Project Report: Bearing Health Detection using Deep Learning and LLM

📊 Title:

Robust Bearing Health Classification using Spectrogram-Based CNN and LLM-Enhanced Knowledge Reasoning

---

💡 Abstract

This project presents a comprehensive pipeline for **automatic bearing health classification** from vibration signals using deep learning. The approach converts raw vibration signals into spectrograms and trains a CNN to classify health conditions (Normal, IR, OR, Ball). To improve performance and robustness, adaptive **signal augmentations** were used: RMS-scaled Gaussian noise, amplitude scaling, and frequency masking. Further, a **retrieval-augmented generation (RAG) system with an LLM** was integrated to support real-time fault classification using vector-based reasoning. The final system offers high classification accuracy with scalable prediction support suitable for industrial deployment.

📚 Dataset and Preprocessing

- **Source**: CWRU (Case Western Reserve University) bearing dataset
- **Format**: `.mat` files, each containing raw vibration signals
- **Sampling Rate**: 12,000 Hz
- **Segmentation**: 5-second windows
- **Spectrogram Conversion**:
    - Shape: 128×128 or 256×256
    - Function: `scipy.signal.spectrogram`

- Spectrograms are computed using `nperseg=256`, `noverlap=128` and converted to log scale using `10 * log10(Sxx + 1e-8)`

🔄 Augmentation Techniques

1. RMS-Scaled Gaussian Noise

```
def add_rms_scaled_noise(signal, noise_ratio=0.2):
    rms = np.sqrt(np.mean(signal**2))
    noise = np.random.normal(0, noise_ratio * rms, size=signal.shape)
    return signal + noise
```

- Adds realistic noise relative to signal energy
- Prevents under/over-noising and supports generalization
- Ensures weak signals aren't overwhelmed and strong signals are realistically disturbed

2. Amplitude Scaling

```
def amplitude_scaling(signal, scale_range=(0.5, 1.5)):
    scale = np.random.uniform(*scale_range)
    return signal * scale
```

- Simulates signal variation due to physical load changes
- Creates synthetic samples with similar structure but varied intensities

3. Variable Frequency Masking

```
def frequency_masking(spec, num_masks=1, freq_masking_max_percentage=0.15):
    spec = spec.copy()
    num_freqs = spec.shape[0]
    for _ in range(num_masks):
        f = int(np.random.uniform(0.1, freq_masking_max_percentage) * num_freqs)
        f0 = np.random.randint(0, num_freqs - f)
        spec[f0:f0 + f, :] = 0
    return spec
```

- Randomly zeros out horizontal frequency bands
- Mimics missing sensor range or interference
- Forces CNN to rely on broader spectral features instead of fixed bands

---

🧠 CNN Architecture

- **Input**: Spectrogram (128x128x1)
- **Layers**:
    - Conv2D (32, 64, 128) + BatchNorm + MaxPooling
    - Flatten + Dense (128) + Dropout (0.3)
    - Dense(4), softmax output
- **Loss**: Sparse Categorical Crossentropy
- **Optimizer**: Adam (lr = 0.0005)
- **EarlyStopping**: Patience = 5 to 15 depending on setup
- Designed to balance performance and trainability with relatively shallow depth

---

🧠 LLM + Knowledge Base Integration

To enable fast and scalable prediction support, a **RAG (Retrieval-Augmented Generation)**-based architecture was implemented. This allows an external LLM (LLaMA 3) to reference embeddings generated from labeled training signals and support **natural language queries** about classification predictions.

- **LLM**: Groq-hosted LLaMA 3 70B (used for inference only)
- **Embeddings**: Jina embeddings generated from descriptions of:
    - Fault types (IR, OR, Ball)
    - Signal segments and their associated class

- **Vector Store**: Indexed using FAISS
- **LLM Behavior**:
    - Retrieves most similar training case(s) based on the input test signal embedding
    - Predicts the fault class using semantic similarity
    - Outputs the predicted class directly

> ⚠️ The LLM is not used for explanation or insights, but strictly for **prediction support using vector-based semantic retrieval**.

This provides a flexible alternative to CNN predictions, particularly useful for edge deployments or cases where model predictions must be double-checked with known historical examples.

🔬 Experimental Comparison

Experiment	Noise Type	Masking Type	Spectrogram Size	Epochs	Accuracy
EXP-1	RMS (0.2x)	Variable	128x128	39	**0.9814**
EXP-2	Fixed (0.08)	Fixed (100px)	128x128	60	0.9224
EXP-3	RMS (0.1x)	Variable	256x256	65	0.9400

> Best performance came from **EXP-1** due to adaptive augmentation and efficient 128x128 spectrograms.

