Consiglio Nazionale delle Ricerche

EFC2025 - 11-12.12.2025 Wrocław

From Distributional to Quantile Neural Basis Models: the case of Electricity Price Forecasting

A.Brusaferri, D.Ramin, A.Ballarino

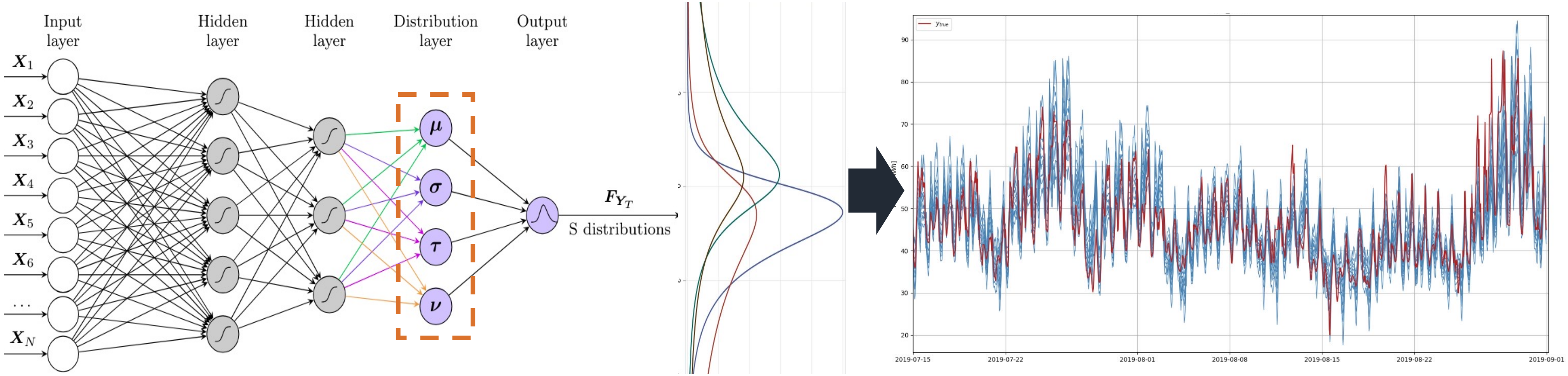
Focus: Probabilistic Forecasting | Neural Networks | Explainable AI

Context and challenges

Neural probabilistic forecasting

- **Focus:** Distributional/QR-neural networks for probabilistic forecasting ([Marcjasz et al, 2023], [Woo et al 2024], [Brusaferri et al 2025],…)
- Leverage **NNs to parameterize** flexible conditional **densities/quantiles**

$$p(y_{t+1} \dots y_{t+h} | y_{t-k:t}, z_{t-k:t}, x_{t+h}) = f_{\Theta}\left(y_{t-k:t}, z_{t-k:t}, x_{t+h}\right)$$

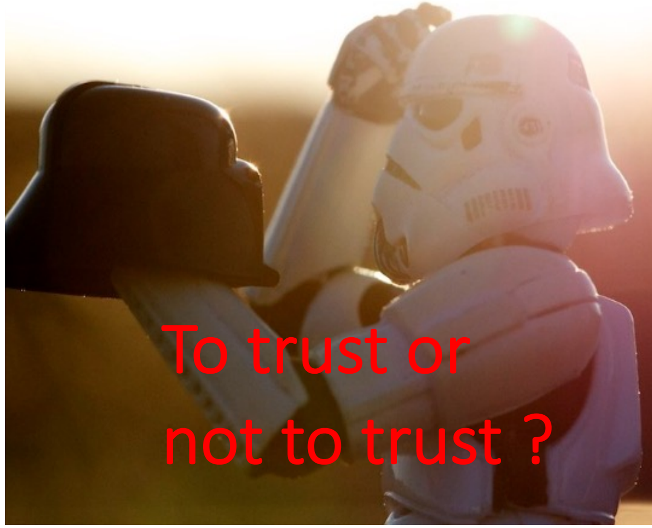


XAI challenge

- **Focus:** Distributional/QR neural networks for probabilistic forecasting ([Marcjasz et al, 2023], [Woo et al 2024], [Brusaferri et al 2025],…)
- Leverage **NNs to parameterize** flexible conditional **densities/quantiles**

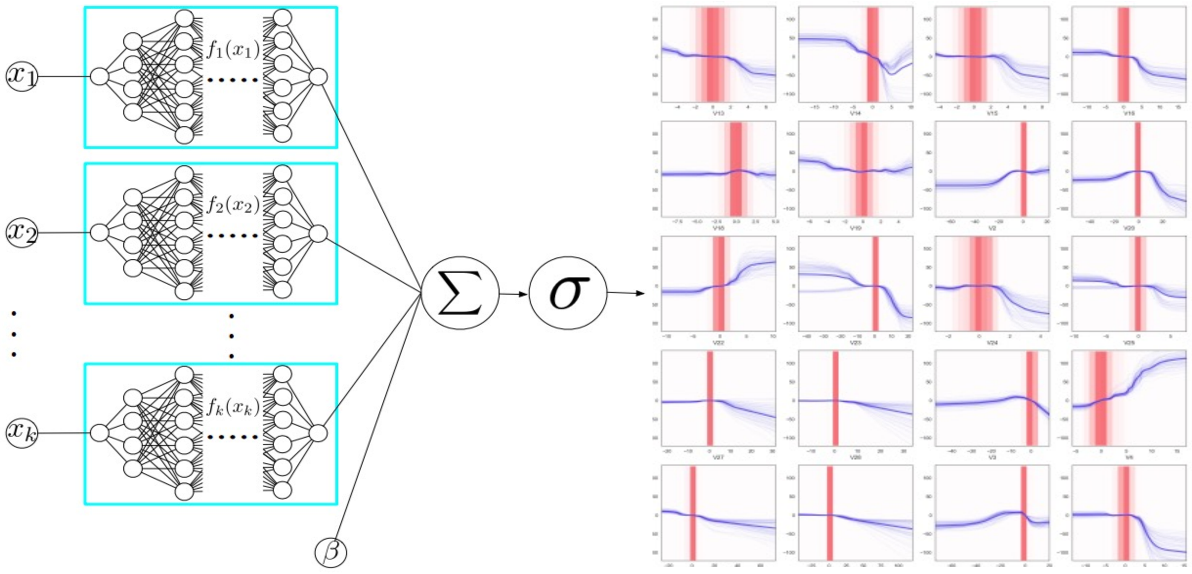
- **XAI challenge:** NNs flexible but inherently **black box**
- Learned **relation** between input variables and CDF parameters/quantiles **hidden** to the user

Goal: reveal the **underlying** mechanism leading to the predicted **feature-conditioned** distribution **param/quant**



Recent "Glass-box" NNs research momentum

- **NAMs** class: taking inspiration from GAM design [Hinton et al, 2021]
- NAM for **distributional regression** [Thielmann et al, 2024]

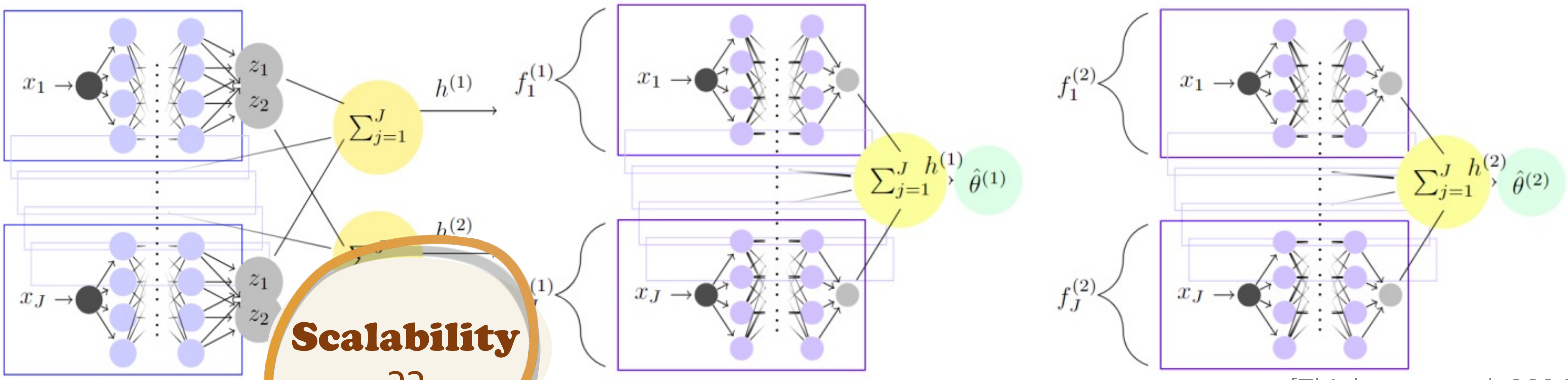


$$\mathbb{E}[y_d^h \mid \mathbf{x}_d] = \beta + f_1(x_{d,1}) + \cdots + f_{n_f}(x_{d,n_f})$$

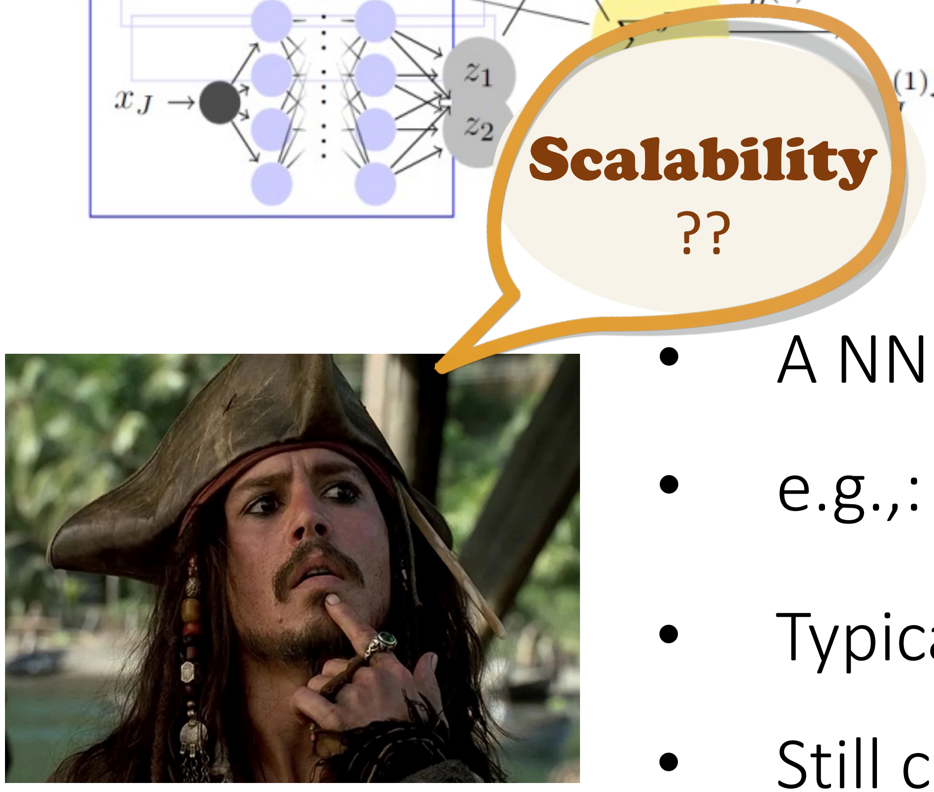
Recent "Glass-box" NNs research momentum

- **NAMs** class: taking inspiration from GAM design [Hinton et al, 2021]
- NAM for **distributional regression** [Thielmann et al, 2024]
- Still **understudied** in probabilistic forecasting (PF) context
- Explored for **point forecasting** by [Jo, 2023][Feddersen, 2024]
- NAMs challenging **scalability** to real world PF applications

