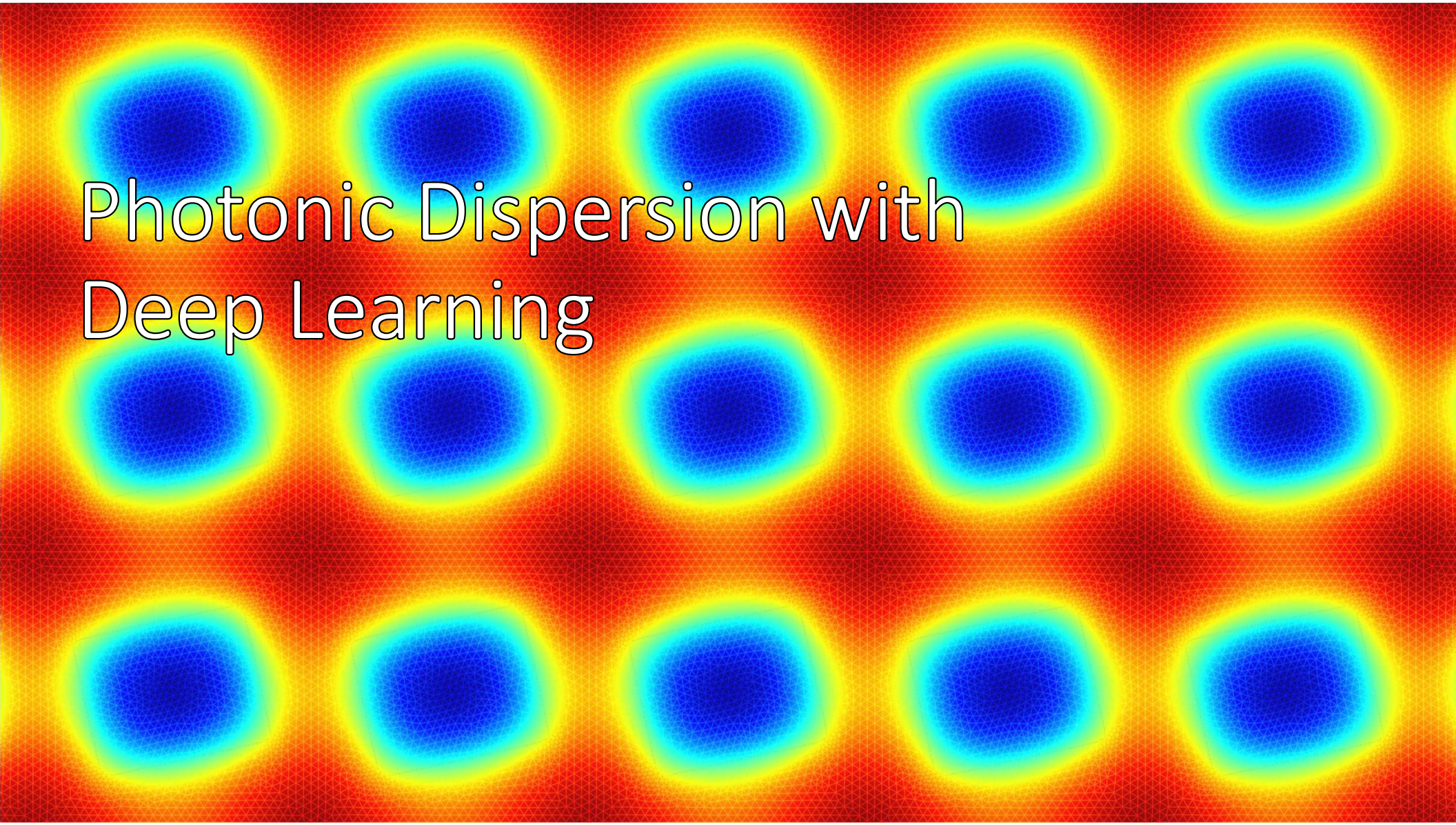
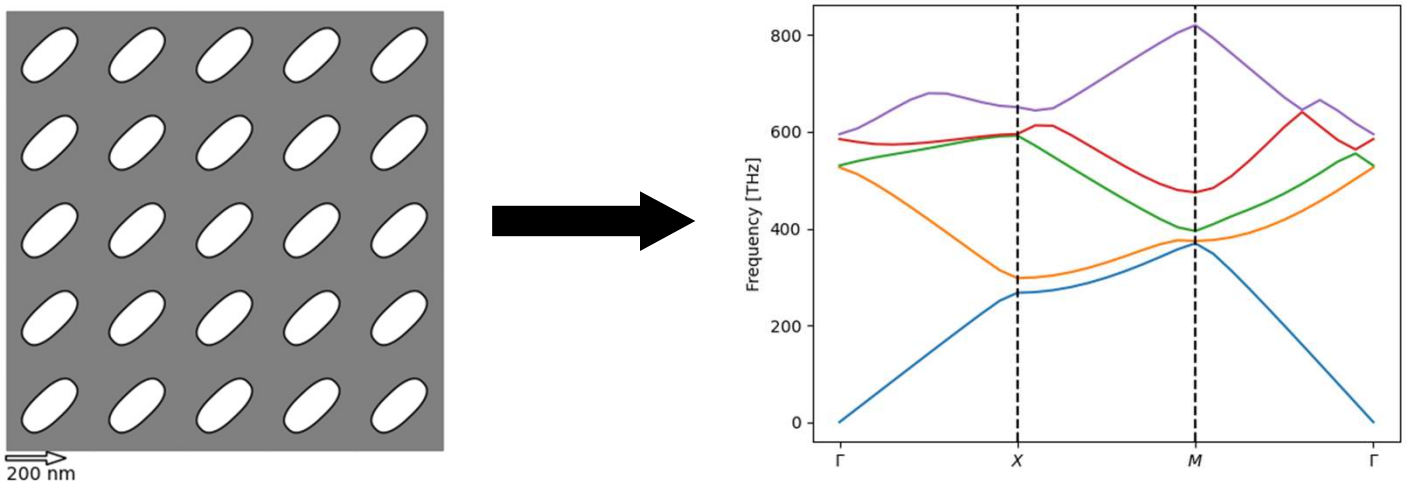
