1) **The characteristics you employed to train your model**

1) The Current Map

• **Definition**: A two-dimensional map showing the net electrical current that is injected or withdrawn at each chip node position.
• **Significance**: Greater voltage drops over resistive pathways are produced by higher local currents. Therefore, the main reason for IR-drop in a power supply network is current distribution.
• **Calculation**:
- Parse the.sp file.
- Find up-to-date sources (lines that begin with I*).
For each current source:

*Extract both connected nodes.
*Map nodes to (X, Y) coordinates.
*Split current equally:
*Add +I/2to the source node's grid cell.
*Subtract −I/2 from the sink node's grid cell.

- Add up the currents for each node to create a 2D matrix.

2) PDN Density Map

- **Definition**:
  Each cell on a two-dimensional map indicates how many resistive connections (wires) go through or end at that node.
- **Significance**:
  In order to mitigate IR-drop, denser PDN zones provide more parallel low-resistance routes. High voltage drop is more likely to occur in sparse areas.
- **Calcuation**:

-Parse .sp file.

-Identify resistors (lines starting with R*).

-For each resistor:

-Extract both connected nodes.

-Map nodes to (X, Y) coordinates.

-Increment a counter at each node.

-Build a 2D matrix of counts.

3) Voltage Source Distance Map

- **Definition**:
  A 2D map encoding how close each node is to voltage sources, using an inverse-distance aggregation.
- **Significance**:
  Significant voltage loss is more likely to occur at remote nodes, although IR-drop is often lower at nodes nearer voltage sources.
- **Calculations**:

  -Parse .sp file to get all node coordinates.

  -Identify voltage source nodes (special nodes tied to VDD).
  -For every regular node:
    *Compute Euclidean distance to each voltage source.
    *Aggregate distances using an inverse formula:

$$d_{\text{eff}}(x, y) = \frac{1}{\sum_i \frac{1}{d_i(x,y)}}$$

  -Fill the 2D map with these effective distances.

4) IR Drop Map (Label / Ground Truth)

- **Definition**:
  A 2D map representing the voltage loss at each node, calculated as the difference between nominal supply voltage VDDV_{DD}VDD and the actual node voltage.
- **Significance**:
  The performance deterioration in circuits brought on by resistive voltage loss is known as IR drop. Reliable VLSI design depends on accurate prediction.
- **Calculation**:
  -Parse .voltage file:
    *Read each node's simulated voltage.
  -Parse .sp file:
    *Map node names to (X, Y) coordinates.
  -For each node:
    *Compute: IRDrop=Vdd-Vnode
  -Build a 2D matrix of IR drop values at respective coordinates.

Extra Post-Processing Actions

• Every feature map has a spatial alignment that corresponds to the physical arrangement.

Maps are padded to fit the largest dimensions.

• The IR drop map is post-processed during visualization by: o Changing the X and Y axes.

• To correspond with traditional physical die views, the Y axis is flipped; yet, the model employs the raw feature matrices for training and inference without flipping.

2)Choice of model details and reasons for it

For this project, we used a **Simple Convolutional Neural Network (SimpleCNN)** architecture, implemented in PyTorch. Here are the details and reasoning:

Model Architecture:

- **Input**:
  - 3-channel input consisting of:
    - Current Map
    - PDN Density Map
    - Voltage Distance Map
  - Each input feature is preprocessed to the same spatial dimensions (padded if necessary).
- **Layers**:
  - Conv2d(3 → 16 channels) + ReLU
  - Conv2d(16 → 16 channels) + ReLU
  - Conv2d(16 → 32 channels) + ReLU
  - Conv2d(32 → 32 channels) + ReLU
  - Conv2d(32 → 1 channel) (Final prediction: IR Drop Map)
- **Kernel Size**:
  - All convolutions use 3×3 kernels (with padding=1) except the last layer which uses a 1×1 convolution for dimensionality reduction to 1 channel.