Computationally intensive for PF implementation



[Thielmann et al, 2024]



- A NN for each **stage-wise input/density** param **map**
- e.g.,: H=24, |X|=100 --> **2400 NNs** (with param sharing)
- Typically **recalibrated** in PEPF apps (+ ensembling)
- Still computationally **"intensive"** for target PEPF tasks

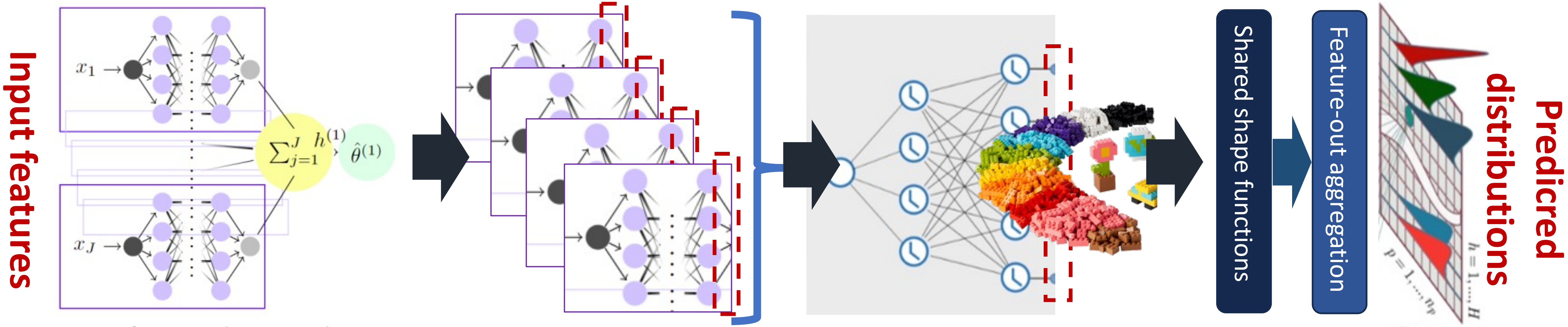
arXiv GitHub

From **NAM** to **D/Q-NBM**

NN inspired by GAMLSS/QGAM for PF

From NAMs to D/Q-NBM

- Leveraging **basis decomposition** of shape functions [Radenovic et al, 2022]
- Learn a set of **shared latent features** in a **multi-step PEPF** setup
- Exploit a cheap **unique NN** for the different feature-output maps
- Combined by **affine projections** supporting dedicated **step-wise** and **param/quantile-wise** feature shape functions **aggregations**



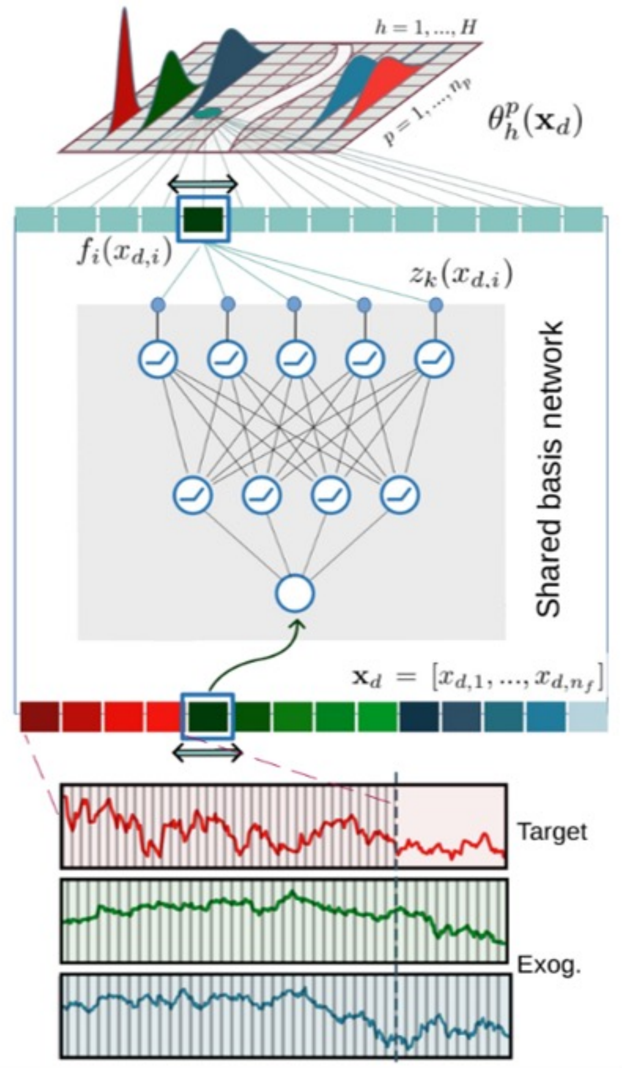
Set of NNs to learn each input-param mapping

Latent features identified by each NN individually

Exploit the NN flexibility to learn a shared features set

Step-wise distribution param/quant projections

D/Q-NBM architecture (in math)



$$z_k(x_{d,i}) = \mathbf{a}\left[\sum_{j=1}^{n_u} \omega_{j,k}^{(2)} \mathbf{a}\left[\omega_j^{(1)} x_{d,i}\right] + \omega_{0,k}^{(2)}\right], k = 1, \dots, n_z$$

$$f_i(x_{d,i}) = \sum_{k=1}^{n_z} W_{(i,k)} z_k(x_{d,i}), i = 1, \dots, n_f$$

$$\hat{\theta}_h^p(\mathbf{x}_d) = \mathbf{g}^p\left[\beta_h^p + \sum_{i=1}^{n_f} V_{(h,\gamma,i)} f_i(x_{d,i})\right], h = 1, \dots, H; p = 1, \dots, n_p$$

$$\boldsymbol{\Theta}(\mathbf{x}_d) = [\theta_1^1(\mathbf{x}_d), \dots, \theta_H^1(\mathbf{x}_d), \dots, \theta_1^{n_p}(\mathbf{x}_d), \dots, \theta_H^{n_p}(\mathbf{x}_d)]$$

$$\lambda_d^h = \boldsymbol{\Theta}(\mathbf{x}_d)^{[h]}$$

$$\sigma_d^h = \epsilon + \varrho \text{ Softplus}\left(\boldsymbol{\Theta}(\mathbf{x}_d)^{[H+h]}\right)$$

$$\tau_d^h = 1 + \varrho \text{ Softplus}\left(\boldsymbol{\Theta}(\mathbf{x}_d)^{[2\cdot H+h]}\right)$$

$$\varsigma_d^h = \boldsymbol{\Theta}(\mathbf{x}_d)^{[3\cdot H+h]}$$

$$d^h(\chi; \mathbf{x}_d) = \frac{\tau_d^h}{\sigma_d^h\sqrt{2\pi}} \frac{1}{\sqrt{1 + \left(\frac{\chi-\lambda_d^h}{\sigma_d^h}\right)^2}} e^{-\frac{1}{2}\left[\varsigma_d^h + \tau_d^h \sinh^{-1}\left(\frac{\chi-\lambda_d^h}{\sigma_d^h}\right)\right]^2}$$

D*

$$\sum_\gamma (y_d^h - \hat{q}_h^\gamma(\mathbf{x}_d))\gamma 1\{y_d^h > \hat{q}_h^\gamma(\mathbf{x}_d)\} + (\hat{q}_h^\gamma(\mathbf{x}_d) - y_d^h)(1-\gamma)1\{y_d^h \leq \hat{q}_h^\gamma(\mathbf{x}_d)\}$$

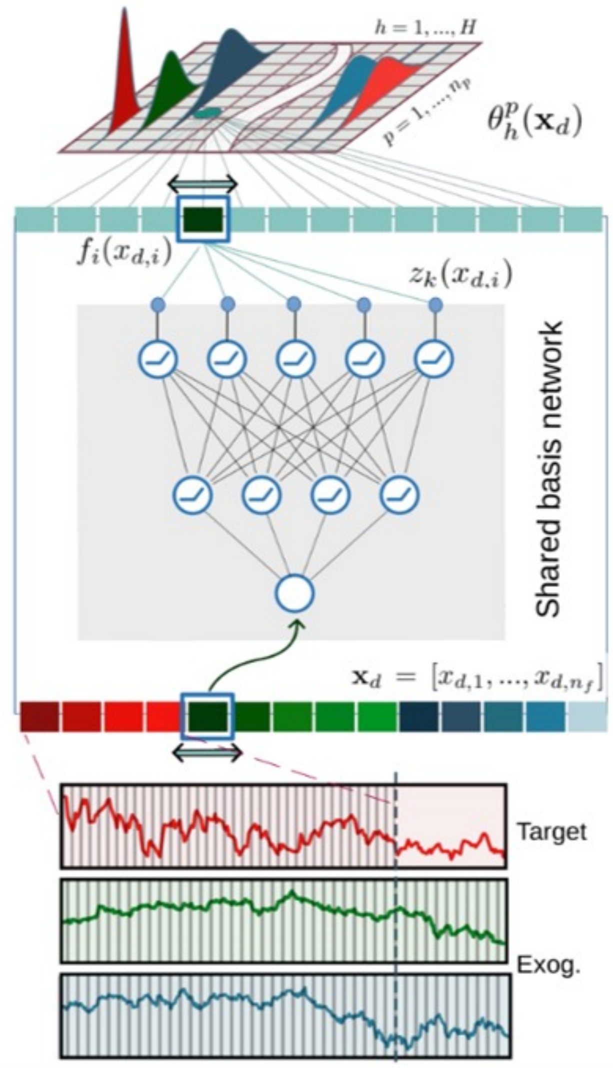
Q*

$$M \approx AB^\top, \text{ with: } A \in \mathbb{R}^{m\times r}, B \in \mathbb{R}^{n\times r}, r \ll m, n$$

Major ingredients:

- Last hidden layer operates as "shared basis" functions
- Shared basis aggregated in input-specific shape functions
- Shape functions combined in stage-wise parameterization
- Stage-wise link function
- Link fun. parameterizations
- Step-wise distrib. (e.g., JSU)
- Quantile mapping/loss
- Basis dropout
- Low-rank factorization for scalable mapping

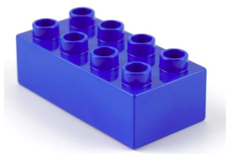
D/Q-NBM as NN building block




```python
class DQNBMRegressor:
    def __init__(self, settings, loss):
        self.settings = settings
        self.settings['add_res'] = False
        self.__build_model__(loss)
        self.loss = loss

    def __build_logit__(self, x_in, out_size):
        def concat_with_batch_size(inputs):
            t1, t2 = inputs
            batch_size = tf.shape(t1)[0]
            t2 = tf.tile(t2, [batch_size, 1, 1])
            return tf.concat([t1, t2], axis=-1)

        if self.settings['basis_mode'] == 'full':
            # [B,nf] --> [B,1,1,nf]
            x_b = tf.expand_dims(x_in, axis=1)
            x_b = tf.expand_dims(x_b, axis=1)
            # [B,1,1,nf] --> [B,h,p,nf]
            x_b = tf.tile(x_b, [1, self.settings['pred_horiz'], self.out_size, 1])
            # [B,h,p,nf] --> [B,h,p,nf,1]
            x_b = tf.expand_dims(x_b, axis=-1)
            # [B,h,p,nf] --> [B,h,p,nf,nh]
            x_b = tf.keras.layers.Dense(self.settings['hidden_size'],
                                        activation=self.settings['activation'],
                                        name='l0-basis')(x_b)
```



```python
elif self.settings['PF_method'] == 'STU':
    self.out_size = 3
    logit = self.__build_logit__(x_in=x_in, out_size=self.out_size)

    output = tfp.layers.DistributionLambda(
        lambda l: tfp.distributions.TransformedDistribution(
            distribution=tfd.StudentT(
                loc=l[0][..., :self.settings['pred_horiz']],
                scale=1e-3 + 3*tf.math.softplus(l[0][...,self.settings['pred_horiz']:self.settings['pred_horiz']
                df=1 + 3*tf.math.softplus(l[0][..., self.settings['pred_horiz'] * 2:])),
            bijector=tfp.bijectors.Chain([tfp.bijectors.Shift(shift=l[2]),tfp.bijectors.Scale(scale=l[1])]))
    )([logit,target_scales_ex[:,0],target_locs_ex[:,0]])

elif self.settings['PF_method'] == 'qr':
    self.out_size = len(self.settings['target_quantiles'])
    logit = self.__build_logit__(x_in=x_in, out_size=self.out_size)
```