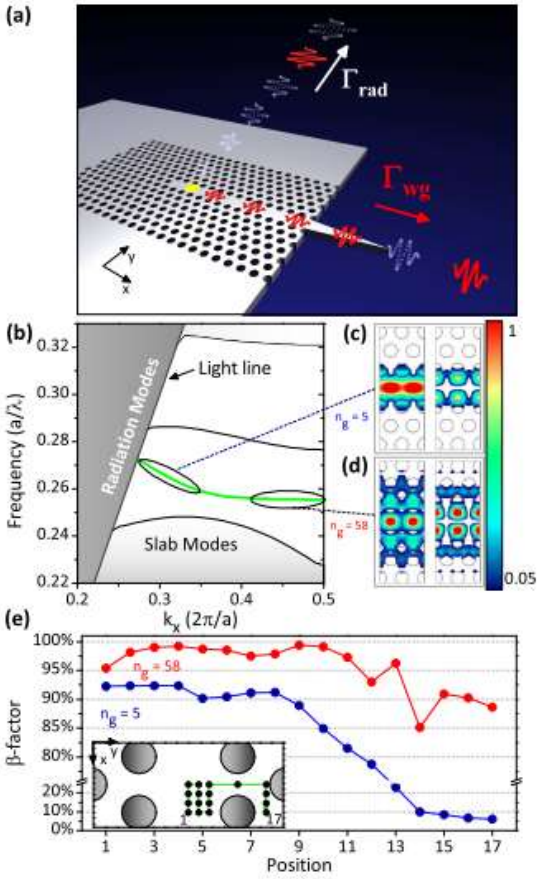
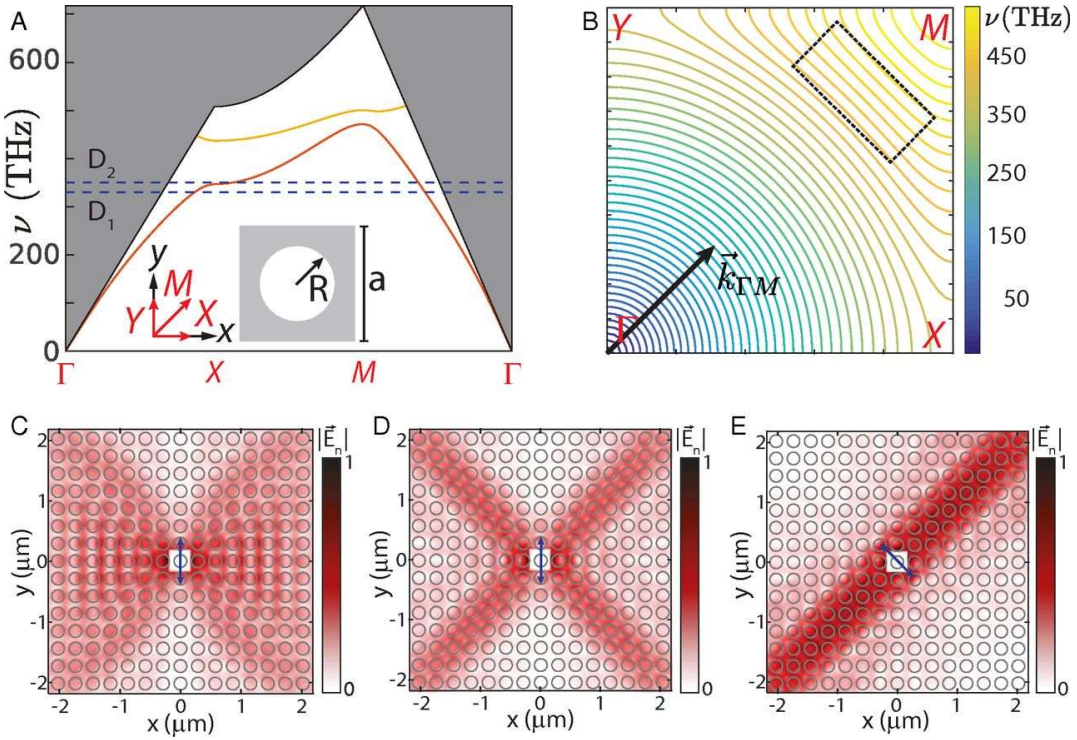

