# Conditional Distributional Learning with Non-crossing Quantile Network and applications

Hongtu Zhu, Ph.D.
Kenan Distinguished Professor
University of North Carolina at Chapel Hill

My collaborators: Guohao Shen (PolyU), Shuai Huang (UNC), Shikai Luo (Bytedance), and Chengchun Shi (LSE)

https://www.med.unc.edu/bigs2/

## Content

# Content

## Conditional Distributional Learning: What and Why?

**Objective:** Estimate the full conditional distribution P(Y | X = x), not just the mean.

**Motivations.**

- ❖ **Risk Management:** Value-at-Risk and Conditional VaR quantify tail losses.
- ❖ **Decision Support:** Design quantile-based policies (e.g., alarm thresholds).
- ❖ **Uncertainty Quantification:** Build prediction intervals with guaranteed coverage.

**Benefits:** Captures heteroskedasticity, multimodality, tail behavior, and complex noise structures.

**Applications:**

- ➢ **Finance:** Tail risk (VaR), portfolio optimization under distributional forecasts.
- ➢ **Healthcare:** Personalized survival distributions, disease progression quantiles.
- ➢ **Engineering:** Reliability analysis, failure time distributions, safety thresholds.
- ➢ **Ride-Sharing Platforms:** Supply and demand, matching, dispatching, price, and subsidies.

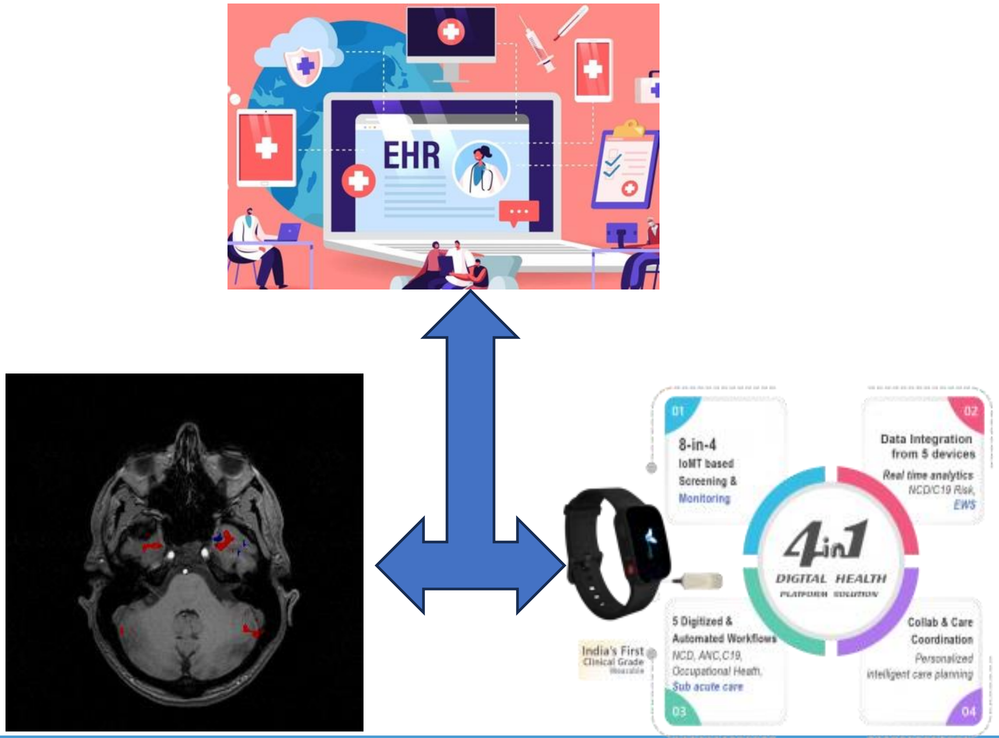
# Computational Challenges

## Major Approaches

- **Quantile Regression:** $\arg\min_{f \in \mathcal{F}} \mathbb{E}_{X,Y}[\rho_\tau(Y - f(X))]$ Techniques: linear, spline-based, deep networks with non-crossing constraints.
- **Distribution Regression:** Model the conditional CDF via classification or monotonic splines.
- **Conditional Density Estimation:** Mixture density networks & normalizing flows for flexible parametric/nonparametric densities.
- **Bayesian Nonparametrics:** GP conditional density models and Dirichlet process mixtures for adaptive tail modeling.

## Challenges

- **Quantile Crossing:** Enforce monotonicity across quantiles.
- **Complexity:** Reduce complexity in high-dimensional X via embeddings or sparsity.
- **Scalability:** Efficiently fit many quantiles or large flows on big data.
- **Calibration:** Validate predictive intervals (coverage, sharpness) on held-out data.



## Our Own References

- Li, T., Shi, C., Lu, Z., Li, Y., & Zhu, H. T. (2024). Evaluating dynamic conditional quantile treatment effects with applications in ridesharing. *Journal of the American Statistical Association, 119*(538), 1736–1750.
- Wu, G., Song, G., Lv, X., Luo, S., Shi, C., & Zhu, H. T. (2023). DNet: Distributional network for distributional individualized treatment effects. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.
- Sui, Y., Huang, Y., Zhu, H., & Zhou, F. (2022). Adversarial learning of distributional reinforcement learning. In *Proceedings of the International Conference on Machine Learning* (pp. 32783–32796).
- Zhou, F., Wang, J., & Feng, X. (2020). Non-Crossing Quantile Regression for Distributional Reinforcement Learning. In *Advances in Neural Information Processing Systems, 33* (pp. 55–65).
- Zhou, F., Lu, C., Tang, X., Zhang, F., Qin, Z., Ye, J., & Zhu, H. (2021). Multi-objective distributional reinforcement learning for large-scale order dispatching. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*(pp. 1541–1546).
- Zhou, F., Zhu, Z., Kuang, Q., & Zhang, L. (2021). Non-decreasing quantile function network with efficient exploration for distributional reinforcement learning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*.

## Content

## Question of Interest

*How to establish the theoretical properties of our Non-crossing Quantile Network from classical non-parametric quantile regression to distributional reinforcement learning (RL)?*

**Shen, G., Dai, R., Wu, G., Luo, S., Shi, C., and Zhu, H.. (2025). Deep Distributional Learning with Non-crossing Quantile Network. https://arxiv.org/abs/2504.08215**

# Background and Problem Formulation

- **Quantile Regression**

$$\arg\min_{f\in\mathcal{F}} \mathbb{E}_{X,Y}[\rho_\tau(Y - f(X))]$$

- **Distributional learning:**

- Let $\tau = (\tau_1, \ldots, \tau_K)$ be a non-decreasing sequence of K quantile levels.
- To estimate $Q_Y^{\tau_1}(x), \ldots, Q_Y^{\tau_K}(x)$ the corresponding conditional quantile functions

$$\sum_{k=1}^{K} \mathbb{E}_{X,Y}[\rho_{\tau_k}(Y - f_k(X))]$$

❓**Longstanding Crossing Quantiles Problem**

$$\hat{f}_1(x) \leq \hat{f}_2(x) \leq \ldots \leq \hat{f}_K(x)$$

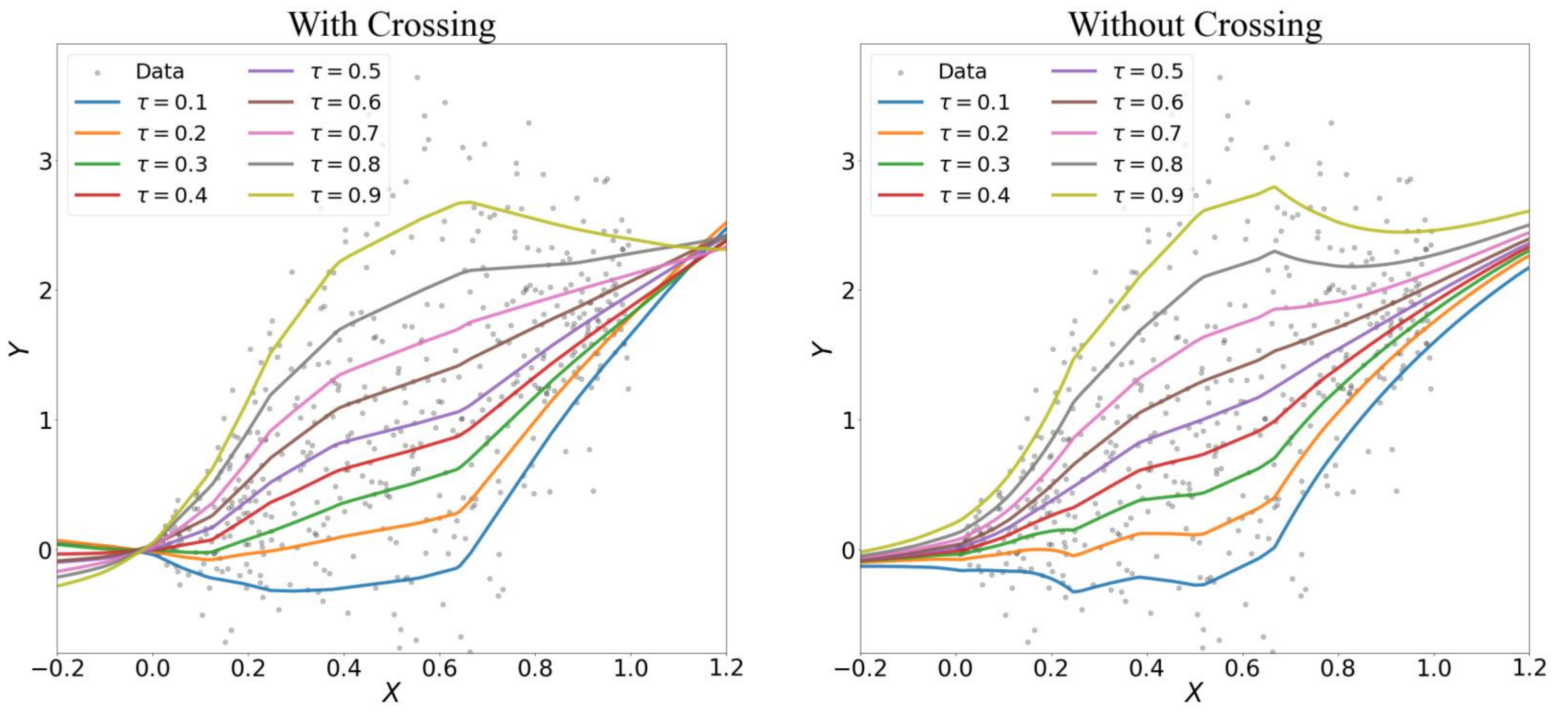


**Fig. 1**. A demonstration of quantile crossing on a simulated dataset. The estimated quantile curves at τ = 0.1, 0.2, . . . , 0.9 and the observations are depicted. The left panel presents the estimates from the deep quantile regression without any constraint and there appear crossings. The right figure presents our proposed quantile estimations with non-crossing constraints, and there is no crossing.

# Non-Crossing Quantile Networks

1. **One-Step:** our estimation of the quantile neural network is implemented in one-step without multiple iterative learning algorithms
2. **Non-crossing:** our quantile neural network can guarantee instead of encourage the estimated quantile curves are non-crossing
3. **Theory and Optimal Rate:** our quantile neural network is supported by learning theory and achieves minimax optimal rate of convergence for non-parametric estimation
4. **Low-dimensional Result:** our quantile neural network can adapt to low-dimensional structures of data to achieve faster convergence rate.

**Table 1.** A comparison of recent results on quantile neural networks.

| Paper | One-step | Non-cross | Theory | Low-dim Result | Optimal Rate |
|---|---|---|---|---|---|
| Zhou et al. (2020) | ✓ | ✓ | ✗ | ✗ | ? |
| Padilla et al. (2022) | ✓ | ✓ | ✗ | ✗ | ? |
| Yan et al. (2023) | ✗ | ✓ | ✗ | ✗ | ? |
| Shen et al. (2024) | ✓ | ? | ✓ | ✗ | ✗ |
| This paper | ✓ | ✓ | ✓ | ✓ | ✓ |

# Non-Crossing Quantile Networks

- **Mean network:** $v(\cdot;\theta)$
- **Gaps network:** $\sigma(g(\cdot;\theta))$
  - Pre-activated Gaps $g = (g_1, \ldots, g_K)$ can take negative values
  - Apply $\sigma(x) = ELU(x) + 1$ to get non-negative gaps $\sigma(g)$ where

$$ELU(x) = I(x \geq 0) \cdot x + I(x < 0) \cdot (\exp(x) - 1)$$

- **Final Output:** $f = (f_1, \ldots, f_K)$

$$f_k = v - \bar{g} + \sum_{i=1}^{k} \sigma(g_i) \quad \bar{g} = \frac{1}{K} \sum_{j=1}^{K} (K + 1 - j) \cdot \sigma(g_j)$$

- Verifies: Mean $v = K^{-1} \sum_{k=1}^{K} f_k$

  Gaps $\sigma(g_{k+1}) = f_{k+1} - f_k$

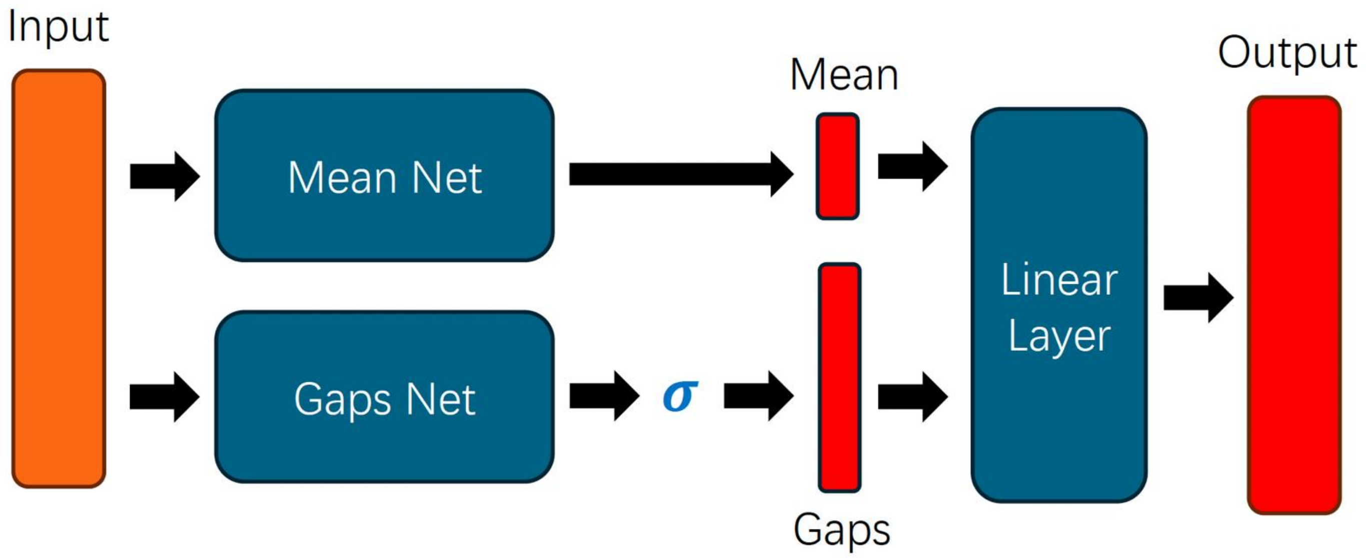


**Fig. 2**. A graphical illustration of the Non-Crossing Quantile Network. The "Mean Net" aims to learn the average of all quantiles, and the "Gaps Net" aims to learn the differences between adjacent quantiles.

# Non-Crossing Quantile Networks

➢ **We use the figures to show how to formulate non-crossing estimation of quantiles.**

- Output of a neural network: Pre-activated Gaps
- Apply the activation function $\sigma(x) = ELU(x) + 1$
- Calculate the cumsum for non-crossing quantiles.
- Center the outputs

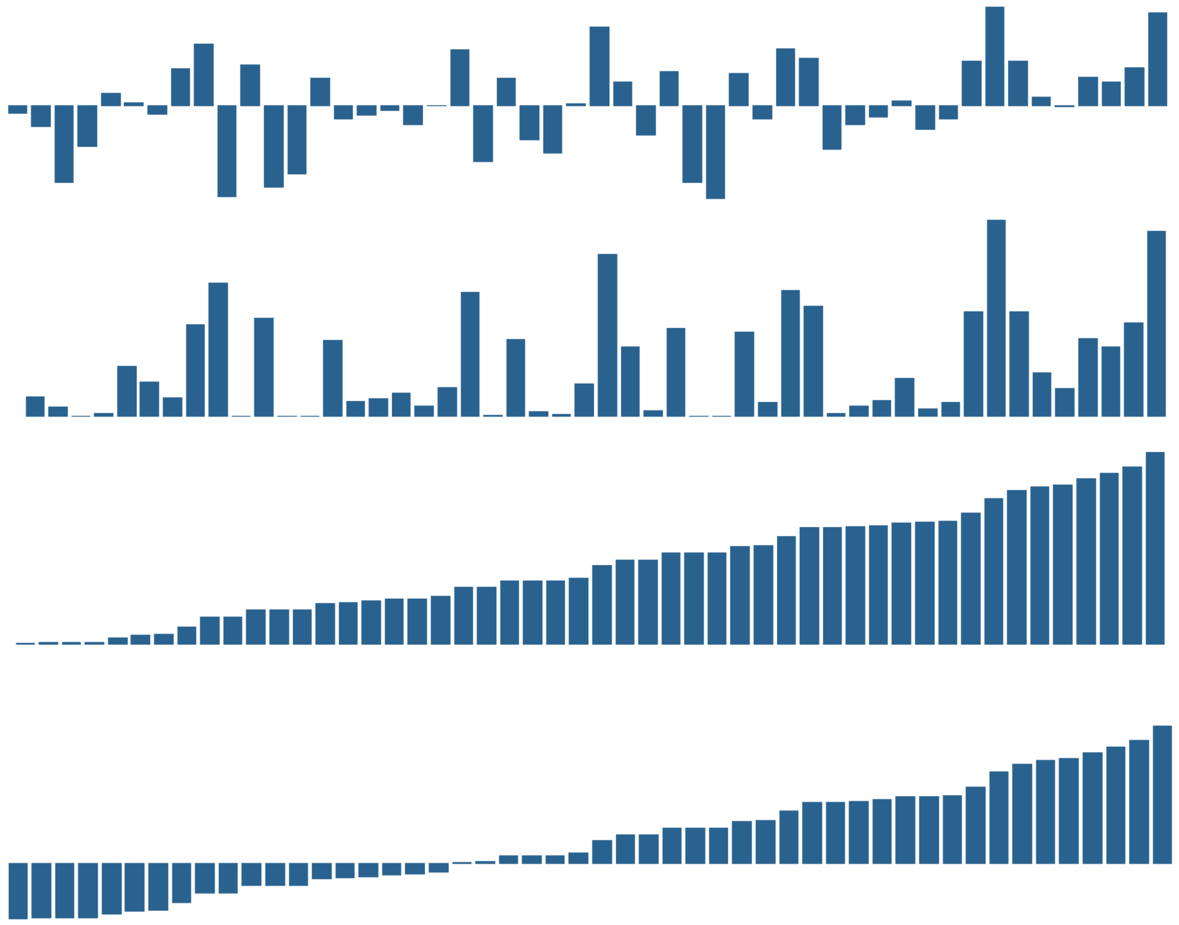


**Fig. 3**. How to formulate non-crossing estimation of quantiles

# Training of NQ Networks

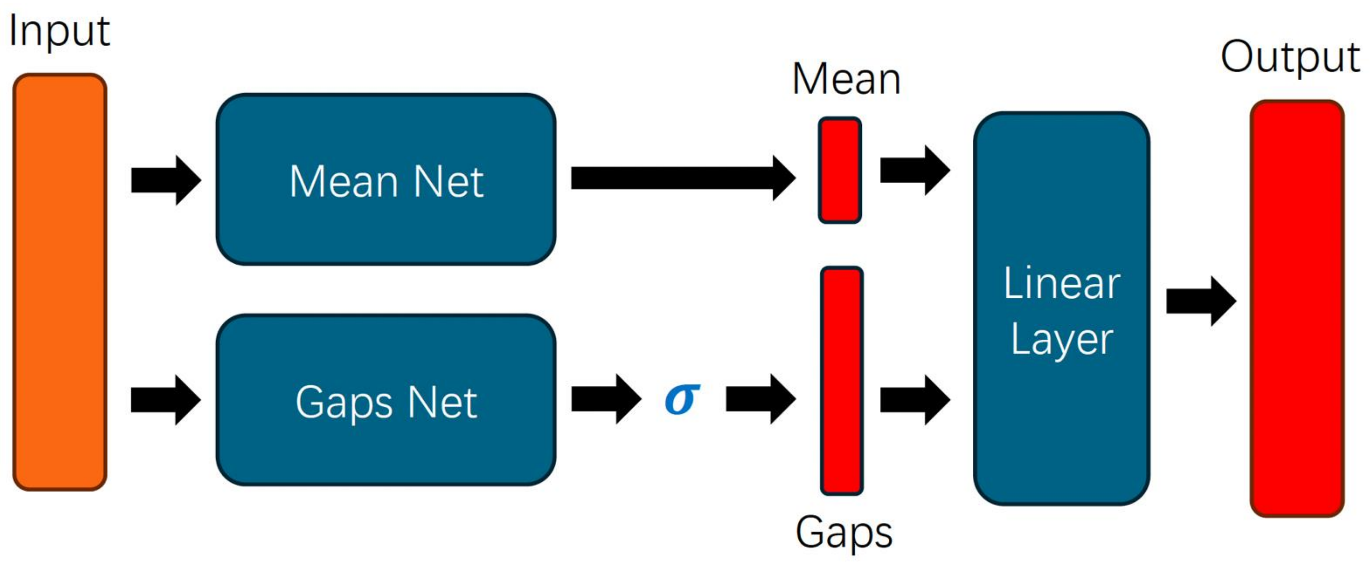
- **Implement NQ network as a feedforward ReLU network** $f : \mathbb{R}^{d_0} \to \mathbb{R}^{K+1}$
- **Mean:** $v(\cdot;\theta)$

  first output coordinate
- **Raw Gaps scores:** $g = (g_1, \ldots, g_K)$

  remaining K coordinates
- **Class of NQ networks:** $\mathcal{F}_N := \mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{U},\mathcal{S},\mathcal{B}}$

- Width $\mathcal{W}$: maximum number of neurons in the hidden layers
- Depth $\mathcal{D}$ : number of hidden layers
- Size $\mathcal{S}$ : total number of parameters (weights and biases)
- Number of Neurons $\mathcal{U}$
- Bound $\mathcal{B}$ : for each entry of NC network output

**ReLU-activated feedforward neural network**



- **Empirical risk for a NQ network f:**

$$\mathcal{L}_N(f) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{k=1}^{K}\rho_{\tau_k}(Y_i - f_k(X_i))$$

- **Empirical risk Minimizer (ERM):** $\hat{f}_N = \arg\min_{f\in\mathcal{F}_N} \mathcal{L}_N(f)$
- **Risk Minimizer (Target, ground truth):**

$$Q_Y := (Q_Y^{\tau_1}, \ldots, Q_Y^{\tau_K}) = \arg\min \mathbb{E}_{X,Y}[K^{-1}\sum_{k=1}^{K}\rho_{\tau_k}(Y - f_k(X))]$$

# Learning Guarantees

📌 **Target: Excess risk** $\mathcal{R}(\hat{f}_N) := \mathcal{L}(\hat{f}_N) - \mathcal{L}(Q_Y)$

🔷 **Reason:**

**Positivity:** Since $\mathcal{L}(Q_Y) \leq \mathcal{L}(f)$ for any estimator f.

**Self-calibration:** $\mathcal{R}(\hat{f}_N)$ towards 0 implies $\hat{f}_N$ towards $Q_Y$

💡**Error Decomposition** $\mathcal{R}_N(f) := \mathcal{L}_N(f) - \mathcal{L}_N(Q_Y)$

$$\mathbb{E}[\mathcal{R}(\hat{f}_N)] \leq \mathbb{E}[\mathcal{R}(\hat{f}_N) - 2\mathcal{R}_N(\hat{f}_N)] + 2\inf_{f\in\mathcal{F}_N} \mathcal{R}(f),$$

