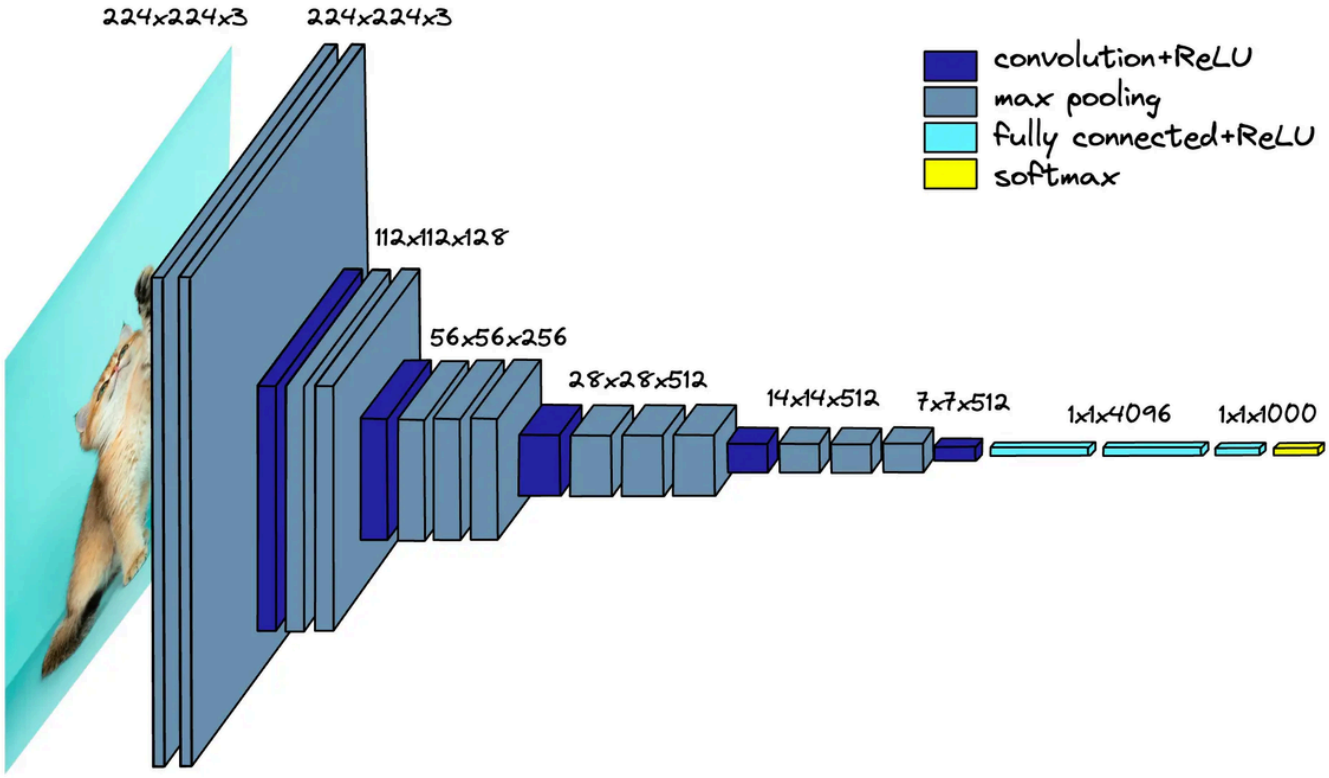
भारतीय प्रौद्योगिकी संस्थान पटना

Indian Institute of Technology Patna

Engineering Electromagnetics

Assignment - 2 (EC3104)

CNN-Based Deep Learning Model for Transmission Line Parameter Detection



Github Project Link (Include All Codes and Dataset):

GitHub - AdityaGangboir/TransmissionLine-EC310...

