**: 16.4 dB average **- SAR (Source-to-Artifacts Ratio)**: 11.2 dB average

---

4.4 Restoration Module

Components: 1. **Super-Resolution Upsampling** (AudioSR) - Increases audio bandwidth for degraded recordings - Reconstructs high-frequency content using learned priors - Particularly effective for telephone-quality audio

1. **Artifact Reduction**
2. Median filtering for transient click removal
3. Spectral subtraction for residual noise
4. De-essing for sibilance control
5. **Dynamic Range Optimization**
6. Adaptive compression to maintain vocal presence
7. Limiting to prevent clipping
8. Gate expansion for background silence
9. **Spectral Balancing**
10. EQ curve optimization based on spectrum analyzer output
11. Formant preservation
12. Presence enhancement (2-5 kHz boost)

Configurable Parameters: - Restoration strength (0-100%) - High-frequency enhancement level - Compression ratio and threshold - EQ curve selection (neutral, warm, bright, broadcast)

---

5. Machine Learning Models

5.1 Denoiser Model Architecture

```
Input: Log-Magnitude Spectrogram [B, 1, F, T]
|
├─ Encoder (Downsampling Path)
|  ├─ Conv2D(1, 32) + BatchNorm + ReLU
|  ├─ Conv2D(32, 64) + BatchNorm + ReLU + MaxPool2D
|  ├─ Conv2D(64, 128) + BatchNorm + ReLU + MaxPool2D
|  └─ Conv2D(128, 256) + BatchNorm + ReLU + MaxPool2D
|
├─ Bottleneck
|  └─ Conv2D(256, 512) + BatchNorm + ReLU
|
├─ Decoder (Upsampling Path)
|  ├─ TransposeConv2D(512, 256) + BatchNorm + ReLU
|  ├─ Concatenate(Skip256) + Conv2D(512, 256)
|  ├─ TransposeConv2D(256, 128) + BatchNorm + ReLU
|  ├─ Concatenate(Skip128) + Conv2D(256, 128)
|  ├─ TransposeConv2D(128, 64) + BatchNorm + ReLU
|  └─ Concatenate(Skip64) + Conv2D(128, 64)
|
└─ Output Layer
   └─ Conv2D(64, 1) + Sigmoid

Output: Denoised Magnitude Spectrogram [B, 1, F, T]

Legend:
B = Batch size
F = Frequency bins
T = Time frames
Skip* = Skip connections from encoder
```

5.2 Model Training Infrastructure

Hardware: - Training: 4x NVIDIA A100 40GB GPUs - Distributed Data Parallel (DDP) training - Mixed precision (FP16) training for efficiency

Software Stack: - PyTorch 2.0.1 - TorchAudio 2.0.2 - CUDA 12.1 - cuDNN 8.9.0

Hyperparameters: - Learning rate: 1e-4 with cosine annealing - Batch size: 32 per GPU (effective 128) - Optimizer: AdamW (β1=0.9, β2=0.999, weight_decay=1e-5) - Gradient clipping: max_norm=1.0

5.3 Model Versioning and Updates

Trinity employs a continuous improvement cycle: - **Quarterly Retraining**: Models are retrained with expanded datasets every 3 months - **A/B Testing**: New model versions undergo blind listening tests before deployment - **Rollback Capability**: Previous model weights retained for 12 months - **Version Control**: Models tracked via Git LFS and Weights & Biases

Current Model Versions (v5.1.0a): - Denoiser: `trinity-denoiser-v5.1.0a.pt` - Vocal Isolator: `demucs-v4-htdemucs_ft.pt` - Audio SR: `audiosr-48khz.ckpt`

---

6. Processing Pipeline

6.1 Input Validation

Before processing begins, Trinity validates: - **File Format**: WAV, MP3, FLAC, M4A, OGG - **Sample Rate**: 16kHz - 192kHz (resampled to 44.1kHz internally) - **Bit Depth**: 16-bit or 24-bit - **Channels**: Mono or stereo (stereo converted to mono for vocal processing) - **Duration**: Maximum 72 hours per file - **File Size**: Up to 2GB

Error Handling: Corrupted or unsupported files trigger graceful error messages with suggested remediation.

6.2 Preprocessing

Steps: 1. **Normalization**: Peak normalize to -3dBFS to prevent clipping 2. **Resampling**: Convert to 44.1kHz if necessary 3. **Channel Conversion**: Downmix stereo to mono using weighted average 4.