Photonic Dispersion with
Deep Learning

Photonic Crystals





M. Acari et al. (2014)



Su-Peng Yu et al. (2019)

Deep learning for the design of photonic structures

Wei Ma, Zhaocheng Liu, Zhaxylyk A. Kudyshev, Alexandra Boltasseva ✉, Wenshan Cai ✉ & Yongmin Liu ✉

Nature Photonics **15**, 77–90 (2021) | Cite this article

Plasmonic nanostructure design and characterization via Deep Learning

Itzik Malkiel, Michael Mrejen, Achiya Nagler, Uri Arieli, Lior Wolf & Haim Suchowski ✉

Light: Science & Applications **7**, Article number: 60 (2018) | Cite this article

Machine learning and applications in ultrafast photonics

Goëry Genty ✉, Lauri Salmela, John M. Dudley, Daniel Brunner, Alexey Kokhanovskiy, Sergei Kobtsev & Sergei K. Turitsyn

Nature Photonics **15**, 91–101 (2021) | Cite this article

Training Data

- Generate data with simulations
- COMSOL – FEM
- ~21,000 unique geometries

Parameterize structure

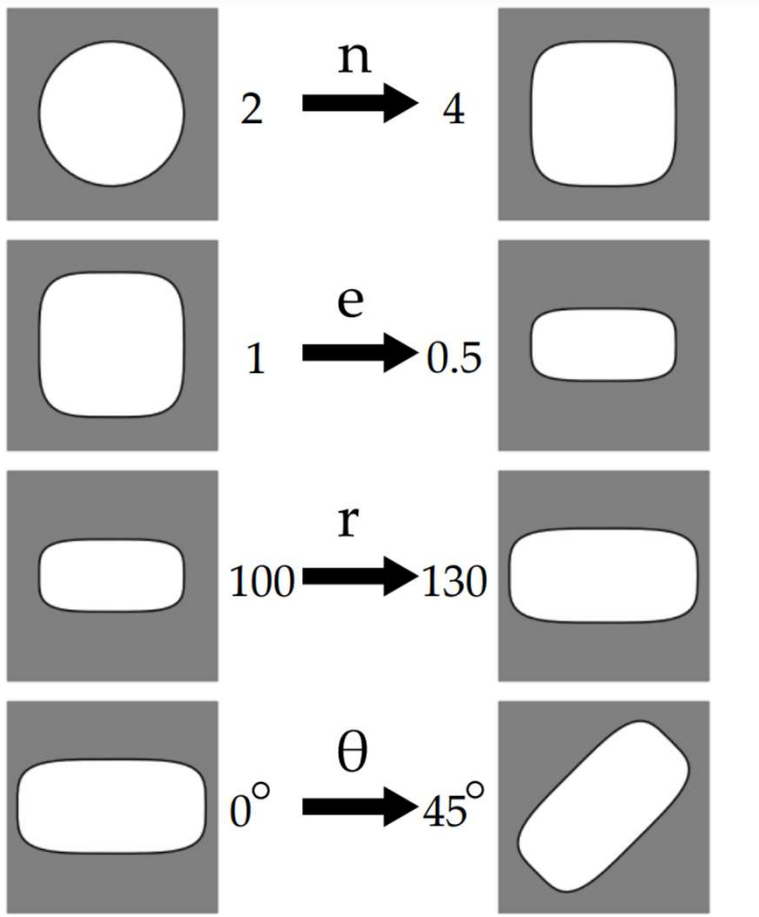
- Superellipse

$$x(t) = \left|cos^{2/n}t\right| r \operatorname{sgn}(\cos(t))$$

$$\mathrm{y}(t) = e\left|sin^{2/n}t\right| r \operatorname{sgn}(\sin(t))$$

$$t \in \{0, 2\pi\}$$

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$



Parameterize structure

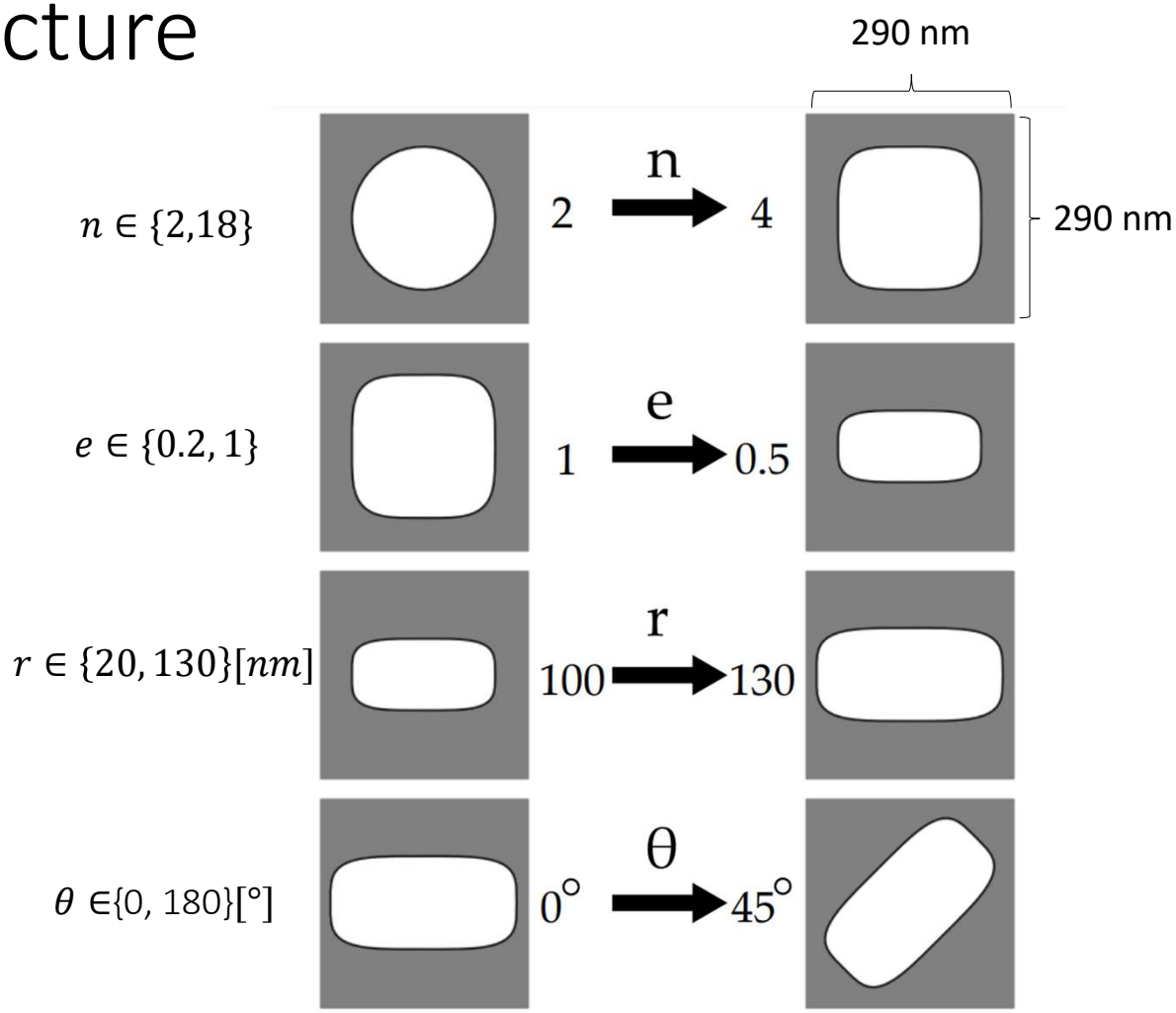
- Superellipse

$$x(t) = \left|cos^{2/n}t\right| r\ \mathrm{sgn}(\cos(t))$$

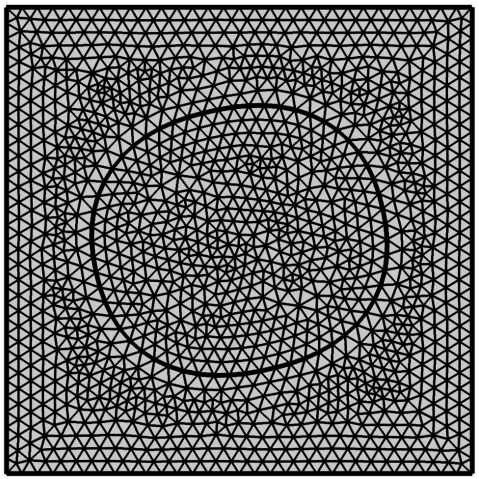
$$\mathrm{y}(t) = e\left|sin^{2/n}t\right| r\ \mathrm{sgn}(\sin(t))$$