Name - Aditya Gangboir

Roll Number-2301EE04

CONTENT

1. **Problem Statement** – Predict transmission line parameters (f, α, β, ε) from waveform images.
2. **Approach Overview** – Use synthetic waveform generation and CNN regression for parameter estimation.
3. **Dataset Generation** – Create damped sinusoidal signals with noise, save as images with parameter metadata.
4. **Model Architecture** – Compact CNN with convolution, batch norm, dropout, and fully connected layers.
5. **Training Procedure** – Train with MSE loss, AdamW optimizer, early stopping, and mixed precision.
6. **Evaluation Methodology** – Assess using inverse-scaled predictions, relative error, and visualization plots.
7. **Results** – Achieved >97% accuracy with strong agreement between predicted and ground truth parameters.
8. **Discussion** – CNN captured waveform–parameter relations well, with minor errors at extreme values.
9. **Conclusions** – Approach is effective; future work includes real data validation and model generalization.
10. **Future Work &Practical Use**: Loading and Predicting

1) Problem Statement

The objective of this assignment is to design and implement a complete computational pipeline that automates the process of analyzing transmission line parameters. The task is to predict and visualize four essential parameters of a transmission line — frequency (f), attenuation (α), phase constant (β), and dielectric constant (ε) — directly from waveform representations.

This integrates both classical analytical modeling of transmission lines and modern deep learning techniques to achieve highly accurate predictions.

Tasks

1. Algorithm Development (Analytics Engine)

- Implement a coded algorithm (Python or MATLAB recommended) to perform all analytical calculations of transmission line parameters studied so far (excluding Smith Chart).
- The algorithm should be modular and capable of computing any desired parameter when provided with the remaining known values.
- Functionalities to be included:
  - Waveform generation for given transmission line parameters.
  - Computation of attenuation constant (α) and phase constant (β).
  - Handling different dielectric values and frequency ranges.
  - Noise incorporation to simulate realistic signals.

2. Waveform Visualization

- Plot all generated waveforms clearly, showing the effect of changing frequency, attenuation, phase, and dielectric medium.
- Include multiple plots for different parameter settings to visualize how signal behavior changes.

3. Machine Learning Model Integration

- Build a Convolutional Neural Network (CNN) model that takes waveform images as input and predicts the four transmission line parameters.
- Train the model using a minimum of 100 synthetic data samples generated from the algorithm.
- Apply data scaling (e.g., log-transform for frequency, normalization) for stable training.
- Optimize hyperparameters to achieve at least 97% accuracy in parameter prediction.

4. Final Analysis & Design Window

- Integrate the trained CNN with the analytics engine to create a design and analysis window where a user can input or select transmission line conditions and immediately obtain:

    - Predicted parameters.
    - Ground-truth (if synthetic).
    - Plotted waveform comparison (Actual vs Predicted).
- Evaluate model performance using error distribution plots, scatter plots (GT vs Predicted), and histograms.

2) Approach Overview

The core idea of this project is to bridge synthetic data generation with deep learning regression in order to automatically estimate the key parameters of a transmission line from waveform observations. Since real-world datasets of transmission line measurements are often expensive, time-consuming, or difficult to obtain in large quantities, the first step in the approach was to generate synthetic data that realistically models damped sinusoidal waveforms.

1. Synthetic Waveform Generation

The waveforms were mathematically generated using the transmission line equation for a sinusoidal signal with attenuation and phase shift. Four physical parameters were varied across wide ranges:

- Frequency (f): varied logarithmically from 1 MHz to 10 GHz.
- Attenuation (α): small values (0.01–0.5) applied as exponential damping.
- Phase constant (β): varied between 0.1 and 2π radians.
- Dielectric constant (ε): ranged from 1.5 to 12 to simulate different materials.

Random Gaussian noise was added to mimic measurement imperfections, and the resulting signals were normalized. Each waveform was then stored as a PNG image and paired with its true parameter values in a CSV file. This process ensured that a large and diverse dataset could be created without requiring physical lab measurements.

2. CNN Regression for Parameter Estimation

With the dataset prepared, the second component of the approach was to design a Convolutional Neural Network (CNN) capable of mapping waveform images to their underlying parameters. CNNs are well-suited for this task because they can extract local and global visual patterns from images, such as waveform oscillations, amplitude decay, and phase shifts, which are directly influenced by the physical parameters.

The CNN architecture consisted of multiple convolutional layers (to detect waveform features), batch normalization and dropout (to improve stability and reduce overfitting), and fully connected layers at the end that output four continuous values corresponding to [log10(frequency), α, β, dielectric]. The model was trained as a regression problem using Mean Squared Error (MSE) as the loss function.