DC Offset Removal: High-pass filter at 20Hz 5. **Chunking**: Segment long files into 30-second overlapping chunks for parallel processing

6.3 Core Processing Workflow

```
# Pseudocode representation of Trinity pipeline

def process_audio(input_file):
    # Stage 1: Analysis
    spectrum_data = spectrum_analyzer.analyze(input_file)

    # Stage 2: Denoising
    denoised = ml_denoiser.process(
        input_file,
        conditioning=spectrum_data
    )

    # Stage 3: Isolation
    vocal_stem = vocal_isolator.separate(
        denoised,
        model="demucs-v4"
    )

    # Stage 4: Restoration
    restored = restoration_suite.enhance(
        vocal_stem,
        spectrum_guide=spectrum_data,
        strength=config.restoration_strength
    )

    # Stage 5: Output
    output = format_converter.export(
        restored,
        format=config.output_format,
        bitrate=config.output_bitrate
    )

    return output
```

6.4 Postprocessing

Final Steps: 1. **Gain Staging**: Normalize output to target LUFS (default: -16 LUFS) 2. **Metadata Embedding**: Preserve and update ID3/Vorbis tags 3. **Quality Validation**: Automated checks for clipping, silence, and artifacts 4. **Checksum Generation**: MD5 hash for output integrity verification

---

7. Technical Specifications

7.1 System Requirements

Cloud Deployment: - **AWS**: p3.2xlarge, g4dn.xlarge, or g5.xlarge instances - **Google Cloud**: n1-standard-4 with NVIDIA T4 or A100 - **Azure**: NC6s_v3 with Tesla V100 or NVv4-series - **OS**: Ubuntu 20.04+, Container-Optimized OS - **Container Runtime**: Docker 24+ with NVIDIA Container Toolkit

macOS (Local): - **Apple Silicon (M1/M2/M3)**: - RAM: 16GB minimum, 32GB recommended - Storage: 20GB free SSD space - macOS: 12.0 (Monterey) or later - Note: GPU acceleration via Metal Performance Shaders

- **Intel-based Mac**:
- CPU: Intel Core i7 8th gen or later
- RAM: 16GB minimum, 32GB recommended
- Storage: 20GB free SSD space
- macOS: 11.0 (Big Sur) or later
- Note: CPU-only processing (no GPU acceleration)

Windows (Local): - **With NVIDIA GPU**: - GPU: NVIDIA GPU with 8GB+ VRAM (RTX 2060 or better) - CUDA: 12.1+ with compatible drivers - CPU: Intel Core i5 10th gen or AMD Ryzen 5 5000 series - RAM: 16GB minimum, 32GB recommended - Storage: 20GB free SSD space - OS: Windows 10 (build 19041+) or Windows 11

- **CPU-Only Mode**:
- CPU: Intel Core i7 10th gen or AMD Ryzen 7 5000 series
- RAM: 16GB minimum
- Storage: 20GB free SSD space

- OS: Windows 10 (build 19041+) or Windows 11

Minimum (CPU-Only - All Platforms): - CPU: 4 cores, 2.5GHz base clock - RAM: 8GB (processing limited to shorter files) - Storage: 10GB free space - Network: 10Mbps for cloud features

7.2 Performance Benchmarks

File Length	CPU (i7-10700)	GPU (RTX 3060)	GPU (T4)
1 minute	72 seconds	8 seconds	12 seconds
10 minutes	12 minutes	1.2 minutes	1.8 minutes
1 hour	72 minutes	7 minutes	11 minutes

Benchmarks include all pipeline stages with default settings

7.3 Model Size and Memory Footprint

Component	Model Size	RAM (Inference)	VRAM (GPU)
Denoiser	47 MB	1.2 GB	2.1 GB
Demucs v4	523 MB	3.8 GB	4.5 GB
AudioSR	231 MB	2.1 GB	3.2 GB
Total	**801 MB**	**7.1 GB**	**9.8 GB**