$$t \in \{0, 2\pi\}$$

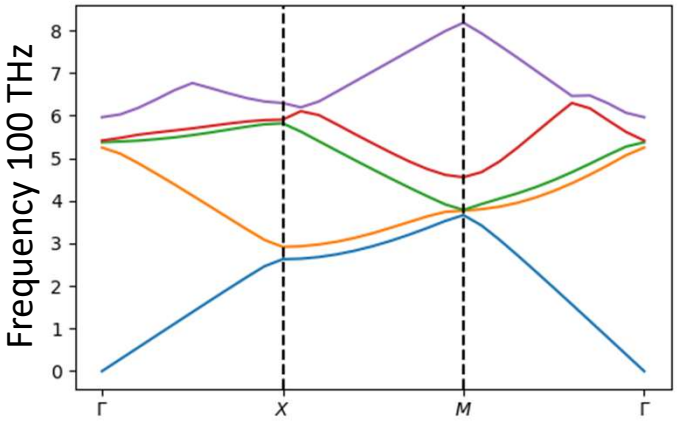
$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$



The simulation



- Build geometry from sampled parameters
- Discretize it
- Sweep boundary conditions
- Solve Maxwell's equations for first 5 eigenfrequencies



[31 x 5] array

The flow of data

- Parameters sampled in MATLAB
- COMSOL called from MATLAB and geometry is simulated
- Solution exported to .txt file on local PC
- Moved to Erda intermittently
- Can be accessed with sftp, e.g. with Paramiko python module

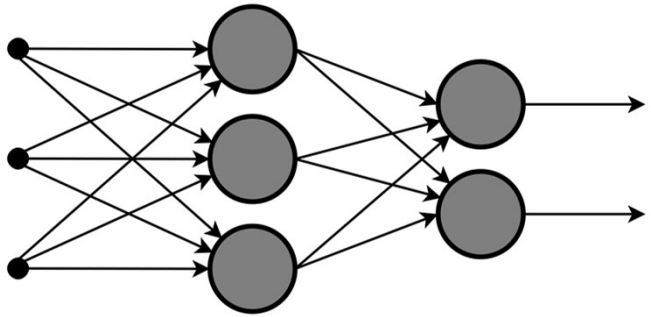
Architecture of NN

- **Problem type**: Multi-regression.
- Independent bands ⇨ **Block approach**.
- Normalized data through **Quantile Transformation**.
- Bidirectional NN [1] : Inverse + Direct
- Loss function:

$$\text{Loss} = \texttt{nn.MSELoss()}_{\text{Inverse}} + \texttt{nn.MSELoss()}_{\text{Direct}}$$

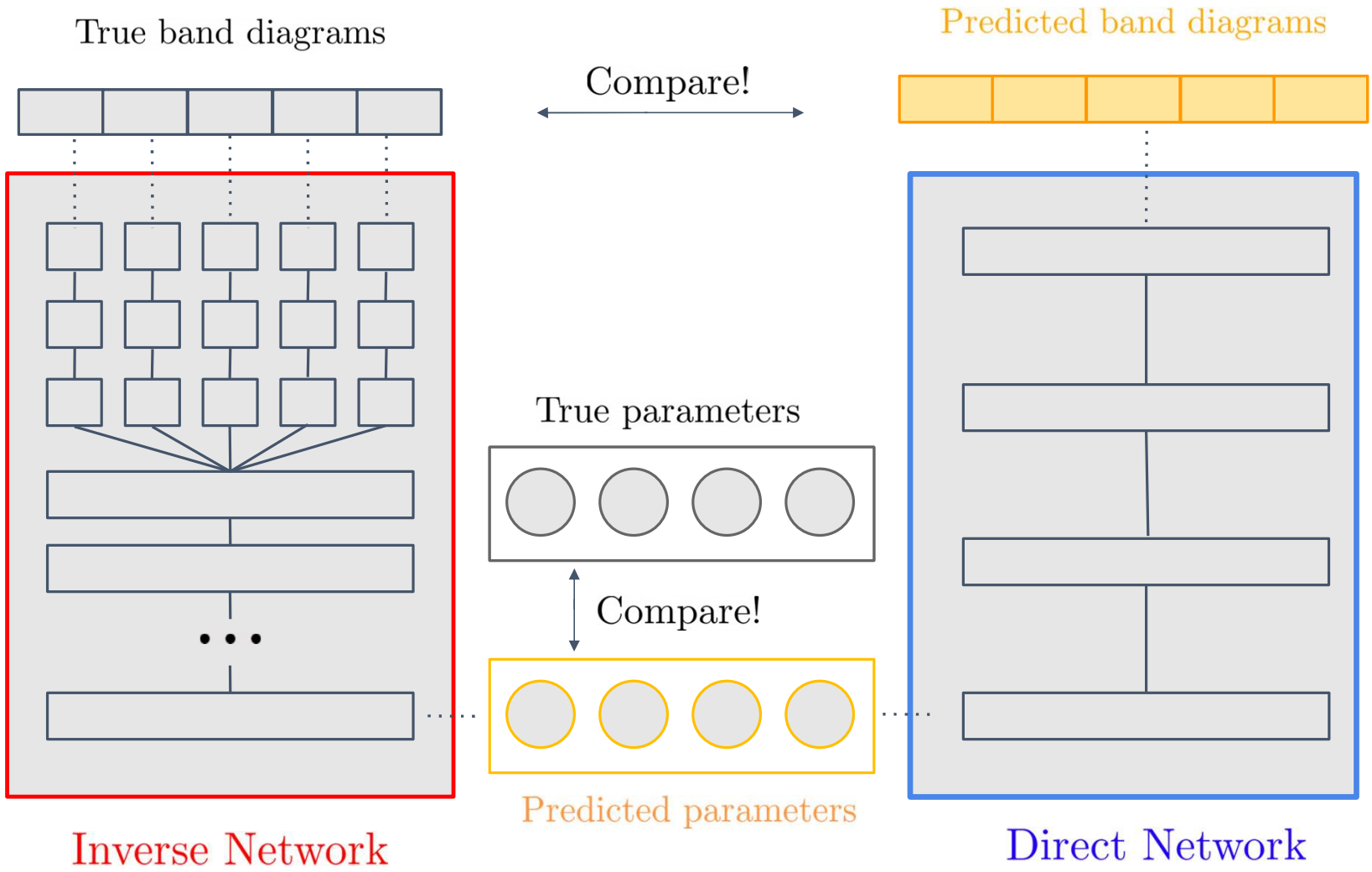
- Easy to implement **GPU** use: CUDA. (GTX 3070 available)

MLP : Multilayer Perceptron



PyTorch

[1] Malkiel, I. et al. Plasmonic nanostructure design and characterization via deep learning. Light Sci. Appl. 7, 60 (2018).