⚠️ Design Trade-offs and Insights

❌ Drawbacks of Using 256×256 Spectrograms

- **Increased Memory & Time**: Training takes longer, consumes more GPU.
- **Overfitting Risk**: More pixels, same data → model may memorize noise or background.
- **Diminishing Returns**: Accuracy does not improve significantly beyond 128×128.

- **Inference Cost**: Slower predictions in real-time scenarios.

❌ Problems with Standard Gaussian Noise (Fixed σ)

```
noisy = signal + np.random.normal(0, 0.08, size=signal.shape)
```

- Adds the same noise level regardless of signal strength.
- **Weak signals** become overpowered, causing false positives.
- **Strong signals** barely change, reducing augmentation effectiveness.
- Inconsistent behavior across samples → unstable learning signal.

✅ How RMS-Scaled Noise Solves It

```
def add_rms_scaled_noise(signal, noise_ratio=0.2):
    rms = np.sqrt(np.mean(signal**2))
    noise = np.random.normal(0, noise_ratio * rms, size=signal.shape)
    return signal + noise
```

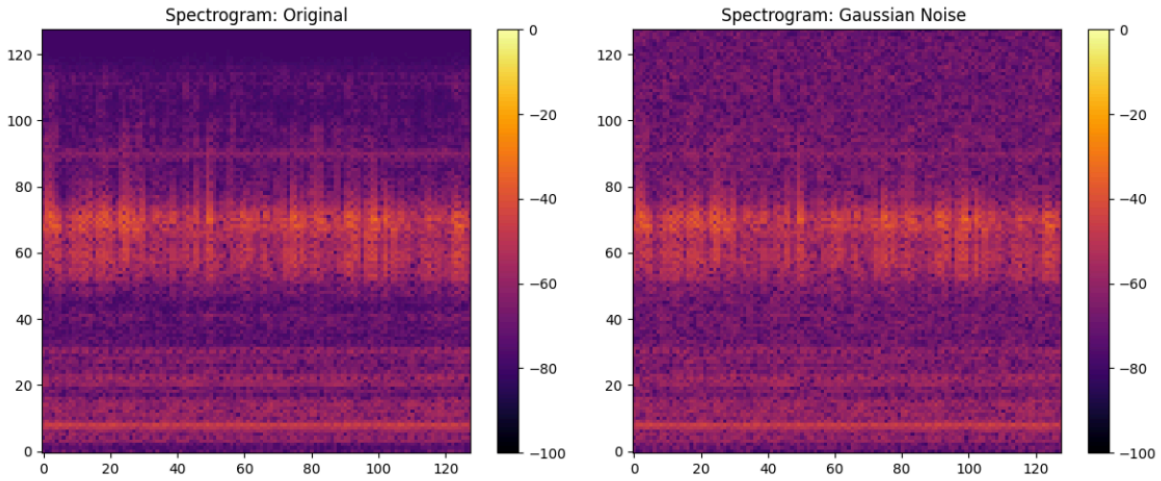
- **Signal-aware**: More noise for high-RMS signals; less for low-RMS
- **Realistic**: Matches real-world vibration sensor noise behavior
- **Balanced augmentation**: Every sample is augmented to the right extent
- **Robust model behavior**: Improves generalization on noisy/real-world data