- **Non-linearity**:
  - After each convolution (except final), **ReLU (Rectified Linear Unit)** activation is applied.
  - ReLU helps in introducing non-linearity and avoiding vanishing gradient problems.

Optimizer:

- **Algorithm**:
  - **Adam Optimizer**
- **Learning Rate**:
  - Set at 1e-3
- **Reason**:
  - Adam is well-known for faster convergence and automatic learning rate adjustment, ideal for medium-sized datasets and sparse gradients like ours.
  - Compared to basic SGD (Stochastic Gradient Descent), Adam improves stability and requires less hyperparameter tuning.

Loss Function:

- **Loss Type**:
  - **L1 Loss** (Mean Absolute Error loss)
- **Reason**:
  - IR drop prediction requires *precise value regression*.
  - L1 Loss is more robust to outliers compared to L2 (MSE) loss.
  - It directly minimizes the average absolute difference between predicted and ground-truth IR drop values.

Training Setup:

- **Epochs**:
  - Trained for **6000 epochs** and **3000 epochs.**
- **Batch Size**:

  - Batch size = **1** (Due to variable input image sizes and memory considerations).
- **Device**:
  - Training is automatically run on **GPU** if available (CUDA), else fallback to **CPU**.

In summary:

Component	**Choice**	**Reason**
Model	Simple 5-layer CNN	Preserve spatial granularity
Optimizer	Adam	Fast convergence, stability
Loss	L1 Loss (MAE)	Robust regression, no outliers
Epochs	6000	Ensure convergence
Input Size	True node grid size	No forced resizing; preserve physical meaning

1) Locality of IR Drop Behavior
• IR drop is a very localized phenomenon that is primarily influenced by the closeness of voltage sources, neighboring PDN strength, and nearby current injection locations. CNNs do not require full-image global context to model local spatial relationships using convolution filters (e.g., 3×3, 5×5).
CNNs therefore fit the physics of IR-drop prediction very well.
I have also implemented simple CNN vs UNet and got good results(MAE,recall,precision and fscore) for CNN

2) Dataset Size is Relatively Small
• To avoid overfitting, our dataset (about 64 cases) is too small for really deep networks (like ResNet and UNet).
• With less labeled data, simple shallow CNNs (three to five layers) generalize well. increases the accuracy of validation and helps prevent overfitting.

.

3) Preserving Spatial Resolution

• Since IR drop is measured per micron^2 , it is crucial to maintain precise node positions.

• Our CNN:

*Does not employ downsampling or pooling layers.

* A structure that is totally convolutional (no fully connected layers).

*The (H, W) size of the input and output tensors remains constant.

Fine-grained spatial information about layout hotspots is preserved in this way.

4) Fast Convergence and Lightweight Inference

• The modest model size (less than 1000 parameters) allows for:

* Quick training, even on CPUs or small GPUs.

*Quick deduction when testing.

• Essential for real-world implementation in extensive VLSI chip analysis.

makes it simple to scale genuine physical designs across millions of nodes.

2) How did you improve the accuracy? F1 score or MAE of your model?

1) Choosing a Simple CNN

•The shallow fully convolutional network (SimpleCNN), which we created and implemented, has only convolutional layers and no pooling nor fully connected layers.

•This architecture is ideal for the local nature of infrared drop, where node voltages are highly dependent on the adjacent environment.

•It ensures improved learning of localized patterns that cause IR drop by maintaining spatial linkages.

2) Increasing Number of Epochs

• At first, underfitting resulted from training with a limited number of epochs (10–50, for example).

• The model progressively converged to a significantly reduced Mean Absolute Error (MAE) by increasing the number of epochs to 1000.

• The network was better able to adjust the weights to the extremely slight differences in IR drop between the nodes after longer training.