- **Trained** multi-step, **end-to-end**
- **1 NN** by tensor **broadcasting**
- **Easy** auto build from **settings**

- Pure TF (Torch) code
- GPU/TPU ready
- **Composable** in pipeline
- Multimodel ensembles, etc

EPF experiments

Datasets

Open benchmark structured by [Aliyon et al 2024]:

- **Regions**: Germany, Belgium, Spain, Sweden-Stockholm (SE3)
- **Extent**: January 2019 - September 2024
- **Exog. vars**: load pred; wind/solar generation pred; calendar (sin-cos)
- **Test sets**: 1/10/2023 -30/9/2024
- **Validation**: previous year for hypertuning, 20% for early stopping
- **Conditioning**: day-ahead exog + d-1,d-2,d-7 hourly prices => 147 feat

147*24*4 = **14000 NNs** under conventional feature-wise NAM setup

Experiments setup

Baselines: D-DNN (N, JSU, STU), Q-DNN

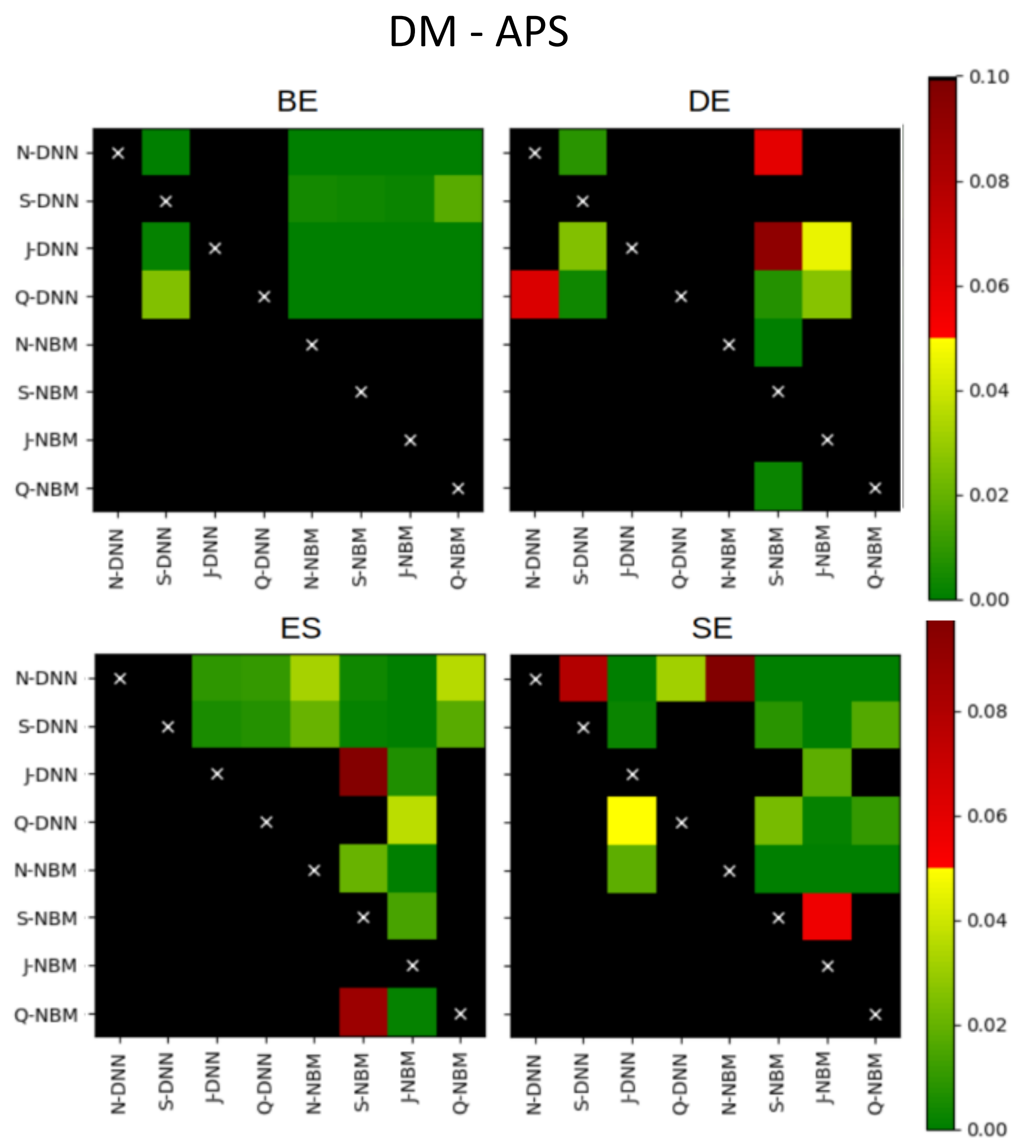
Consistent training/hypertuning:

- **Learning**: Adam, max 800 epochs, patience 20, batch size 32
- **Hyperparam** search by Optuna

N-DNN	BE	DE	ES	SE
n_u	512	768	640	768
l_r	1e-3	5e-5	1e-3	1e-3
d_r	0.3	0.3	0.3	0.3
S-DNN	BE	DE	ES	SE
n_u	768	640	640	512
l_r	1e-4	1e-4	5e-4	1e-3
d_r	0.3	0.1	0.5	0.3
J-DNN	BE	DE	ES	SE
n_u	768	512	640	762
l_r	5e-4	5e-4	1e-4	1e-4
d_r	0.3	0.3	0.3	0.3
Q-DNN	BE	DE	ES	SE
n_u	128	640	512	128
l_r	5e-4	1e-4	5e-4	1e-3
d_r	0.1	0.1	0.1	0.3

N-NBM	BE	DE	ES	SE
n_u	256	128	64	256
l_r	5e-4	5e-4	5e-4	1e-4
d_r	0.5	0.5	0.3	0.3
S-NBM	BE	DE	ES	SE
n_u	128	64	128	128
l_r	5e-4	5e-4	1e-4	1e-4
d_r	0.5	0.1	0.5	0.1
J-NBM	BE	DE	ES	SE
n_u	128	64	32	64
l_r	5e-4	1e-4	5e-4	5e-4
d_r	0.5	0.3	0.3	0.1
Q-NBM	BE	DE	ES	SE
n_u	64	64	64	32
l_r	5e-4	5e-4	5e-4	1e-4
d_r	0.3	0.1	0.3	0.1

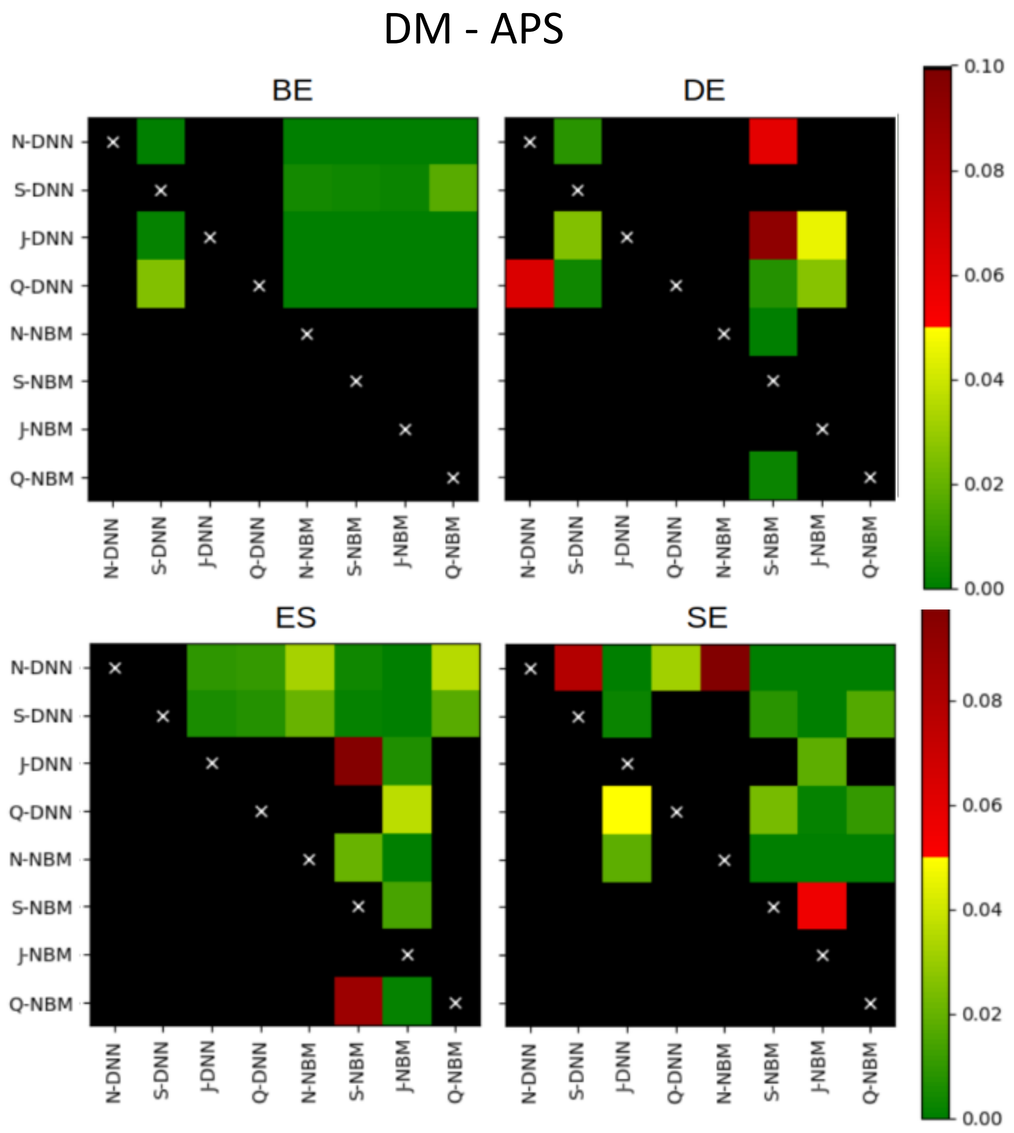
Test set results



APS	BE	DE	ES	SE
N-DNN	4.860	3.785	4.318	4.351
S-DNN	4.776	3.727	4.350	4.280
J-DNN	4.847	3.809	4.253	4.151
Q-DNN	4.863	3.858	4.225	4.236
N-NBM	4.634	3.787	4.225	4.279
S-NBM	4.632	3.711	4.188	4.097
J-NBM	4.644	3.728	4.137	4.035
Q-NBM	4.653	3.789	4.224	4.096

- **D/Q-NBMs** has achieved PF scores comparable (in some cases slightly improved) to **D/Q-DNNs**
- Best **distribution/quantile** form dataset specific
- Selection depending on **application** needs