---

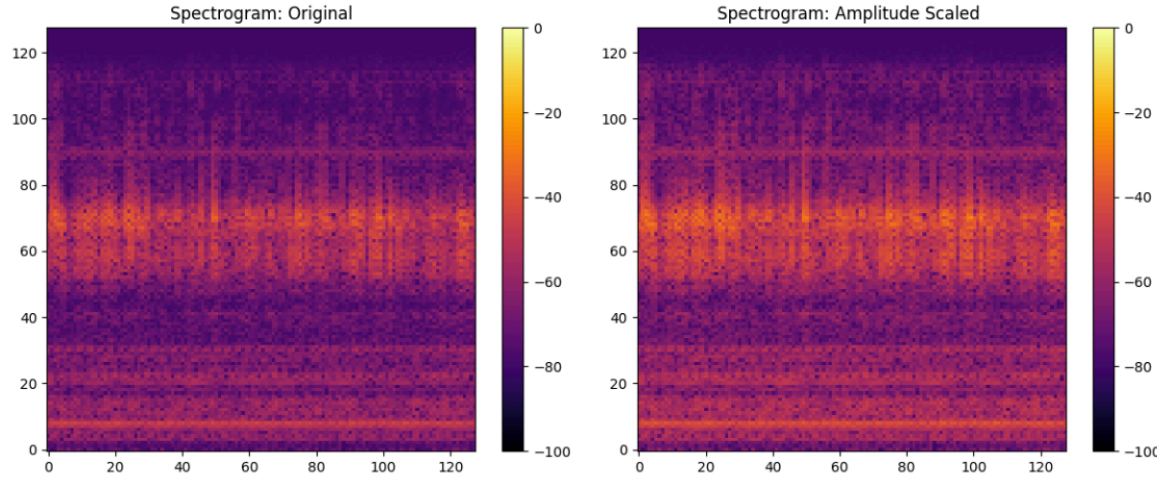
📊 Visual Results

Spectrogram Comparison

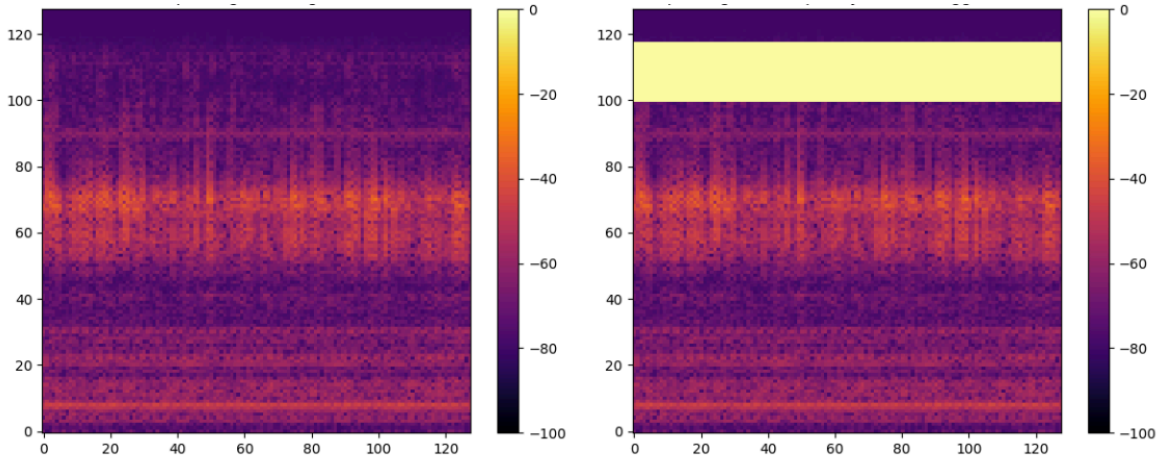
- Original vs Gaussian Noise



- Original vs Amplitude Scaling