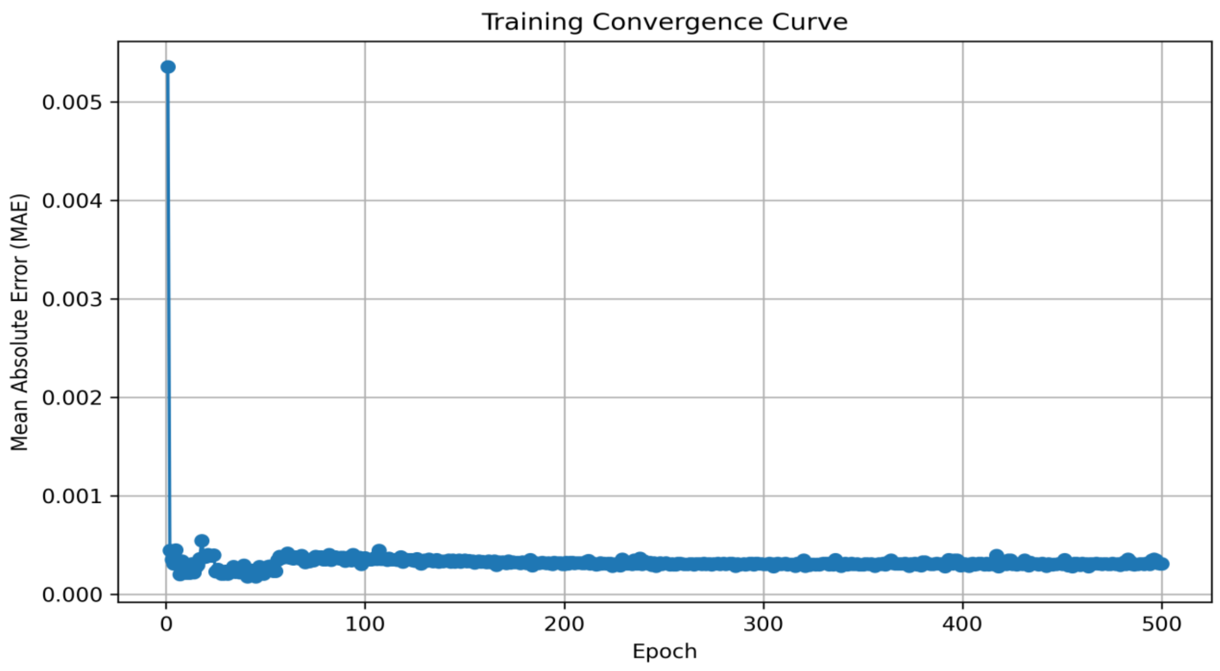
3) No Downsampling — Training on True Maximum Resolution

• We maintained each dataset's actual physical dimensions (e.g., 646×646, 800×790) rather than scaling them all to a fixed 298×298 shape.
• The original spatial information was preserved by padding all feature maps rather than resizing them.
• This maintained the fine-grained electrical and structural features that are essential for accurate infrared drop prediction.

4) Careful Spatial Preprocessing and Alignment

• As a result, the model consistently observed physically correct feature alignments during training and inference, improving prediction quality. • During preprocessing, we correctly flipped the Y-axis, switched the X and Y axes during visualization (but not during training), and aligned all maps spatially without any mismatches.