Test set results

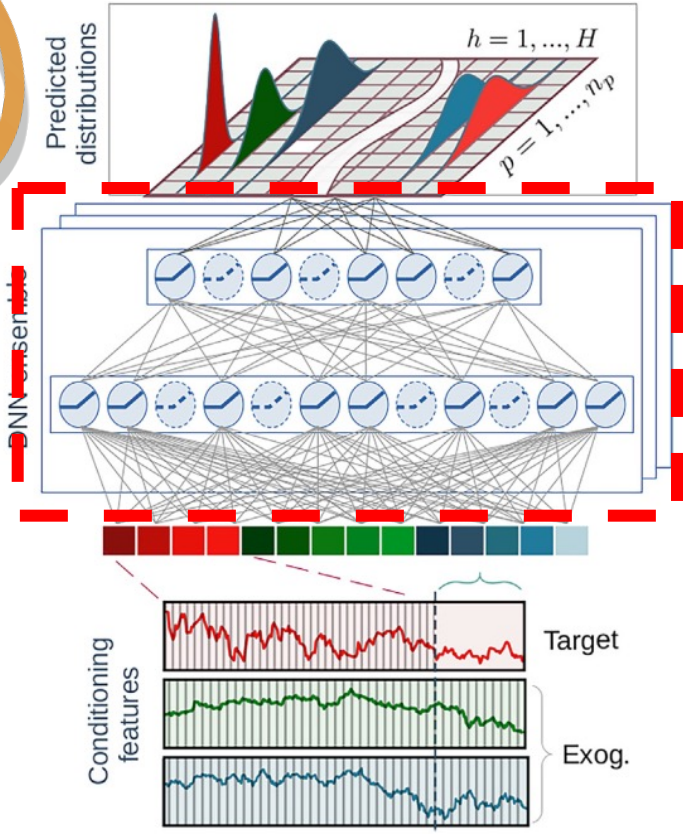


D/Q-NBMs has achieved PF scores comparable (in some cases slightly improved) to **D/Q-DNNs**

From "black-box" to "glass-box" NNs

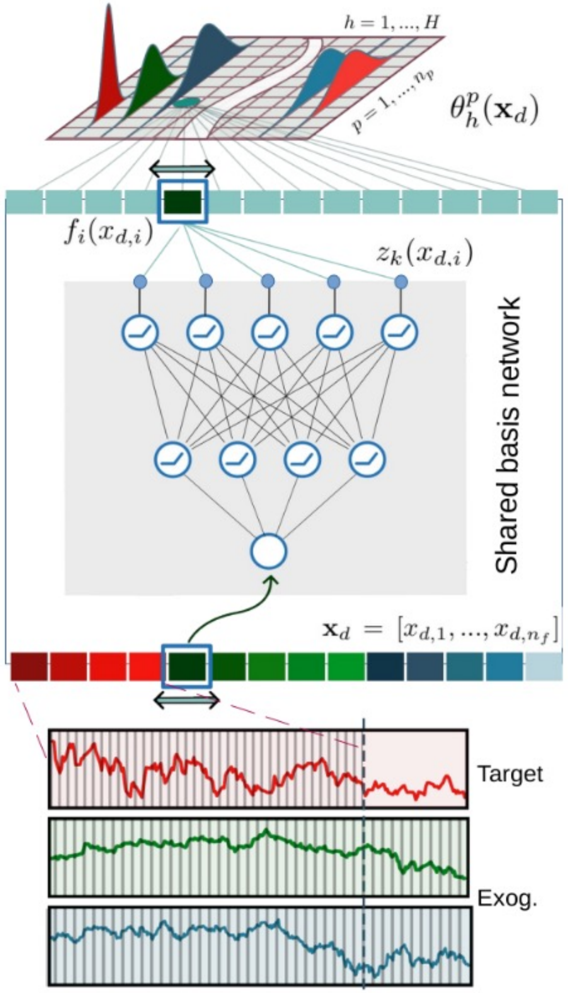
What did my NN learn?

Fully **black box** NN



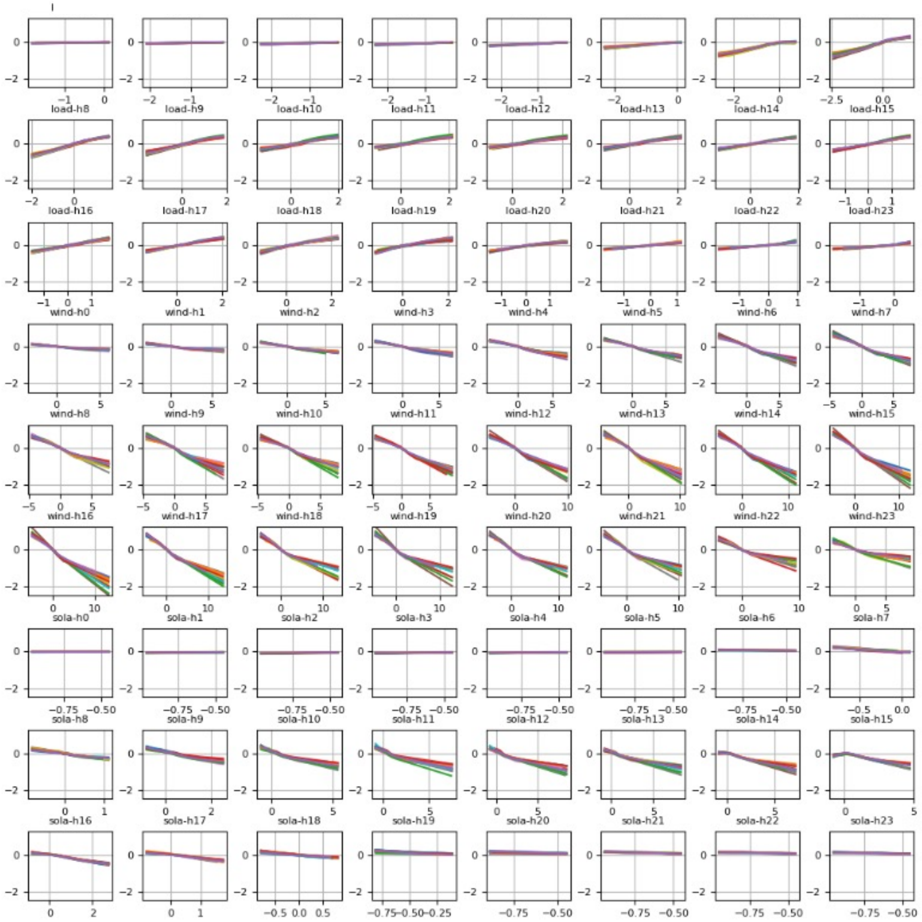
Feature-out relations
hidden within the
computational graph

Revealing identified feature shape maps



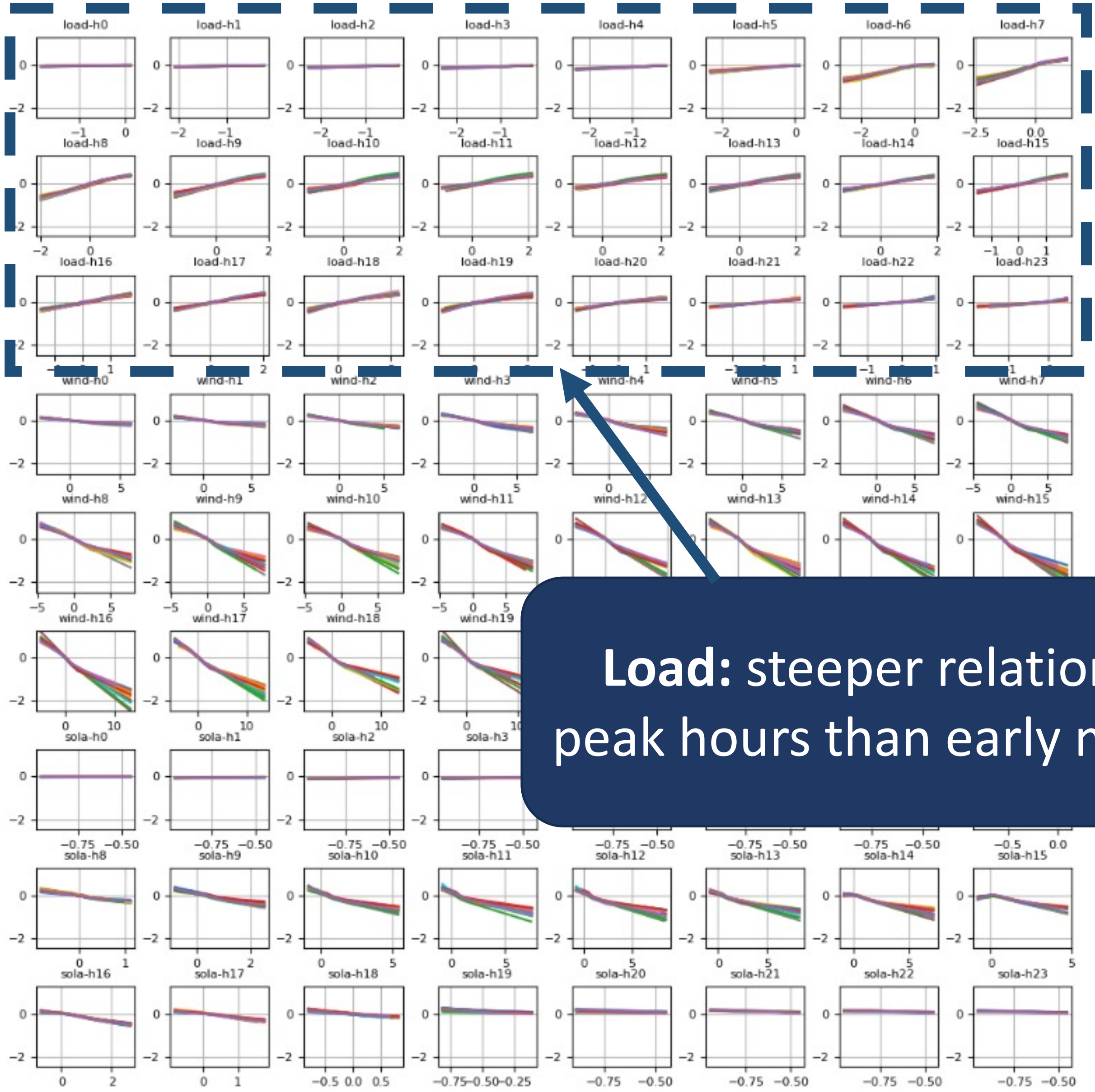
Gaining insights into what the NN is doing under the hood