---

8. Voxis Integration

8.1 Architecture Overview

Trinity V5.1 serves as the backend processing engine for Voxis, a cloud-deployed web application. The integration follows a microservices architecture:

```
┌────────────────────────────────────────────────────────┐
│                 VOXIS WEB INTERFACE                    │
│              (React Frontend - Port 80)                │
└──────────────────────────┬─────────────────────────────┘
                           │ HTTP/REST API
                           ▼
┌────────────────────────────────────────────────────────┐
│              VOXIS BACKEND API LAYER                   │
│           (Flask + Gunicorn - Port 5001)               │
├────────────────────────────────────────────────────────┤
│  • File upload/download endpoints                     │
│  • Job queue management (Redis)                       │
│  • Authentication and rate limiting                   │
│  • Progress tracking and webhooks                     │
└──────────────────────────┬─────────────────────────────┘
                           │ Python API
                           ▼
┌────────────────────────────────────────────────────────┐
│                 TRINITY V5.1 PIPELINE                  │
│                    (Core Engine)                       │
├────────────────────────────────────────────────────────┤
│  • Spectrum Analyzer                                  │
│  • ML Denoiser (custom trained)                       │
│  • Vocal Isolation (Demucs v4)                        │
│  • Restoration Suite (AudioSR)                        │
│  • Output Generation                                  │
└────────────────────────────────────────────────────────┘
```

8.2 API Endpoints

Base URL: `https://voxis.glasstone.com/api`

Authentication: Bearer token (JWT)

Endpoints:

Method	Endpoint	Description
POST	`/upload`	Upload audio file for processing
GET	`/job/{job_id}`	Check job status
GET	`/download/{job_id}`	Download processed audio
GET	`/health`	Service health check
GET	`/stats`	Processing statistics

Example Request:

```
curl -X POST https://voxis.glasstone.com/api/upload \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -F "file=@input.wav" \
  -F "config={\"restoration_strength\": 75}"
```

Example Response:

```
{
  "job_id": "a3f2b8c9-1234-5678-90ab-cdef12345678",
  "status": "queued",
  "estimated_time": 120,
  "created_at": "2025-11-05T14:30:00Z"
}
```

8.3 Job Queue and Scaling

Queue Management: - Redis-backed job queue for asynchronous processing - Priority queuing (premium users processed first) - Automatic retry on transient failures (max 3 attempts) - Dead letter queue for failed jobs

Horizontal Scaling: - Kubernetes-based deployment with auto-scaling - Scale workers based on queue depth - Load balancer distributes requests across backend pods - Shared storage (S3/GCS) for file uploads and outputs

Concurrency: - Default: 4 concurrent jobs per worker - GPU workers: 2 concurrent jobs (memory constraints) - Maximum global concurrency: 50 jobs (configurable)

---

9. Performance Metrics

9.1 Quality Metrics

Trinity achieves the following quality benchmarks on standard test datasets:

DNS Challenge 2020 Dataset: - PESQ: 3.85 (vs 3.21 baseline) - STOI: 0.94 (vs 0.78 baseline) - SI-SDR: 18.2 dB (vs 8.1 dB baseline)

LibriSpeech Test-Clean: - WER (Word Error Rate) improvement: 23% reduction when used as ASR preprocessing - Perceptual quality: 4.2/5.0 MOS (Mean Opinion Score)

Custom Church Acoustics Test Set (Glass Stone internal): - Reverb reduction: 68% RT60 reduction - Intelligibility improvement: 89% of listeners rated "significantly improved" - Background noise suppression: 15-22 dB SNR improvement

9.2 Speed and Efficiency

Processing Speed (RTX 3060 GPU): - Real-time factor: 0.12 (8.5x faster than real-time) - Latency: 450ms startup + processing time - Throughput: 8.5 hours of audio per hour of wall-clock time

Resource Utilization: - CPU: 65-80% on single worker thread during I/O - GPU: 90-95% utilization during model inference - Memory: Stable usage, no leaks detected over 72-hour continuous operation

9.3 Reliability Metrics

Uptime and Availability (Last 90 Days): - Service uptime: 99.7% - Successful job completion rate: 98.3% - Average error rate: 1.7%

Error Categories: - Unsupported format: 0.8% - Timeout (>72 hours): 0.3% - Out of memory: 0.4% - Unexpected failures: 0.2%

---

10. API Reference

10.1 Python API

Trinity can be used as a standalone Python library:

```
from trinity import TrinityPipeline

# Initialize pipeline
pipeline = TrinityPipeline(
    device="cuda",  # or "cpu"
    restoration_strength=75,
    output_format="wav"
)

# Process audio file
result = pipeline.process(
    input_path="noisy_audio.wav",
    output_path="clean_audio.wav",
    progress_callback=lambda p: print(f"Progress: {p}%")
)

# Access metadata
print(f"SNR Improvement: {result.snr_improvement} dB")
print(f"Processing Time: {result.processing_time} seconds")
```

10.2 Configuration Options

TrinityPipeline Parameters:

Parameter	Type	Default	Description
`device`	str	"cpu"	Device for inference ("cpu", "cuda", "mps")
`restoration_strength`	int	75	Restoration intensity (0-100)
`output_format`	str	"wav"	Output format (wav, mp3, flac)
`sample_rate`	int	44100	Output sample rate (Hz)
`denoise_model`	str	"default"	Denoiser model version
`isolation_model`	str	"demucs-v4"	Vocal isolation model
`enable_super_res`	bool	True	Enable AudioSR upsampling

10.3 Event Callbacks

```
def progress_callback(progress: int, stage: str):
    """
    Called during processing to report progress.

    Args:
        progress: Percentage complete (0-100)
        stage: Current stage ("analysis", "denoise", "isolate", "restore")
    """
    print(f"{stage}: {progress}%")

def error_callback(error: Exception, context: dict):
    """
    Called when an error occurs.

    Args:
        error: The exception that was raised
        context: Dictionary with error context
    """
    logging.error(f"Processing failed: {error}")
```

11. Deployment Guide

11.1 Cloud Deployment (Docker)

Trinity is distributed as a Docker container for consistent deployment:

Pull Image:

```
docker pull glasstonellc/voxis:latest
```

Run Container:

```
docker run -d \
  --name voxis \
  --gpus all \
  -p 5001:5001 \
  -v /path/to/audio:/app/audio \
  -e VOXIS_GPU_ENABLED=true \
  glasstonellc/voxis:latest
```

Environment Variables: - `VOXIS_GPU_ENABLED`: Enable GPU acceleration (true/false) - `VOXIS_MAX_FILE_SIZE`: Maximum upload size in bytes - `VOXIS_JOB_TIMEOUT`: Job timeout in hours - `VOXIS_DEBUG`: Enable debug logging (true/false)

11.2 macOS Local Installation

Prerequisites:

```
# Install Homebrew (if not already installed)
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Install Python 3.10+
brew install python@3.10

# Install system dependencies
brew install ffmpeg portaudio
```

Install Trinity:

```
# Clone repository
git clone https://github.com/glasstonellc/trinity.git
cd trinity

# Create virtual environment
python3 -m venv venv
source venv/bin/activate

# Install Python dependencies
pip install -r requirements.txt

# Download model weights
python scripts/download_models.py
```

Apple Silicon (M1/M2/M3) Optimization:

```
# Install Metal Performance Shaders support
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu

# Enable MPS acceleration (automatic on supported hardware)
export PYTORCH_ENABLE_MPS_FALLBACK=1
```

Run Trinity on macOS:

```
# CPU mode
python trinity_cli.py --input audio.wav --output clean.wav

# MPS mode (Apple Silicon)
python trinity_cli.py --input audio.wav --output clean.wav --device mps
```

11.3 Windows Local Installation

Prerequisites: 1. Install Python 3.10+ - Check "Add Python to PATH" during installation 2. Install FFmpeg - Extract to `C:\ffmpeg` and add to PATH 3. (Optional) Install NVIDIA CUDA Toolkit 12.1+

Install Trinity:

```
# Open PowerShell as Administrator

# Clone repository
git clone https://github.com/glasstonellc/trinity.git
cd trinity

# Create virtual environment
python -m venv venv
.\venv\Scripts\Activate.ps1

# Install Python dependencies
pip install -r requirements.txt

# For GPU support, install CUDA-enabled PyTorch
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu121

# Download model weights
python scripts\download_models.py
```

Run Trinity on Windows:

```
# CPU mode
python trinity_cli.py --input audio.wav --output clean.wav

# GPU mode (if CUDA available)
python trinity_cli.py --input audio.wav --output clean.wav --device cuda
```

Windows-Specific Notes: - Long path support may need to be enabled in Windows Registry - Windows Defender may flag model downloads - add exclusion for Trinity directory - GPU acceleration requires compatible NVIDIA drivers (version 525+)

11.4 Kubernetes Deployment

Example Deployment Manifest:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: voxis-trinity
spec:
  replicas: 3
  selector:
    matchLabels:
      app: voxis
  template:
    metadata:
      labels:
        app: voxis
    spec:
      containers:
      - name: voxis
        image: glasstonellc/voxis:v5.1.0a
        resources:
          limits:
            nvidia.com/gpu: 1
            memory: "12Gi"
            cpu: "4"
          requests:
            memory: "8Gi"
            cpu: "2"
        env:
        - name: VOXIS_GPU_ENABLED
          value: "true"
        ports:
        - containerPort: 5001
```

11.3 Cloud Provider Specific Guides

AWS EC2: 1. Launch p3.2xlarge instance with Deep Learning AMI 2. Install Docker and NVIDIA Container Toolkit 3. Pull and run Voxis container 4. Configure security groups (port 5001, HTTPS 443)

Google Cloud Platform: 1. Create Compute Engine VM with NVIDIA T4 GPU 2. Use Container-Optimized OS 3. Deploy via Cloud Run or GKE 4. Configure Cloud Load Balancing

Azure: 1. Create NC-series VM with NVIDIA GPU 2. Install Docker and dependencies 3. Deploy container 4. Configure Application Gateway for SSL termination

12. Troubleshooting

12.1 Common Issues

Issue: GPU not detected despite being available

Solution:

```
# Verify NVIDIA drivers
nvidia-smi

# Check Docker GPU access
docker run --rm --gpus all nvidia/cuda:12.1.0-base nvidia-smi

# Ensure NVIDIA Container Toolkit is installed
sudo apt-get install -y nvidia-container-toolkit
sudo systemctl restart docker
```

Issue: Out of memory errors during processing

Solution: - Reduce batch size in configuration - Process shorter audio chunks - Disable AudioSR super-resolution for memory-constrained systems - Upgrade to instance with more VRAM

```
# Reduce memory usage
pipeline = TrinityPipeline(
    device="cuda",
    enable_super_res=False,  # Disable memory-intensive SR
    chunk_duration=15  # Process 15-second chunks instead of 30
)
```

Issue: Processing is slower than expected

Diagnostic Steps:

```
# Check GPU utilization
nvidia-smi -l 1

# Monitor system resources
htop

# Check for CPU throttling
cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_cur_freq

# Verify I/O performance
iotop
```

Solutions: - Ensure GPU is actually being used (check logs for "Using device: cuda") - Verify no other processes are consuming GPU memory - Use SSD storage for input/output files - Enable GPU persistence mode: `sudo nvidia-smi -pm 1`

Issue: Poor audio quality in output

Diagnostics:

```
# Enable verbose logging
import logging
logging.basicConfig(level=logging.DEBUG)

# Inspect intermediate outputs
pipeline = TrinityPipeline(save_intermediates=True)
result = pipeline.process("input.wav", "output.wav")

# Check saved intermediate files:
# - input_analyzed.json (spectrum data)
# - input_denoised.wav (after denoising)
# - input_isolated.wav (after isolation)
# - input_restored.wav (final output)
```

Common Causes: - Input audio is too degraded (extreme noise levels) - Sample rate mismatch causing artifacts - Restoration strength set too high (try 50-60 instead of 75-85) - Source contains multiple overlapping speakers (not supported)

12.2 Logging and Debugging

Enable Debug Mode:

```
from trinity import TrinityPipeline
import logging

logging.basicConfig(
    level=logging.DEBUG,
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s'
)

pipeline = TrinityPipeline(verbose=True)
```

Log Locations: - Application logs: `/var/log/voxis/app.log` - Processing logs: `/var/log/voxis/processing.log` - Error logs: `/var/log/voxis/error.log`

Log Rotation: Logs are automatically rotated daily and compressed after 7 days.

12.3 Performance Profiling

Enable Profiling:

```
from trinity import TrinityPipeline

pipeline = TrinityPipeline(enable_profiling=True)
result = pipeline.process("input.wav", "output.wav")