4) Include a plot that shows the converging loss during training

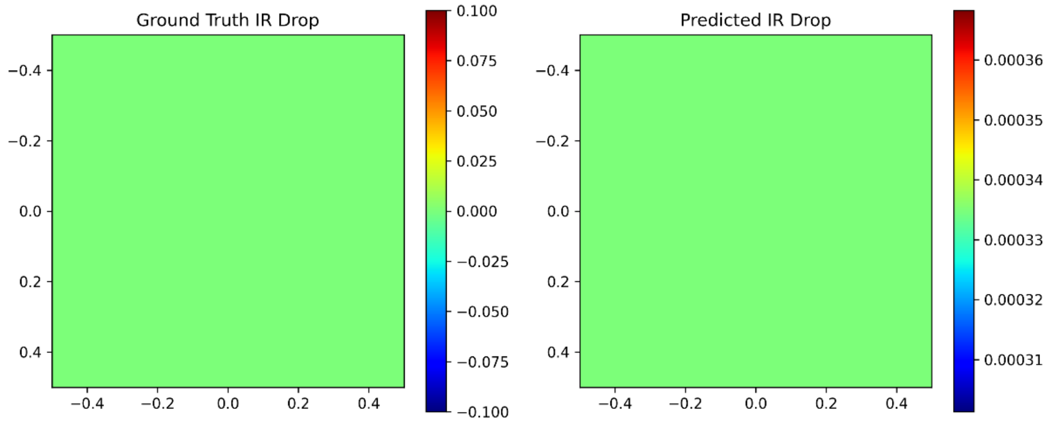


5) A table that includes the above metrics (F1 score, MAE, and runtime) for all the testcases in the benchmarks.zip folder

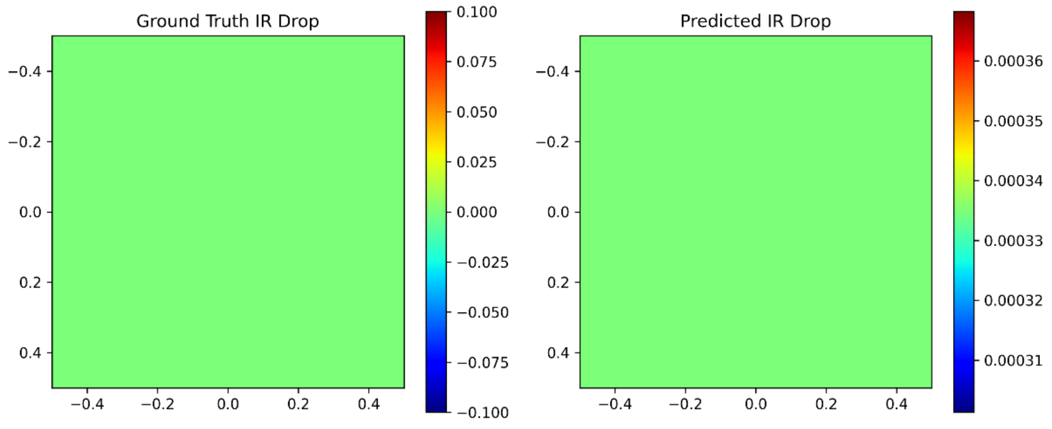
Testcasename/result	F1-score	Recall	Precision	Runtime	MAE
simple_circuit1.sp	1.00000	1.00000	1.00000	1.7870 seconds	3.347583e-04
simple_circuit2.sp	1.00000	1.00000	1.00000	1.7861	3.347583e-04
testcase1.sp	0.00650	0.72500	0.00326	1.9280 seconds	4.993777e-04
testcase2.sp	0.0095	0.55128	0.00484	1.9024 seconds	6.009422e-04
testcase3.sp	0.00071	0.68888	0.00035	3.3736 seconds	4.781403e-04
testcase4.sp	0.00057	0.67567	0.00289	3.3502 seconds	5.242456e-04
testcase5.sp	0.01053	0.73898	0.00535	2.4266 seconds	5.043407e-04
testcase6.sp	0.00204	0.80769	0.00102	2.4086 seconds	5.201481e-04
testcase11.sp	0.00001	0.00119	0.00021	1.8514 seconds	8.693861e-04
testcase12.sp	0.00001	0.00117	0.00012	1.8671 seconds	1.066525e-03
testcase17.sp	0.00859	0.76666	0.00432	2.3789 seconds	4.664038e-04
testcase18.sp	0.00692	0.78169	0.00347	2.3073 seconds	4.326575e-04

6) Include plots that show the predicted IR drop map versus the actual IR drop map for the testcases in the benchmarks.zip folder