**Variance (or Stochastic error):**

$$\mathbb{E}[\mathcal{R}(\hat{f}_N) - 2\mathcal{R}_N(\hat{f}_N)]$$

**Bias (or Approximation error):**

$$2\inf_{f\in\mathcal{F}_N} \mathcal{R}(f)$$

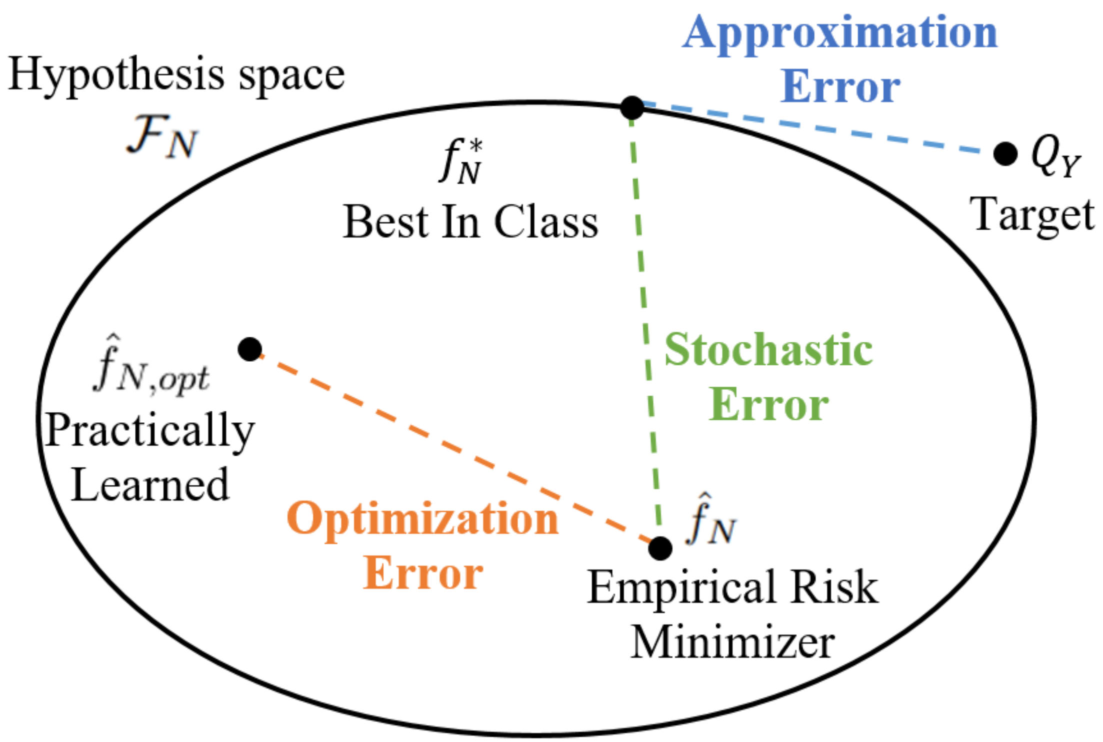

Hypothesis space $\mathcal{F}_N$
$f_N^*$ Best In Class
Approximation Error
$Q_Y$ Target
Stochastic Error
$\hat{f}_{N,opt}$ Practically Learned
Optimization Error
$\hat{f}_N$ Empirical Risk Minimizer

**Fig. 4.** An error decomposition of empirical risk minimization problems

## Assumptions

### Assumption 1: Smooth Target quantiles

ASSUMPTION 1. *(i) The target quantile curves $Q_Y = (Q_Y^{\tau_1}, \dots, Q_Y^{\tau_K})$ are $\beta$-Hölder smooth with constant $B$. (ii) The probability distribution of covariate $X$ with domain $\mathcal{X} = [0,1]^d$ is absolutely continuous with respect to the Lebesgue measure.*

### Assumption 2: Local Quadratic Bound of The Excess Risk

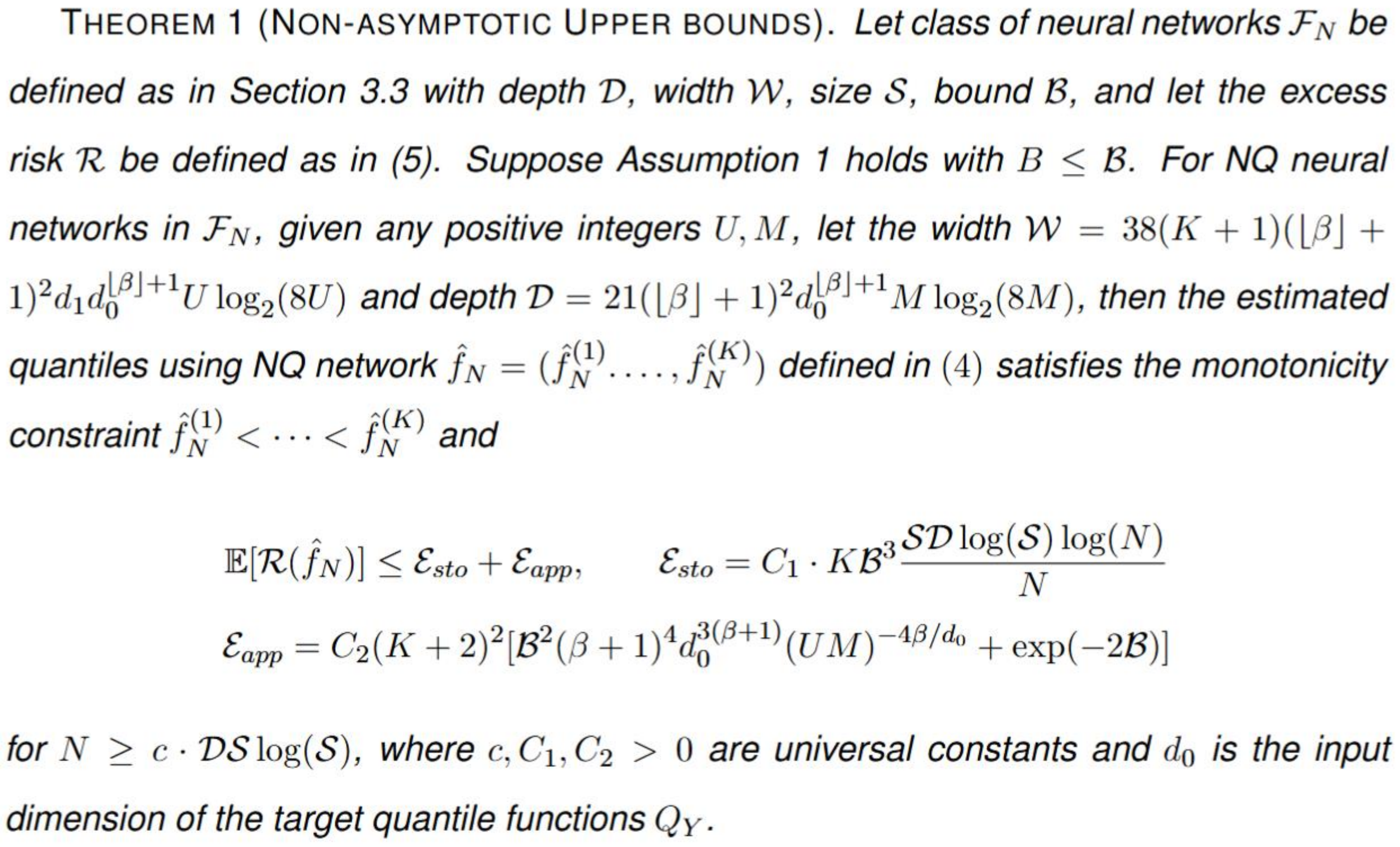
ASSUMPTION 2 (LOCAL QUADRATIC BOUND OF THE EXCESS RISK). *There exist some constants $c^0 = c^0(X,Y) > 0$ and $\delta^0 = \delta^0(X,Y) > 0$ such that*

$$\mathcal{R}(f) - \mathcal{R}(Q_Y) \leq \frac{c^0}{K} \sum_{k=1}^{K} \mathbb{E}\| f_{\tau_k}(X) - Q_Y^{\tau_k}(X) \|^2$$

*for any $f = (f_{\tau_1}, \dots, f_{\tau_K})$ satisfying $\| f_{\tau_k} - Q_Y^{\tau_k} \|_{L^\infty(\mathcal{X}^0)} \leq \delta^0$ for $k = 1, \dots, K$, where $\mathcal{X}^0$ is any subset of $\mathcal{X}$ such that $\mathbb{P}(X \in \mathcal{X}^0) = \mathbb{P}(X \in \mathcal{X})$.*

## Main Results

### Theorem 1 (Non-Asymptotic Upper Bound)



THEOREM 1 (NON-ASYMPTOTIC UPPER BOUNDS). *Let class of neural networks $\mathcal{F}_N$ be defined as in Section 3.3 with depth $\mathcal{D}$, width $\mathcal{W}$, size $\mathcal{S}$, bound $\mathcal{B}$, and let the excess risk $\mathcal{R}$ be defined as in (5). Suppose Assumption 1 holds with $B \leq \mathcal{B}$. For NQ neural networks in $\mathcal{F}_N$, given any positive integers $U, M$, let the width $\mathcal{W} = 38(K+1)(\lfloor\beta\rfloor+1)^2 d_1 d_0^{\lfloor\beta\rfloor+1} U \log_2(8U)$ and depth $\mathcal{D} = 21(\lfloor\beta\rfloor+1)^2 d_0^{\lfloor\beta\rfloor+1} M \log_2(8M)$, then the estimated quantiles using NQ network $\hat{f}_N = (\hat{f}_N^{(1)}, \ldots, \hat{f}_N^{(K)})$ defined in (4) satisfies the monotonicity constraint $\hat{f}_N^{(1)} < \cdots < \hat{f}_N^{(K)}$ and*

$$\mathbb{E}[\mathcal{R}(\hat{f}_N)] \leq \mathcal{E}_{sto} + \mathcal{E}_{app}, \qquad \mathcal{E}_{sto} = C_1 \cdot K\mathcal{B}^3 \frac{\mathcal{S}\mathcal{D}\log(\mathcal{S})\log(N)}{N}$$

$$\mathcal{E}_{app} = C_2(K+2)^2[\mathcal{B}^2(\beta+1)^4 d_0^{3(\beta+1)}(UM)^{-4\beta/d_0} + \exp(-2\mathcal{B})]$$

*for $N \geq c \cdot \mathcal{D}\mathcal{S}\log(\mathcal{S})$, where $c, C_1, C_2 > 0$ are universal constants and $d_0$ is the input dimension of the target quantile functions $Q_Y$.*

- **Stochastic Error: $\mathcal{E}_{sto}$**
  - Linear in sample size N
  - Square faster to those derived from standard concentration inequalities (Takeuchi et al., 2006;Sangnier et al., 2016; Mohri et al., 2018; Padilla et al., 2022; Shen et al., 2024)

- **Approximation Error: $\mathcal{E}_{app}$**
  - Differences from traditional results on quantile and robust regression (Lederer, 2020; Padilla et al., 2022; Shen et al., 2024)
  - Bias is proportional to $K^2$, number of quantile curves being estimated.
  - Error term exp(−2B) emerges due to used truncation technique to manage the unbounded preimage associated with ELU activation. Negligible if B increases appropriately

## Main Results

### Trade-off between Stochastic Error and approximation Error

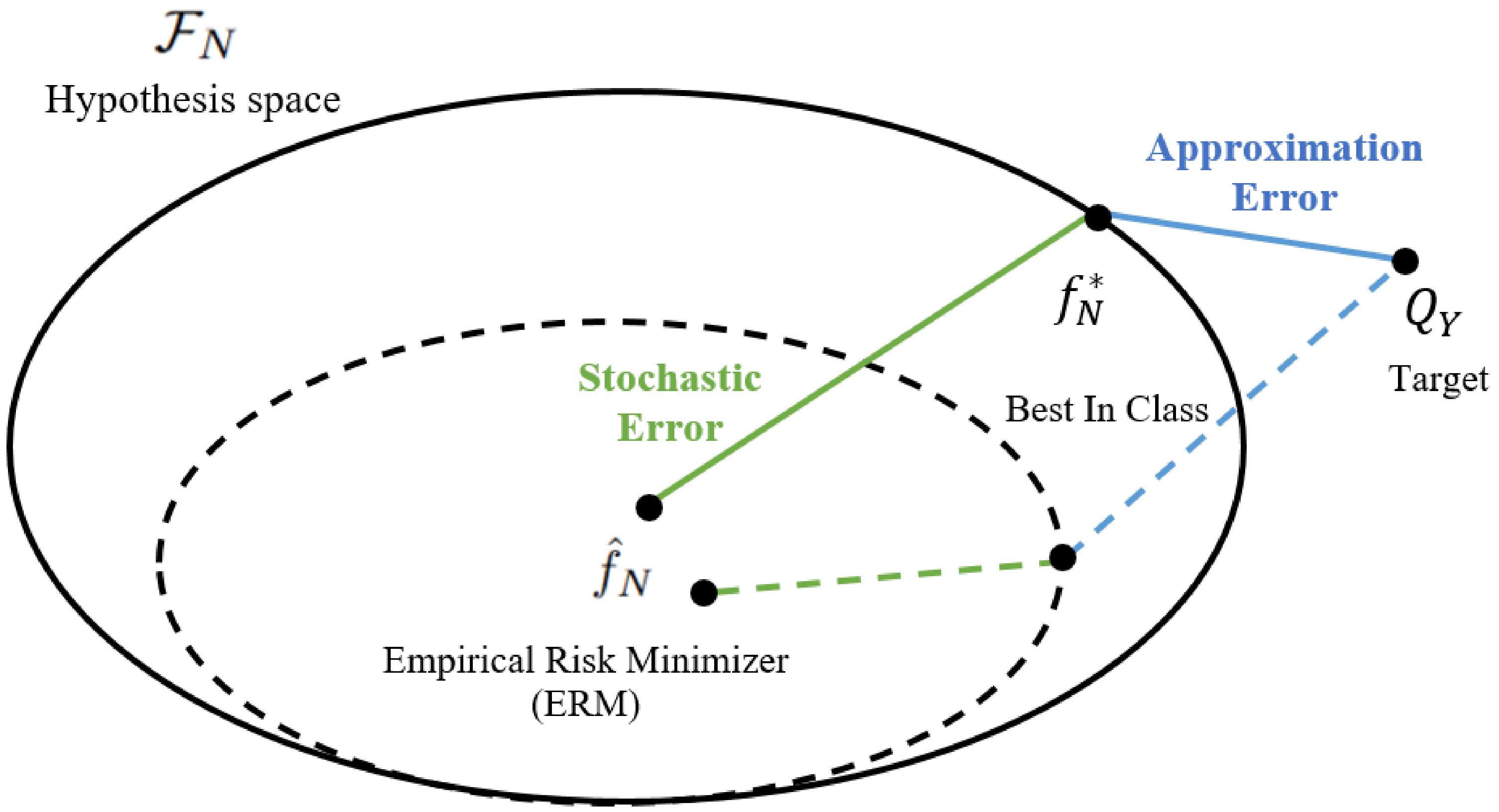


**Fig. 5**. between Stochastic Error and approximation Error

➢ **Stochastic Error:**

- Increasing in network size (depth, width and number of neurons)
- Decreasing in sample size N

➢ **Approximation Error:**

- Decreasing in network size (depth, width and number of neurons)
- Decreasing in the smoothness $\beta$ of the target quantiles $Q_Y$
- Increasing in input dimension $d_0$

➢ **Choose Proper Width and Depth:**

- With respect to N, $\beta$, $d_0$
- To optimize convergence rates

## Main Results

### Corollary 1 (Convergence Rate)

COROLLARY 1. *Suppose that the conditions in Theorem 1 hold. Let* $U = 1$, $M = N^{d_0/[2(d_0+2\beta)]}$ *and* $\mathcal{B} = \log(N)$. *Then the estimated quantiles* $\hat{f}_N = (\hat{f}_N^{(1)}, \dots, \hat{f}_N^{(K)})$ *defined in* (4) *satisfies the monotonicity constraint* $\hat{f}_N^{(1)} < \cdots < \hat{f}_N^{(K)}$ *and*

$$\mathbb{E}[\mathcal{R}(\hat{f}_N)] \leq C \cdot K^2 (\log N)^7 N^{-\frac{2\beta}{d_0+2\beta}},$$

*where* $C > 0$ *is a constant depending only on* $\beta$ *and* $d_0$.

- ➢ **Minimax optimal Rate:** (up to logarithms) for nonparametric regression (Stone, 1982)
- ➢ **Network Architecture:**
  - Choice of U = 1 and M = $N^{\gamma}$ for NQ network architecture is for parameter efficiency rather than as a strict requirement as network size scales faster with increasing width than with depth ($S \approx DW^2$).
  - Various network architectures, including those with varying widths and depths, can achieve the optimal rate as long as the total number of parameters in the network scales properly with the sample size.

# Main Results

Under self-calibration conditions (Padilla et al., 2022)

$$\sum_{k=1}^{K} \mathbb{E}|\hat{f}_N^{\tau_k}(X) - Q_Y^{\tau_k}(X)|^2 \leq C' \cdot K^2 (\log N)^7 N^{-\frac{2\beta}{d_0+2\beta}}$$

📌 **Curse of Dimensionality**

🔷 **In scenarios common to many machine learning tasks, where the input dimension $d_0$ is large, the convergence rate of the NQ network estimator slows markedly.**

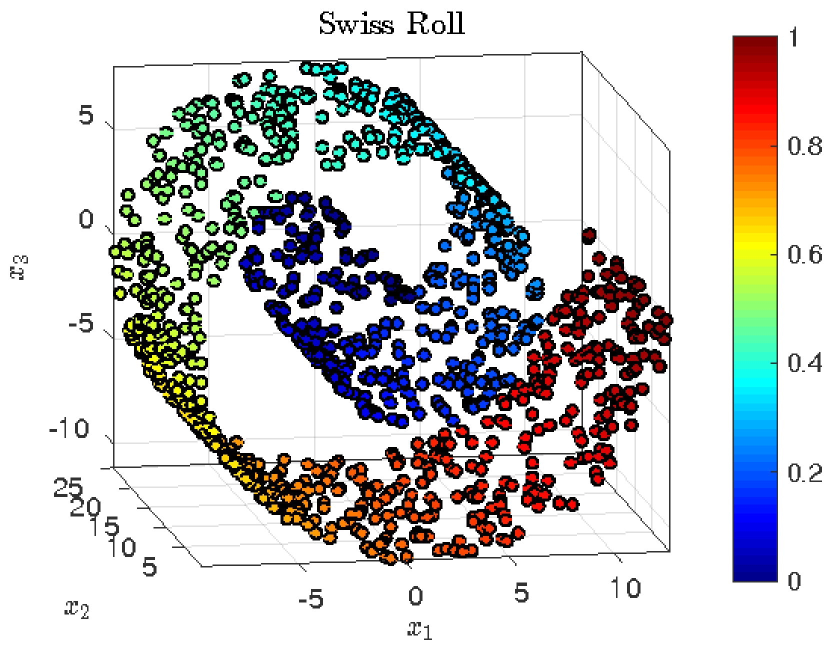
🔷 **Such a slow convergence rate needs a significantly larger sample size to achieve the desired theoretical accuracy, often proving impractical in real-world settings.**

## Main Results

### Assumption 3: Low-dimensional Support of Data (Manifold Hypothesis)

ASSUMPTION 3. *The covariate $X$ is supported on $\mathcal{M}_\rho$, a $\rho$-neighborhood of $\mathcal{M} \subset [0,1]^{d_0}$, where $\mathcal{M}$ is a compact $d_{\mathcal{M}}$-dimensional Riemannian sub-manifold and*

$$\mathcal{M}_\rho = \{x \in [0,1]^{d_0} : \inf\{\|x-y\|_2 : y \in \mathcal{M}\} \le \rho\},\ \rho \in (0,1).$$



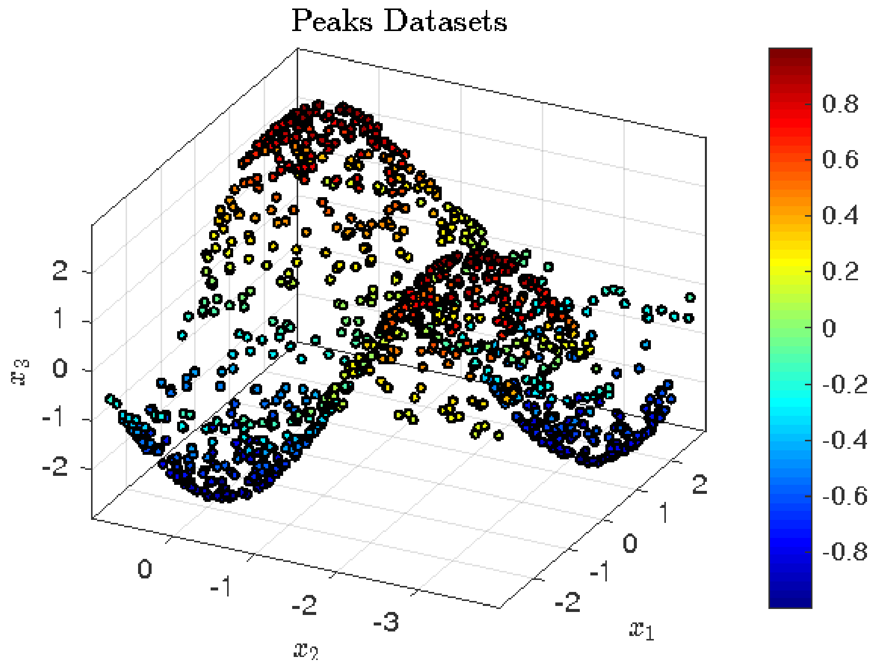


**Fig. 6**. An example of data with low-dimensional support.

## Main Results

### Corollary 2 (Improved Convergence Rate)

COROLLARY 2. *Suppose the conditions in Theorem 1 and Lemma 2 hold. Let* $U = 1$ *and* $M = N^{d_0^*/[2(d_0^*+2\beta)]}$. *Then the NQ network estimator* $\hat{f}_N = (\hat{f}_N^{(1)}, \dots, \hat{f}_N^{(K)})$ *defined in* $(4)$ *satisfies the monotonicity constraint* $\hat{f}_N^{(1)} < \cdots < \hat{f}_N^{(K)}$ *and*

$$\mathbb{E}[\mathcal{R}(\hat{f}_N)] \leq C \cdot K^2 (\log N)^7 N^{-\frac{2\beta}{d_0^*+2\beta}},$$

*where* $C > 0$ *is a constant depending only on* $\beta$, $d_0$ *and* $d_0^*$.

📌 **NQ Networks Mitigate the Curse of Dimensionality**

🔷 **The convergence rate adapts to the intrinsic dimension $d^*$ of the data, which can be significantly faster especially when $d^* \ll d_0$**

🔷 **The NQ networks can mitigate the curse of dimensionality**

# Applications to Deep Reinforcement Learning

## 📌 Markov decision process (MDP)

- $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces
- $R$ is the r.v. reward function

  $\gamma \in [0, 1)$ is the discount factor
- $P(s' \mid s, a)$ is the transition probability
- A policy $\pi(\cdot \mid s)$ maps each state $s \in \mathcal{S}$ to a distribution over $\mathcal{A}$
- For a fixed $\pi$, the return is a r.v. of the sum of discounted rewards observed along one trajectory of states while following $\pi$.

$$Z^{\pi} = \sum_{t=0}^{\infty} \gamma^t R_t$$

## 💡Problem definition

We want to estimate the distribution of $Z^{\pi}$ as well as get an optimal one $Z^{\pi^*}$ in the sense that $[E[Z^{\pi^*}] \geq EZ^{\pi}]$ for any $\pi$.