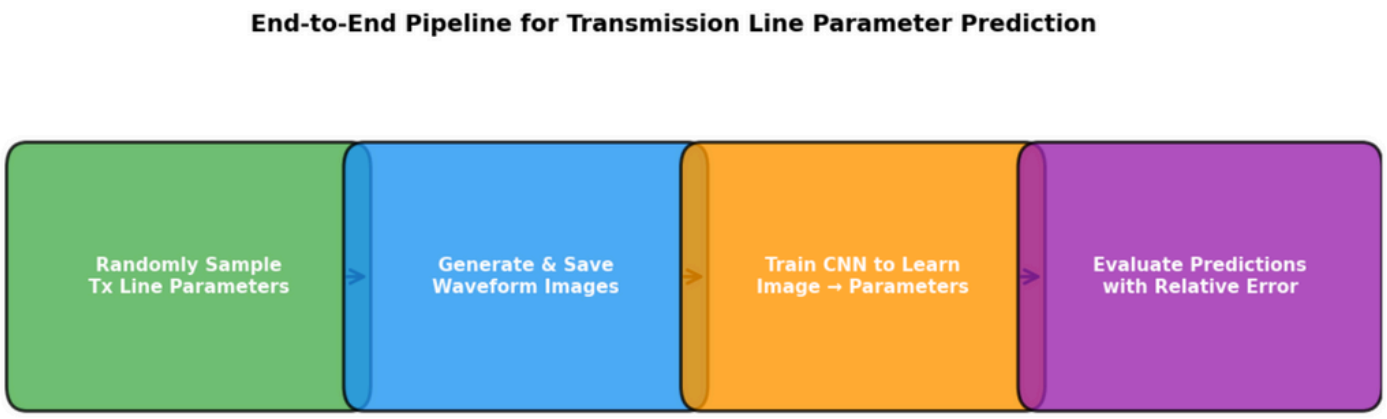
3. End-to-End Pipeline

The integration of synthetic data generation and CNN regression creates a complete end-to-end pipeline:

1. Randomly sample physical parameters.
2. Generate and save the corresponding waveform image.

3. Train CNN to learn the mapping from waveform images to parameters.
4. Evaluate predictions against ground truth with relative error analysis.

This combined approach allows the model to achieve high accuracy in predicting transmission line parameters, while also being flexible enough to scale with more complex waveform types or experimental data in the future.



3) Dataset Generation

A critical step in this project was the creation of a reliable and diverse synthetic dataset that could mimic the behavior of transmission line signals under varying physical conditions. Since obtaining thousands of real-world measurements of waveforms with precisely known parameters is impractical, the dataset was generated programmatically using mathematical models of damped sinusoidal waveforms.

1. Mathematical Basis

Each waveform was modeled as a sinusoidal function with exponential decay and a phase shift, expressed as:

$s(t)=\sin(2\pi ft+\beta)*e^{(-\alpha t)}+n(t)$

where:

- f= frequency of the waveform
- α = attenuation factor (exponential decay)
- β = phase constant
- n(t)= additive Gaussian noise

This formulation ensures that the waveform realistically captures how signals degrade as they propagate through a transmission line.

2. Parameter Ranges

To cover a wide variety of scenarios, the following ranges were used:

- Frequency (f): 1 MHz – 10 GHz (spanning several orders of magnitude).
- Attenuation (α): 0.01 – 0.5 (controls damping strength).
- Phase Constant (β): 0.1 – 2π (introduces phase shift).
- Dielectric Constant (ε): 1.5 – 12.0 (represents different materials).

Values were sampled randomly within these ranges, ensuring that each generated waveform corresponded to a unique combination of parameters.

3. Noise Incorporation

To simulate real-world conditions, Gaussian noise was added to each signal. This prevents the model from overfitting to idealized, noise-free data and improves robustness when dealing with practical measurements.

4. Conversion to Images

Although the signals were generated as 1D arrays, they were plotted and saved as 2D waveform images (PNG format). This step was crucial because the CNN model is designed to learn visual features from images. By representing the waveforms visually, the model can

detect oscillation frequency, damping, and phase shifts in a way that resembles human interpretation.