True band diagrams
Compare!
Predicted band diagrams
True parameters
Compare!
Predicted parameters
Inverse Network
Direct Network

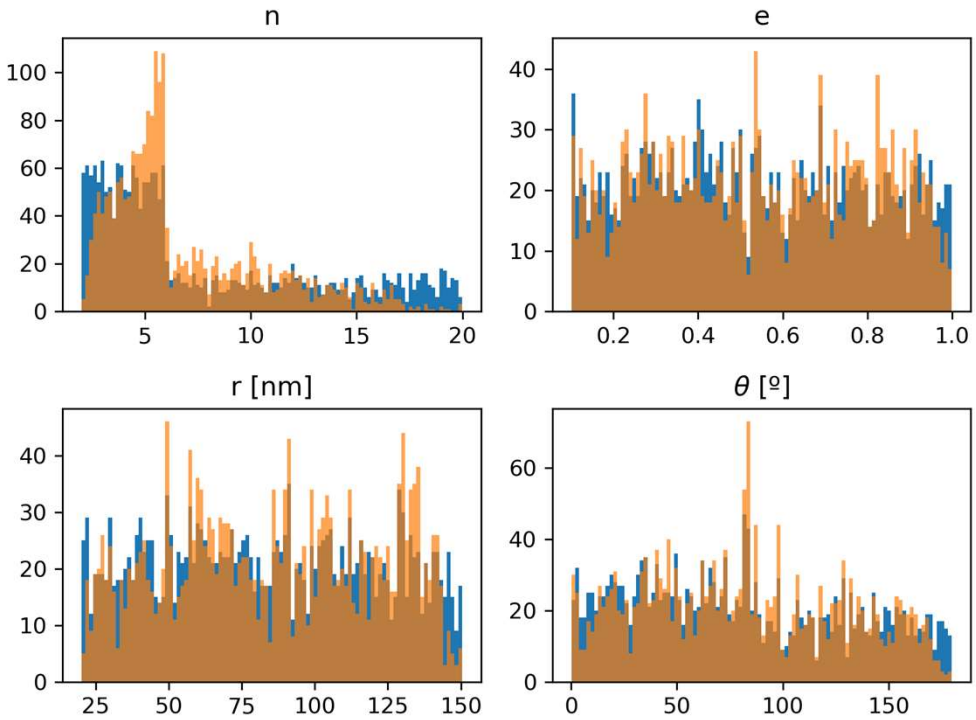
Some of the techniques

- Dropout
- Batch normalization
- Weights inizialization, easier for optimizer
- Different optimizer choices (and results)

Parameter & band predictions

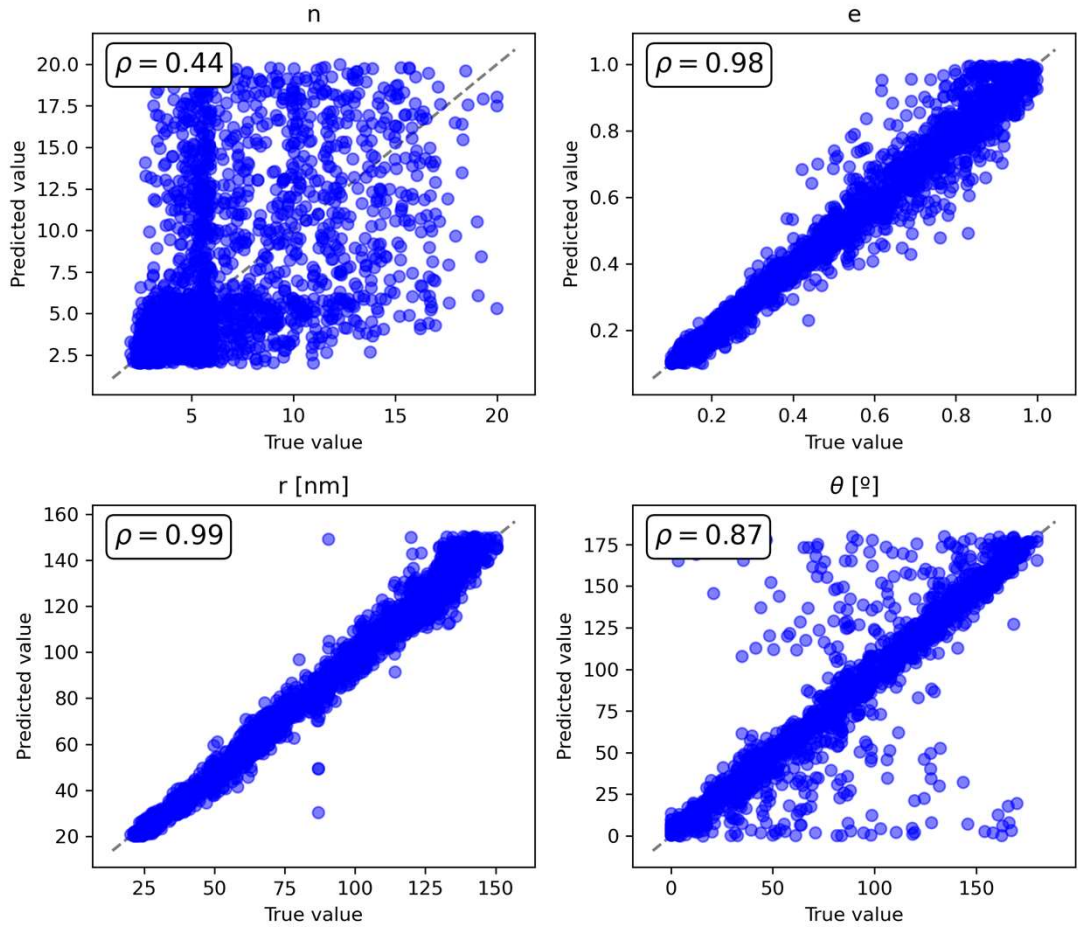
- In the beginning we were getting the **mean value predicted** for parameter **n**.
    - Lack of complexity and weak correlation to band structure.