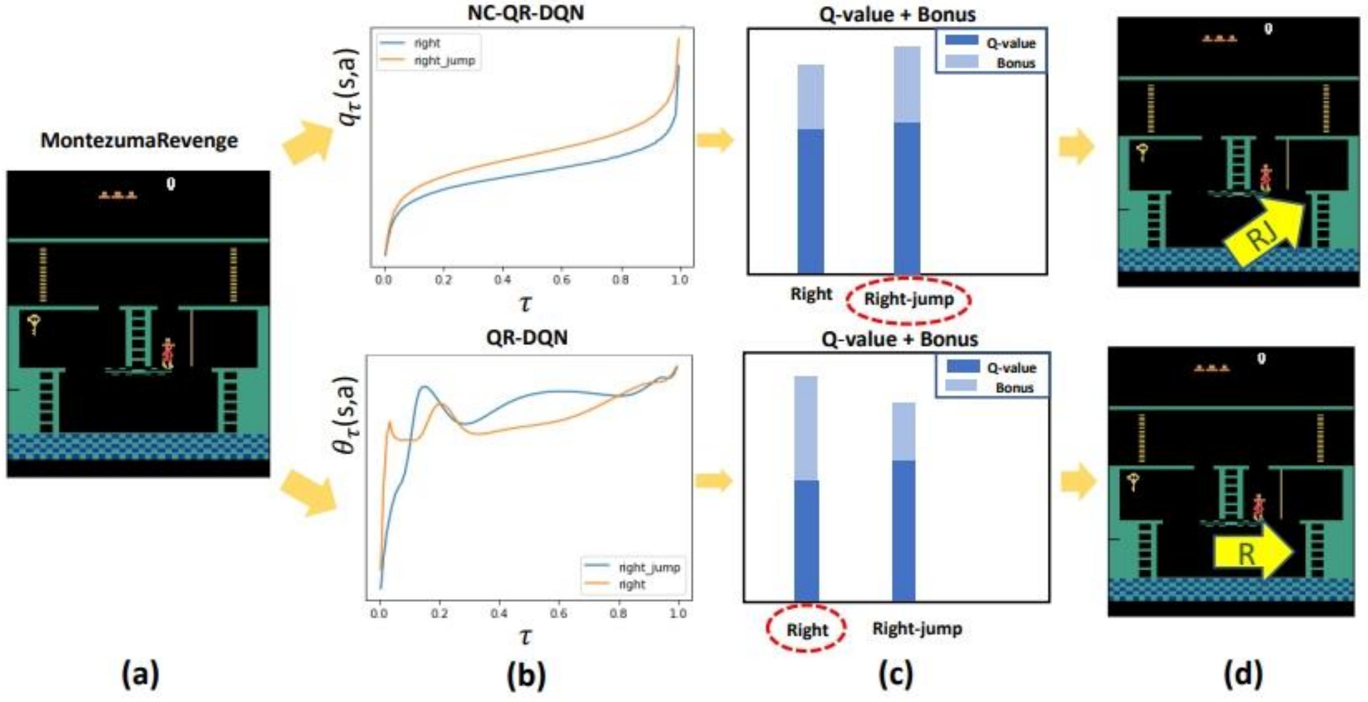


**Fig. 7**. An Atari example to show how the crossing issue may affect the exploration efficiency.

# Applications to Deep Reinforcement Learning

**Distributional Bellman optimality equation**

$$(\mathcal{T}Z)(s,a) = R(s,a) + \gamma Z(s',a')$$

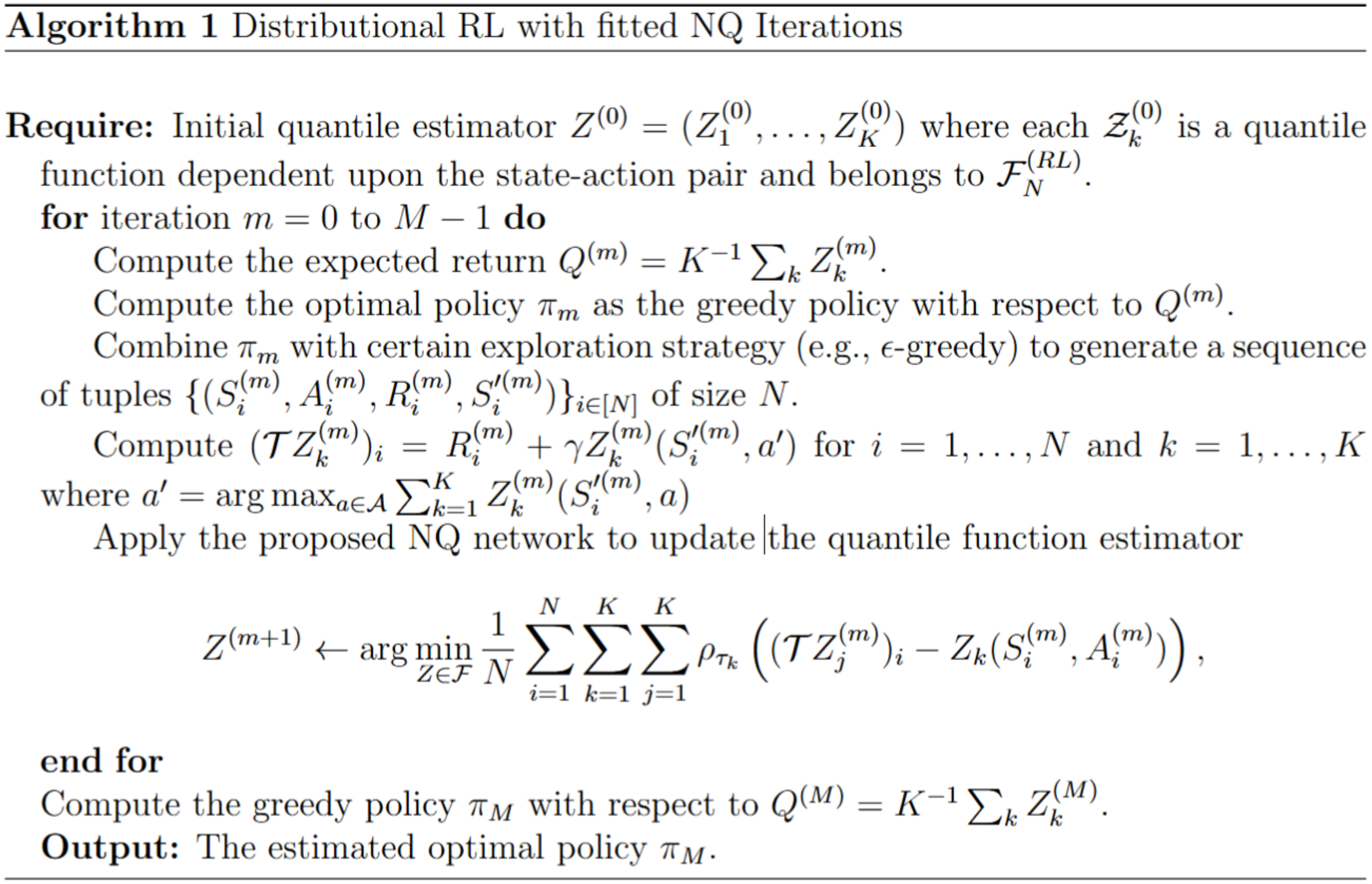
with $s' \sim P(\cdot \mid s,a)$ , $a' = \arg\max_{a \in \mathcal{A}} \mathbb{E}[Z(s',a)]$

- Quantile regression to represent the entire distribution of Z(s, a)
- NQ Network is iteratively trained to solve Bellman optimality equation
- In the m-th iteration, update NQ network using newly generated data

$$\{(S_i^{(m)}, A_i^{(m)}, R_i^{(m)}, S_i'^{(m)})\}_{i \in [N]}$$

- Modify the original NQ networks for the value distribution estimation

$$\mathcal{F}_N^{(RL)} = \{f : \mathcal{S} \times \mathcal{A} \to \mathbb{R} : f(\cdot, a) \in \mathcal{F}_N \text{ for any } a \in \mathcal{A}\}$$



**Algorithm 1** Distributional RL with fitted NQ Iterations

```
Require: Initial quantile estimator $Z^{(0)} = (Z_1^{(0)}, \dots, Z_K^{(0)})$ where each $Z_k^{(0)}$ is a quantile
  function dependent upon the state-action pair and belongs to $\mathcal{F}_N^{(RL)}$.
  for iteration $m = 0$ to $M - 1$ do
      Compute the expected return $Q^{(m)} = K^{-1} \sum_k Z_k^{(m)}$.
      Compute the optimal policy $\pi_m$ as the greedy policy with respect to $Q^{(m)}$.
      Combine $\pi_m$ with certain exploration strategy (e.g., $\epsilon$-greedy) to generate a sequence
  of tuples $\{(S_i^{(m)}, A_i^{(m)}, R_i^{(m)}, S_i'^{(m)})\}_{i \in [N]}$ of size $N$.
      Compute $(\mathcal{T}Z_k^{(m)})_i = R_i^{(m)} + \gamma Z_k^{(m)}(S_i'^{(m)}, a')$ for $i = 1, \dots, N$ and $k = 1, \dots, K$
  where $a' = \arg\max_{a \in \mathcal{A}} \sum_{k=1}^K Z_k^{(m)}(S_i'^{(m)}, a)$
      Apply the proposed NQ network to update the quantile function estimator
          $Z^{(m+1)} \leftarrow \arg\min_{Z \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K \sum_{j=1}^K \rho_{\tau_k}\left((\mathcal{T}Z_j^{(m)})_i - Z_k(S_i^{(m)}, A_i^{(m)})\right)$,
  end for
  Compute the greedy policy $\pi_M$ with respect to $Q^{(M)} = K^{-1} \sum_k Z_k^{(M)}$.
  Output: The estimated optimal policy $\pi_M$.
```

# Assumptions

## Assumption 4: Unbounded Reward with 1st+ moment

ASSUMPTION 4. *There exist some $p > 1$ and $0 < C_{p,R} < \infty$ such that $(\mathbb{E}|R(s,a)|^p)^{1/p} < C_{p,R}$ for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$.*

## Assumption 5: Smooth Target Function

ASSUMPTION 5. *For any $f \in \mathcal{F}_N^{(RL)}$ and any $a, a' \in \mathcal{A}$, the function $R_\tau(\cdot, a) + \gamma f(\cdot, a')$ is $\beta$-Hölder smooth with constant $B$, where $R_\tau(s,a)$ denotes the $\tau$-th conditional quantile of the reward given the state $s$ and action $a$.*

## Assumption 6: Self-Calibration

ASSUMPTION 6. *There exist constants $c, C > 0$ such that for any $|\delta| \le C$ and $m = 0, \dots, M-1$, it holds*

$$\left| P_m\left(\tilde{Z}_\tau^{(m)}(s+\delta, a) \middle| s, a\right) - P_m\left(\tilde{Z}^{(m)}(s,a) \middle| s, a\right)\right| \ge c|\delta|,$$

*for all $\tau \in (0,1)$, $s \in \mathcal{S}$, and $a \in \mathcal{A}$ up to a negligible set. Here $P_m(\cdot|s,a)$ denotes the conditional distribution function of $\tilde{Z}^{(m)} := \mathcal{T} Z^{(m)}$ and $\tilde{Z}_\tau^{(m)}(s', a')$ denotes the $\tau$th conditional quantile of $\tilde{Z}^{(m)}$ given state $s'$ and action $a'$.*

## 📌 Relaxation over Existing studies

🔷 We do not require the data to be **i.i.d.** like the existing literature (see e.g., Chen and Jiang, 2019; Fan et al., 2020; Li et al., 2021)

as it is often violated in MDPs due to the temporal dependence between the observations(Hao et al., 2021).

🔷 Nor do we require **stationarity, ergodicity, or mixing** conditions (see e.g., Shi et al., 2022; Ramprasad et al., 2023)

🔷 We only requires the reward function to have **bounded absolute moments of order p > 1**, no matter how closely p approaches 1. unlike those bounded (see e.g., Chen and Jiang, 2019; Fan et al., 2020;Shi et al., 2022; Ramprasad et al., 2023) or sub-Gaussian (see e.g., Rowland et al., 2023).

# Theoretical Results

THEOREM 2. *Suppose that Assumptions 4, 5, and 6 hold. Given sample size $N$, let the NQ networks in $\mathcal{F}_N^{(RL)}$ have depth $\mathcal{D} = 21(\lfloor\beta\rfloor + 1)^2(d_0)^{\lfloor\beta\rfloor+1}N^{d_0/[d_0+4\beta]}\log_2(8N^{d_0/[d_0+4\beta]})$, width $\mathcal{W} = 114(\lfloor\beta\rfloor + 1)^2(K+1)(d_0)^{\lfloor\beta\rfloor+1}$, and bound $\mathcal{B} = \log(N)$. Then the expected cumulative reward following $\pi_M$ (the greedy policy of $Z^{(M)}$) satisfies*

$$J(\pi^*) - J(\pi_M) \leq \frac{2c_M\gamma}{(1-\gamma)^2}|\mathcal{A}|(\log N)^4 N^{-2\beta/(4\beta+d_0)} + \frac{8\gamma^{M+1}}{(1-\gamma)^2}C_{p,R} + \frac{C_1 \times C_{p,R}}{(1-\gamma)K^{(p-1)/p}},$$

*where $C_1 > 0$ is a universal constant and $c_M > 0$ is the concentration coefficient (Chen and Jiang, 2019; Fan et al., 2020) depending on the distribution of data.*

- The expected cumulative reward $J(\pi) = \mathbb{E}[\pi(a \mid S_0)Z^\pi(S_0, a)]$
- Approximation error occurs since we use the average of K quantiles to approximate the mean.
- Algorithmic error converges to zero at a linear rate with respect to the number of fitted iterations M
- Estimation error intrinsically relates to distributional learning with quantile regression using NQ network.
- Estimation error dominates when iterations M and number of quantiles K are large, with rate $|\mathcal{A}|N^{-2\beta/(4\beta+d_0)}$

# Real-Data Experiments

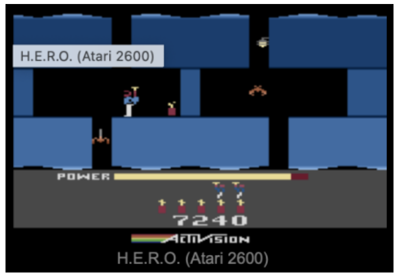
📌 **Evaluation on Atari 2600**

🔷 On six selected Atari game environments

🔷 To learn an optimal policy as a function of the snapshots of the game interface

🔷 Compare NQ-Net against NC-QR-DQN (Zhou et al., 2020) utilizing the same image-embedding network architecture and downstream networks with similar scales.

🔷 Specifically, we employ ReLU activation for the "Gaps net" in our model, denoted by NQ-Net*.



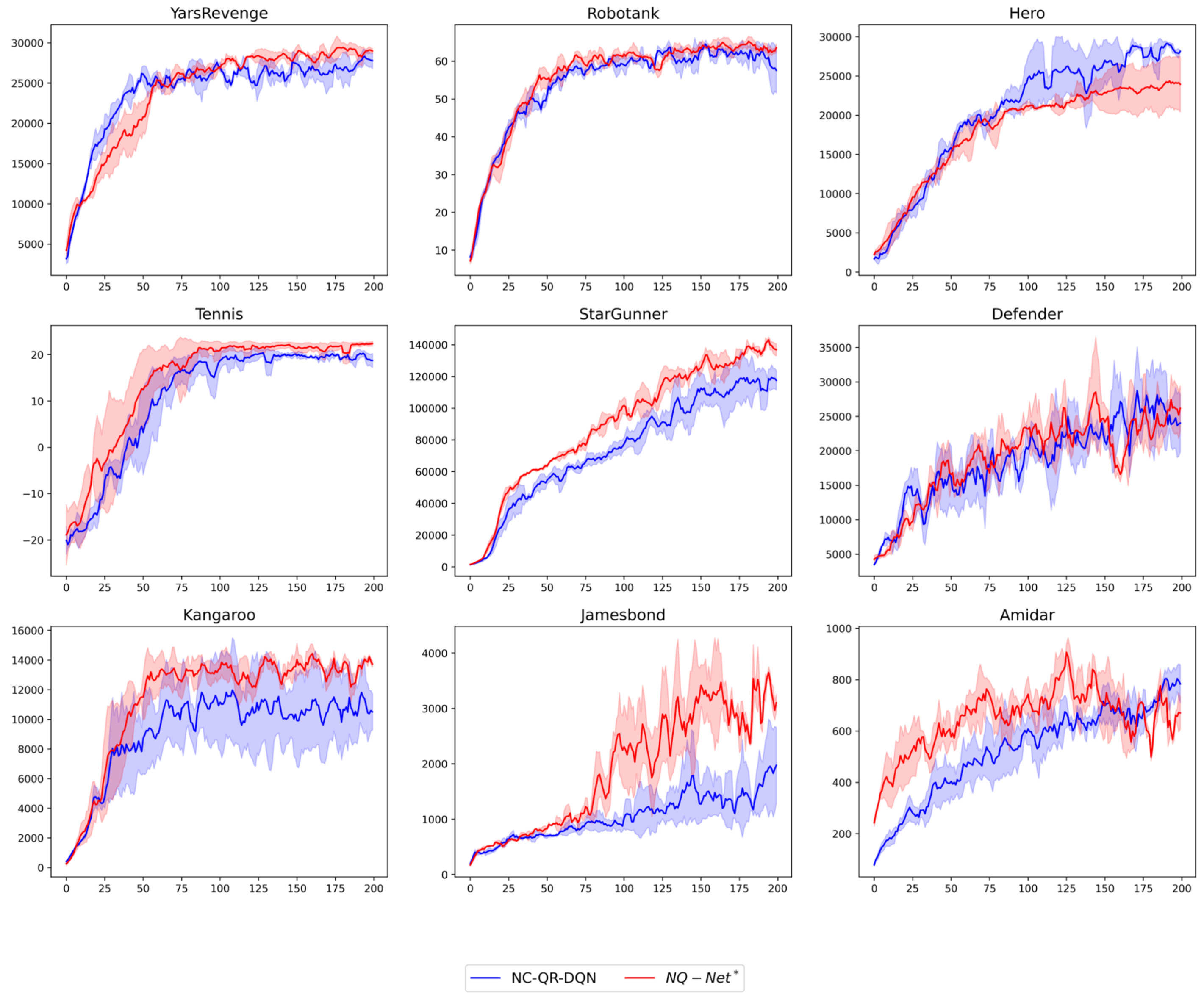


**Fig. 8**. Testing scores for NQ-Net* and NC-QR-DQN along the training process.

# Real-Data Experiments

**Evaluation on Atari 2600**

- On six selected Atari game environments
- Compare NQ-Net against NC-QR-DQN (Zhou et al., 2020) utilizing the same image-embedding network architecture and downstream networks with similar scales. We employ ReLU activation for the "Gaps net" in our model, denoted by NQ-Net*.
- Configuration

Number of quantiles K = 200

Sample size N=200million frames

Learning rate 5 × 10−5

Linear ϵ-greedy exploration strategy starting with ϵ = 1 at 0.5 million frames gradually decreasing to ϵ = 0.01 by 1 million frames

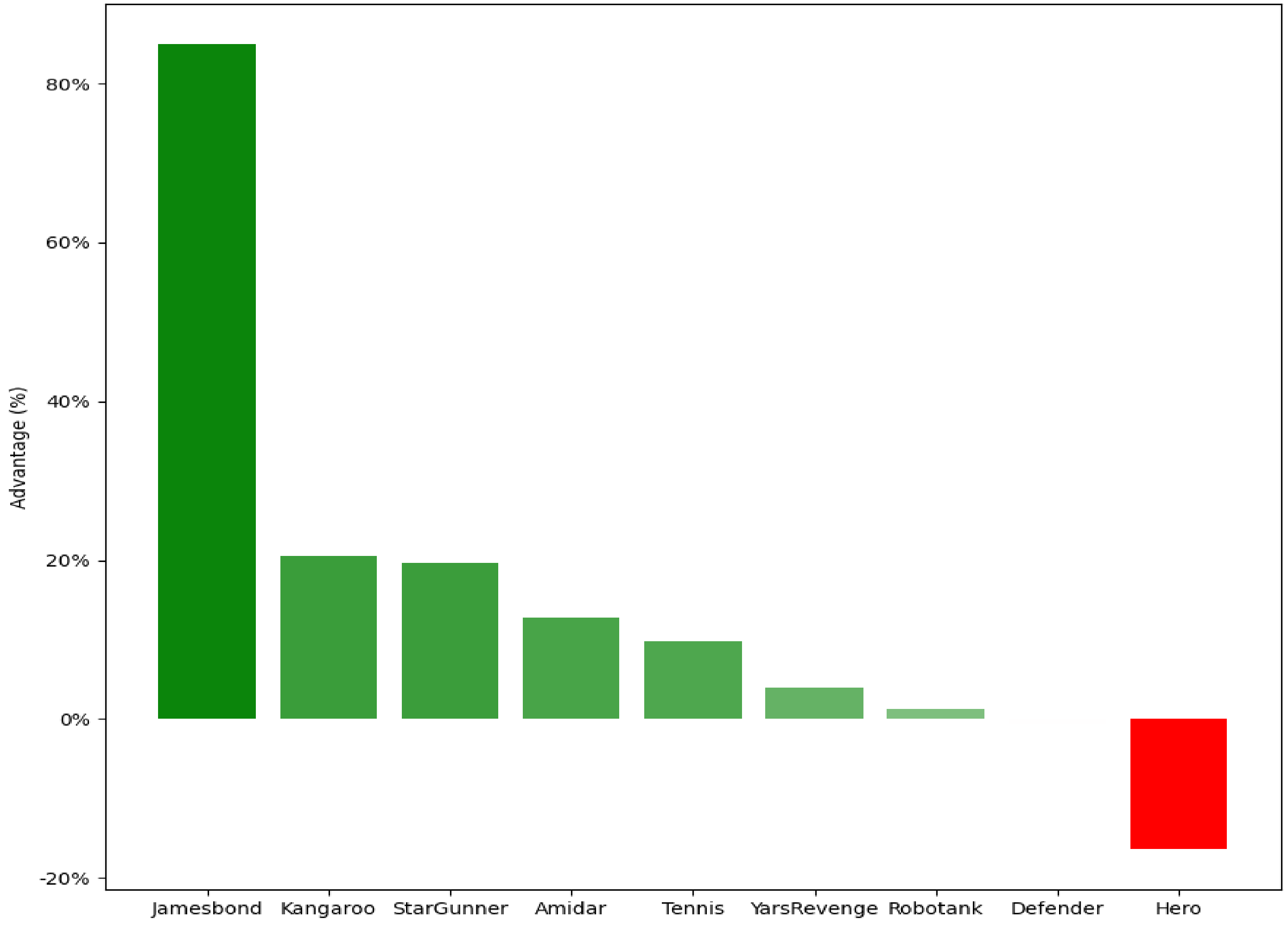


**Fig. 9**. The advantage of the best-performing NQ-Net* model over NC-QR-DQN. The advan-tage (%) on the y-axis is defined as (ScoreNQ − ScoreNC )/ScoreNC , where ScoreNQ and ScoreNC denote the highest testing scores achieved by the trained NQ-Net* and NC-QR-DQNmodels, respectively.

# Simulation Studies

## 📌 Comparison Methods

◆ **DQR:** Deep quantile regression (Padilla et al., 2022). As benchmark: estimates quantiles using ReLU NN without non-crossing constraints.

◆ **DQR*:** Extension of **DQR** with log(1 + exp(·)) activation for non-crossing multiple quantile estimation.

◆ **DQRP:** Deep quantile regression process using rectified quadratic unit networks (Shen et al.,2024). Estimates the quantile process with a non-crossing penalty. Implemented with tuning parameter $\lambda = \log(n)$ and $\xi \sim \mathrm{Unif}\,(0, 1)$.

◆ **NC-QR-DQN:** (Zhou et al., 2020) A non-crossing quantile network proposed for optimal policy learning in DRL.

◆ **NQ-Net:** The proposed method.

## 📌 Training and Testing

◆ **Training Data:** Sample size $N = 512$ or $2048$ with N/4 validation data for early stopping in the training.