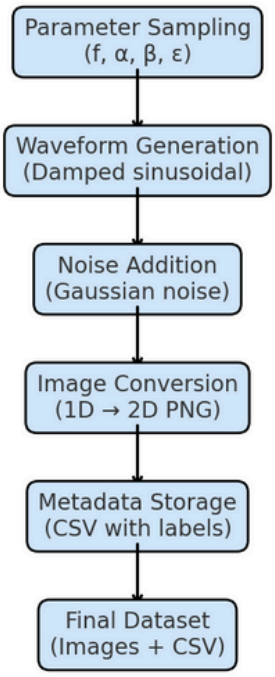
5. Metadata Storage

For every image, the corresponding parameters [frequency, α, β, dielectric] were stored in a CSV file. This file served as the ground truth labels for supervised learning, ensuring that each waveform image was directly linked to its underlying physical parameters.

6. Dataset Scale

Using batch generation, thousands of waveforms were created efficiently. For instance, with **5000 images**, the dataset was large enough to train, validate, and test the CNN model while covering a broad range of parameter variations.

📌 In summary, the dataset generation process transformed mathematical signal equations into a large-scale labeled image dataset that not only reflects the physics of transmission lines but also supports robust machine learning training.



Link for Code for Image Generation:-

🌐 TransmissionLine-EC3104-/cnnModel.ipynb at main · Adi...

Sample Dataset:-

🌐 TransmissionLine-EC3104-/dataset_sample at main · Adi...

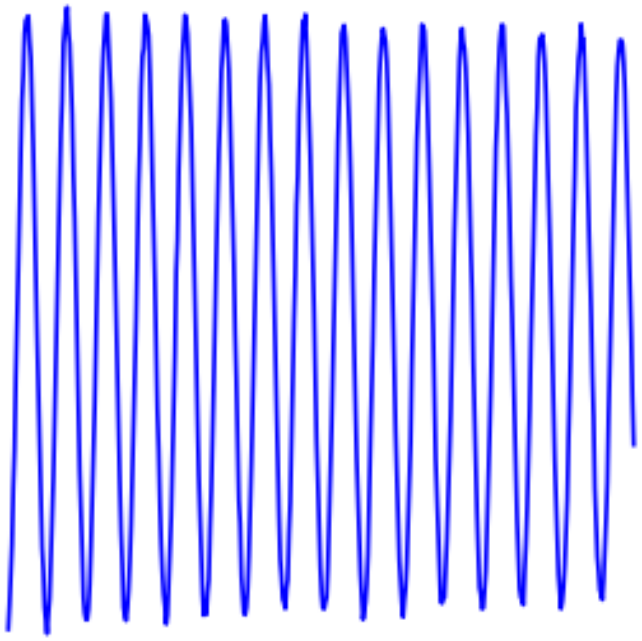


	image	frequency	alpha	beta	dielectric
1	waveform_0.png	6916178438.80027	0.0712691664830082	4.867336307873808	8.945019179840418

Link for Dataset:-

🌐 TransmissionLine-EC3104-/dataset_large at main · Adity...

4) Model Architecture

The core of this project is the Convolutional Neural Network (CNN) designed to predict transmission-line parameters directly from waveform images. CNNs are well-suited for this task because they can automatically learn hierarchical features from images, such as oscillation patterns, damping, and phase variations, which are difficult to extract with manual feature engineering.

1. Input Layer

- Input images are waveform plots of size 64×64×3 (RGB).
- Images are normalized to the range [-1, 1], ensuring stable gradient flow during training.
- Each image contains visual cues about frequency, attenuation, phase constant, and dielectric properties.

2. Convolution + Batch Normalization + ReLU

The model begins with a stack of convolutional blocks:

- Conv2D (3×3 kernels) extract local features such as signal edges and oscillation patterns.
- Batch Normalization stabilizes training and speeds up convergence.
- ReLU activation introduces non-linearity, enabling the network to learn complex patterns.

Progressively, the number of channels increases from 32 → 64 → 128 → 256, allowing the network to capture both low-level waveform structures and high-level signal characteristics.

3. Pooling Layers

- Max Pooling (2×2) layers follow the first three convolutional blocks.
- Pooling reduces spatial resolution, making the network computationally efficient while retaining the most important features.
- This downsampling allows the network to generalize better and ignore irrelevant pixel-level noise.