- Parameter distributions and predictions:

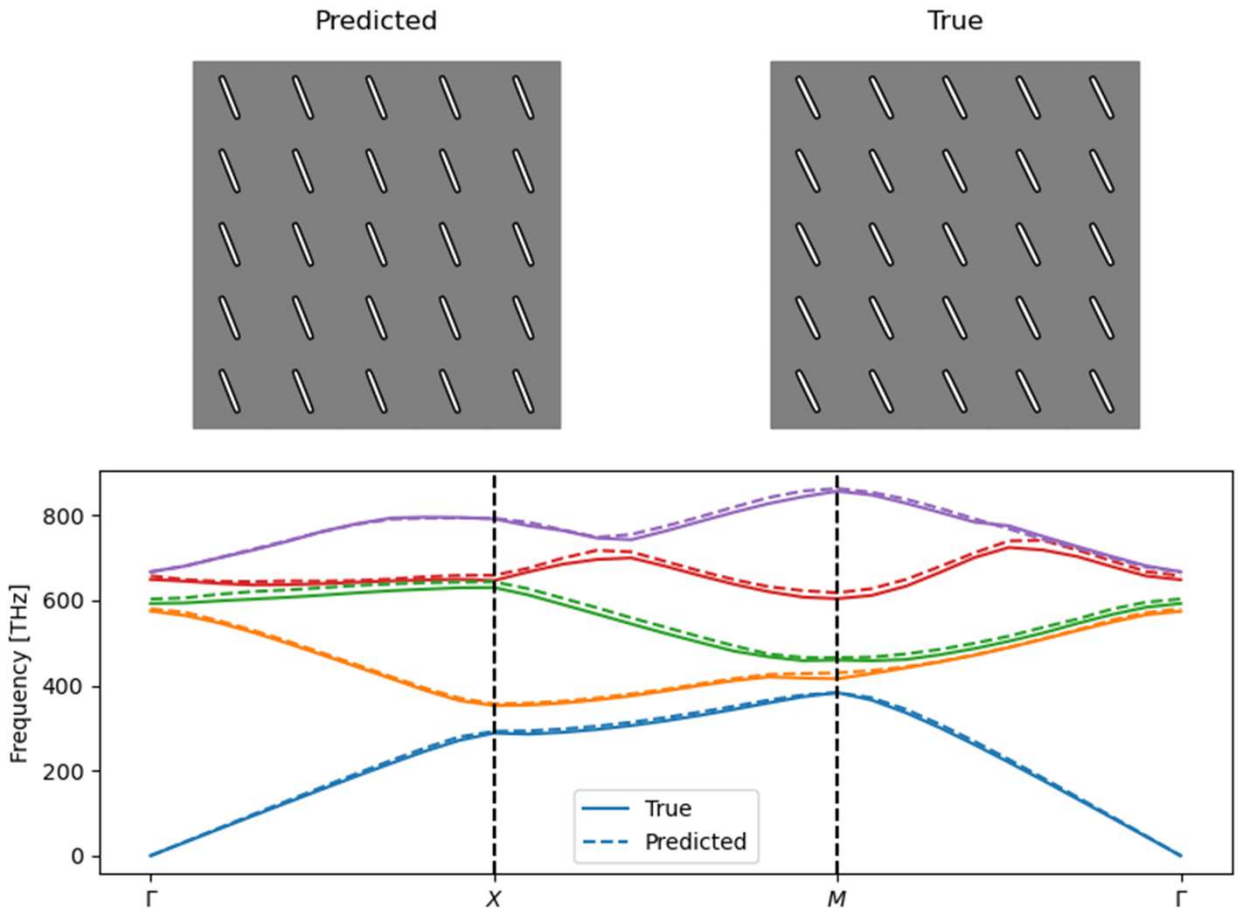


Parameter & band predictions

- N dependence is very weak
- θ errors may be due to symmetric geometries



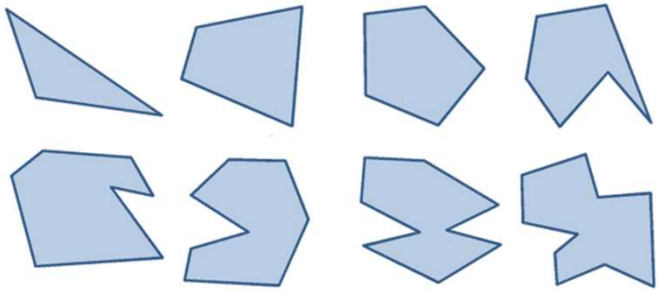
Example

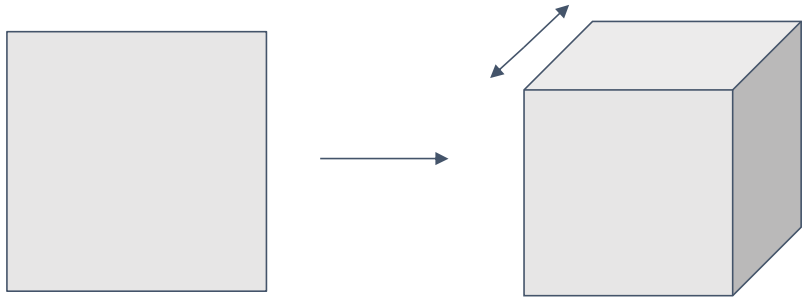

Predicted
True
800
600
400
200
0
Frequency [THz]
Γ
X
M
Γ
True
Predicted

Further work

How can we improve on this approach?

- More complex shapes.
- Add TE modes.
- Include more physical inputs (i.e. derivatives of bands).
- Include more simulation data.
- Go from 2D to 3D and add thickness parameter.