◆ **Testing Data:** Sample size $T = 10^5$

◆ **Architecture:** Rectangular neural networks with the same architecture for all methods:

- **hidden layers width [128, 128, 128] for univariate target**
- **hidden layers width [256, 256, 256] for multivariate target**

◆ **Optimization:** Pytorch with Adam :

- default hyperparameters, learning rate 0.001.
- Batch size 128, maximum epochs 1000

◆ **Estimate:**

- Quantile curves at 19 different levels$(\tau_1, \ldots, \tau_{19})$ =(0.05, 0.1, . . . , 0.9, 0.95)

◆ **Metric:** Mean and standard deviation of L1 and L2 distances over R = 100replications.

# Simulation Studies

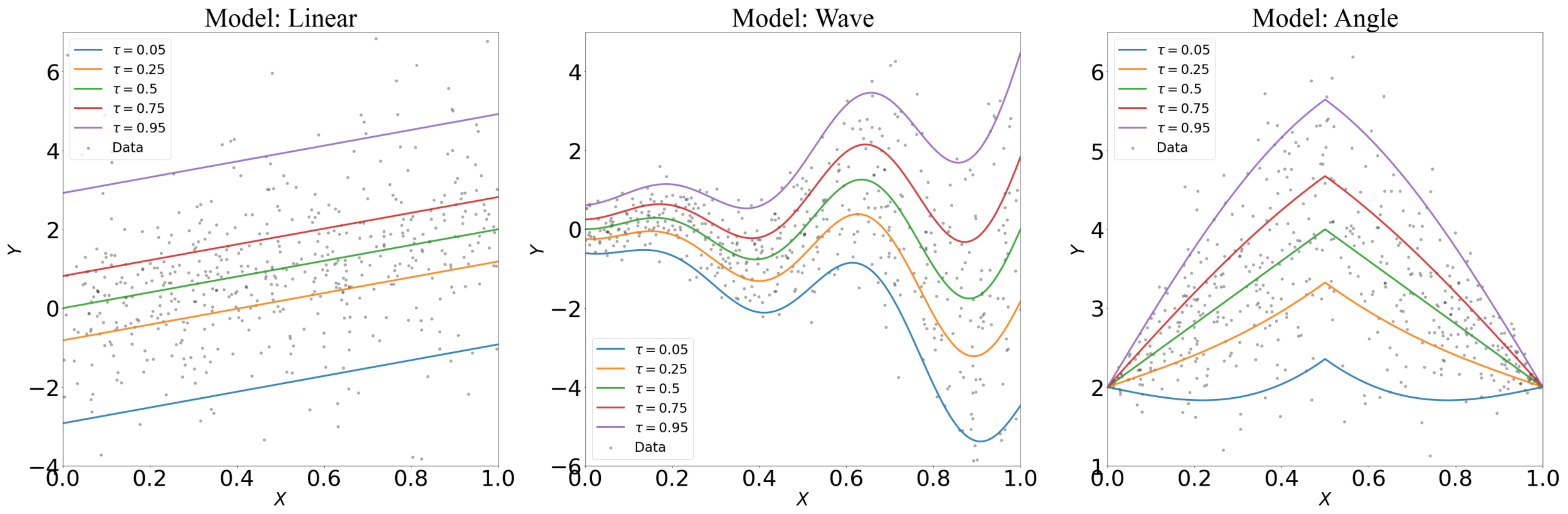


**Fig. 10**. The simulated univariate models. The sample data with size N = 512 is depicted asgrey dots. Five conditional quantile curves at levels τ =0.05 (blue), 0.25 (orange), 0.5 (green),0.75 (red), and 0.95 (purple) are depicted as solid curves.

## Simulation Studies

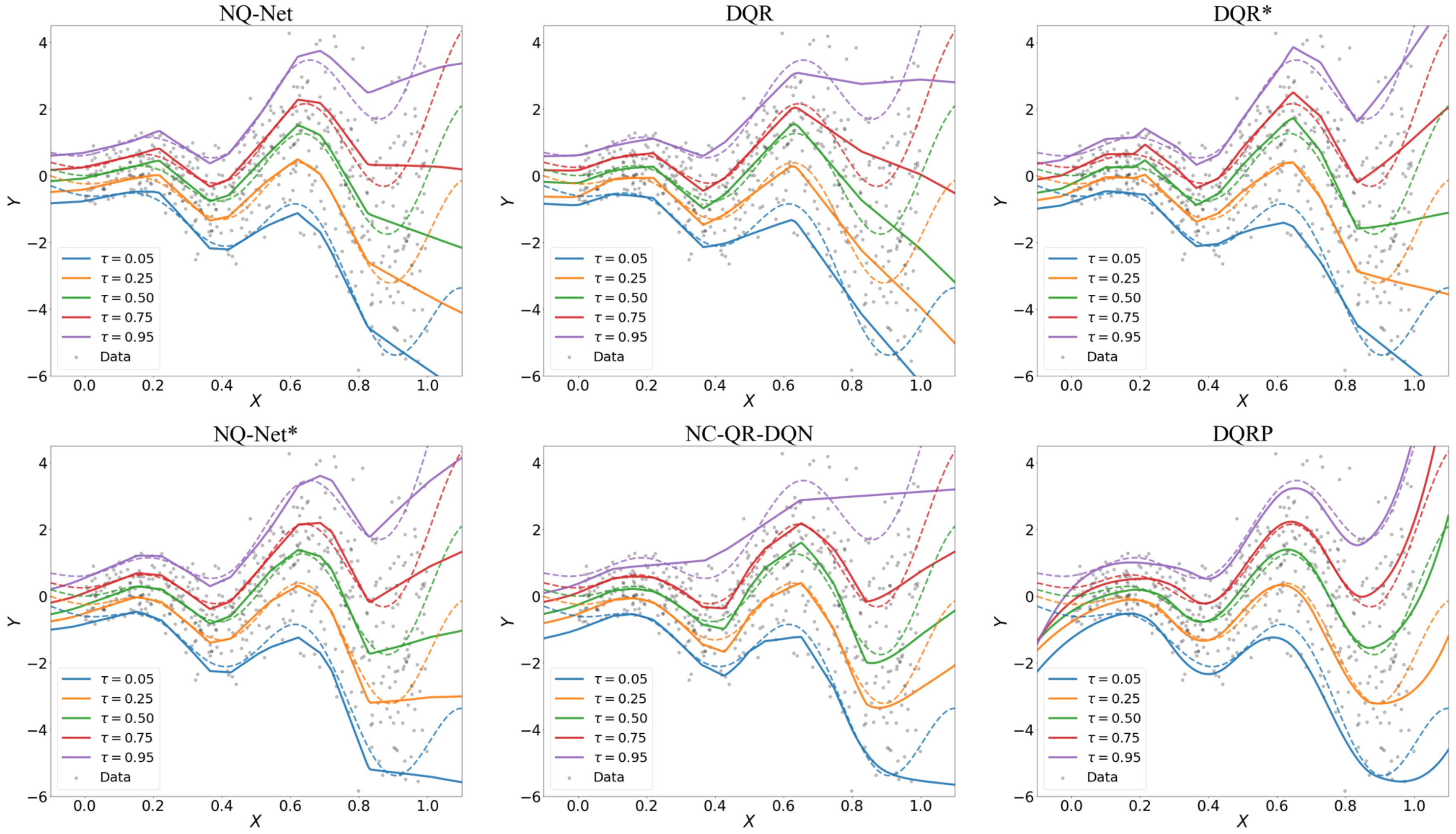


**Fig. 11**. An instance of the fitted quantile curves under "Wave" model when N = 512. The training data is depicted as grey dots. The target (estimated) quantile curves are depicted as dashed(solid) curves at levels τ =0.05 (blue), 0.25 (orange), 0.5 (green), 0.75 (red), 0.95 (purple). The NQ-Net* is a variant of the NQ-Net, employing ReLU activation instead of ELU + 1.

## Simulation Studies: “Wave” model

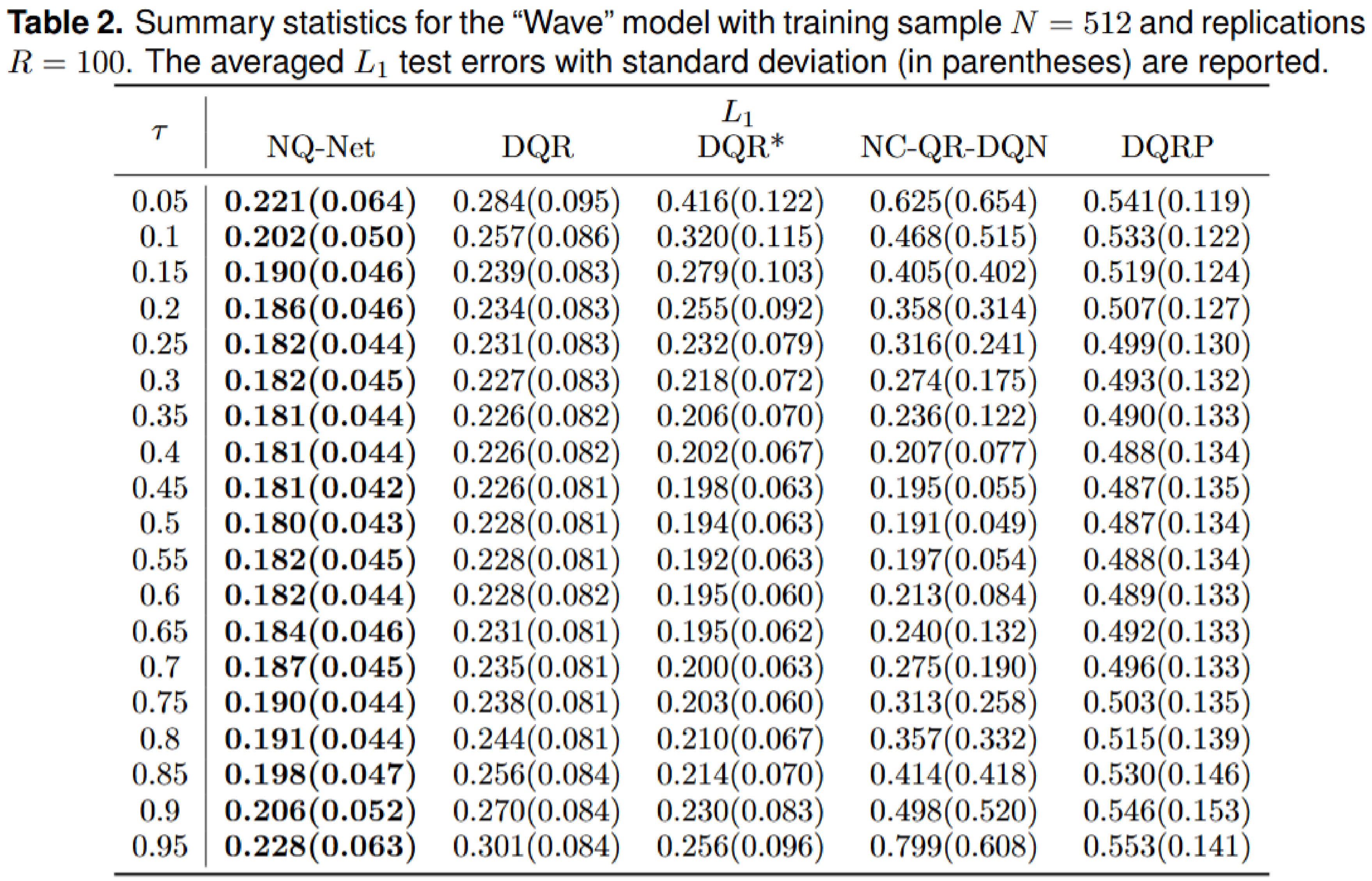


**Table 2.** Summary statistics for the “Wave” model with training sample $N = 512$ and replications $R = 100$. The averaged $L_1$ test errors with standard deviation (in parentheses) are reported.

| $\tau$ | | | $L_1$ | | |
|---|---|---|---|---|---|
| | NQ-Net | DQR | DQR* | NC-QR-DQN | DQRP |
| 0.05 | **0.221(0.064)** | 0.284(0.095) | 0.416(0.122) | 0.625(0.654) | 0.541(0.119) |
| 0.1 | **0.202(0.050)** | 0.257(0.086) | 0.320(0.115) | 0.468(0.515) | 0.533(0.122) |
| 0.15 | **0.190(0.046)** | 0.239(0.083) | 0.279(0.103) | 0.405(0.402) | 0.519(0.124) |
| 0.2 | **0.186(0.046)** | 0.234(0.083) | 0.255(0.092) | 0.358(0.314) | 0.507(0.127) |
| 0.25 | **0.182(0.044)** | 0.231(0.083) | 0.232(0.079) | 0.316(0.241) | 0.499(0.130) |
| 0.3 | **0.182(0.045)** | 0.227(0.083) | 0.218(0.072) | 0.274(0.175) | 0.493(0.132) |
| 0.35 | **0.181(0.044)** | 0.226(0.082) | 0.206(0.070) | 0.236(0.122) | 0.490(0.133) |
| 0.4 | **0.181(0.044)** | 0.226(0.082) | 0.202(0.067) | 0.207(0.077) | 0.488(0.134) |
| 0.45 | **0.181(0.042)** | 0.226(0.081) | 0.198(0.063) | 0.195(0.055) | 0.487(0.135) |
| 0.5 | **0.180(0.043)** | 0.228(0.081) | 0.194(0.063) | 0.191(0.049) | 0.487(0.134) |
| 0.55 | **0.182(0.045)** | 0.228(0.081) | 0.192(0.063) | 0.197(0.054) | 0.488(0.134) |
| 0.6 | **0.182(0.044)** | 0.228(0.082) | 0.195(0.060) | 0.213(0.084) | 0.489(0.133) |
| 0.65 | **0.184(0.046)** | 0.231(0.081) | 0.195(0.062) | 0.240(0.132) | 0.492(0.133) |
| 0.7 | **0.187(0.045)** | 0.235(0.081) | 0.200(0.063) | 0.275(0.190) | 0.496(0.133) |
| 0.75 | **0.190(0.044)** | 0.238(0.081) | 0.203(0.060) | 0.313(0.258) | 0.503(0.135) |
| 0.8 | **0.191(0.044)** | 0.244(0.081) | 0.210(0.067) | 0.357(0.332) | 0.515(0.139) |
| 0.85 | **0.198(0.047)** | 0.256(0.084) | 0.214(0.070) | 0.414(0.418) | 0.530(0.146) |
| 0.9 | **0.206(0.052)** | 0.270(0.084) | 0.230(0.083) | 0.498(0.520) | 0.546(0.153) |
| 0.95 | **0.228(0.063)** | 0.301(0.084) | 0.256(0.096) | 0.799(0.608) | 0.553(0.141) |

## Simulation Studies: Multivariate Linear model

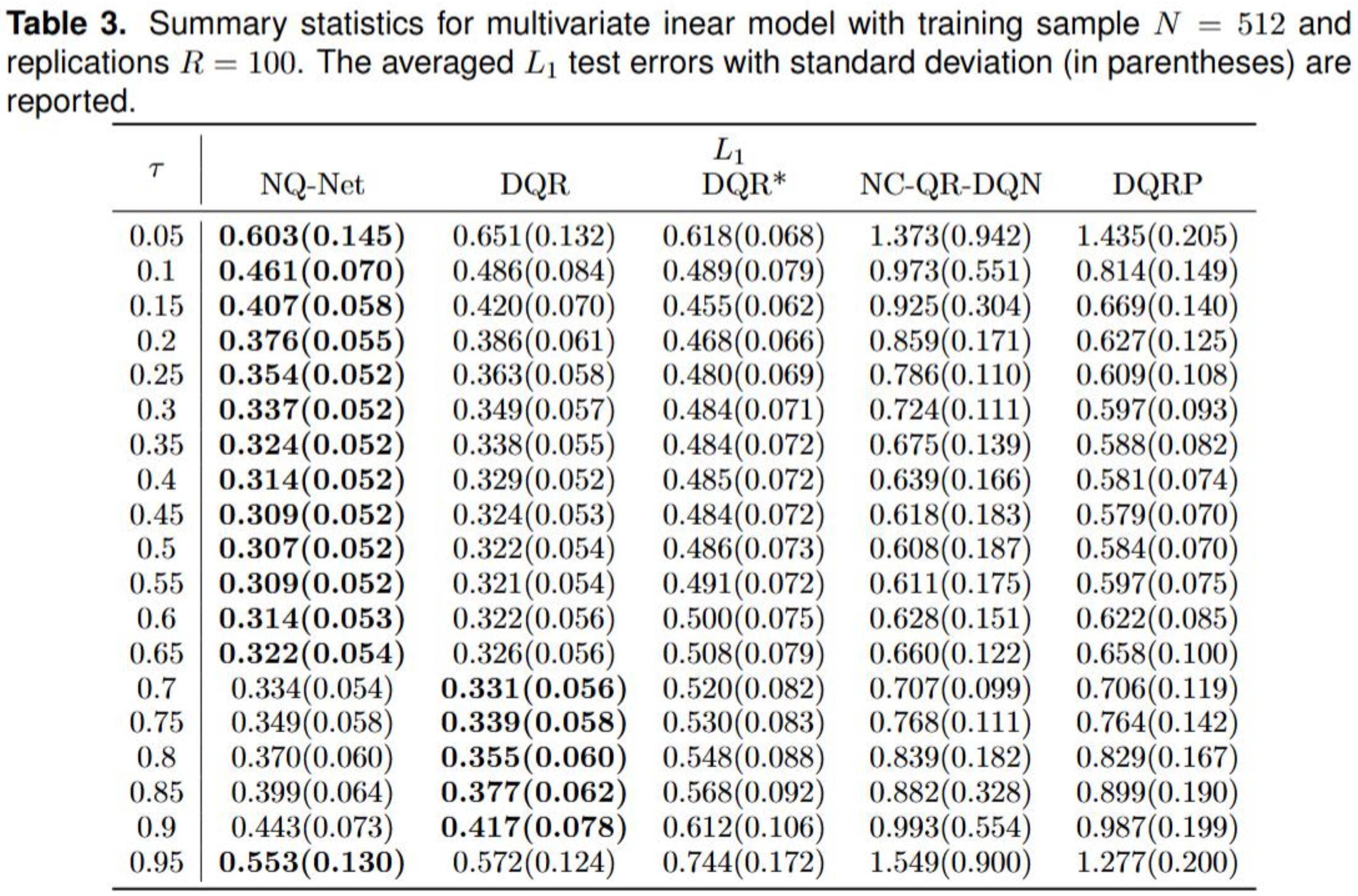


**Table 3.** Summary statistics for multivariate inear model with training sample $N = 512$ and replications $R = 100$. The averaged $L_1$ test errors with standard deviation (in parentheses) are reported.

| $\tau$ | | | $L_1$ | | |
|---|---|---|---|---|---|
| | NQ-Net | DQR | DQR* | NC-QR-DQN | DQRP |
| 0.05 | **0.603(0.145)** | 0.651(0.132) | 0.618(0.068) | 1.373(0.942) | 1.435(0.205) |
| 0.1 | **0.461(0.070)** | 0.486(0.084) | 0.489(0.079) | 0.973(0.551) | 0.814(0.149) |
| 0.15 | **0.407(0.058)** | 0.420(0.070) | 0.455(0.062) | 0.925(0.304) | 0.669(0.140) |
| 0.2 | **0.376(0.055)** | 0.386(0.061) | 0.468(0.066) | 0.859(0.171) | 0.627(0.125) |
| 0.25 | **0.354(0.052)** | 0.363(0.058) | 0.480(0.069) | 0.786(0.110) | 0.609(0.108) |
| 0.3 | **0.337(0.052)** | 0.349(0.057) | 0.484(0.071) | 0.724(0.111) | 0.597(0.093) |
| 0.35 | **0.324(0.052)** | 0.338(0.055) | 0.484(0.072) | 0.675(0.139) | 0.588(0.082) |
| 0.4 | **0.314(0.052)** | 0.329(0.052) | 0.485(0.072) | 0.639(0.166) | 0.581(0.074) |
| 0.45 | **0.309(0.052)** | 0.324(0.053) | 0.484(0.072) | 0.618(0.183) | 0.579(0.070) |
| 0.5 | **0.307(0.052)** | 0.322(0.054) | 0.486(0.073) | 0.608(0.187) | 0.584(0.070) |
| 0.55 | **0.309(0.052)** | 0.321(0.054) | 0.491(0.072) | 0.611(0.175) | 0.597(0.075) |
| 0.6 | **0.314(0.053)** | 0.322(0.056) | 0.500(0.075) | 0.628(0.151) | 0.622(0.085) |
| 0.65 | **0.322(0.054)** | 0.326(0.056) | 0.508(0.079) | 0.660(0.122) | 0.658(0.100) |
| 0.7 | 0.334(0.054) | **0.331(0.056)** | 0.520(0.082) | 0.707(0.099) | 0.706(0.119) |
| 0.75 | 0.349(0.058) | **0.339(0.058)** | 0.530(0.083) | 0.768(0.111) | 0.764(0.142) |
| 0.8 | 0.370(0.060) | **0.355(0.060)** | 0.548(0.088) | 0.839(0.182) | 0.829(0.167) |
| 0.85 | 0.399(0.064) | **0.377(0.062)** | 0.568(0.092) | 0.882(0.328) | 0.899(0.190) |
| 0.9 | 0.443(0.073) | **0.417(0.078)** | 0.612(0.106) | 0.993(0.554) | 0.987(0.199) |
| 0.95 | **0.553(0.130)** | 0.572(0.124) | 0.744(0.172) | 1.549(0.900) | 1.277(0.200) |

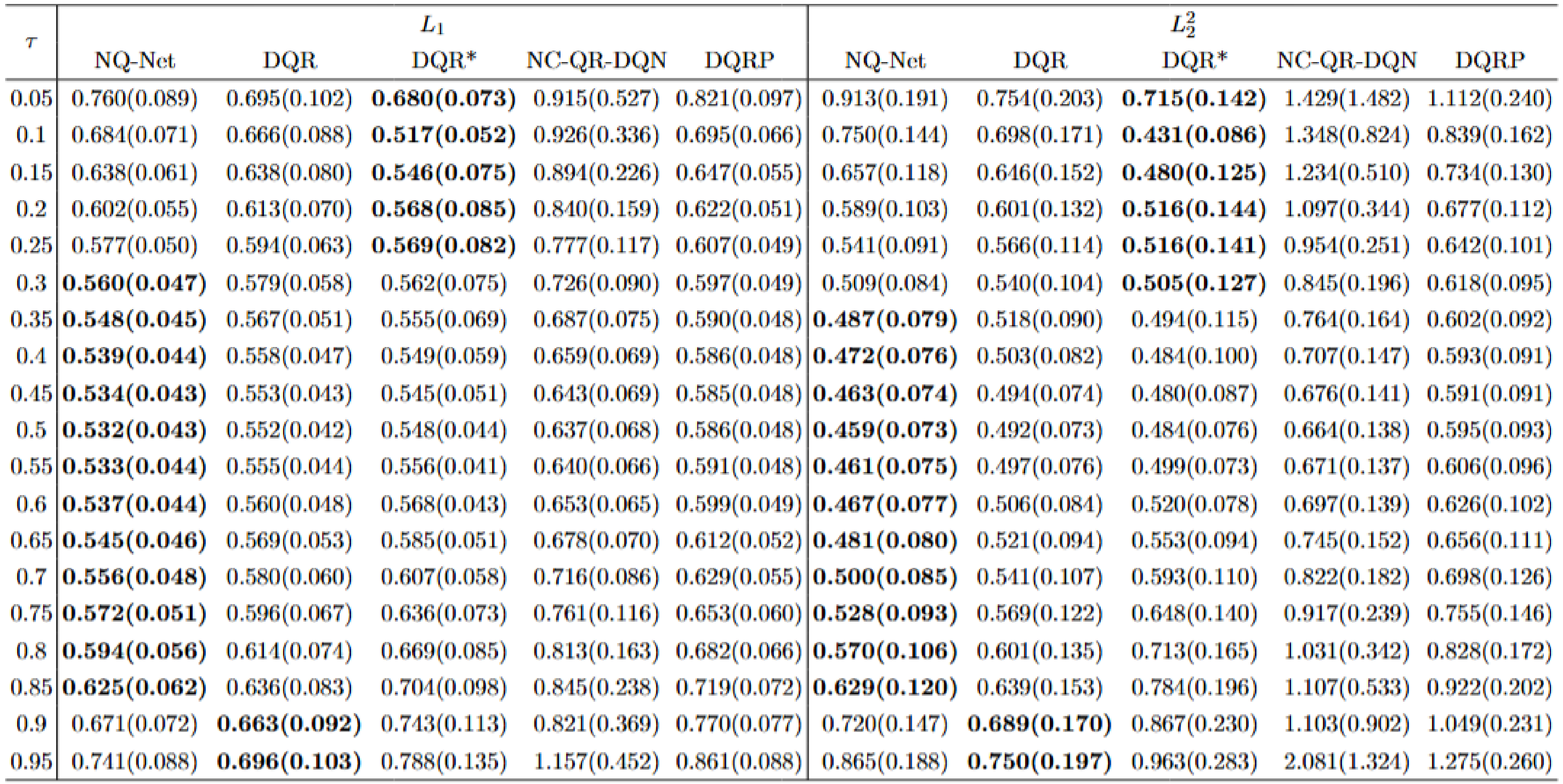
## Simulation Studies: Univariate Linear model