Identified feature-out relations

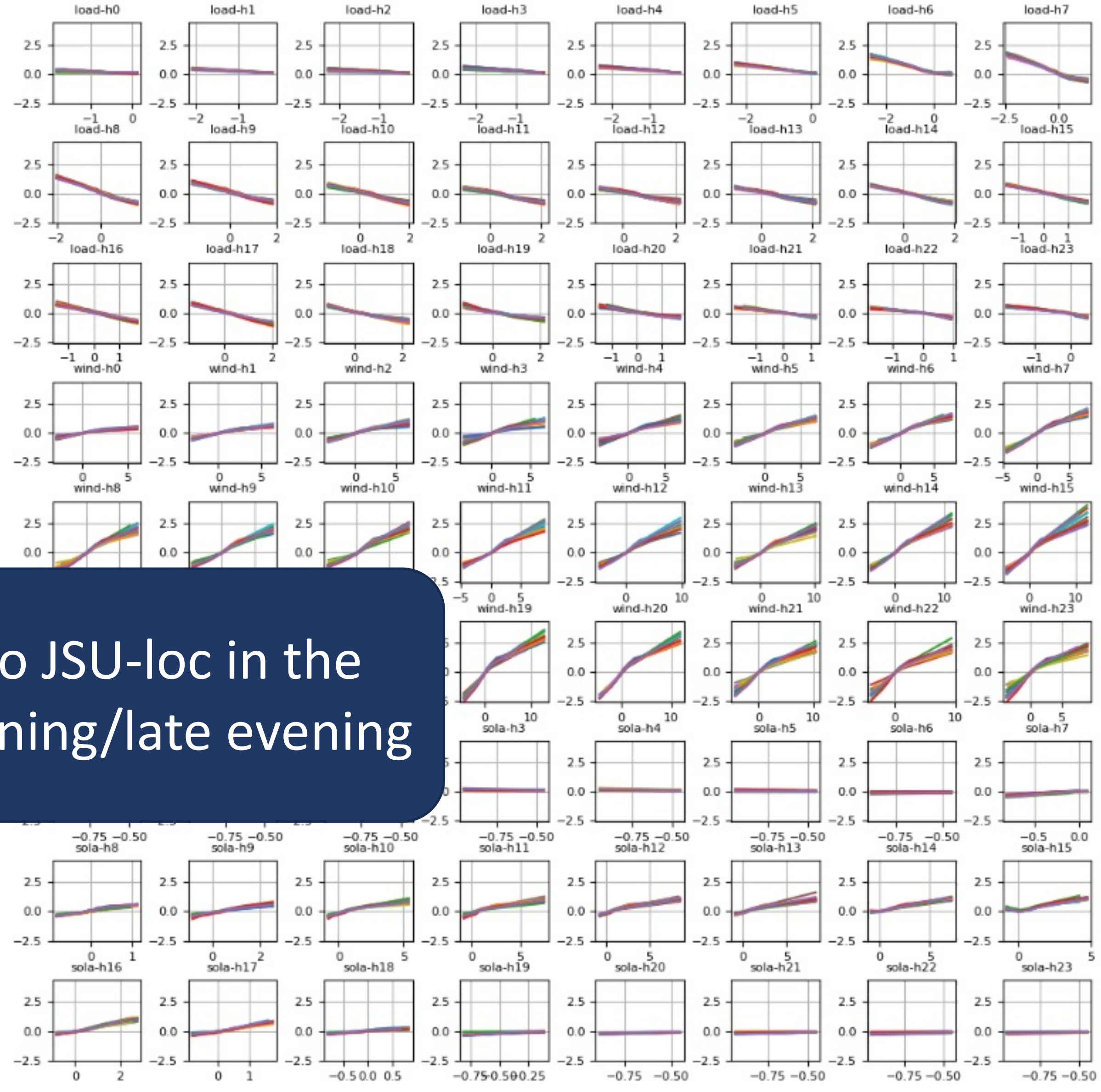


Revealing identified feature shape maps

DE – JSU location



DE – JSU skeweness



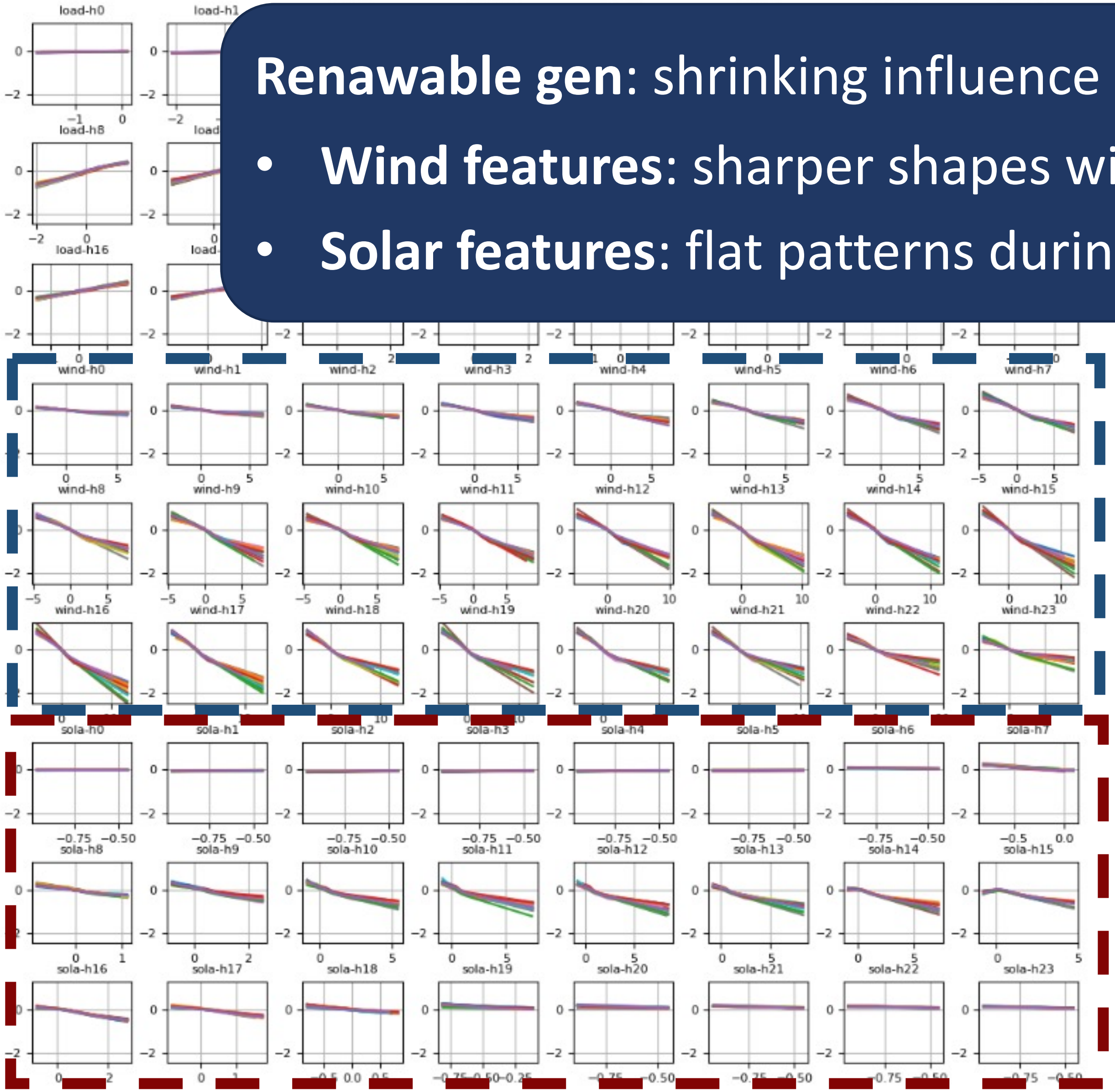
Load: steeper relations to JSU-loc in the peak hours than early morning/late evening

Revealing identified feature shape maps

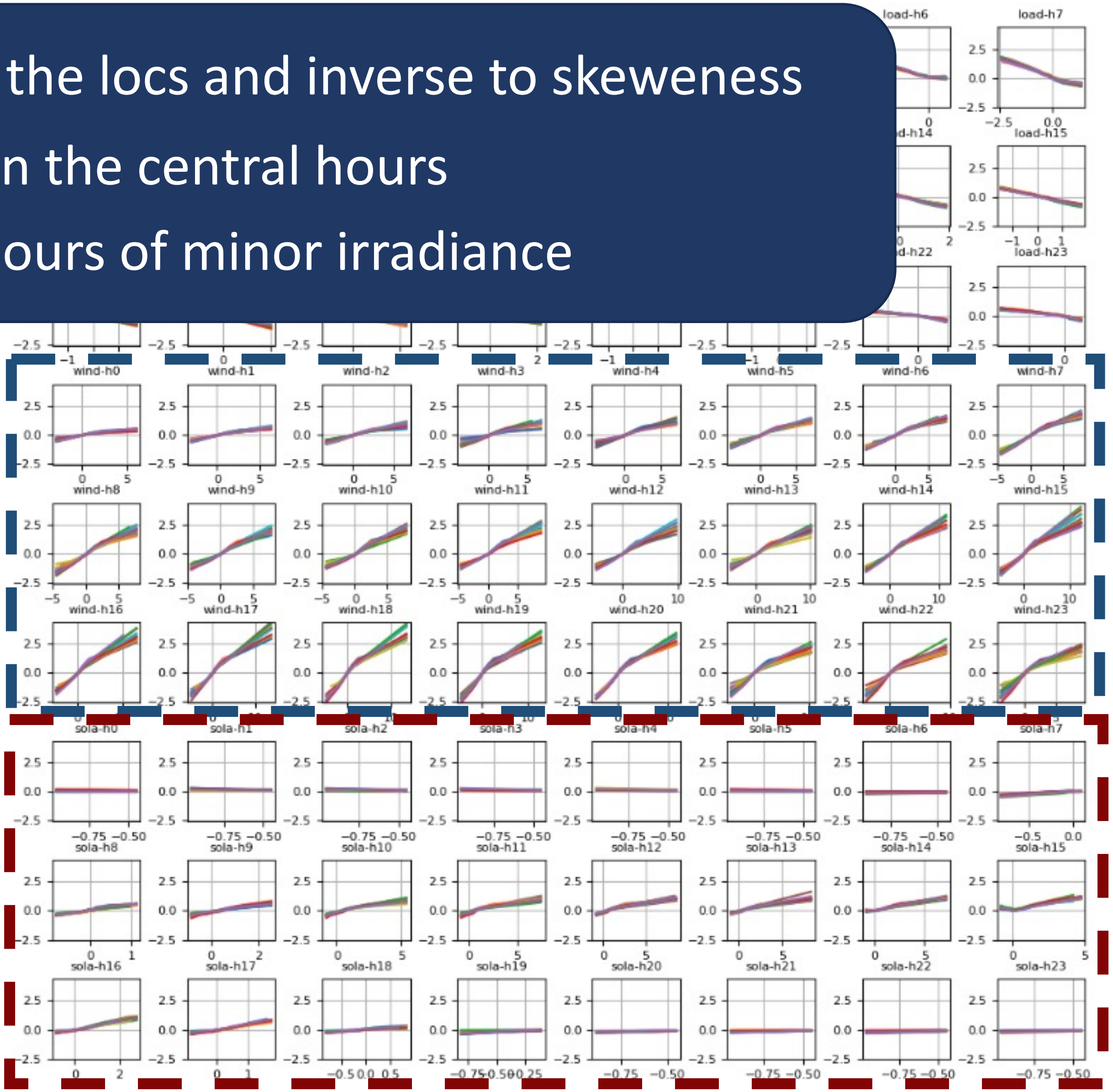
Renawable gen: shrinking influence on the locs and inverse to skeweness

- **Wind features**: sharper shapes within the central hours
- **Solar features**: flat patterns during hours of minor irradiance

