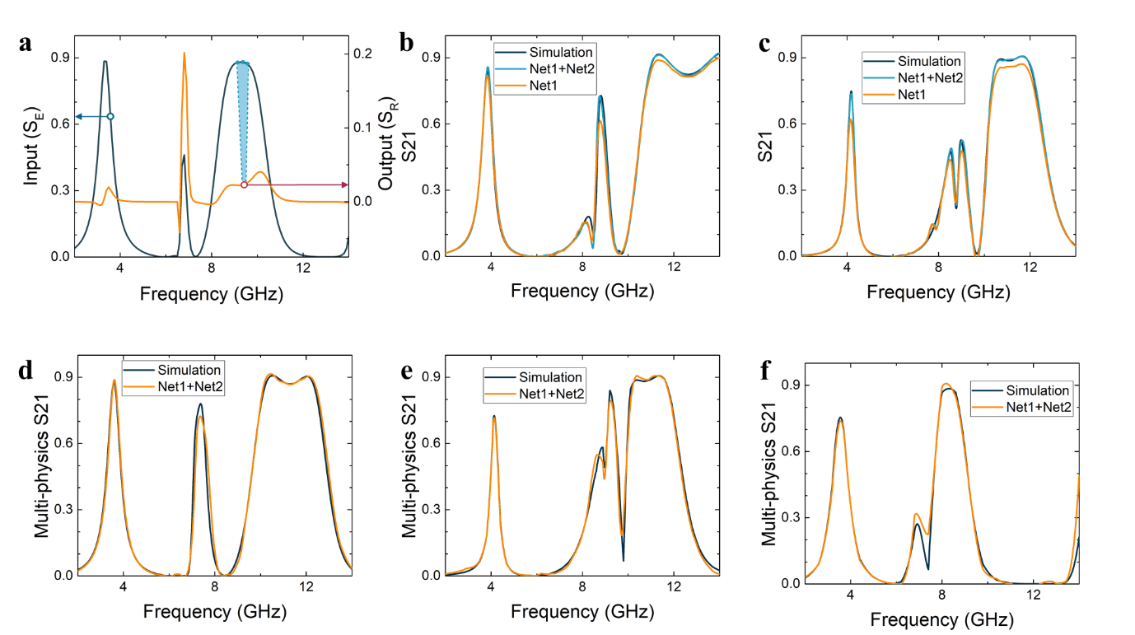
Multi-physical metasurface NN

**Configurations:**

**Suggested Environment**: Python 3.8.18, tensorflow 2.13.0, deepxde 1.10.2

**Dataset: All datasets for training and testing are contained in the subdirectory.**

**EM response by multi-fidelity DeepOnet:**



**There are 3 steps in total in this shared code：**

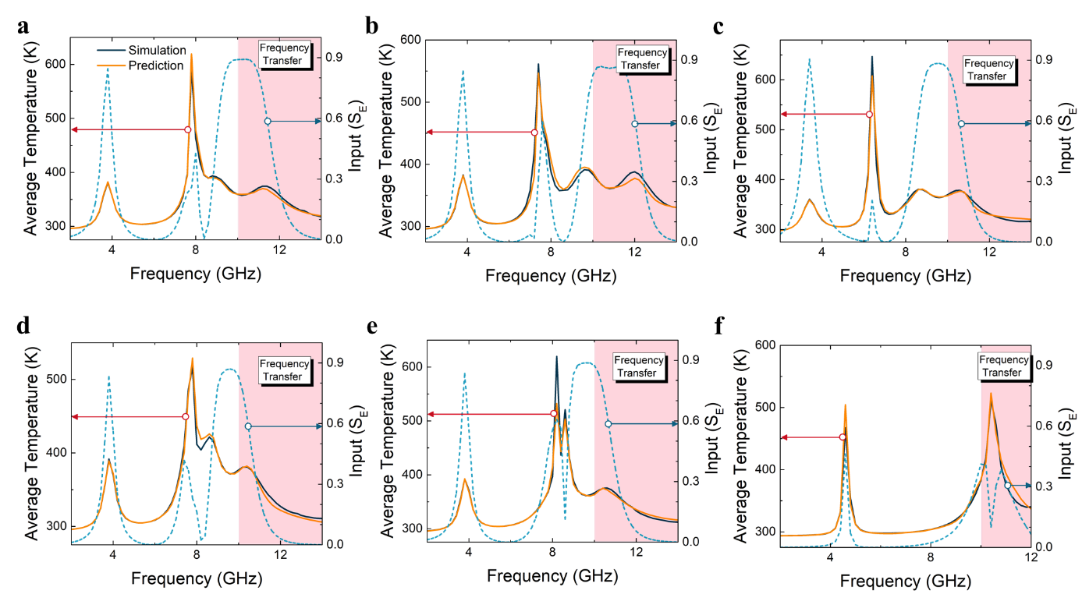
**Step 1. Train/test the EM predictor with parameterized\_strcuture/forward\_code /electro\_predictor.py.**

**Step 2. Train/test the Multiphysics predictor with** **parameterized\_strcuture /forward\_code/ electro\_thermal\_predictor.py.**

**Step 3. Validate the model in validation set with electro\_thermal\_predictor.py: Remember to replace the trained model name at Line 135 and set the variable *train* to be False at Line 127, the results are in variable *results* and the ground truth are in variable *y\_2\_test*.**

**The output is in results/EM\_predictor\_result.csv with a specific geometry whose parameter is defined in data/** **e-t-dataset/test. And the results in results/ EM\_test\_result.csv is the predicted results in validation set.**