**Table S2.** Summary statistics for the "Linear" model with training sample size $N = 512$ and replications $R = 100$. The averaged $L_1$ and $L_2^2$ test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.

| $\tau$ | $L_1$ | | | | | $L_2^2$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | NQ-Net | DQR | DQR* | NC-QR-DQN | DQRP | NQ-Net | DQR | DQR* | NC-QR-DQN | DQRP |
| 0.05 | **0.296(0.175)** | 0.363(0.166) | 0.604(0.205) | 0.506(0.239) | 0.143(0.206) | **0.143(0.206)** | 0.216(0.234) | 0.492(0.264) | 0.423(0.439) | 0.497(0.193) |
| 0.1 | **0.171(0.082)** | 0.194(0.094) | 0.251(0.081) | 0.318(0.215) | 0.050(0.059) | **0.050(0.059)** | 0.067(0.069) | 0.099(0.061) | 0.205(0.286) | 0.495(0.187) |
| 0.15 | **0.136(0.060)** | 0.142(0.066) | 0.230(0.085) | 0.222(0.145) | 0.031(0.030) | **0.031(0.030)** | 0.035(0.036) | 0.087(0.059) | 0.098(0.133) | 0.473(0.168) |
| 0.2 | 0.117(0.049) | **0.115(0.050)** | 0.192(0.063) | 0.162(0.091) | 0.023(0.021) | **0.023(0.021)** | 0.023(0.025) | 0.060(0.035) | 0.049(0.059) | 0.452(0.156) |
| 0.25 | 0.105(0.044) | **0.104(0.045)** | 0.158(0.054) | 0.131(0.062) | 0.019(0.016) | **0.019(0.016)** | 0.019(0.016) | 0.040(0.024) | 0.030(0.028) | 0.436(0.15) |
| 0.3 | 0.099(0.041) | **0.095(0.041)** | 0.135(0.048) | 0.116(0.048) | 0.017(0.015) | 0.017(0.015) | **0.016(0.014)** | 0.029(0.018) | 0.023(0.018) | 0.426(0.147) |
| 0.35 | 0.093(0.039) | **0.087(0.038)** | 0.114(0.045) | 0.111(0.043) | 0.015(0.014) | 0.015(0.014) | **0.013(0.011)** | 0.022(0.015) | 0.020(0.015) | 0.419(0.146) |
| 0.4 | 0.088(0.039) | **0.082(0.039)** | 0.102(0.042) | 0.111(0.041) | 0.014(0.013) | 0.014(0.013) | **0.012(0.012)** | 0.017(0.014) | 0.020(0.014) | 0.415(0.146) |
| 0.45 | 0.086(0.036) | **0.078(0.034)** | 0.095(0.038) | 0.115(0.042) | 0.014(0.012) | 0.014(0.012) | **0.011(0.010)** | 0.015(0.011) | 0.021(0.015) | 0.413(0.147) |
| 0.5 | 0.085(0.035) | **0.078(0.036)** | 0.085(0.034) | 0.120(0.044) | 0.013(0.011) | 0.013(0.011) | **0.011(0.011)** | 0.012(0.009) | 0.023(0.016) | 0.412(0.147) |
| 0.55 | 0.085(0.037) | **0.082(0.038)** | 0.085(0.036) | 0.124(0.045) | 0.013(0.012) | 0.013(0.012) | **0.012(0.012)** | 0.013(0.011) | 0.024(0.018) | 0.411(0.147) |
| 0.6 | 0.089(0.037) | **0.086(0.037)** | 0.092(0.035) | 0.130(0.046) | 0.014(0.012) | 0.014(0.012) | **0.013(0.012)** | 0.014(0.012) | 0.027(0.019) | 0.413(0.148) |
| 0.65 | 0.091(0.038) | **0.089(0.041)** | 0.101(0.039) | 0.138(0.050) | 0.015(0.013) | 0.015(0.013) | **0.014(0.014)** | 0.017(0.014) | 0.031(0.023) | 0.417(0.151) |
| 0.7 | 0.096(0.041) | **0.094(0.045)** | 0.112(0.046) | 0.154(0.064) | 0.016(0.014) | **0.016(0.014)** | 0.016(0.015) | 0.022(0.017) | 0.041(0.034) | 0.426(0.156) |
| 0.75 | 0.107(0.044) | **0.106(0.049)** | 0.132(0.052) | 0.182(0.009) | 0.019(0.015) | **0.019(0.015)** | 0.020(0.017) | 0.029(0.022) | 0.059(0.054) | 0.443(0.167) |
| 0.8 | 0.123(0.050) | **0.120(0.054)** | 0.156(0.063) | 0.221(0.121) | 0.025(0.018) | 0.025(0.018) | **0.024(0.020)** | 0.040(0.029) | 0.090(0.085) | 0.468(0.186) |
| 0.85 | 0.145(0.063) | **0.143(0.068)** | 0.180(0.073) | 0.261(0.147) | 0.034(0.028) | **0.034(0.028)** | 0.035(0.033) | 0.053(0.037) | 0.125(0.124) | 0.503(0.214) |
| 0.9 | **0.185(0.085)** | 0.200(0.093) | 0.220(0.085) | 0.285(0.148) | 0.053(0.043) | **0.053(0.043)** | 0.063(0.057) | 0.074(0.050) | 0.143(0.146) | 0.539(0.245) |
| 0.95 | **0.296(0.168)** | 0.346(0.165) | 0.298(0.157) | 0.529(0.236) | 0.132(0.124) | **0.132(0.124)** | 0.189(0.181) | 0.138(0.126) | 0.384(0.301) | 0.553(0.238) |

## Simulation Studies: Additive model

**Table S8.** Summary statistics for the additive model with training sample size $N = 512$ and the number of replications $R = 100$. The averaged $L_1$ and $L_2^2$ test errors with the corresponding standard deviation (in parentheses) are reported for the estimators trained by different methods.



| $\tau$ | $L_1$ | | | | | $L_2^2$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | NQ-Net | DQR | DQR* | NC-QR-DQN | DQRP | NQ-Net | DQR | DQR* | NC-QR-DQN | DQRP |
| 0.05 | 0.760(0.089) | 0.695(0.102) | **0.680(0.073)** | 0.915(0.527) | 0.821(0.097) | 0.913(0.191) | 0.754(0.203) | **0.715(0.142)** | 1.429(1.482) | 1.112(0.240) |
| 0.1 | 0.684(0.071) | 0.666(0.088) | **0.517(0.052)** | 0.926(0.336) | 0.695(0.066) | 0.750(0.144) | 0.698(0.171) | **0.431(0.086)** | 1.348(0.824) | 0.839(0.162) |
| 0.15 | 0.638(0.061) | 0.638(0.080) | **0.546(0.075)** | 0.894(0.226) | 0.647(0.055) | 0.657(0.118) | 0.646(0.152) | **0.480(0.125)** | 1.234(0.510) | 0.734(0.130) |
| 0.2 | 0.602(0.055) | 0.613(0.070) | **0.568(0.085)** | 0.840(0.159) | 0.622(0.051) | 0.589(0.103) | 0.601(0.132) | **0.516(0.144)** | 1.097(0.344) | 0.677(0.112) |
| 0.25 | 0.577(0.050) | 0.594(0.063) | **0.569(0.082)** | 0.777(0.117) | 0.607(0.049) | 0.541(0.091) | 0.566(0.114) | **0.516(0.141)** | 0.954(0.251) | 0.642(0.101) |
| 0.3 | **0.560(0.047)** | 0.579(0.058) | 0.562(0.075) | 0.726(0.090) | 0.597(0.049) | 0.509(0.084) | 0.540(0.104) | **0.505(0.127)** | 0.845(0.196) | 0.618(0.095) |
| 0.35 | **0.548(0.045)** | 0.567(0.051) | 0.555(0.069) | 0.687(0.075) | 0.590(0.048) | **0.487(0.079)** | 0.518(0.090) | 0.494(0.115) | 0.764(0.164) | 0.602(0.092) |
| 0.4 | **0.539(0.044)** | 0.558(0.047) | 0.549(0.059) | 0.659(0.069) | 0.586(0.048) | **0.472(0.076)** | 0.503(0.082) | 0.484(0.100) | 0.707(0.147) | 0.593(0.091) |
| 0.45 | **0.534(0.043)** | 0.553(0.043) | 0.545(0.051) | 0.643(0.069) | 0.585(0.048) | **0.463(0.074)** | 0.494(0.074) | 0.480(0.087) | 0.676(0.141) | 0.591(0.091) |
| 0.5 | **0.532(0.043)** | 0.552(0.042) | 0.548(0.044) | 0.637(0.068) | 0.586(0.048) | **0.459(0.073)** | 0.492(0.073) | 0.484(0.076) | 0.664(0.138) | 0.595(0.093) |
| 0.55 | **0.533(0.044)** | 0.555(0.044) | 0.556(0.041) | 0.640(0.066) | 0.591(0.048) | **0.461(0.075)** | 0.497(0.076) | 0.499(0.073) | 0.671(0.137) | 0.606(0.096) |
| 0.6 | **0.537(0.044)** | 0.560(0.048) | 0.568(0.043) | 0.653(0.065) | 0.599(0.049) | **0.467(0.077)** | 0.506(0.084) | 0.520(0.078) | 0.697(0.139) | 0.626(0.102) |
| 0.65 | **0.545(0.046)** | 0.569(0.053) | 0.585(0.051) | 0.678(0.070) | 0.612(0.052) | **0.481(0.080)** | 0.521(0.094) | 0.553(0.094) | 0.745(0.152) | 0.656(0.111) |
| 0.7 | **0.556(0.048)** | 0.580(0.060) | 0.607(0.058) | 0.716(0.086) | 0.629(0.055) | **0.500(0.085)** | 0.541(0.107) | 0.593(0.110) | 0.822(0.182) | 0.698(0.126) |
| 0.75 | **0.572(0.051)** | 0.596(0.067) | 0.636(0.073) | 0.761(0.116) | 0.653(0.060) | **0.528(0.093)** | 0.569(0.122) | 0.648(0.140) | 0.917(0.239) | 0.755(0.146) |
| 0.8 | **0.594(0.056)** | 0.614(0.074) | 0.669(0.085) | 0.813(0.163) | 0.682(0.066) | **0.570(0.106)** | 0.601(0.135) | 0.713(0.165) | 1.031(0.342) | 0.828(0.172) |
| 0.85 | **0.625(0.062)** | 0.636(0.083) | 0.704(0.098) | 0.845(0.238) | 0.719(0.072) | **0.629(0.120)** | 0.639(0.153) | 0.784(0.196) | 1.107(0.533) | 0.922(0.202) |
| 0.9 | 0.671(0.072) | **0.663(0.092)** | 0.743(0.113) | 0.821(0.369) | 0.770(0.077) | 0.720(0.147) | **0.689(0.170)** | 0.867(0.230) | 1.103(0.902) | 1.049(0.231) |
| 0.95 | 0.741(0.088) | **0.696(0.103)** | 0.788(0.135) | 1.157(0.452) | 0.861(0.088) | 0.865(0.188) | **0.750(0.197)** | 0.963(0.283) | 2.081(1.324) | 1.275(0.260) |

# References

1. Chen, J. and Jiang, N. (2019). Information-theoretic considerations in batch reinforcement learning. International Conference on Machine Learning, 1042–1051
2. Fan, J., Wang, Z., Xie, Y., and Yang, Z. (2020). A theoretical analysis of deep q-learning. Learning for dynamics and control, 486–489.
3. Hao, B., Ji, X., Duan, Y., Lu, H., Szepesvari, C., and Wang, M. (2021). Bootstrapping fitted q-evaluation for off-policy inference. International Conference on Machine Learning,4074–4084.
4. Lederer, J. (2020). Risk bounds for robust deep learning. arXiv preprint arXiv:2009.06202.
5. Li, G., Cai, C., Chen, Y., Gu, Y., Wei, Y., and Chi, Y. (2021). Tightening the dependence on horizon in the sample complexity of q-learning. International Conference on Machine Learning, 6296–6306.
6. Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). Foundations of machine learning. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA.
7. Padilla, O. H. M., Tansey, W., and Chen, Y. (2022). Quantile regression with relu networks:Estimators and minimax rates. Journal of Machine Learning Research, 23(247):1–42.
8. Ramprasad, P., Li, Y., Yang, Z., Wang, Z., Sun, W. W., and Cheng, G. (2023). Online boot-strap inference for policy evaluation in reinforcement learning. Journal of the American Statistical Association, 118(544):2901–2914.
9. Rowland, M., Tang, Y., Lyle, C., Munos, R., Bellemare, M. G., and Dabney, W. (2023). The statistical benefits of quantile temporal-difference learning for value estimation. Interna-tional Conference on Machine Learning, 29210–29231.
10. Sangnier, M., Fercoq, O., and d'Alch ´e Buc, F. (2016). Joint quantile regression in vector-valued rkhss. Advances in Neural Information Processing Systems.
11. Shen, G., Jiao, Y., Lin, Y., Horowitz, J. L., and Huang, J. (2024). Nonparametric estimation of non-crossing quantile regression process with deep requ neural networks. Journal of Machine Learning Research, 25(88):1–75.
12. Shi, C., Zhang, S., Lu, W., and Song, R. (2022). Statistical inference of the value function for reinforcement learning in infinite-horizon settings. Journal of the Royal Statistical Society Series B: Statistical Methodology, 84(3):765–793.
13. Stone, C. J. (1982). Optimal global rates of convergence for nonparametric regression. Annals of Statistics, 10(4):1040–1053.
14. Takeuchi, I., Le, Q. V., Sears, T. D., and Smola, A. J. (2006). Nonparametric quantileestimation. Journal of Machine Learning Research, 7(45):1231–1264.
15. Yan, X., Su, Y., and Ma, W. (2023). Ensemble multi-quantiles: Adaptively flexible distribu-tion prediction for uncertainty quantification. IEEE Transactions on Pattern Analysis andMachine Intelligence, 45(11):13068–13082.
16. Zhou, F., Wang, J., and Feng, X. (2020). Non-crossing quantile regression for distributionalreinforcement learning. Advances in neural information processing systems, 33:15909–15919.

## Content

## Questions of Interest

*How to extend our Non-crossing Quantile Network to time-to-event analysis, while considering more advanced DL architectures?*

**S. Huang, Z. Qu, Z. Hua , G. Shen, Rui Tang, H. Zhu (2025). Non-Crossing Quantile Regression for Time-to-Event Analysis: A Deep Learning Framework with Theoretical Guarantees. In submission.**

# Motivations and resolution

## Motivations

Traditional methods: baseline covariates not incorporated -
Heterogeneity in patient population, prognostic factors

Traditional methods:
model assumptions –
AFT, CoxPH

**Time-to-event prediction**

## Goals

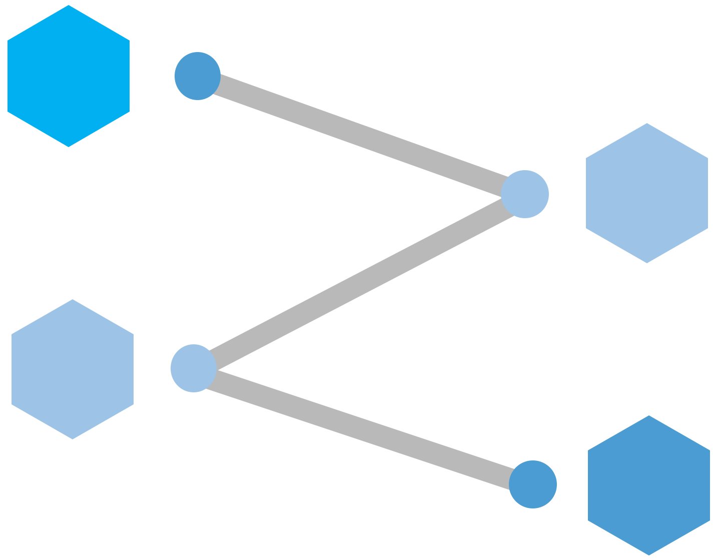
- With covariates incorporated
- Enhance prediction accuracy.

- Distribution – free
- Flexible to non-linear patterns

# Related applications in clinical trials

identify prognostic factors, optimize treatment strategy

clinical trial designs, predicting timeline for efficacy readout



identify patients sensitive to benefit from treatment, those at high risk of severe safety events

Improve quality of life
Optimize resource

# Deep censored quantile regression

**Traditional quantile regression**:

Under right censoring, the estimator $\hat{\beta}_\tau$ can be obtained by minimizing the loss function:

$$\sum_{i=1}^{n} \omega_i \, \rho_\tau(\log(Y_i) - \boxed{\beta_\tau' X_i})$$

**Deep quantile regression**:

Replace the linear functional by the output from the neural network $\log(\widehat{\mathrm{Q}}_{\mathrm{i}}^{(\tau)})$, and train by minimizing the new loss:

$$\sum_{i=1}^{n} \omega_i \, \rho_\tau(\log(Y_i) - \boxed{\log(\widehat{\mathrm{Q}}_{\mathrm{i}}^{(\tau)})})$$

$\omega_i = \delta_i / \hat{G}(Y_i)$, $\hat{G}(Y_i)$: Kaplan-Meier estimator of the censoring distribution;
$\rho_\tau(u)$: check function, $\rho_\tau(u) = u[\tau - I(u \le 0)]$

# Technical challenges: multiple quantile estimation and crossing

**Sequential estimation**

- Estimate a grid of M desired quantiles at a time
- Efficient and resource-saving

**Crossing issues**

- Implicit restriction in quantile regression
- Positive activation function $\log(1+\exp(\cdot))$

**Non-crossing:** $Q_T^{\tau_1}(X) \leq \cdots \leq Q_T^{\tau_K}(X)$

**Positive activation:** $\log(1+\exp(\cdot))$; Base network, Gap networks; Cumsum

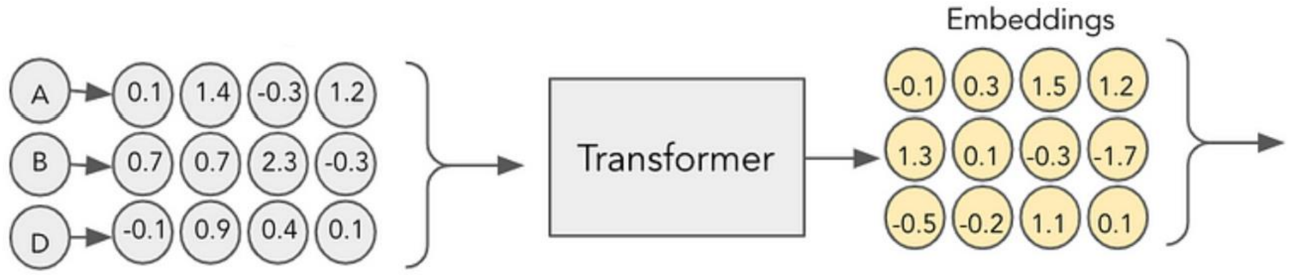
# TabTransformer

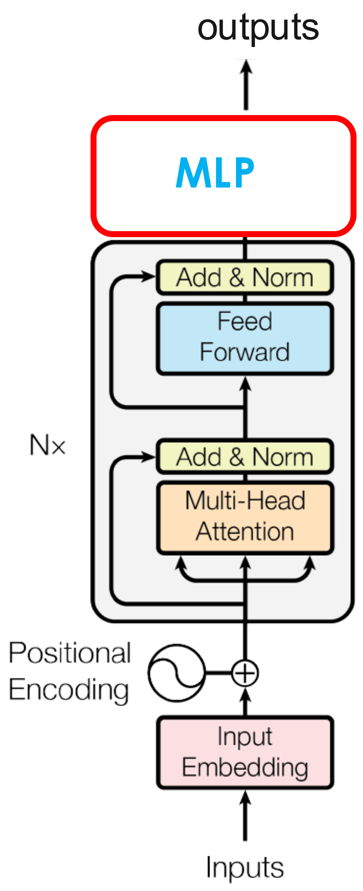
➢**Our data**:

- - Each feature is a scalar.
- - The order of features does not matter

➢**Attentions** are performed on the features

- - Embed each scalar feature to a feature vector





# KAN

- Kolmogorov-Arnold Network (KAN), proposed in May 2024.