# Print profiling report
print(result.profiling_report)
```

Sample Profiling Output:

```
Stage               | Time (s) | % Total
--------------------|----------|--------
Spectrum Analysis   |     2.1  |    8%
ML Denoising        |    12.4  |   48%
Vocal Isolation     |     8.2  |   32%
Restoration         |     2.8  |   11%
Output Generation   |     0.3  |    1%
--------------------|----------|--------
Total               |    25.8  |  100%
```

---

13. Future Roadmap

13.1 Planned Features (Q1-Q2 2026)

Enhanced Models: - Trinity v5.2: Improved denoiser with transformer-based architecture - Support for 48kHz and 96kHz processing without resampling - Real-time streaming mode with <200ms latency

New Capabilities: - Multi-speaker diarization and separation - Accent normalization (experimental) - Automatic speech-to-text integration - Emotional tone preservation metrics

Performance Improvements: - 2x faster processing through model quantization (INT8) - Reduced memory footprint (target: 5GB VRAM) - Apple Silicon (M1/M2) optimization via Core ML

13.2 Research Directions

Active Research Areas: 1. **Few-shot speaker adaptation**: Personalized enhancement based on 30-60 seconds of reference audio 2. **Zero-shot noise removal**: Generalization to unseen noise types without retraining 3. **Perceptually-guided restoration**: Loss functions based on human auditory models 4. **Energy-efficient inference**: Optimizations for edge deployment on mobile devices

Collaboration Opportunities: Glass Stone LLC welcomes research partnerships with academic institutions and industry partners. Contact research@glassstone.com for collaboration inquiries.

---

14. References

14.1 Technical Papers

1. Schröter, H., et al. (2021). "DeepFilterNet: A Low Complexity Speech Enhancement Framework for Full-Band Audio." *arXiv preprint arXiv:2110.05588*.

2. Défossez, A., et al. (2021). "Hybrid Spectrogram and Waveform Source Separation." *Proc. of the ISMIR Conference*.

3. Liu, H., et al. (2023). "AudioSR: Versatile Audio Super-resolution at Scale." *arXiv preprint arXiv:2309.07314*.

4. Jansson, A., et al. (2017). "Singing Voice Separation with Deep U-Net Convolutional Networks." *ISMIR*.

14.2 External Resources

Hugging Face Model Hub: - Trinity Denoiser: https://huggingface.co/glassstone/trinity-denoiser-v5 - Demucs v4: https://huggingface.co/facebook/htdemucs - AudioSR: https://huggingface.co/haoheliu/audiosr-speech-v1

Documentation: - Voxis User Guide: https://docs.voxis.glasstone.com - API Documentation: https://api.voxis.glasstone.com/docs - Trinity Python Package: https://pypi.org/project/trinity-audio/

Community: - GitHub Repository: https://github.com/glasstonellc/trinity - Discord Server: https://discord.gg/glassstone - Support Forum: https://community.glassstone.com

14.3 Datasets

Training Data: - DNS Challenge 2020: https://github.com/microsoft/DNS-Challenge - LibriSpeech: http://www.openslr.org/12/ - MUSDB18: https://sigsep.github.io/datasets/musdb.html - Custom Glass Stone dataset (proprietary)

14.4 Open Source Acknowledgments

Trinity builds upon the following open-source projects: - PyTorch (https://pytorch.org) - BSD License - TorchAudio (https://pytorch.org/audio) - BSD License - NumPy (https://numpy.org) - BSD License - SciPy (https://scipy.org) - BSD License - Librosa (https://librosa.org) - ISC License

Appendix A: Glossary

Deep Learning: Machine learning techniques using neural networks with multiple layers.

Spectrogram: Visual representation of the spectrum of frequencies in an audio signal as they vary with time.

STFT (Short-Time Fourier Transform): Technique to determine the frequency content of local sections of a signal over time.

SNR (Signal-to-Noise Ratio): Measure comparing the level of desired signal to background noise.

PESQ (Perceptual Evaluation of Speech Quality): ITU-T standard for objective speech quality assessment.

STOI (Short-Time Objective Intelligibility): Metric for predicting speech intelligibility.

U-Net: Convolutional neural network architecture originally developed for image segmentation, adapted for audio processing.

Formant: Resonant frequency of the vocal tract, crucial for vowel identification.

Stem: Individual audio component extracted from a mixed audio signal.

GPU (Graphics Processing Unit): Specialized processor designed for parallel computation, essential for deep learning.

Appendix B: Version History

Version	Release Date	Key Changes
5.0.0	June 2025	Initial production release
5.0.1	July 2025	Bug fixes, performance improvements
5.0.2	August 2025	Enhanced error handling
5.1.0a	November 2025	Current version, improved denoiser, expanded dataset

---

Appendix C: License and Legal

Permitted Uses: - Evaluation and testing with express written permission - Production use under valid Voxis subscription or enterprise license - Academic research with signed research agreement

Contact: - **General Inquiries**: info@glassstone.com - **Research Division**: research@glassstone.com - **Sales**: sales@glassstone.com - **Support**: support@voxis.glasstone.com - **Legal**: legal@glassstone.com

Patent Pending: Certain techniques used in Trinity are the subject of pending patent applications.

Document Control

Field	Value
Document ID	TRINITY-TECH-DOC-5.1.0a-ALPHA
Revision	1.1
Author	Gabriel B. Rodriguez, CEO
Organization	Glass Stone LLC / Glass Stone Research
Reviewed By	Glass Stone Engineering Team
Approved By	Gabriel B. Rodriguez, CEO
Created	November 5, 2025
Last Updated	January 11, 2026
Status	Alpha Documentation
Next Review	April 11, 2026

END OF TECHNICAL DOCUMENTATION