THANK YOU

Model summary

Inverse model

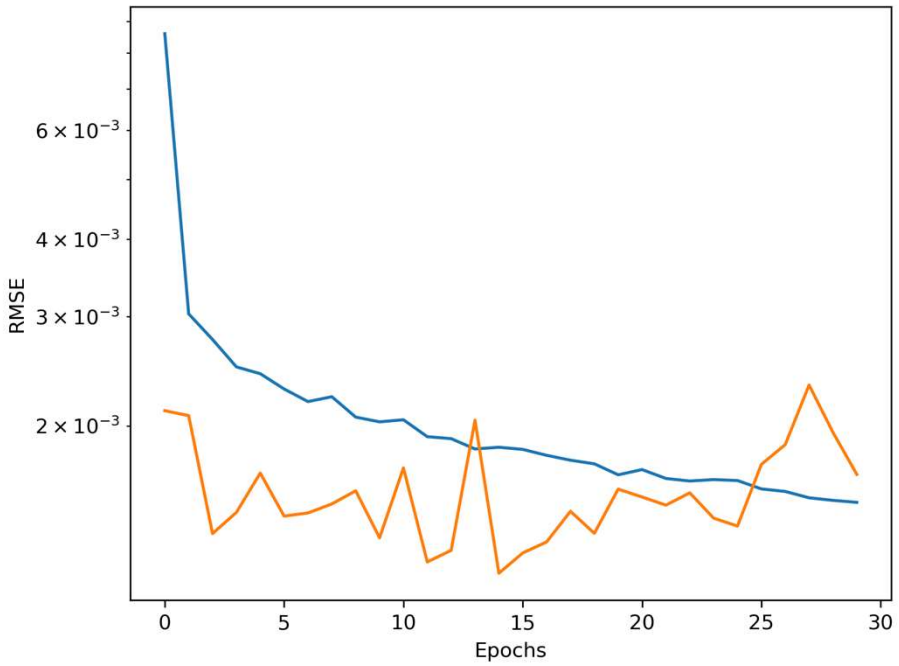
```
==========================================================================
Layer (type:depth-idx)                   Param #
==========================================================================
├─ReLU: 1-1                             --
├─ModuleList: 1-2                       --
|    └─Linear: 2-1                      40,480
|    └─ReLU: 2-2                        --
|    └─Linear: 2-3                      1,593,894
├─ModuleList: 1-3                       --
|    └─Linear: 2-4                      595,361
|    └─ReLU: 2-5                        --
|    └─Dropout: 2-6                     --
├─Sequential: 1-4                       --
|    └─Linear: 2-7                      992
|    └─ReLU: 2-8                        --
|    └─Linear: 2-9                      (recursive)
|    └─Linear: 2-10                     (recursive)
├─Sequential: 1-5                       --
|    └─Linear: 2-11                     7,026,336
|    └─ReLU: 2-12                       --
|    └─Linear: 2-13                     (recursive)
|    └─Dropout: 2-14                    --
|    └─Linear: 2-15                     2,136
==========================================================================
Total params: 9,259,199
Trainable params: 9,259,199
Non-trainable params: 0
```

Direct model

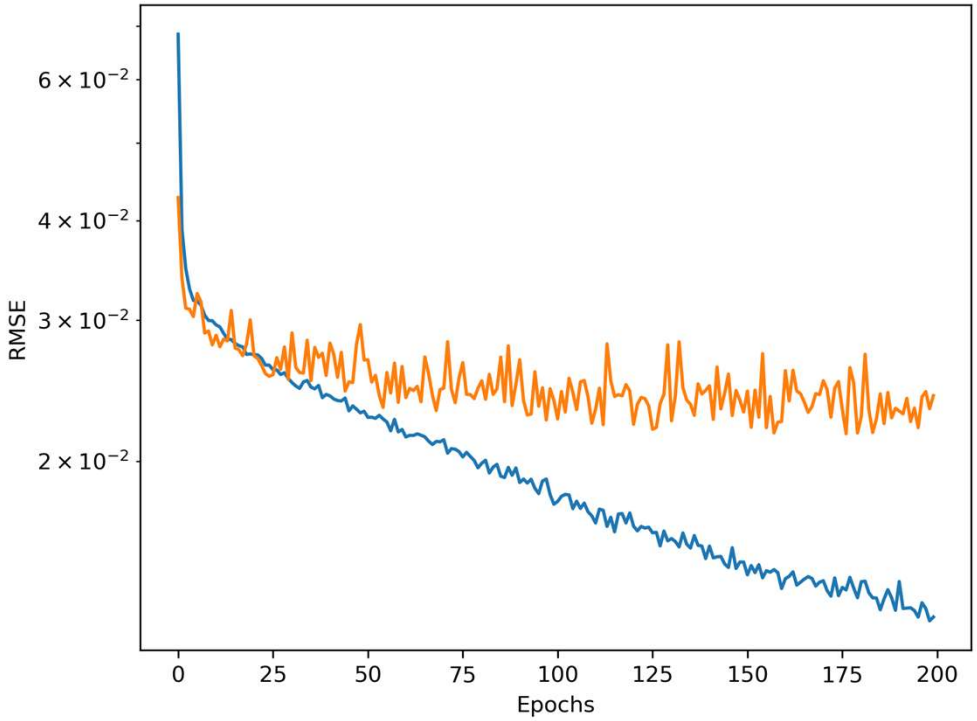
```
==========================================================================
==========================================================================
Layer (type:depth-idx)                   Param #
==========================================================================
├─ReLU: 1-1                             --
├─ModuleList: 1-2                       --
|    └─Linear: 2-1                      999,408
|    └─ReLU: 2-2                        --
|    └─Dropout: 2-3                     --
|    └─Linear: 2-4                      2,098,740
|    └─Dropout: 2-5                     --
|    └─Linear: 2-6                      1,792,165
|    └─Dropout: 2-7                     --
├─Sequential: 1-3                       --
|    └─Linear: 2-8                      3,755
|    └─ReLU: 2-9                        --
|    └─Linear: 2-10                     (recursive)
|    └─Dropout: 2-11                    --
|    └─Linear: 2-12                     (recursive)
|    └─Dropout: 2-13                    --
|    └─Linear: 2-14                     (recursive)
|    └─Dropout: 2-15                    --
|    └─Linear: 2-16                     176,080
==========================================================================
Total params: 5,070,148
Trainable params: 5,070,148
Non-trainable params: 0
==========================================================================
```