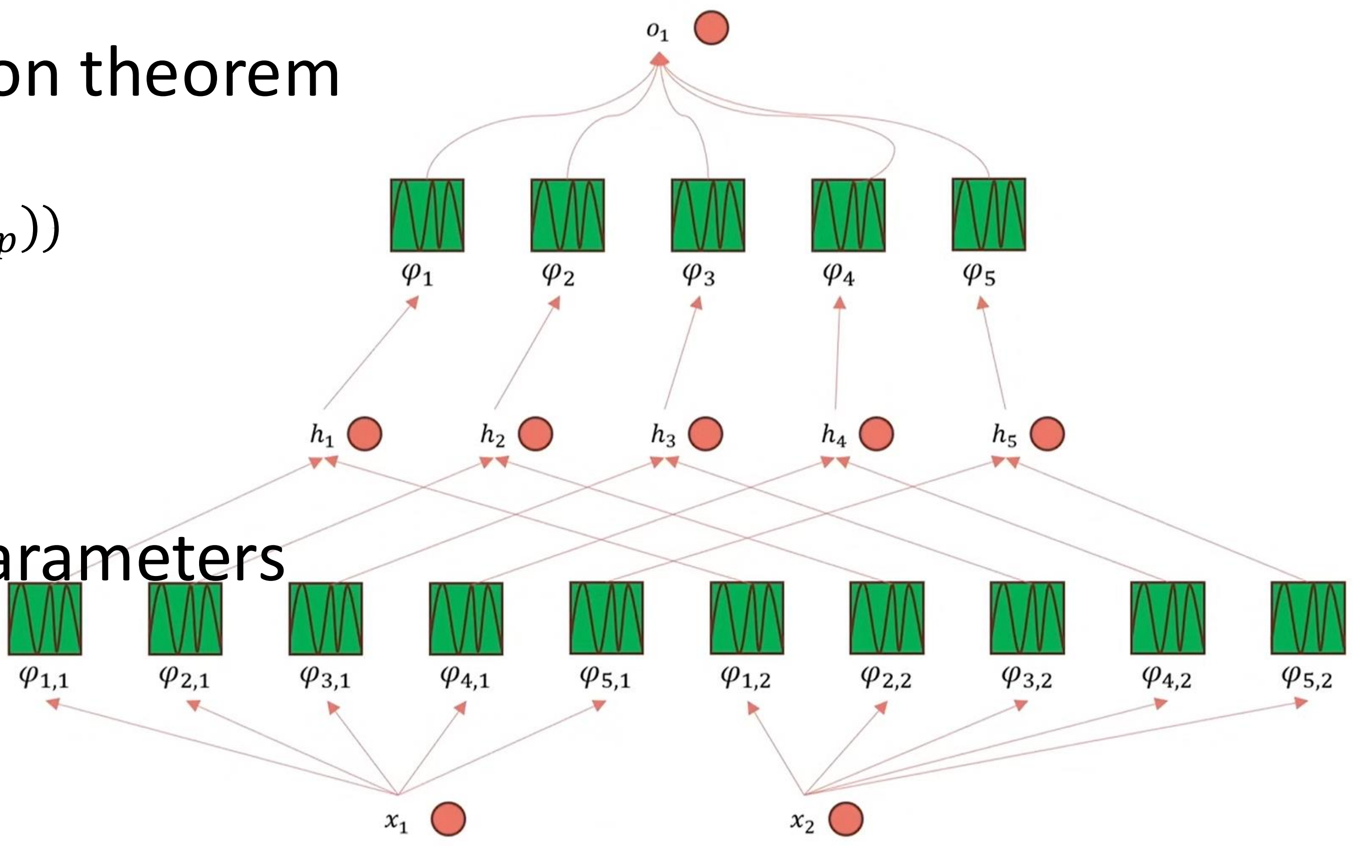
- Kolmogorov-Arnold representation theorem

$$f(x) = f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q\left(\sum_{p=1}^{n} \phi_{q,p}(x_p)\right)$$

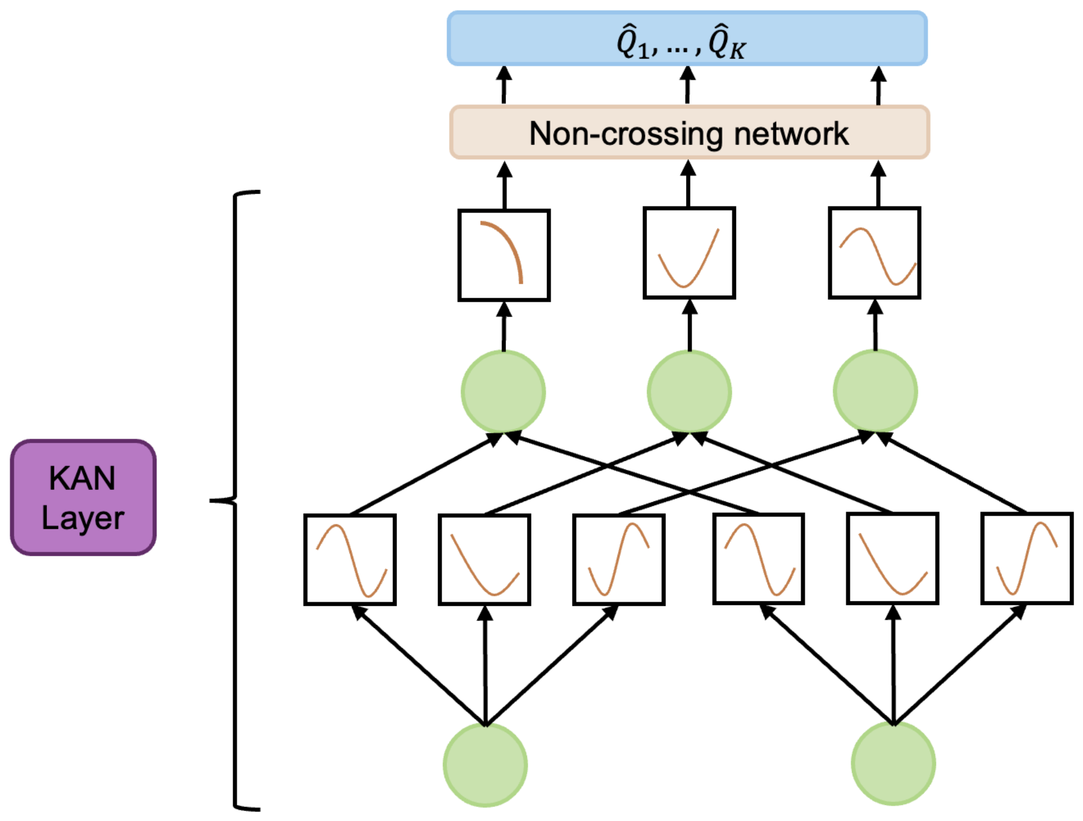
- Learnable functions instead of parameters

- Right figure: n=2, 2n+1=5

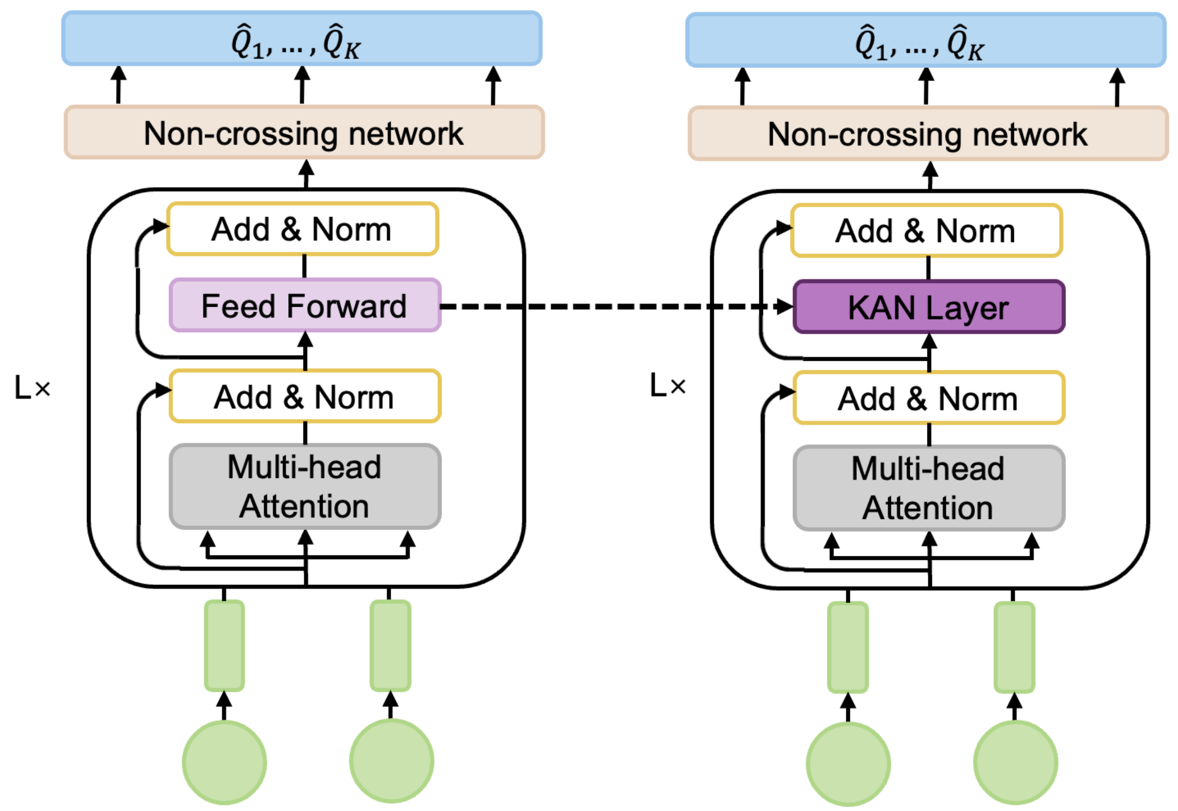
- Replace feed forward layer (MLP) in Transformer



# Model structures



KAN-CNQ



Trans-CNQ

TransKAN-CNQ

## Theoretical guarantees

Theorem 1*: Let $\mathcal{R}$ be the excess risk defined in (11). Then for any $N \geq 1$, and $\eta > 0$, with probability of not less than $1 - \frac{9}{2}e^{-\eta} - \exp(-H_t/3N)$,*

$$\mathcal{R}(\hat{f}_N) = L(\hat{f}_N) - \inf_{f \in \mathcal{F}} L(f)$$

$$\leq \frac{B\left(6\sqrt{\frac{[\eta + \log \mathcal{N}(\mathcal{F}_N, ||\cdot||_\infty, \epsilon)]}{2}} + D_o + 2\right)}{H_t^2\sqrt{N}} + \frac{8\epsilon L_M}{H_t^2}$$

## Real data distributions

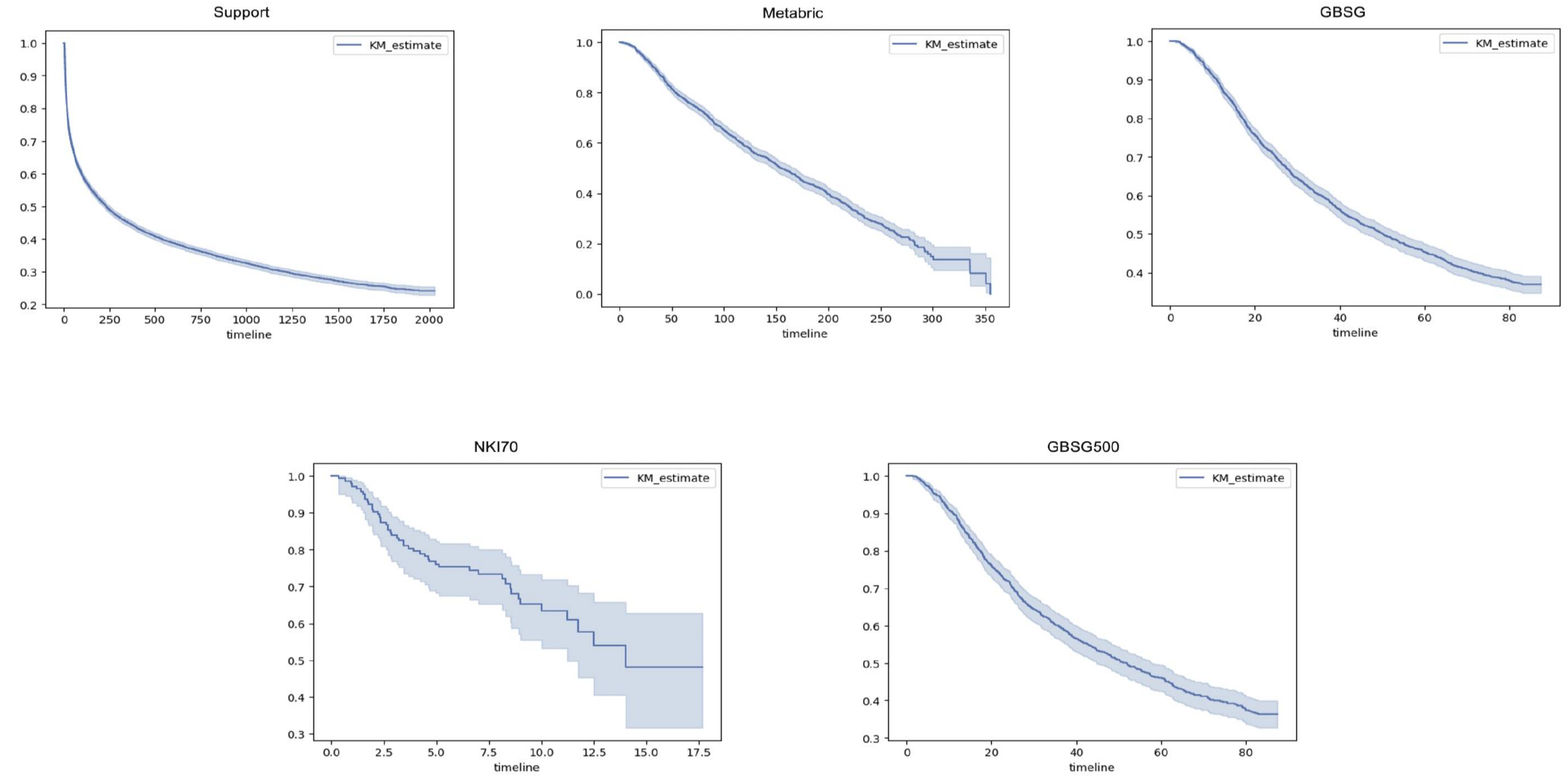
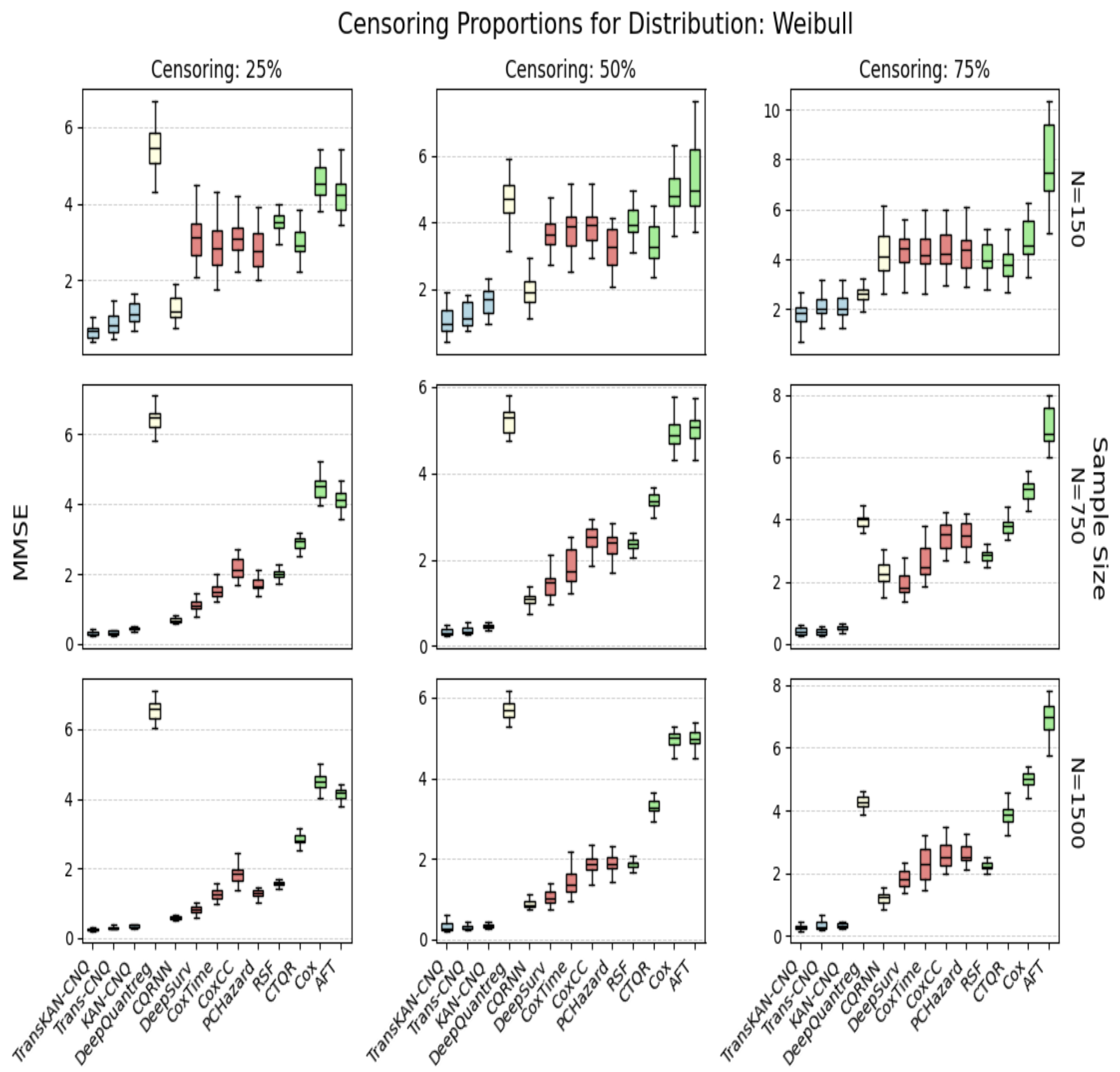
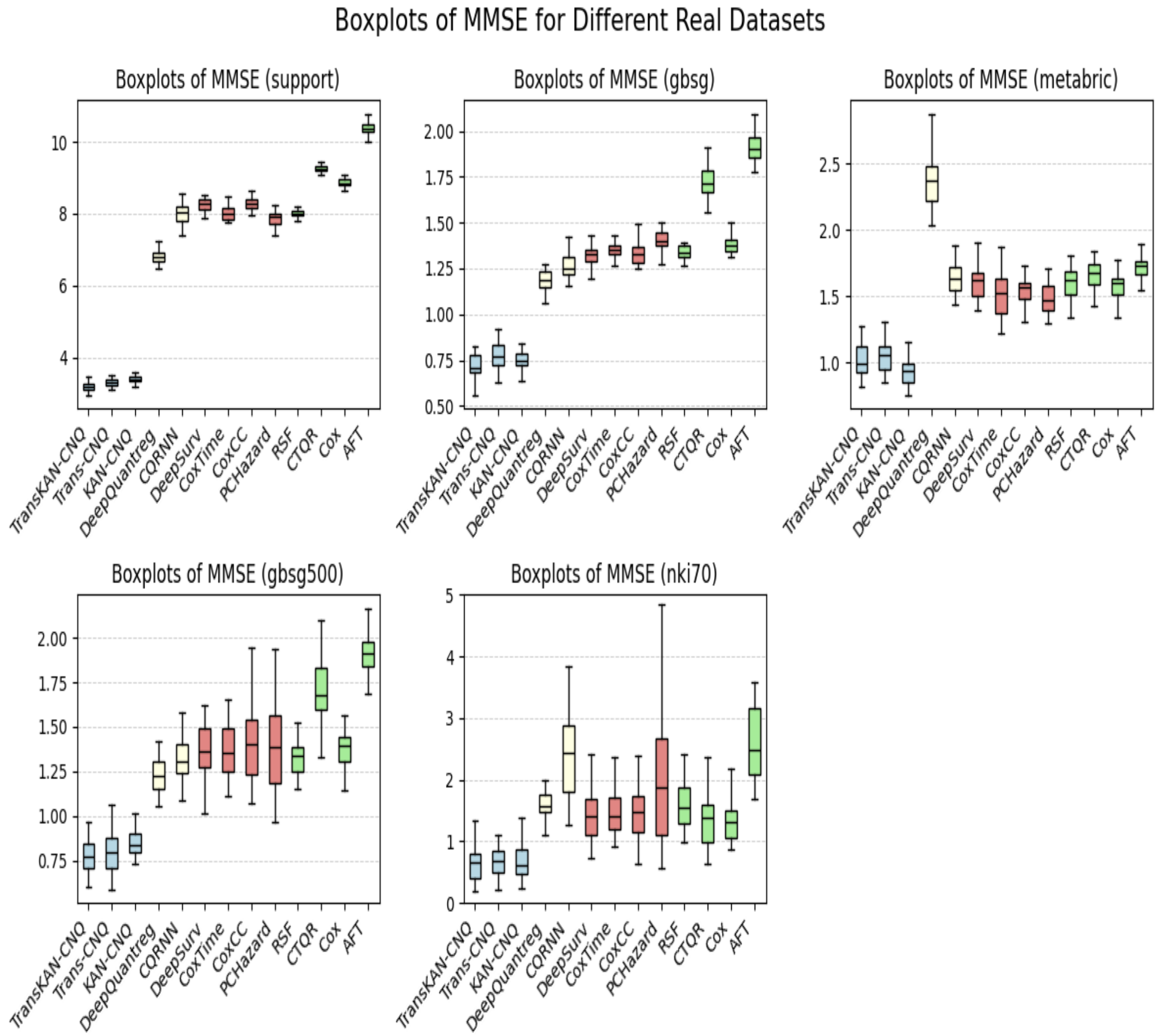


**Figure 7**: KM curves and the confidence band for 5 real datasets

# Results - Simulation and real data





**Censored Non-Crossing Quantile (CNQ) network**

## Content

## Question of Interest

While the **manifold hypothesis** provides a useful guiding intuition, its strict mathematical form is often too brittle for real-world, noisy, heterogeneous datasets. Modern work therefore moves toward looser notions—**mixtures of manifolds**, **fractal supports**, or **purely learned latent distributions**—that can better tolerate noise, intersections, and varying intrinsic dimensions.

***How to address the challenges of unstructured image data in the supervised learning framework?***

**Shen, G. and Zhu, H.. (2025). Understanding Convolutional Neural Networks: Statistical Generative Models for Unstructured Image Data. In submission.**

# Understanding CNNs: Statistical Generative Models for Image Data

- **To understand CNN:** learning theories through the lenses of complexity, approximation, and optimization (training dynamics).
  - Optimization: algorithms, such as SGD can effectively find global minima of the training objectives (Zhou & Feng, 2018; Allen-Zhu et al., 2019).
  - Studies on Complexity lead to generalization guarantees for learning tasks involving CNNs. and approximation (Lin & Zhang, 2019; Feng et al., 2021; Shen,2024; Petersen & Voigtlaender, 2020; Zhou, 2020a;b)
- **Revealing no significant advantage of CNNs over other FNNs on image data**
  - The limitation stems from insufficient calibration of the heterogeneous characteristics of image data, identifying the signals and noises, and understanding their interactions with CNNs during feature extraction.

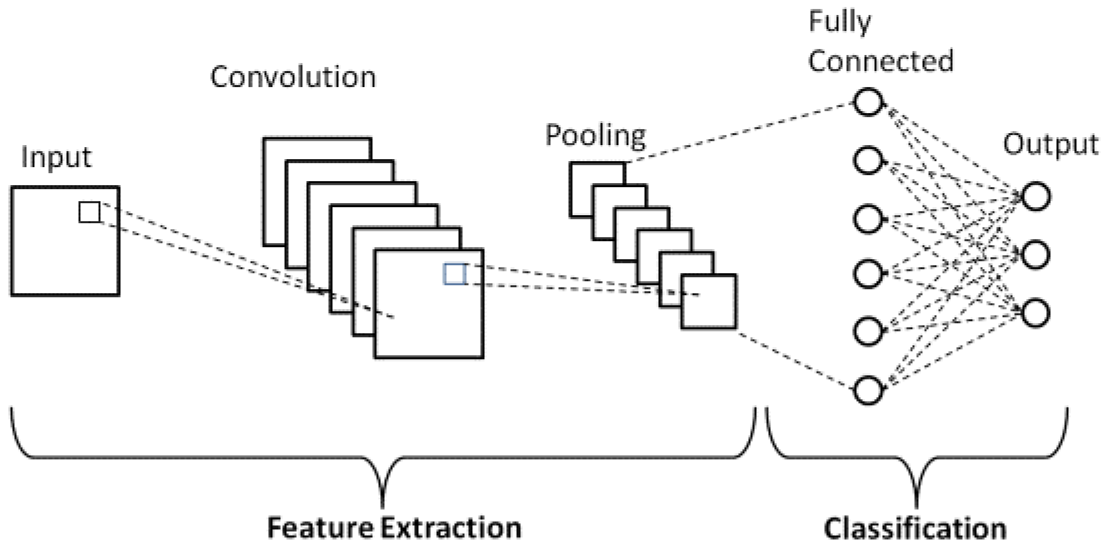


**Fig. 1**. An example of a convolutional neural network(CNN) for classification, comprising a feature extraction stage followed by classification using fully connected layers.

## Understanding CNNs: Statistical Generative Models for Image Data

- **To address these challenges, three critical questions arise:**
- (Q1) How can we model image data statistically, particularly in differentiating between ’signals’ and ’noise’ for downstream tasks?
- (Q2) How can we understand the role of feature maps by using stacked convolutional and pooling layers that trans-form heterogeneous image data into relatively homogeneous feature representations?
- (Q3) How can we evaluate the learning efficiency of CNNs in vision tasks using our image data model and feature mapping approach?