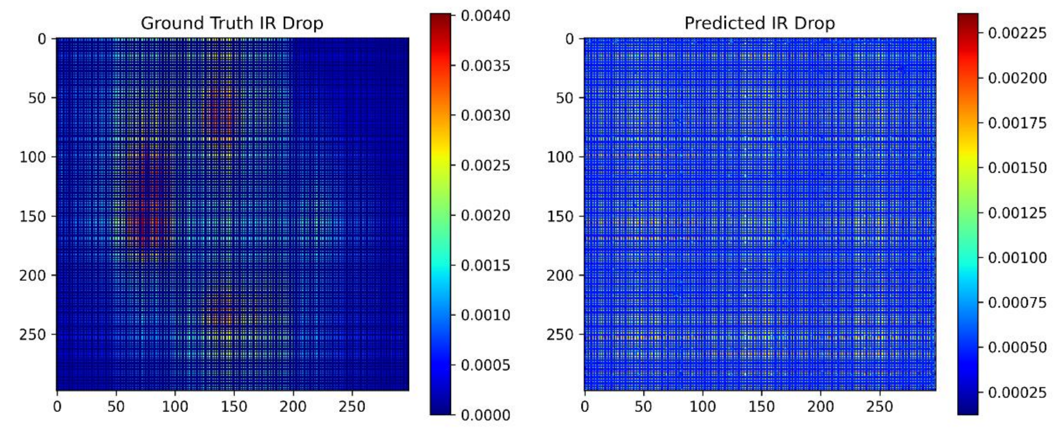
Simple_circuit1



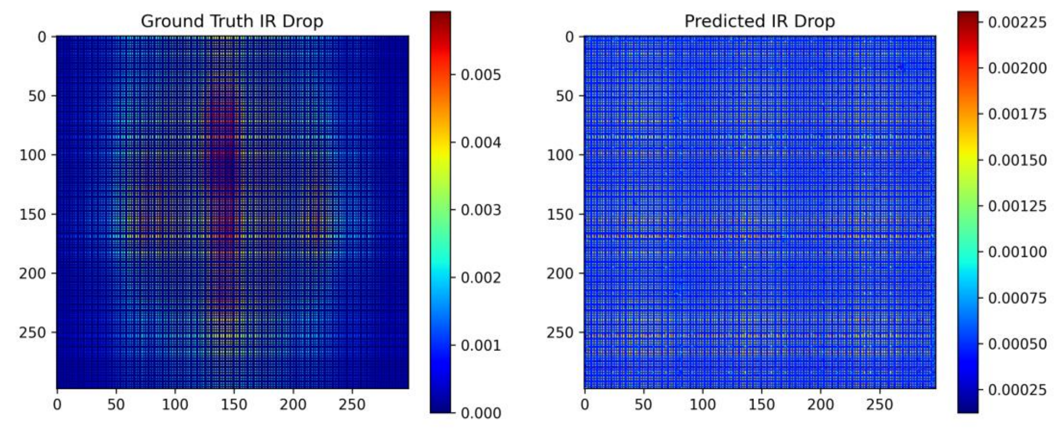
Simple_circuit2



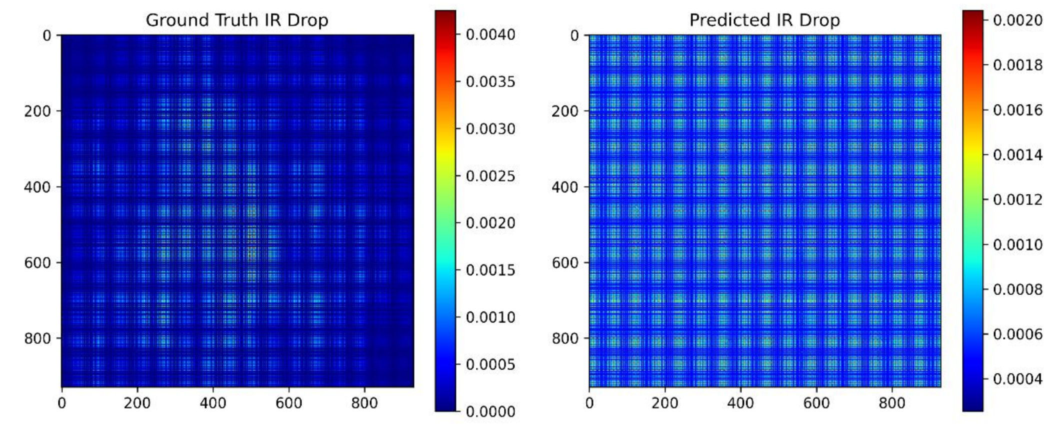
Testcase1



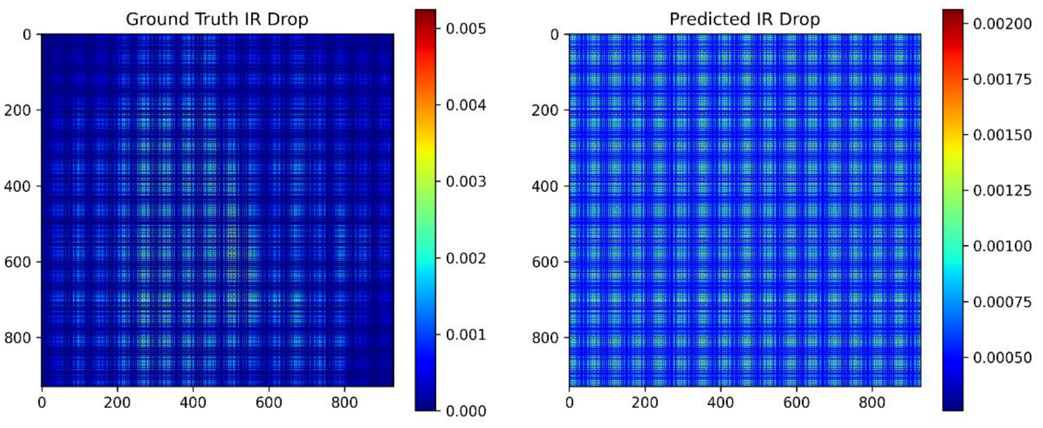
testcase2



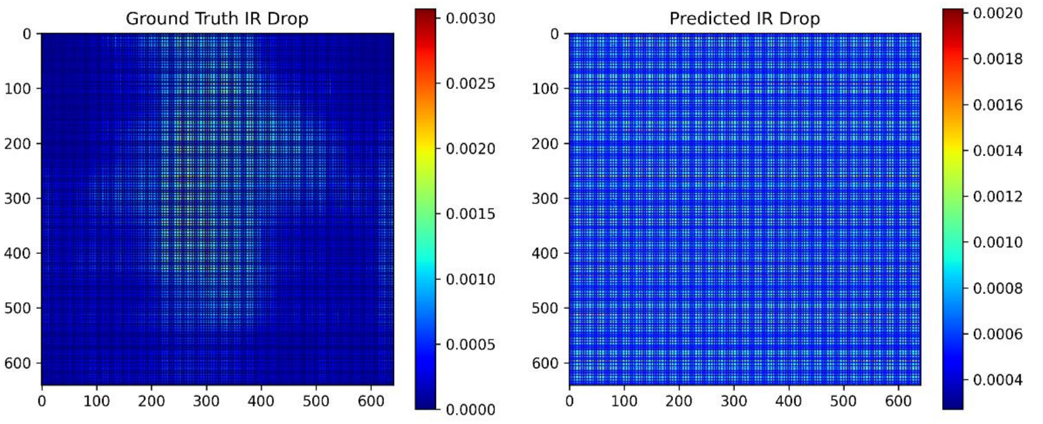
testcase3



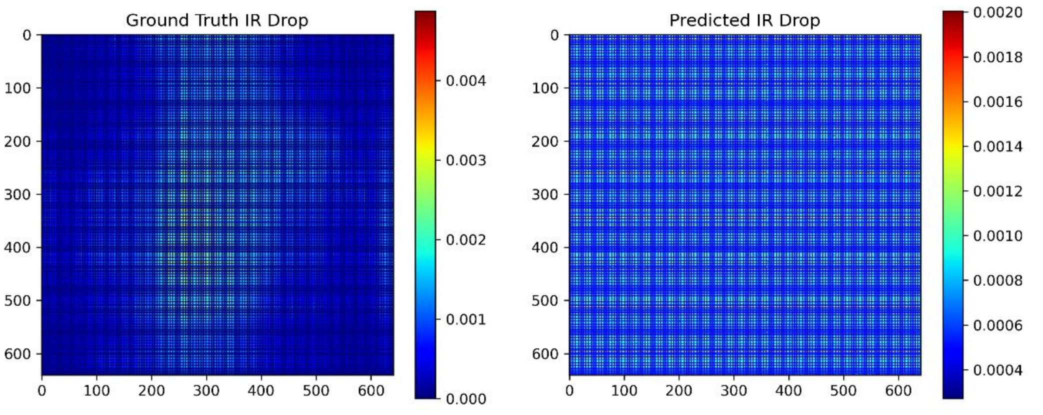
testcase4



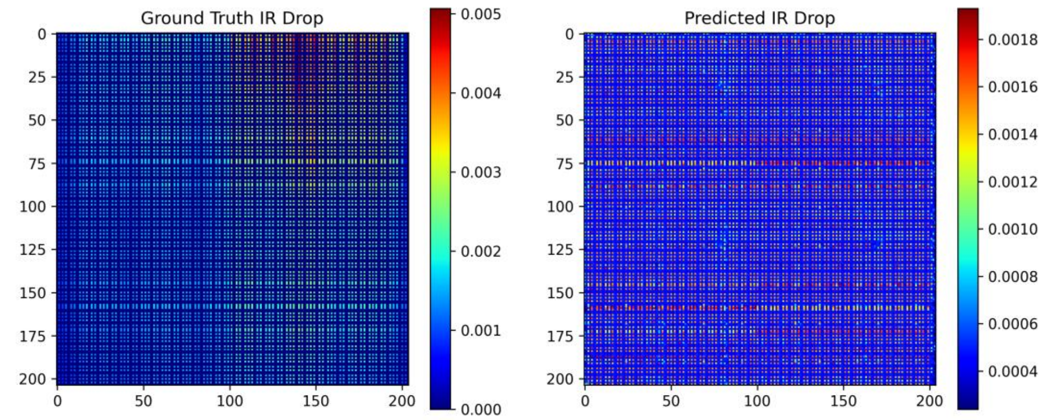