DE – JSU location

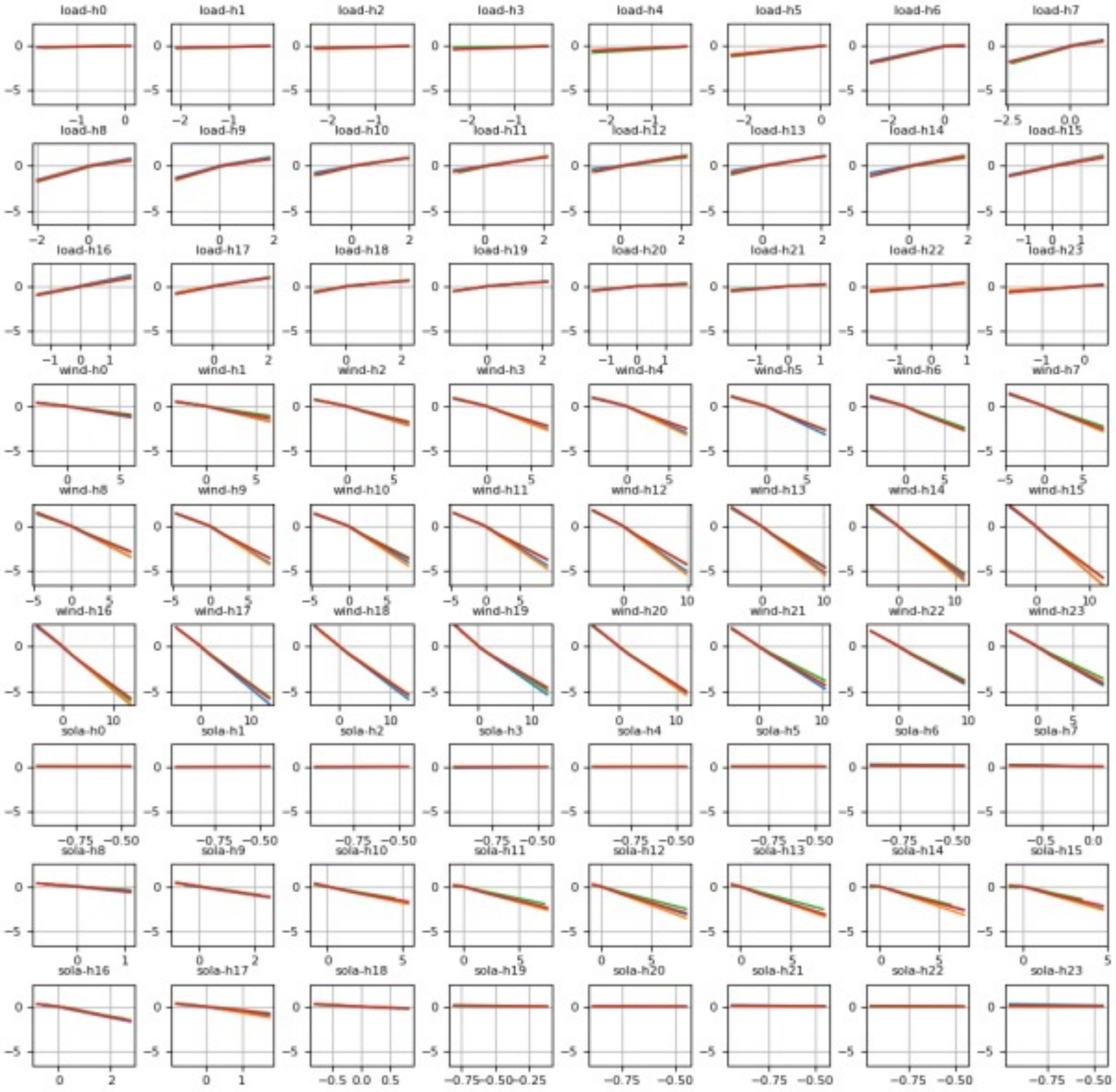


DE – JSU skeweness

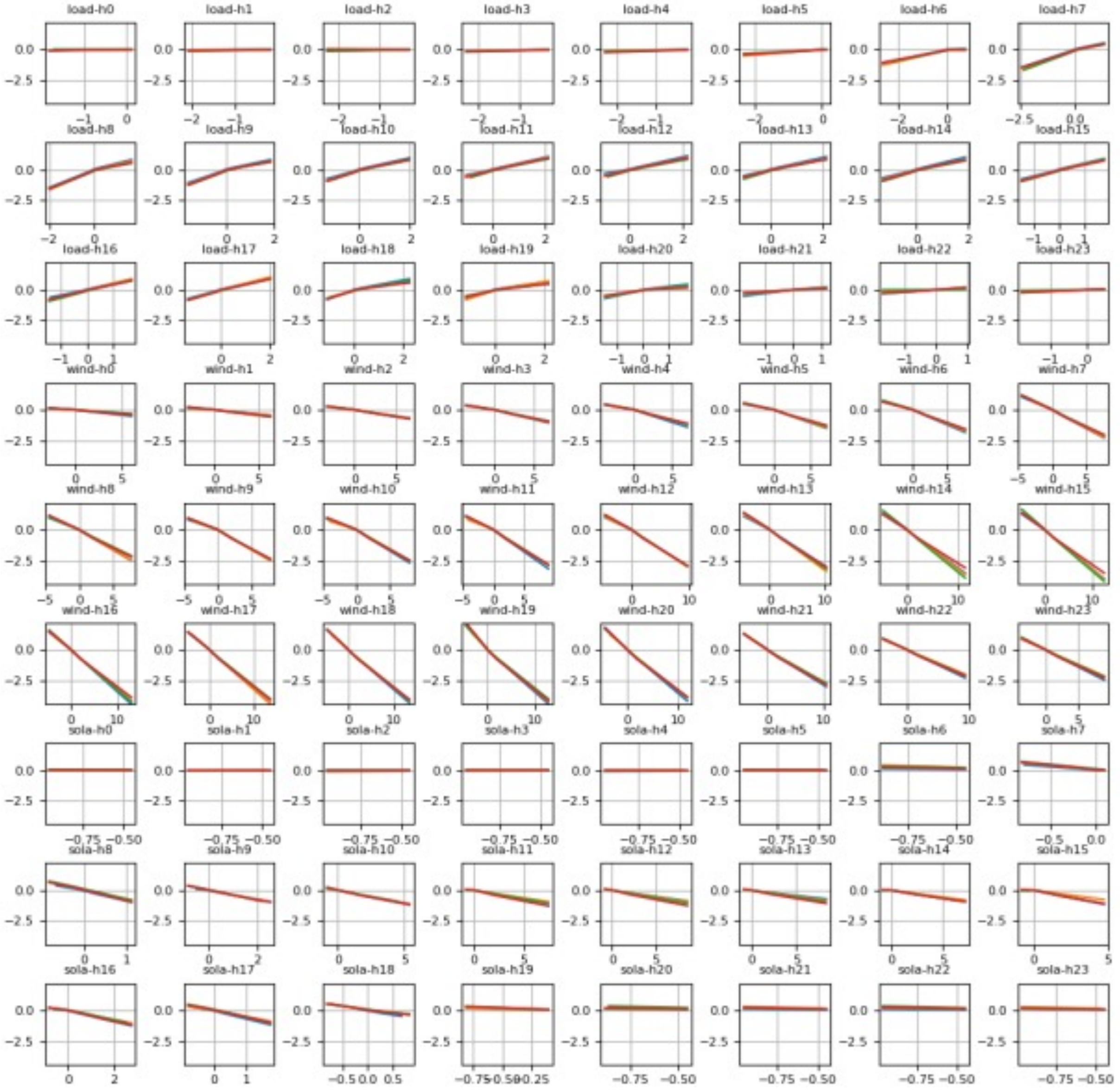


Revealing identified feature shape maps

DE – q0.05

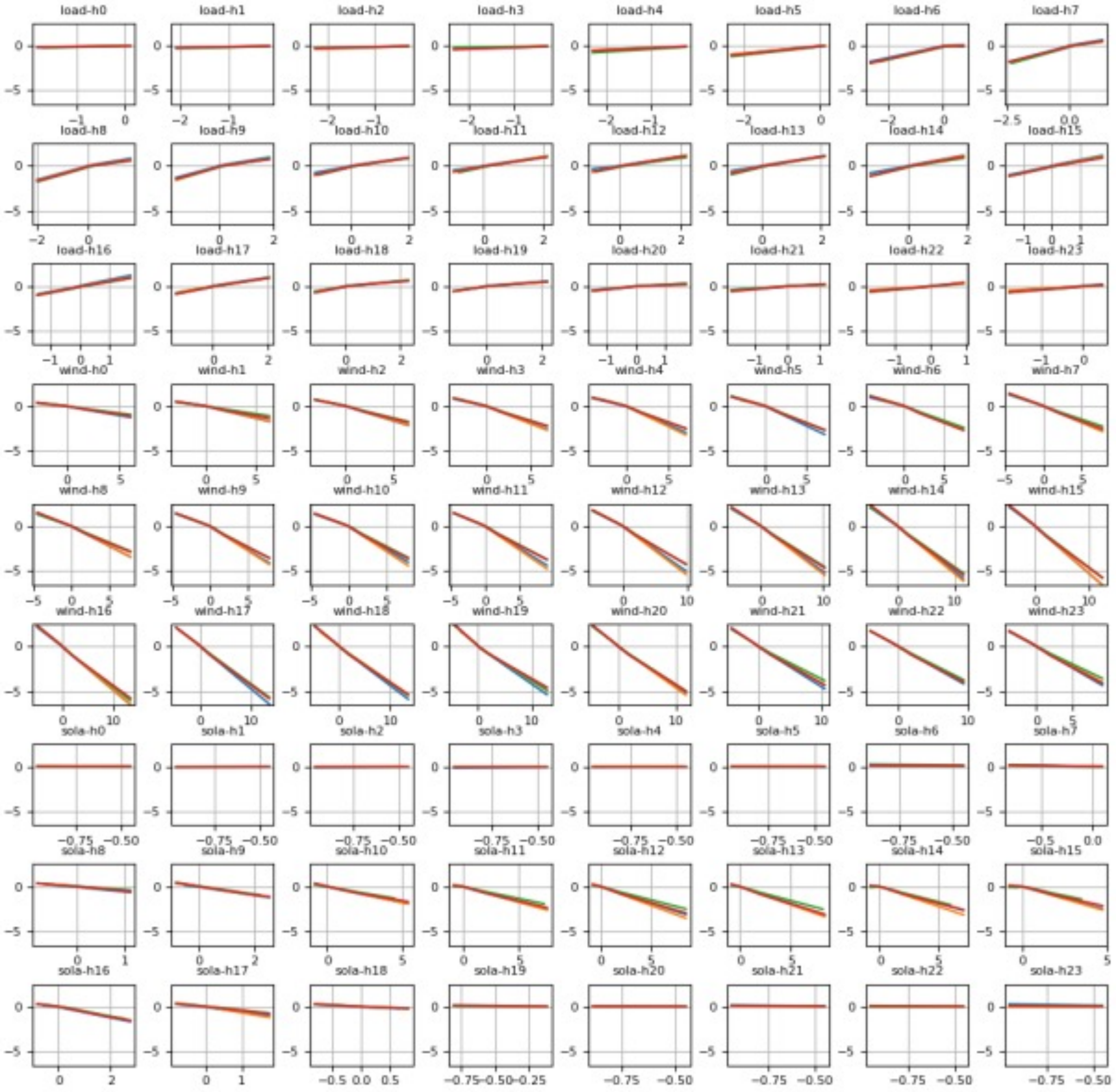


DE – q0.95

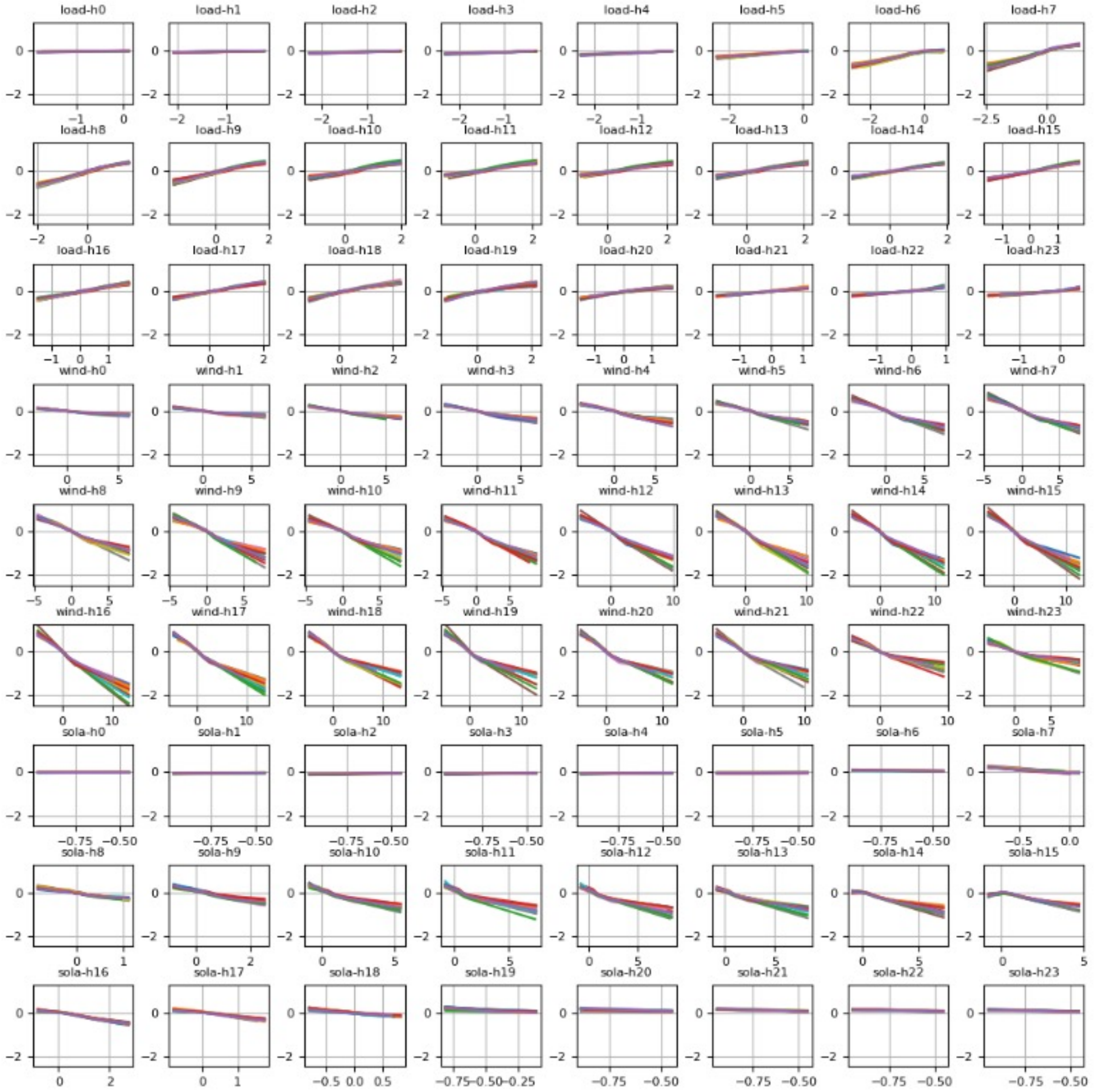