## Statistical Generative Models (SGMs) of Image Data

📌 **Statistical Generative Models (SGMs) of image data**

$$X = X_{\text{obj}}^1(u_1) \oplus \cdots \oplus X_{\text{obj}}^J(u_J) \oplus \epsilon$$

- X: image data with domain $\Omega = Z_n \times Z_m$
- $X_{obj}^1, \dots, X_{obj}^J$: Random Objects in the image
- $\epsilon$: Backgrounds, may correlate with objects
- $\oplus$: Masked addition operation, overlaying objects
- $J$: Number of objects, $J \sim Possion(\Lambda(\Omega))$
- $\lambda$(u): Intensity function, $\Lambda(\Omega) = \int_\Omega \lambda(u)\, du$
- $u_j$: Location of object j, $u_j \sim P_\lambda = \lambda(u)/\Lambda(\Omega)$

💡**Problem definition**

- Flexibility and Adaptability
- Statistical Interpretability
- Enhanced Feature Prioritization

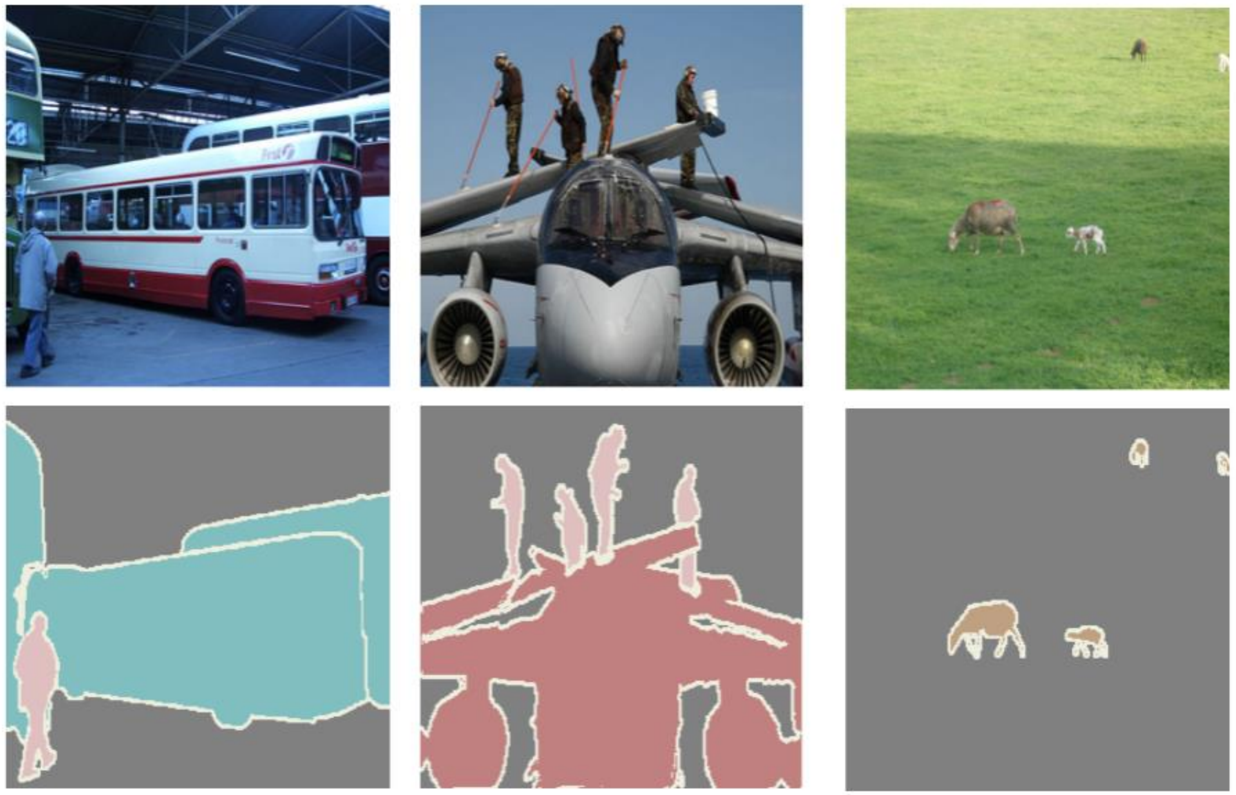


**Fig. 2**. Sample images from the VOC2012 dataset. The top row displays original images, while the bottom row highlights the objects with the background shaded in gray.

# Feature Mapping Approach (FMA) for Learning Tasks

📌 **Predict an attribute Y from image X**

◆ Often modeled as $Y = f^*(X, \epsilon)$, where $f^*$ is the target function and $\epsilon$ is the noise. The efficiency of learning methods, especially FNNs, relies on intrinsic structures in X and $f^*$ and the adaptation of FNNs to these structure(Schmidt-Hieber, 2019; 2020; Jiao et al., 2023; Zhou & Huo,2024).

◆ For image data, we redefine $X$ as $X = X_{obj} \oplus \epsilon$ and express the relationship as $Y = f^*(X_{obj})$ where $X_{obj}$ includes the objects of interest relevant to $Y$ , and $\epsilon$ encompasses irrelevant objects and noise.

◆ $Y = f^*(X_{obj})$ assumes that X contains all necessary information to infer $Y$ through $X_{obj}$, allowing estimators to achieve near-perfect performance. This aligns with the success of deep neural networks in vision tasks (Wang et al., 2017; Langer & Schmidt-Hieber, 2022).

◆ The model fundamentally differs from conventional regression models $Y = f^*(X) + \epsilon$. No estimator can perfectly predict $Y$ , as $E|f(X) - Y|^2 \geq \epsilon^2$.

◆ In CV, $X_{obj}$ can be described using low-dimensional structures like edges and textures (Zhao et al., 2019; Zou et al., 2023). We assume $X_{obj}$ encapsulates all relevant information for inferring $Y$ via $M^*$ local patterns $H_1, \ldots, H_{M^*} \in [0,1]^{k^* \times k^*}$. Each local pattern is a $k^* \times k^*$ sub-image within the larger image domain $\Omega$.

◆ Using the widely adopted sliding window approach in CV, we define $\rho(X_{obj}, H_m) \in R^{(n-k^*+1)\times(n-k^*+1)}$ to represent the feature map of $X_{obj}$ with respect to $H_m$.

◆ The target link $f^*$ for the relationship between Y and $X_{obj}$ is characterized as

$$Y = F^*(\rho(X_{obj}, H_1), \ldots, \rho(X_{obj}, H_{M^*}))$$

where $F^*$ is a nonlinear mapping.

# Informative Metrics of FMA

## Object Size

◆ Quantifies the spatial extent of objects in an image, which is crucial for object recognition and various image processing tasks (Ghassemian & Landgrebe,1988; Bainbridge & Oliva, 2015). Structure (Schmidt-Hieber, 2019; 2020; Jiao et al., 2023; Zhou & Huo,2024).

$$\beta = \frac{|mask(X_{obj})|}{|\Omega|}$$

◆ Larger objects lead to proportionally larger $k^* \times k^*$ local patterns, which in turn reduce the dimensionality of the feature maps to $(n - k^*+1) \times (m - k^*+1)$. This reduction facilitates learning the target function $F^*$ , enhancing both the efficiency and effectiveness of the learning process.

## Number of Objects

◆ Quantifies scene composition and complexity

$$J \sim Possion(\Lambda(\Omega))$$

◆ Images with multiple objects provide stronger and robust signals in the feature map, enhancing the learning of $F^*$ in tasks such as object detection and segmentation (Lempitsky & Zisserman, 2010).

## Spatial distribution of objects

◆ Captures their arrangement within an image, influencing tasks such as localization, segmentation, and scene understanding (Schauerte& Stiefelhagen, 2013; Andrianov et al., 2015).

$$S(\lambda) := -\sum_{u \in \Omega} p_\lambda(u) \log(p_\lambda(u))$$

◆ S(λ) reaches its maximum for a uniform λ, indicating evenly distributed locations, and is minimized at 0 when $p_\lambda$ concentrates at a single point for deterministic placement.

## Informative Metrics of FMA

**Signal-to-Noise Ratio (SNR)**

- $SNR(X, H_m) := \frac{\|\rho(X_{obj}, H_m)\|}{\|\rho(X, H_m) - \rho(X_{obj}, H_m)\|}$

- Noise in the feature maps $\rho(X, H_m) - \rho(X_{obj}, H_m)$, i.e., the difference between the feature maps derived from the image $X$ and the object $X_{obj}$.

- A larger SNR facilitates the learning of the downstream nonlinear map $F^*$ , leading to more accurate predictions, while a smaller SNR increases the difficulty in distinguishing object patterns from noise, reflecting greater learning challenges. SNR can be influenced by factors such as object size and count with larger objects and higher counts naturally in-crease the signal-to-noise ratio.

# Experiments on MNIST Dataset

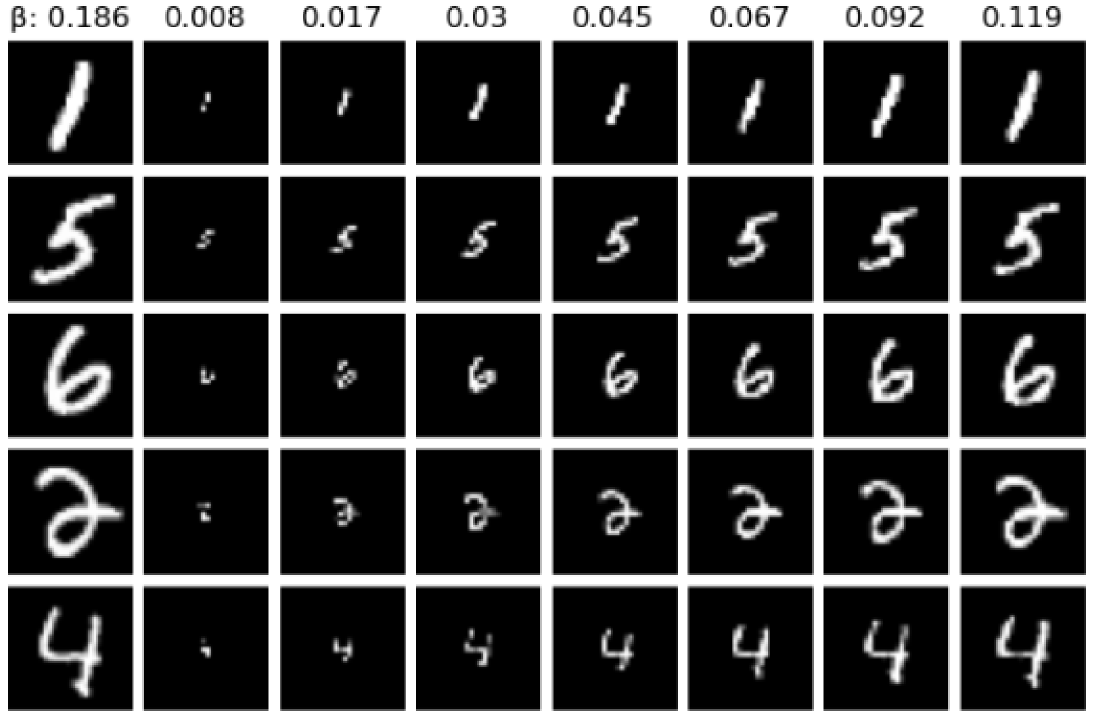
## CNN Configuration

- For our simulations, we train 5-layer CNNs with a fixed architecture under various scenarios.
- The architecture includes the following components:
  - a convolutional layer C(16, 3, 1, 1) with 16 filters of size $3 \times 3$, a stride of 1, and padding of 1;
  - a non-overlapping max-pooling layer with a window size of 2 and a stride of 2;
  - another convolutional layer C(32, 3, 1, 1) with 32 filters of size $3 \times 3$, a stride of 1, and padding of 1;
  - a second non-overlapping max-pooling layer with a window size of 2 and a stride of 2;
  - and a fully connected layer with an input dimension of $32 \times 7 \times 7$ and an output dimension of 10.

## Image Manipulation

- Resize the handwritten digits
- Duplicate the number of objects
- Change spatial distribution patterns for the objects

## Experiments on MNIST Dataset



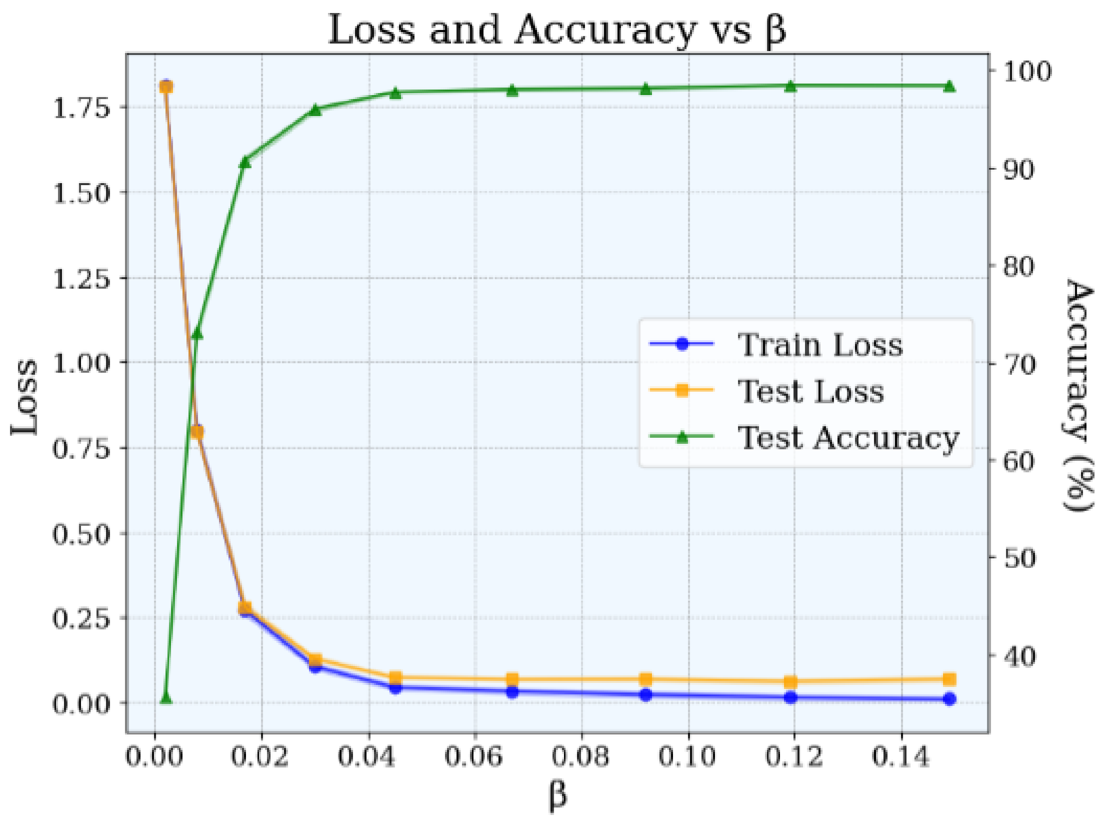
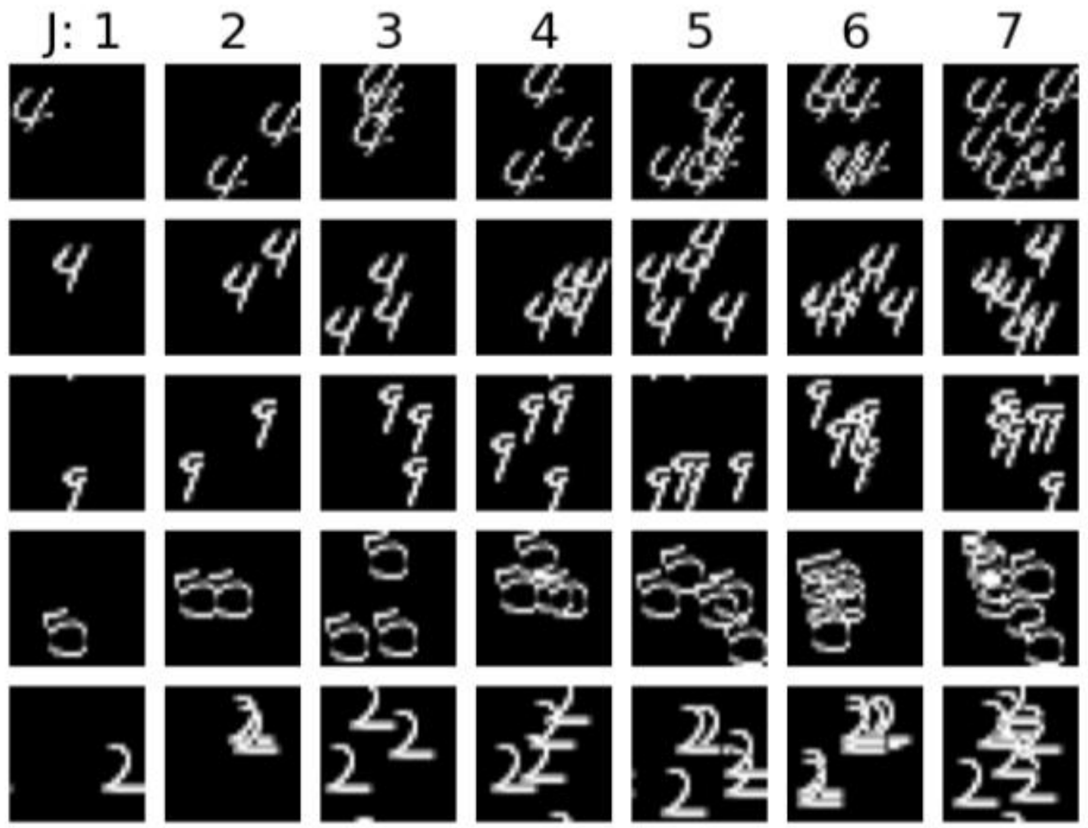


**Fig. 3**. Performance of CNNs trained on modified MNIST datasets. Columns in the left figure present samples of original (β = 0.186) and modified datasets with different sizes of objects where β denotes theaverage size in the datasets.

# Experiments on MNIST Dataset



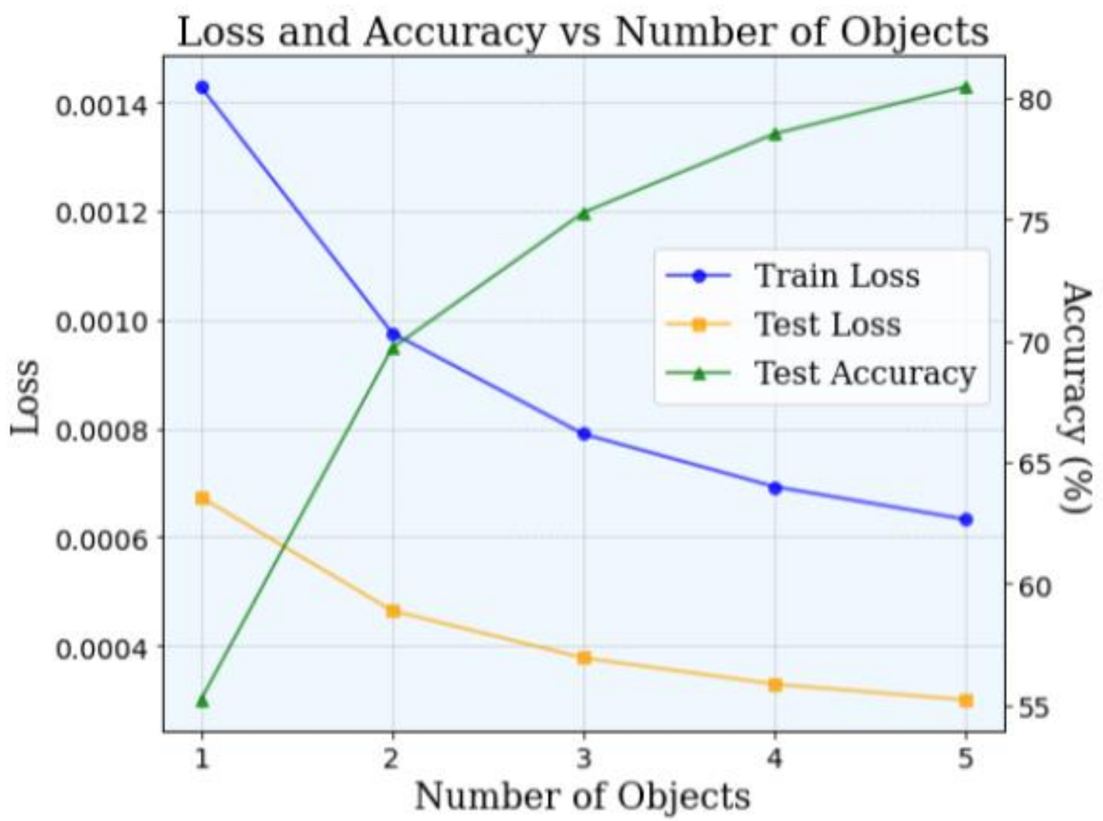
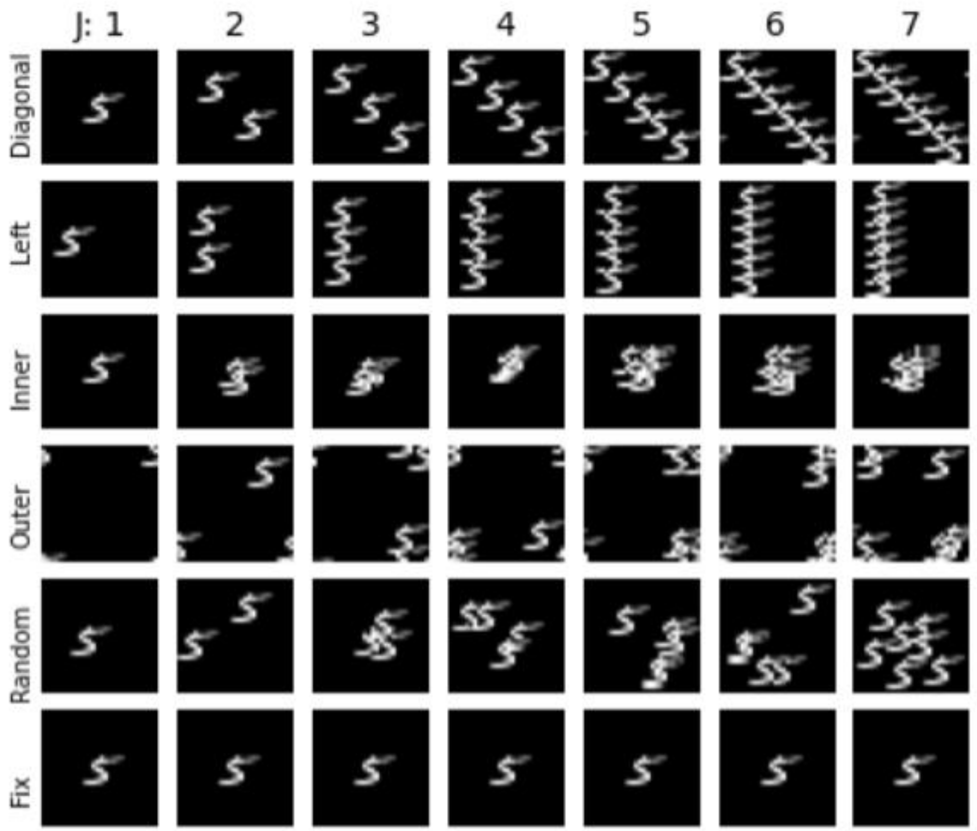


**Fig. 4**. Performance of CNNs trained on modified MNIST datasets. Columns in the left figure present samples of modified datasets with different numbers of objects J.