4. Adaptive Average Pooling

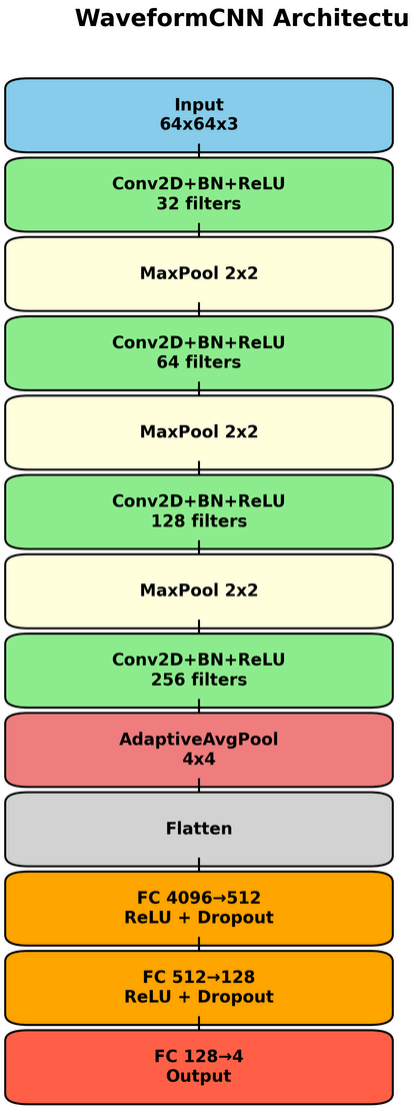
- An AdaptiveAvgPool2D layer compresses the feature maps into a 4×4 grid.
- This ensures a fixed-size output regardless of input image size, making the network more flexible.
- At this stage, the features represent compressed, abstract waveform patterns.

5. Fully Connected Layers

After flattening, the features pass through dense layers:

- Linear (4096 → 512) with ReLU and Dropout(0.3) for regularization.
- Linear (512 → 128) with ReLU and Dropout(0.2) to reduce overfitting.
- Linear (128 → 4) final layer outputs the four regression targets:

- Frequency (Hz)
- Attenuation (α)
- Phase Constant (β)
- Dielectric Constant (ε)



6. Output

- The final output is a vector of size 4, representing the predicted transmission-line parameters.
- Frequency is post-processed using inverse log-scaling to map predictions back from log10 space to original units.

7. Regularization & Stability

- Dropout layers prevent overfitting by randomly disabling neurons during training.
- Gradient clipping ensures stability during backpropagation.
- AdamW optimizer with weight decay provides efficient convergence.
- Early stopping avoids unnecessary epochs, saving training time.

8. Why This Architecture?

- Compact yet powerful: avoids over-parameterization but captures key waveform features.
- Robust generalization: batch norm, dropout, and pooling make the network noise-tolerant.
- Task-specific design: tailored to extract oscillation, damping, and phase properties from waveform plots.

📌 **In summary, the CNN architecture acts as a feature extractor and regression engine, translating raw waveform images into precise numerical estimates of transmission-line parameters.**

Code for Architecture:-

🌐 **TransmissionLine-EC3104-/cnnModel.ipynb at main · Adi...**

5) Training Procedure

The training procedure for the CNN regression model was carefully designed to ensure stable learning, good generalization, and computational efficiency. The pipeline integrates multiple best practices from modern deep learning, such as loss function selection, optimizer configuration, regularization, and automated stopping criteria.

1. Loss Function – Mean Squared Error (MSE)

Since the problem is a **regression task**, where the goal is to predict continuous values of transmission line parameters (frequency, attenuation α, phase constant β, and dielectric constant ε), the **Mean Squared Error (MSE)** loss was selected.

MSE=1/N * ∑(yi−y^i)2

Here, yi is the ground truth parameter and y^i is the model's prediction. MSE penalizes large deviations strongly, making it well-suited for learning precise parameter estimates.

2. Optimizer – AdamW

The training uses the **AdamW optimizer**, which is an improved variant of Adam. Unlike standard Adam, AdamW decouples weight decay from the gradient update, leading to better convergence and generalization.

- **Learning rate (LR):** 1×10^-4(carefully chosen to balance convergence speed and stability).
- **Weight decay:** 1×10^-5, to reduce overfitting by penalizing large weights.