Revealing identified feature shape maps

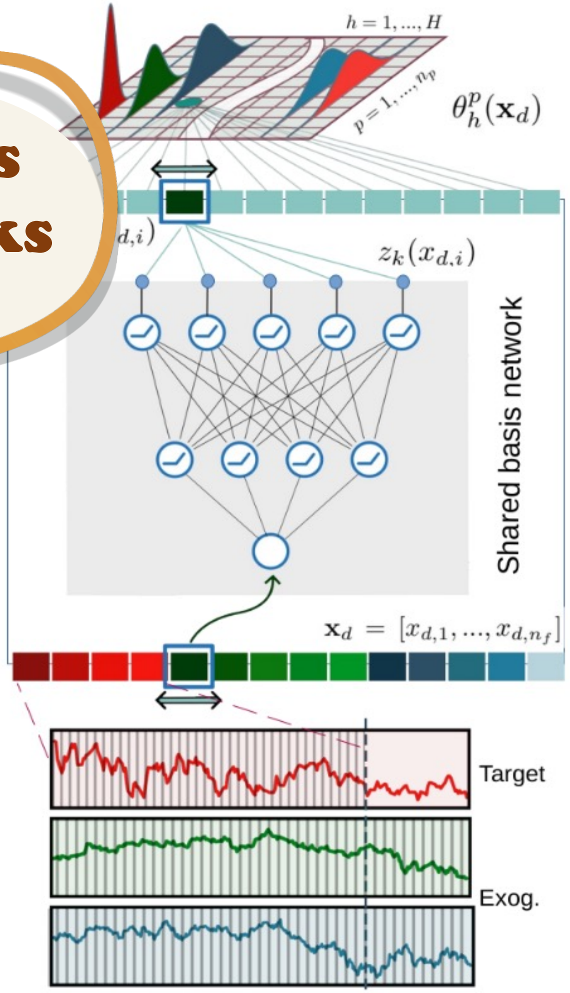
DE – q0.05



DE – JSU location

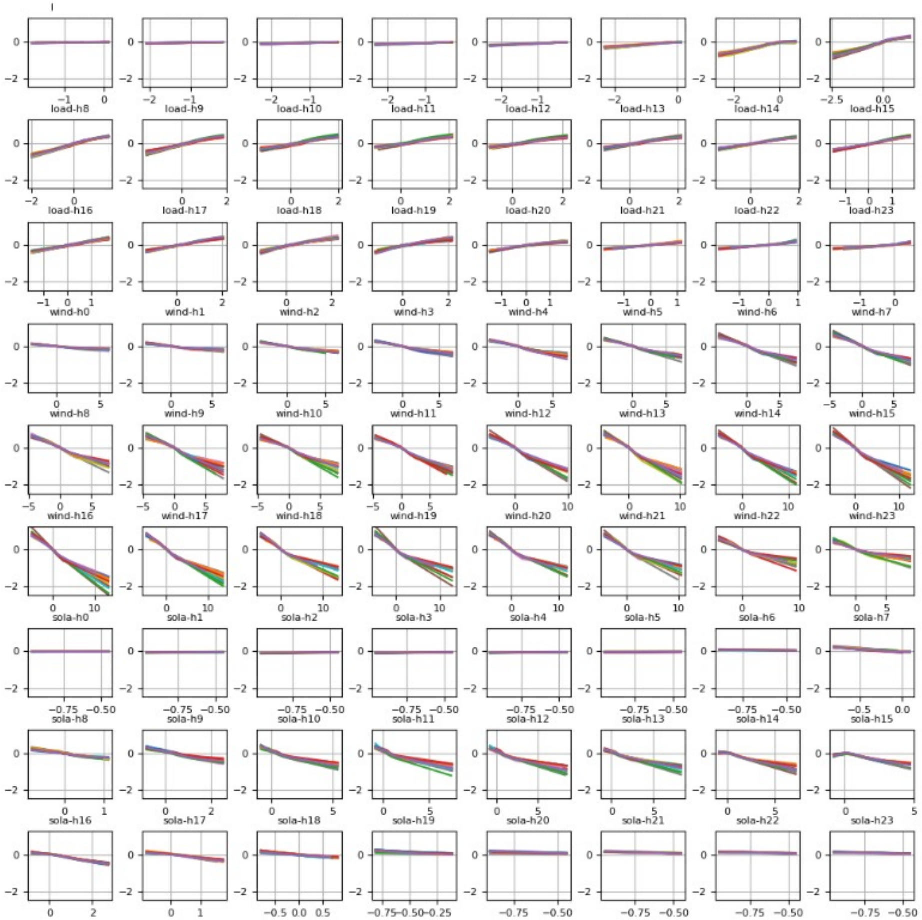


Revealing identified feature shape maps



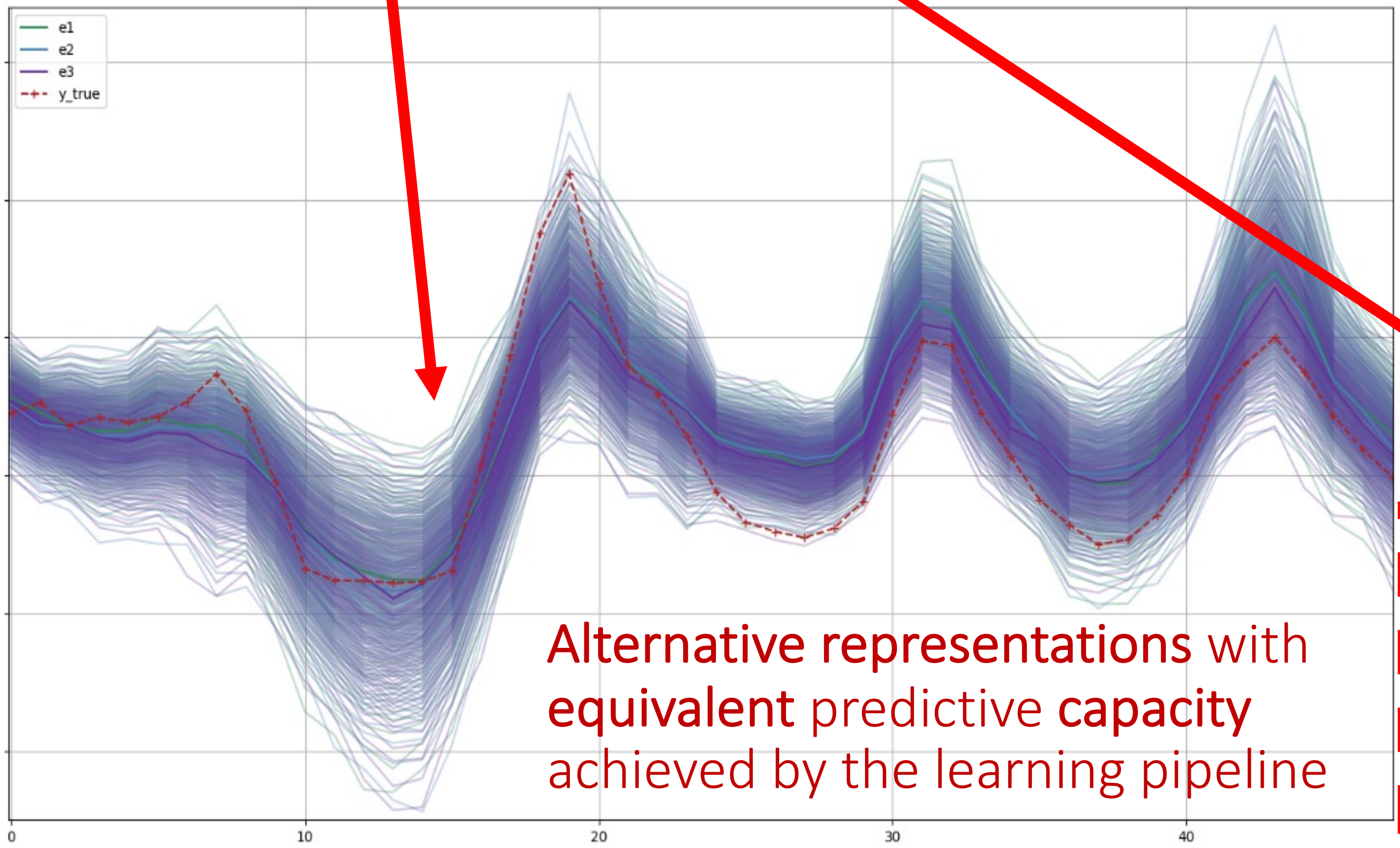
Gaining insights into what the NN is doing under the hood