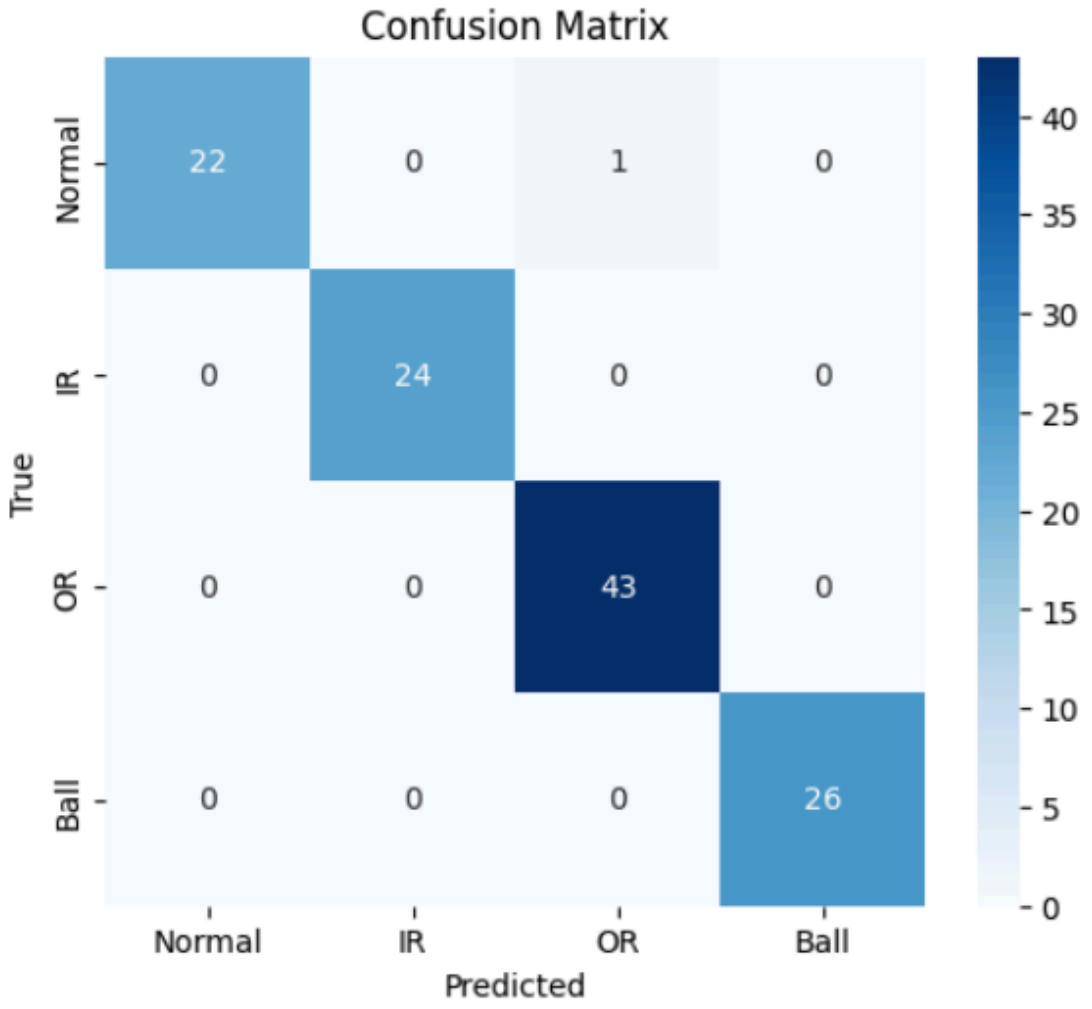


- Original vs Frequency Masking

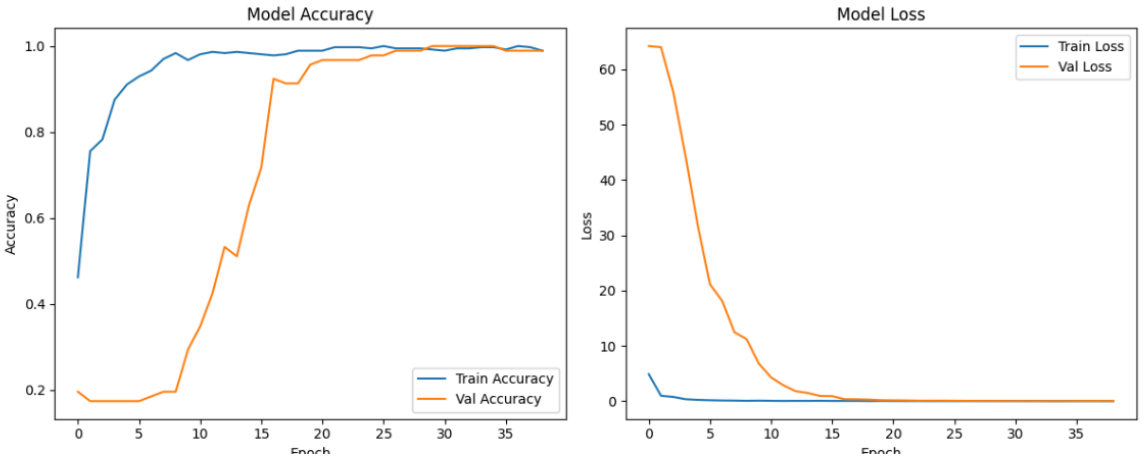


CNN Performance

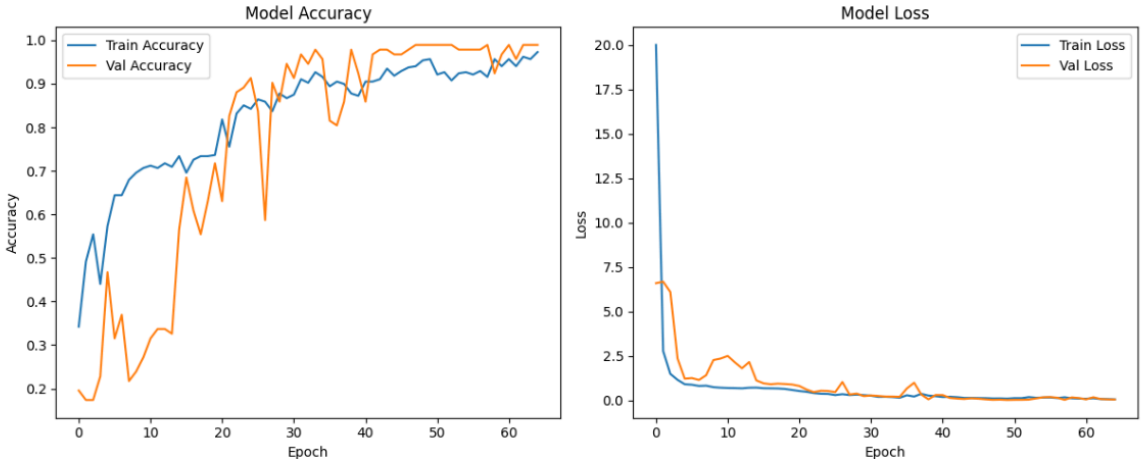
- Confusion Matrix (EXP-1)