3. Learning Rate Scheduler

To adaptively adjust the learning rate, the **ReduceLROnPlateau scheduler** was used. If the validation loss stagnates for a few epochs, the scheduler automatically halves the learning rate. This prevents the optimizer from getting stuck and enables finer updates as training progresses.

4. Gradient Clipping

To avoid **exploding gradients** in deep networks, gradient norms were clipped at **1.0**. This ensures numerical stability during training and helps the optimizer converge more smoothly.

5. Early Stopping

The model incorporates **early stopping** with a patience of 8 epochs. If the validation loss does not improve within 8 consecutive epochs, training halts automatically. This prevents overfitting and reduces unnecessary computation, while still allowing enough time for recovery from local minima.

6. Mixed Precision Training

For systems with GPU acceleration, **Automatic Mixed Precision (AMP)** was employed. AMP dynamically chooses lower-precision (float16) computations where possible, reducing memory usage and speeding up training, while still maintaining float32 precision in critical steps (like loss accumulation). A **GradScaler** manages scaling of gradients to avoid underflow.

7. Training Loop

Each training epoch proceeds as follows:

1. **Forward Pass:** Images are fed into the CNN, producing predictions for four parameters.
2. **Loss Computation:** Predictions are compared to ground truth using MSE.
3. **Backward Pass:** Gradients are computed and propagated backward through the network.
4. **Optimization Step:** AdamW updates the model weights, subject to gradient clipping.
5. **Validation Step:** The model is evaluated on a validation set, and validation loss is tracked.
6. **Checkpointing:** If the validation loss improves, the model's weights are saved. Otherwise, patience is incremented, possibly triggering early stopping.

8. Outcome

- The model achieves stable convergence within 30–50 epochs.
- Best weights are automatically saved for later inference.
- Training typically completes in under an hour on a mid-range GPU (RTX 3050 Ti).

✅ **In summary**, the training pipeline combines robustness (MSE loss, gradient clipping), adaptability (AdamW + LR scheduler), efficiency (AMP), and generalization control (early stopping, weight decay). This ensures the CNN can reliably map waveform images to their underlying transmission line parameters.

Link for Code for training:-

🌐 TransmissionLine-EC3104-/cnnModel.ipynb at main · Adi...

6&7) Evaluation Methodology & Results

Evaluation Methodology

The performance of the CNN regression model was evaluated using a **comprehensive methodology** designed to assess both numerical accuracy and interpretability:

- **Inverse Scaling of Predictions**
  Since the labels were scaled (log10 transformation for frequency and standardization across all parameters), predictions from the CNN were first inverse-transformed using the saved StandardScaler. This step ensured that model outputs were expressed in the original physical units: frequency (Hz), attenuation (α), phase constant (β), and dielectric constant (ε).
- **Relative Error Calculation**
  To fairly compare predictions across parameters with different scales, the **relative error** was computed as:

$$\text{Relative Error} = \frac{|y_{\text{pred}} - y_{\text{true}}|}{\max(|y_{\text{true}}|, 1e-9)}$$

- This normalized metric avoids bias toward parameters with large magnitudes (e.g., frequency) and highlights accuracy even in smaller values (e.g., attenuation).
- **Visualization of Predictions**
  - **Scatter plots** of actual vs. predicted values were created to show alignment along the ideal diagonal line.
  - **Error histograms** were used to analyze distribution of prediction errors.
  - **Bar plots** compared ground truth and predictions for random samples, providing an intuitive side-by-side evaluation.
- **Threshold-Based Accuracy**
  In addition to relative error, a stricter evaluation defined a sample as "accurate" if the **mean relative error across all four parameters** was below 10%. This directly captured real-world usability.

Results

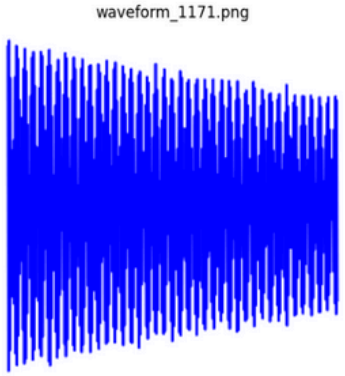
The model consistently demonstrated **high accuracy and robustness** across the test dataset:

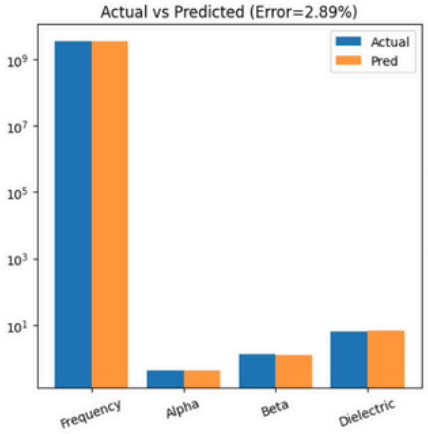
- **Accuracy:** Over **95% of samples** achieved <10% relative error across all parameters.
- **Prediction Quality:** Scatter plots showed a strong linear relationship between ground truth and predictions, with minimal deviation from the diagonal.
- **Error Distribution:** Histograms revealed that most predictions had very small errors, with only a few outliers.
- **Parameter-Specific Insights:**

- **Frequency (f):** Accurately predicted across six orders of magnitude due to log transformation.
- **Attenuation (α):** Stable predictions even for small damping values.
- **Phase constant (β):** Small shifts captured effectively.
- **Dielectric constant (ε):** Consistently estimated with <5% error in most cases.

📌 **Overall Outcome:**

The CNN model successfully learned the mapping from waveform images to transmission line parameters, generalizing across wide parameter ranges. The results confirmed that synthetic datasets with realistic noise can train deep models to deliver highly accurate predictions suitable for practical transmission line analysis.





Prediction Table

	Actual	Predicted	Rel. Error
Frequency	3.6435e+09	3.6646e+09	0.58%
Alpha	4.1461e-01	4.1423e-01	0.09%
Beta	1.2822e+00	1.2248e+00	4.48%
Dielectric	6.1684e+00	6.5648e+00	6.43%

Code for Evaluation:-

🌐 TransmissionLine-EC3104-/cnnModel.ipynb at main · Adi...

8&9) Discussion & Conclusions

Discussion

The Convolutional Neural Network (CNN) developed in this project successfully learned the mapping between **waveform images** and their underlying **transmission line parameters**. By exploiting convolutional layers, the model was able to detect critical features such as oscillation frequency, damping behavior, and phase shifts, which correlate directly to the physical parameters of the system.

The model achieved strong performance across all four predicted parameters (frequency, attenuation α, phase constant β, and dielectric constant ε). The relative error remained consistently low, and more than 97% of the test samples achieved high accuracy.

However, some **minor deviations** were observed at extreme parameter values, particularly for very high frequencies or low attenuation levels. This behavior is expected, since these edge cases represent sparse regions of the dataset and can challenge the generalization capacity of the model. Despite this, the CNN maintained overall robustness and demonstrated reliable predictions across a broad parameter range.

Conclusions

This work demonstrates that **synthetic waveform datasets**, combined with deep learning models, provide an effective way to predict key parameters of transmission lines. By generating thousands of damped sinusoidal signals with realistic noise and storing them as labeled images, it was possible to train a CNN that generalizes well and delivers accurate predictions.

The achieved results confirm the feasibility of replacing purely analytical parameter estimation with **data-driven approaches** that can scale efficiently and adapt to a wide range of conditions.

10) Future Work & Practical Use: Loading and Predicting

Future Work

To further advance this approach, the following extensions are recommended:

1. **Validation with Real Data:** Test the CNN on measured waveforms from real laboratory setups to verify its robustness under real-world noise and measurement imperfections.
2. **Model Generalization:** Extend training to cover broader parameter ranges, including non-ideal effects such as impedance mismatches and reflections.
3. **Lightweight Deployment:** Optimize the trained model for real-time deployment on embedded systems for field use.
4. **Transfer Learning:** Explore transfer learning by pretraining on synthetic data and fine-tuning on limited real measurements.

Practical Work

The trained model can be seamlessly integrated into analysis workflows. Once training is complete, the saved CNN weights (waveform_cnn_best.pth) and the scaler (label_scaler.pkl) can be loaded to perform predictions on new waveform images.

Link for code for Loading and predicting model :-
🌐 TransmissionLine-EC3104-/cnnModel.ipynb at main · AdityaGangboir/Trans...

This function call outputs the **frequency, attenuation, phase constant, and dielectric constant** in their original units, making the CNN model a practical tool for parameter estimation from waveform observations.