Identified feature-out relations



Concurvity issue

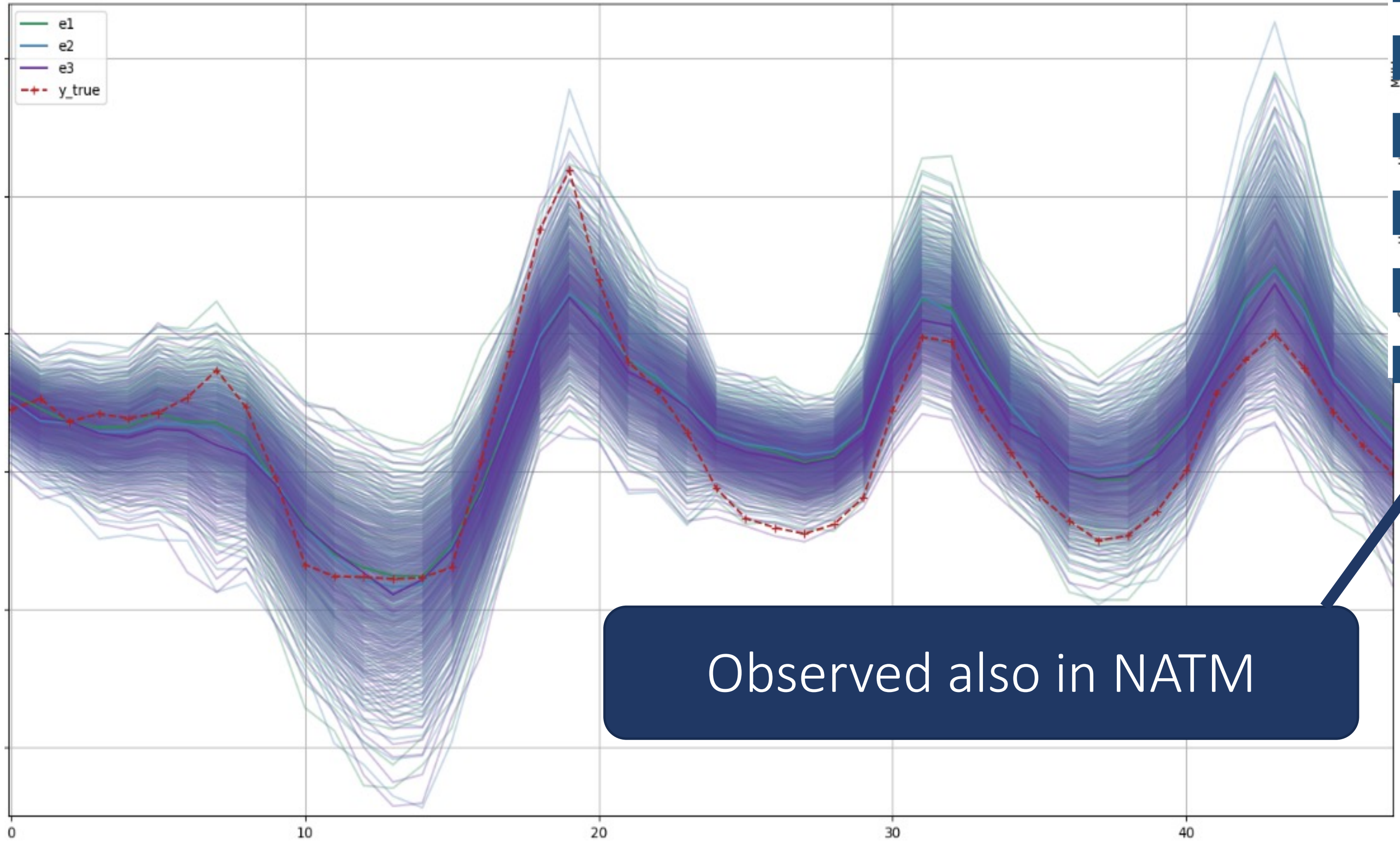
Heterogeneous feature maps providing **equal** predictions



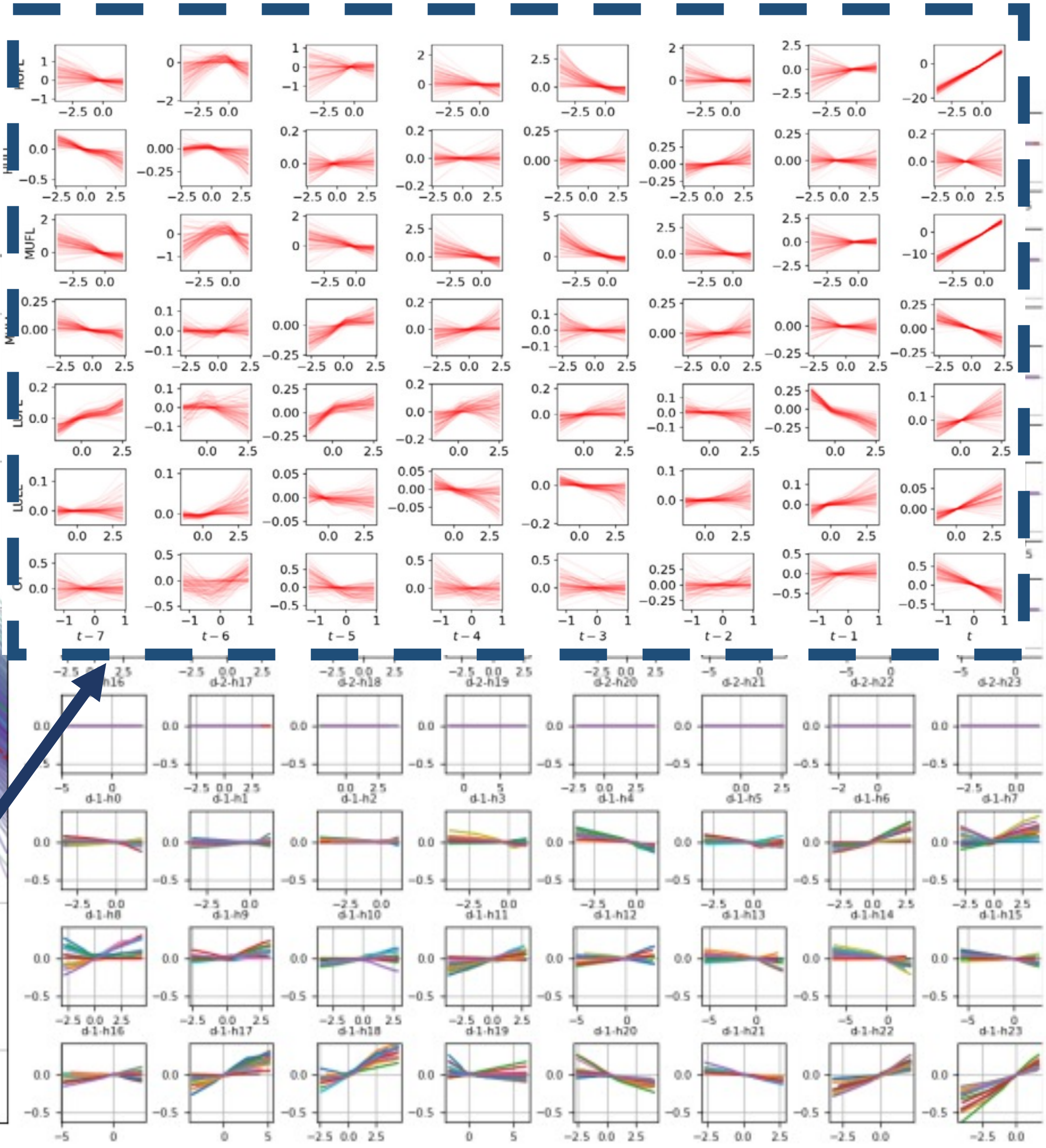
Alternative representations with **equivalent** predictive **capacity** achieved by the learning pipeline

Concurvity issue

Heterogeneous feature maps providing **equal** predictions



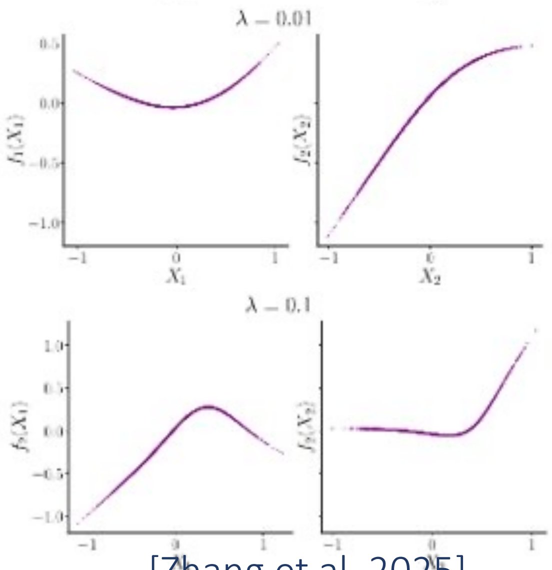
Observed also in NATM



Lags

Current state

- Concurvity **regularizers** to enforce decorrelation
- **NL** shape functions **dependencies** still open issue
- Most **practical trade-off**: offering an ensemble of solutions, rather than a single candidate



[Zhang et al, 2025]

- NAMs can **complement** the **flexibility** of NNs (e.g., hybrid ensembles)
- Offering **insights** into the underlying feature's contribution across the domain
- **Supporting** NNs users during model **design** and assessment

Conclusions and next dev

- D/Q-NBM: **NN proxy** with additional **interpretability**
- Inspired by **GAMLSS/QGAM,** with **TF-GPU** deployment

Experiments on benchmark datasets covering multiple regions:

- Achieving PF performance **comparable to D/Q-NNs**
- Providing further **insights** into the model **behavior**

HORIZON-CL4-24

MEDUSA

Next developments:

- Application to further PF/distributional regression tasks
- Extensions: concurvity, 2nd order interactions, features sparsity, hybrid models

References

- Marcjasz G., Narajewski M., Weron R., Ziel F., Distributional neural networks for electricity price forecasting, Energy Economics, 2023
- G Woo, C Liu, A Kumar, C Xiong, S Savarese, D Sahoo, Unified Training of Universal Time Series Forecasting Transformers, ICML, 2024
- A Brusaferri, A Ballarino, L Grossi, F Laurini, On-line conformalized neural networks ensembles for probabilistic forecasting of day-ahead electricity prices, Applied Energy, 2025
- R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, G. E. Hinton, Neural additive models: interpretable machine learning with neural nets, NIPS, 2021
- A. Thielmann, R.-M. Kruse, T. Kneib, B. Säfken, Neural additive models for location scale and shape: A framework for interpretable neural regres- sion beyond the mean, PMLR, 2024
- S Hirsch, J Berrisch, F Ziel, Online Distributional Regression, arXiv, 2024
- W Jo, D Kim, Neural additive time-series models: Explainable deep learning for multivariate time-series prediction, Expert systems with applications, 2023
- L. Feddersen, C. Cleophas, Hierarchical Neural Additive Models for Interpretable Demand Forecasts, arXiv 2024
- F Radenovic, A Dubey, D Mahajan, Neural basis models for interpretability, NIPS, 2022
- T Kim, J Kim, Y Tae, C Park, JH Choi, J Choo, Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift, ICLR, 2022
- K. Aliyon, J. Ritvanen, Deep learning-based electricity price forecasting: Findings on price predictability and european electricity markets, Energy, 2024
- J.N. Siems, K. Ditschuneit, W. Ripken, A. Lindborg, M. Schambach, J. Otterbach, M. Genzel, Curve your Enthusiasm: Concurvity Regularization in Differentiable Generalized Additive Models, ICML2023
- X. Zhang , J. Martinelli, S.T. John, Challenges in interpretability of additive models, arXiv, 2025

Thanks

Consiglio Nazionale delle Ricerche

Contact: alessandro.brusaferri@cnr.it

www.linkedin.com/in/alessandro-brusaferri-a9548933/