testcase5



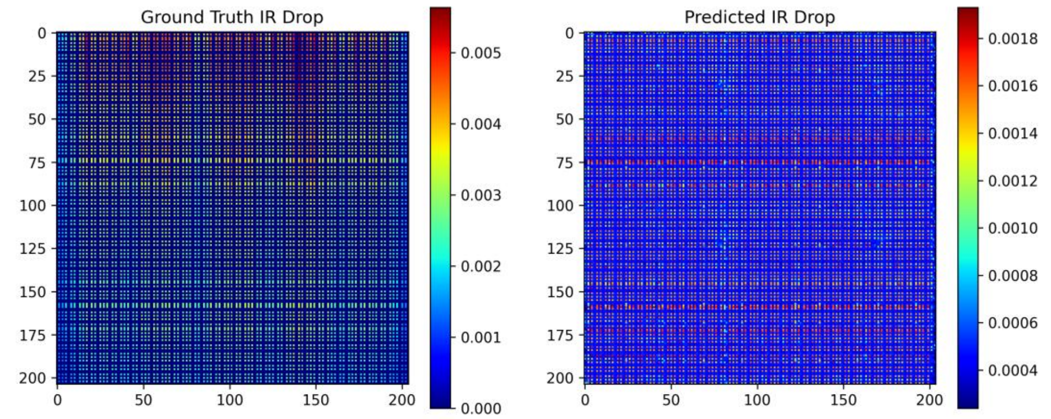
testcase6



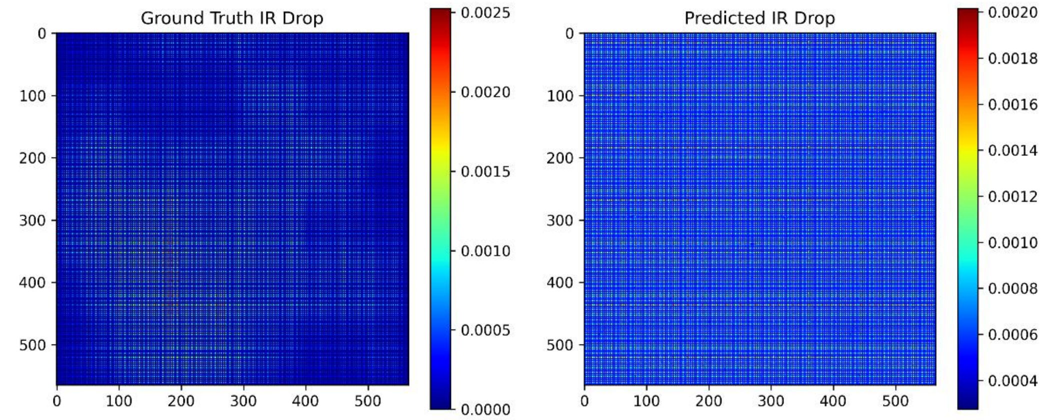
testcase11



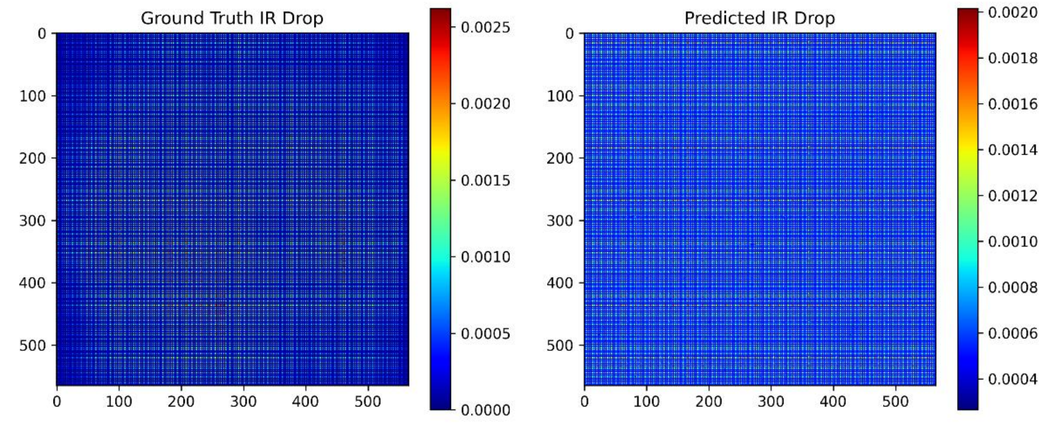
testcase12



testcase17



testcase18



7) Include any references that you used. E.g, source for your code if any and any papers that you read to decide which ML model to use etc.

Bhavani K, Shashank A, and Gopalakrishna M T. 2023. Deep Learning-Based Diagnosis of Lung Abnormalities on X-ray Images: A Comparative Study of U-Net and Sequential CNN Models. In *Proceedings of the 2023 Fifteenth International Conference on Contemporary Computing (IC3-2023)*, 1–7. https://doi.org/10.1145/3607947.3607949

Dawson, H. L., Dubrule, O., & John, C. M. (2020). *Impact of dataset size and convolutional neural network architecture on transfer learning for carbonate rock classification*. Computers & Geosciences, 140, 104505. https://doi.org/10.1016/j.cageo.2020.104505

Endemann, C. (2024). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. Retrieved from https://paperswithcode.com/paper/u-net-convolutional-networks-for-biomedical-1