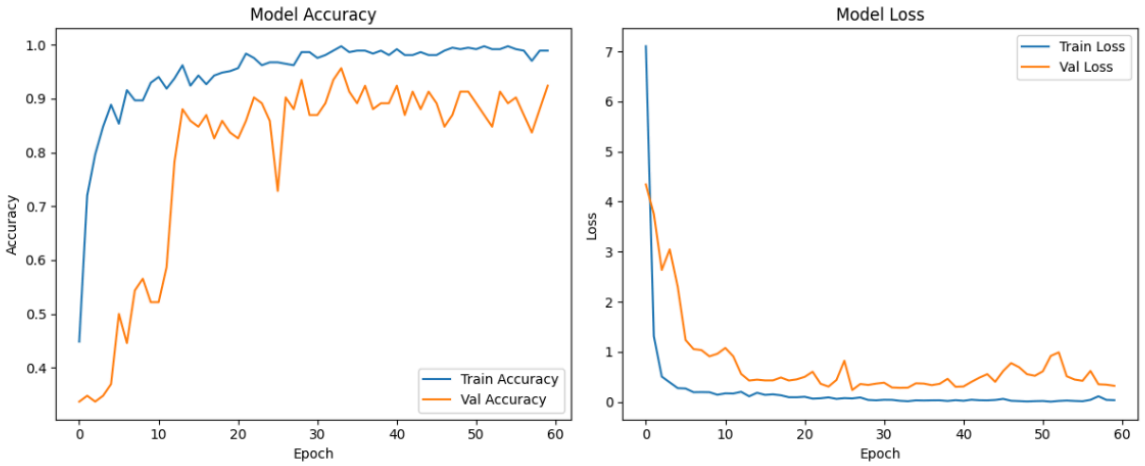
- Training Accuracy & Loss Curves

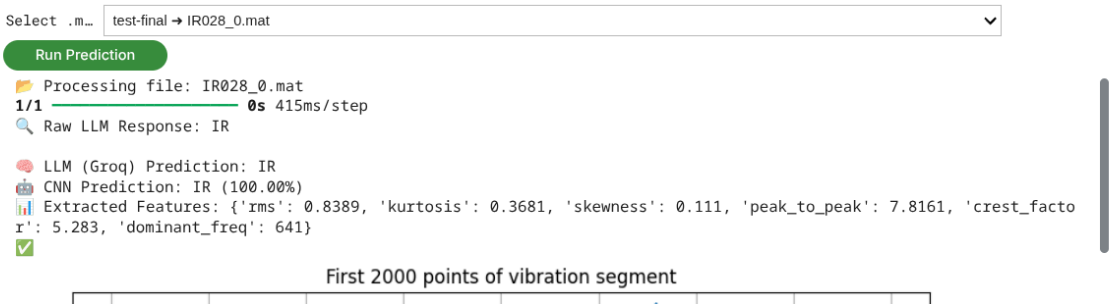


Using spectogram size as 256*256



Using fixed masking





🧠 Insights

- **RMS-based augmentations** improved robustness

- **Variable frequency masking** simulated sensor unpredictability
- **128x128 spectrograms** balanced speed and resolution
- **LLM integration** enabled fault prediction via semantic retrieval of similar cases, improving interpretability of classification under noisy conditions

---

📝 Conclusion

This project demonstrates that combining **signal-aware data augmentation** with **spectrogram-based CNNs** and a **lightweight LLM-based retrieval system** yields a highly effective and deployable solution for real-time bearing health classification.

The best model (EXP-1) achieved **98.14% accuracy** with only 39 epochs, outperforming other configurations with lower compute cost. The LLM system supported auxiliary predictions using embedded vector matching, making this framework not only accurate but also scalable across real-world industrial settings.

-