**Average temperature by latent dynamics networks:**



**There are 2 steps in total in this shared code：**

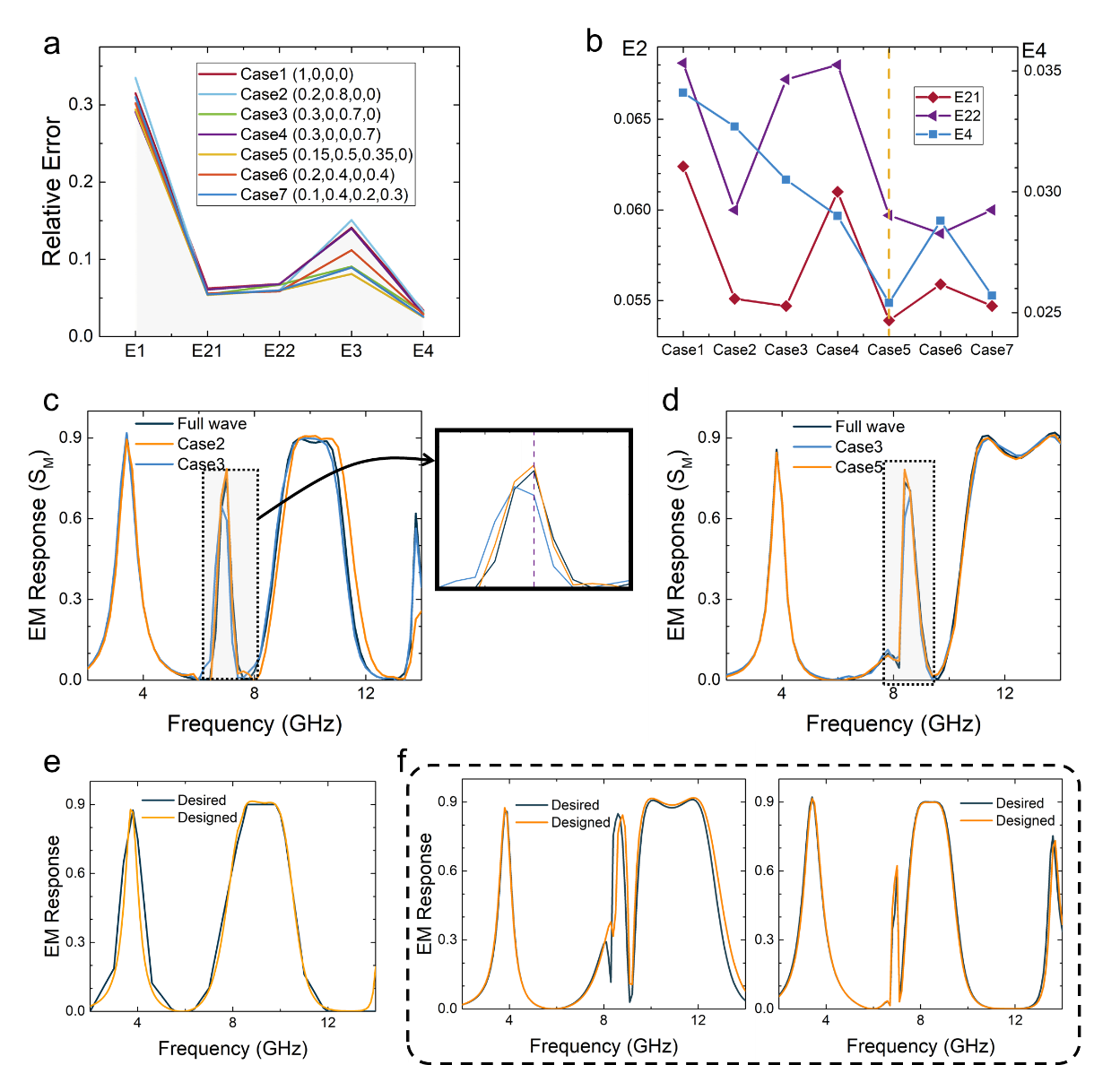
**Step 1. Train/test the Multiphysics predictor with**

**parameterized\_strcuture/forward\_code /temperature\_predictor.py.**

**Step 2. Validate the model with parameterized\_strcuture/ forward\_code/temperature\_predictor.py: Remember to replace the trained model name at Line 100 and set the variable *train* to be False at Line 112, the results are in variable *yy* and the ground truth are in variable *y1*. The average relative error in training set range is e1, and that out of training frequency range is e2.**

**The results are output to results/temperature\_predictor\_result.csv, each** **row refers to one geometry in dataset, and from left to right indicates the predicted temperature from 2-14 GHz (0.2GHz).**

**Inverse design by data-analytical driven networks:**



**In the development of inverse design network, the weights in (4) are important to direct the training. To test the model, just run parameterized\_strcuture/ inverse\_design\_code/inverse\_design.py with the trained model.**

**Model without input\_T is tested in**

**parameterized\_strcuture/inverse\_design\_code/** **inverse\_design\_without\_input\_T.py**

**There are 3 steps in total in this shared code：**

**For a demo of inverse design:**

**Step 1. Train the Inverse model with** **parameterized\_strcuture/ forward\_code/inverse\_design.py.**

**Step 2. Put the desired EM response in parameterized\_strcuture/ data/i-dataset/inverse\_test\_2.csv.**

**Step 2. Validate the model with inverse\_design.py: Remember to replace the trained model name and set the variable *train* to be False. The obtained structure is in variable *yy*.**