Training and validation loss

Direct model



Inverse model



Hyperparameter optimization

- Automatic hyperparameter optimization via **OPTUNA**.
- Implemented Pruning.
- Number of EPOCS: 10
- 200 trials for forward and backwards each

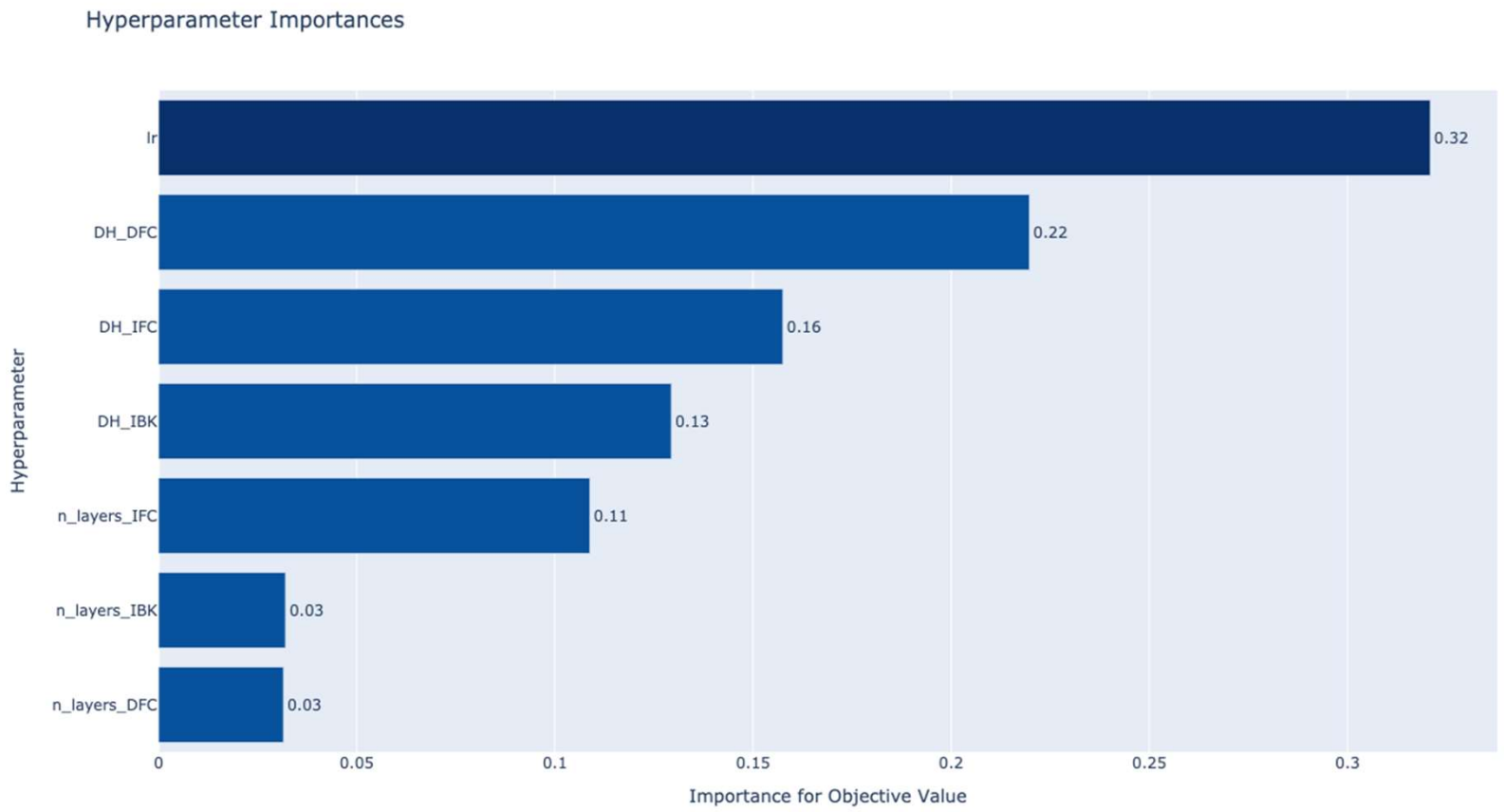
" An open source hyperparameter optimization framework to automate hyperparameter search"

Optimized parameters

- Inverse: {'N_HIDDEN_BK_LAYERS': 13,
'N_HIDDEN_FC_LAYERS': 3,
'N_HIDDEN_BK_UNITS_0': 950,
'N_HIDDEN_BK_UNITS_1': 774,
'N_HIDDEN_BK_UNITS_2': 934,
'N_HIDDEN_BK_UNITS_3': 1287,
'N_HIDDEN_BK_UNITS_4': 475,
'N_HIDDEN_BK_UNITS_5': 1475,
'N_HIDDEN_BK_UNITS_6': 283,
'N_HIDDEN_BK_UNITS_7': 368,
'N_HIDDEN_BK_UNITS_8': 437,
'N_HIDDEN_BK_UNITS_9': 917,
'N_HIDDEN_BK_UNITS_10': 1248,
'N_HIDDEN_BK_UNITS_11': 1429,
'N_HIDDEN_BK_UNITS_12': 918,
'N_HIDDEN_FC_UNITS_0': 1082,
'N_HIDDEN_FC_UNITS_1': 807,
'N_HIDDEN_FC_UNITS_2': 1803,
'P_DROPOUT_0': 0.3312902348149265,
'P_DROPOUT_1': 0.3350978993045471,
'optimizer': 'Adam',
'lr': 0.00021654758913416323}

Forward: {'N_HIDDEN_FC_LAYERS': 3,
'N_HIDDEN_FC_UNITS_0': 924,
'N_HIDDEN_FC_UNITS_1': 856,
'N_HIDDEN_FC_UNITS_2': 1376,
'P_DROPOUT_0': 0.2033220847686216,
'P_DROPOUT_1': 0.11433651601412209,
'optimizer': 'Adam',
'lr': 0.00015406608613430397}

Importance of hyperparameters



Derivatives