## Experiments on MNIST Dataset



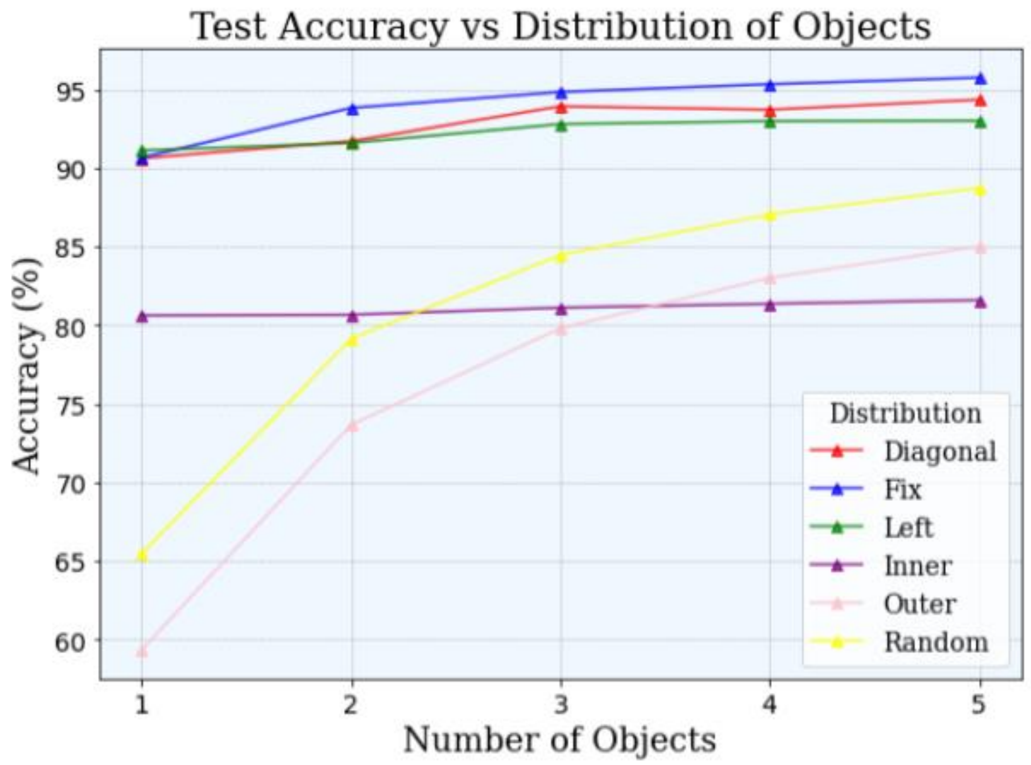
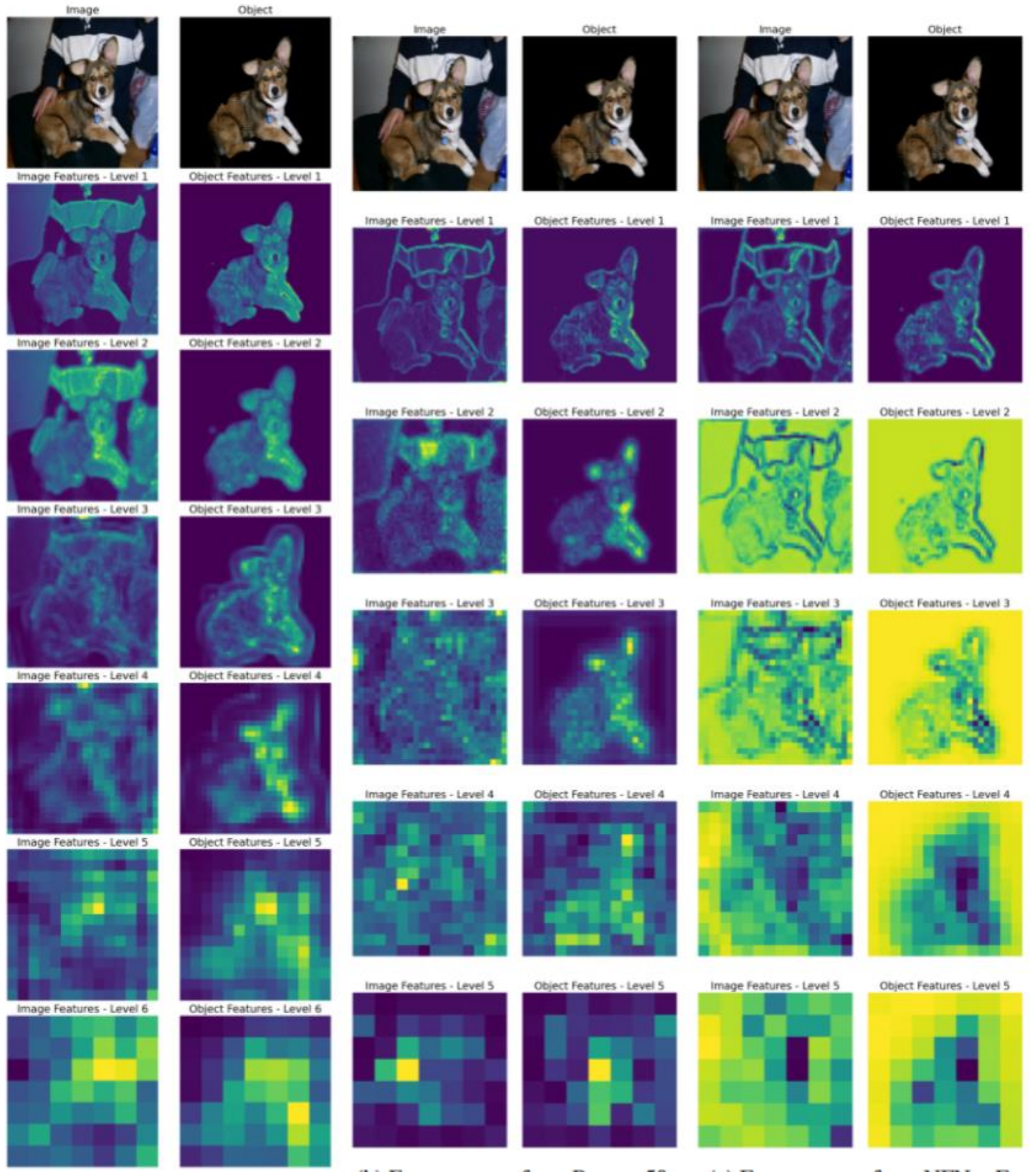


**Fig. 5**. Performance of CNNs trained on modified MNIST datasets. Rows in the left figure present samples of modified datasets with different distributions of objects given different numbers of objects J.

# Experiments on ImageNet Dataset

📌 **Feature Visualization of original image vs object segmentation**

🔷 Sample image from ImageNet. For each subfigure, the top row displays original image and the object segmentation by “SAM2" (Ravi et al., 2024). The rest rows visualize the feature maps calculated from the pretrained models at different layers (levels).



## Experiments on ImageNet Dataset

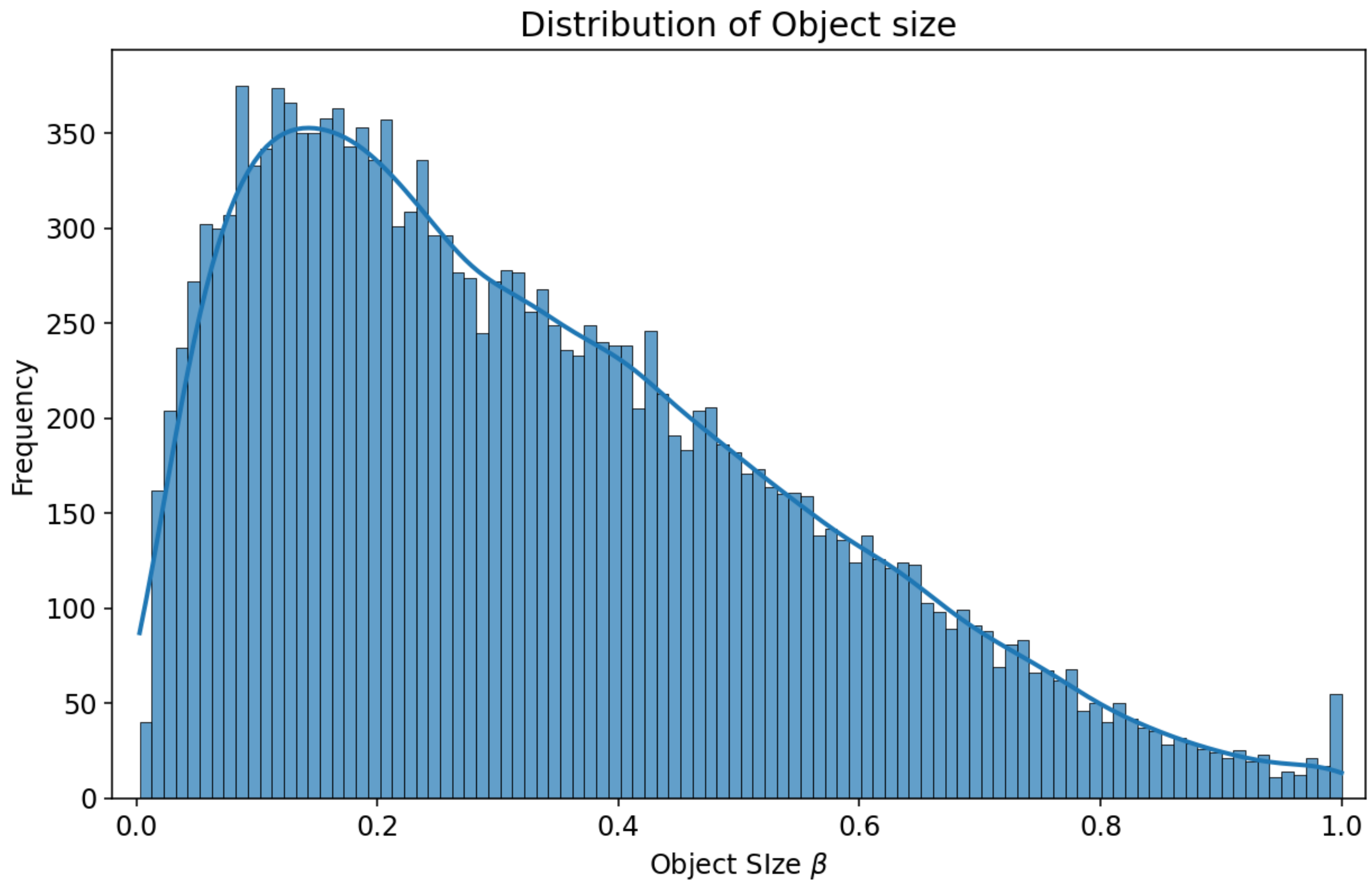


**Fig. 6**. The distribution of the object size of the subset of 17,489 ImageNet images

# Experiments on ImageNet Dataset

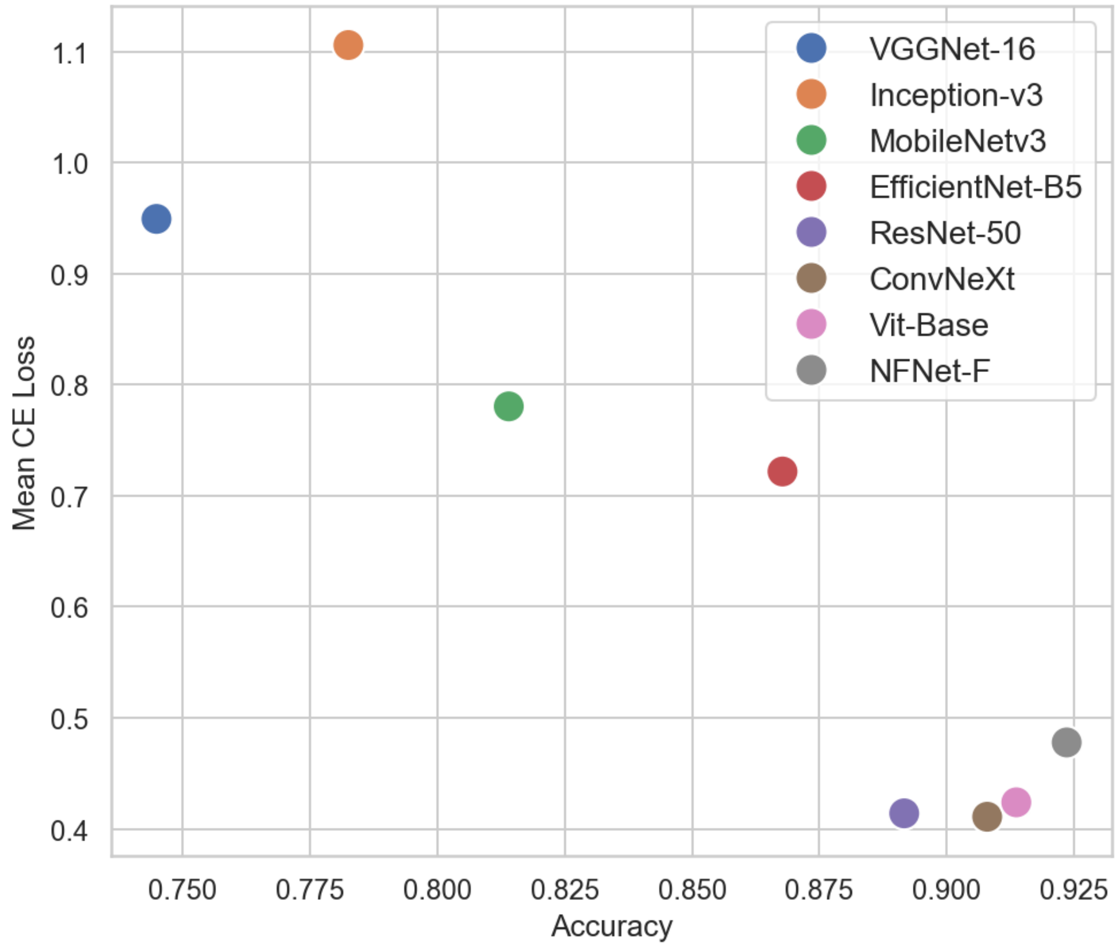


**Fig. 7**. The accuracy and the averaged CE loss of the pretrained models on a subset of 17,489 ImageNet images.

## Experiments on ImageNet Dataset

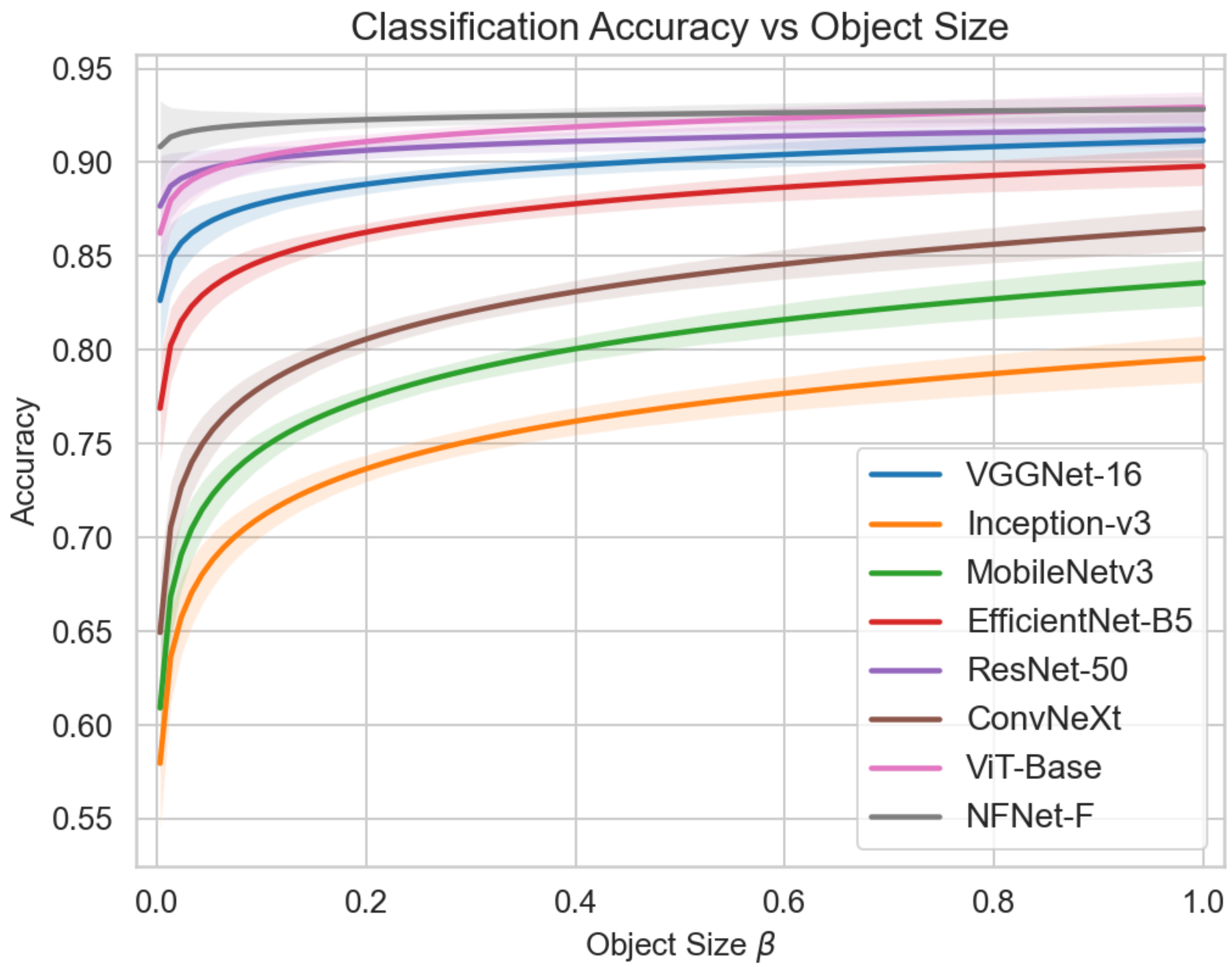


**Fig. 8**. Regression Curves of Classification Accuracy vs. Object Size with 95% confidence bands for pretrained models on a subset of 17,489 ImageNet images.

# Experiments on ImageNet Dataset

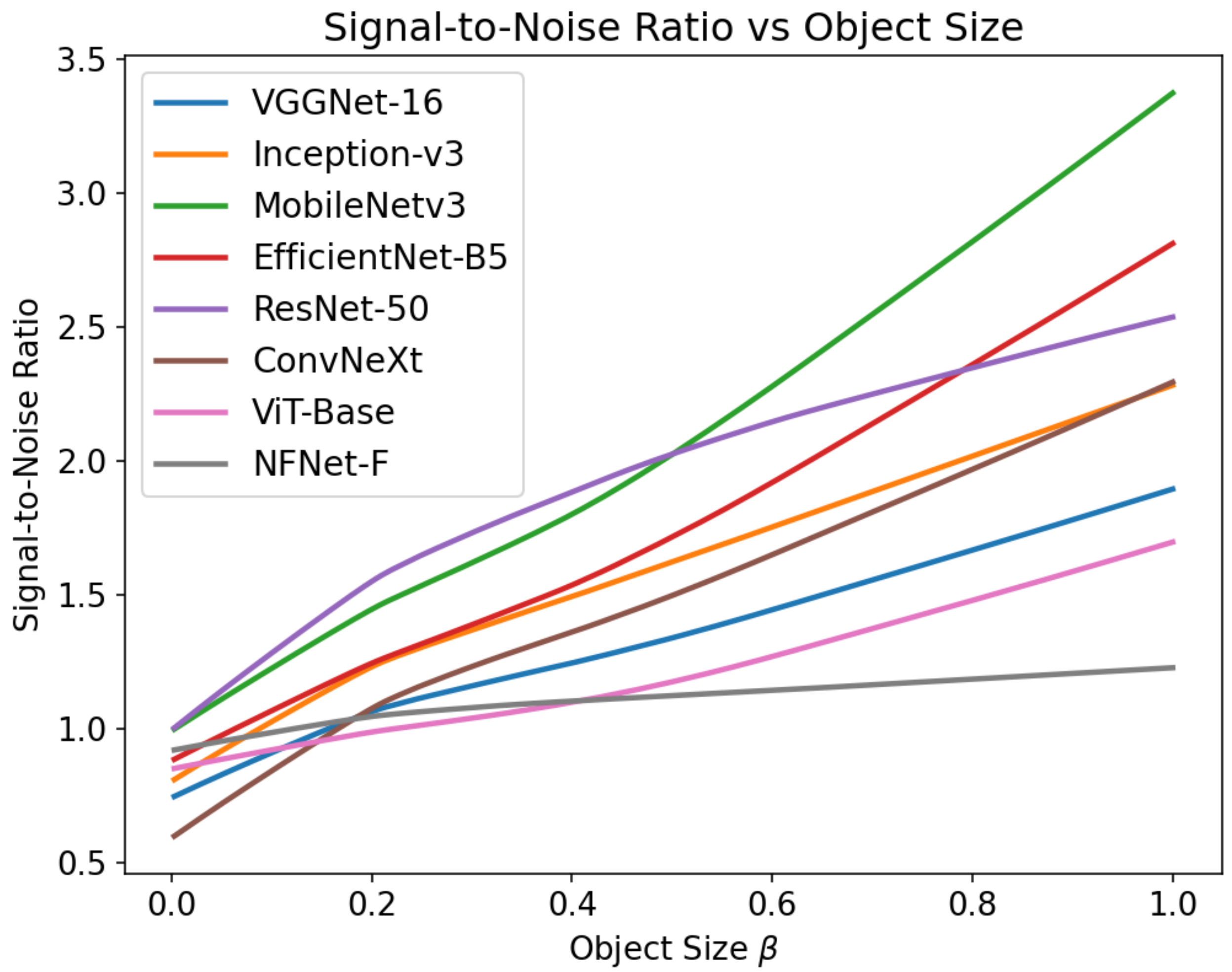


**Fig. 8**. Regression Curves of Signal-to-Noise Ratio vs. Object Size for pretrained models on asubset of 17,489 ImageNet images.

## Experiments on ImageNet Dataset

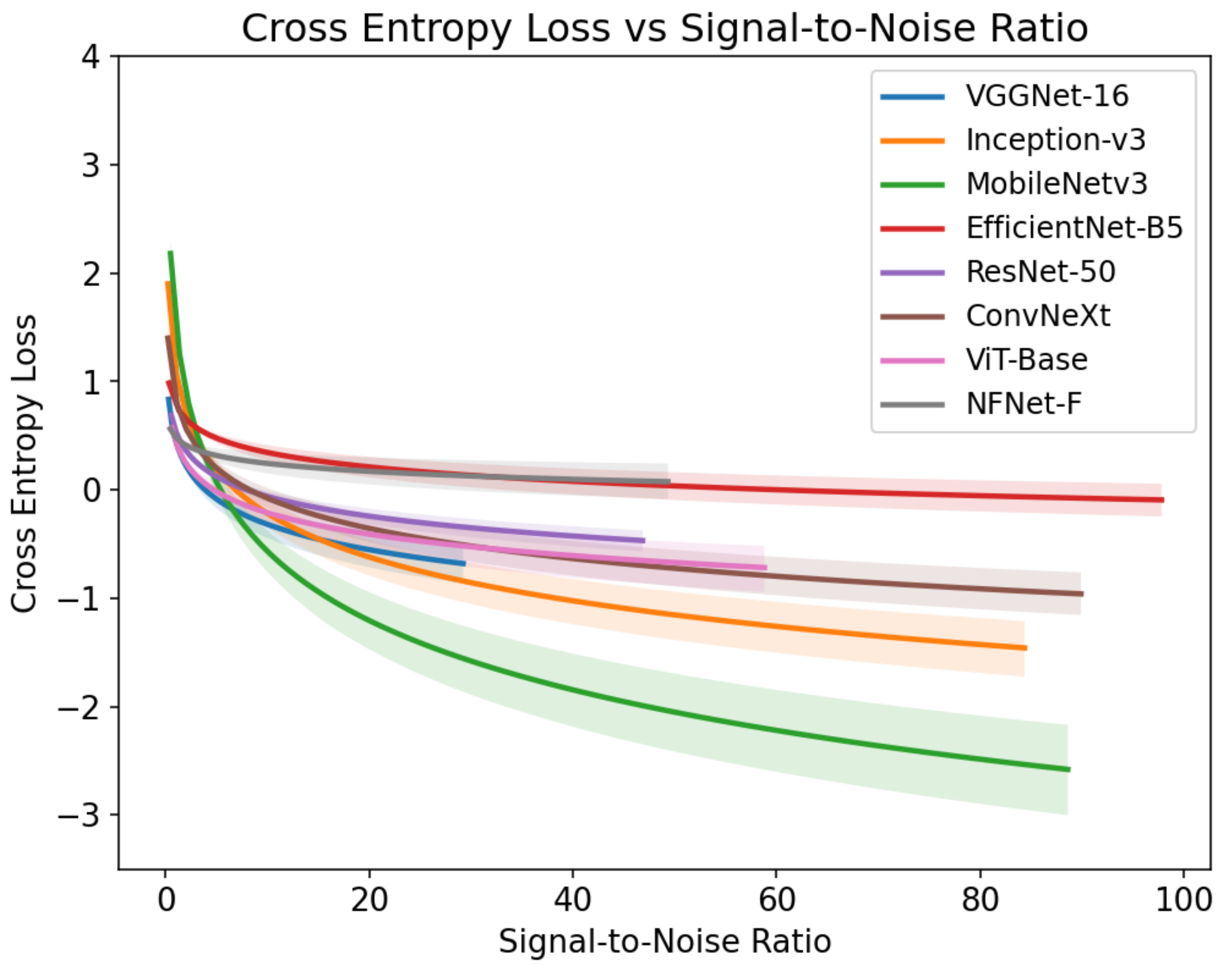


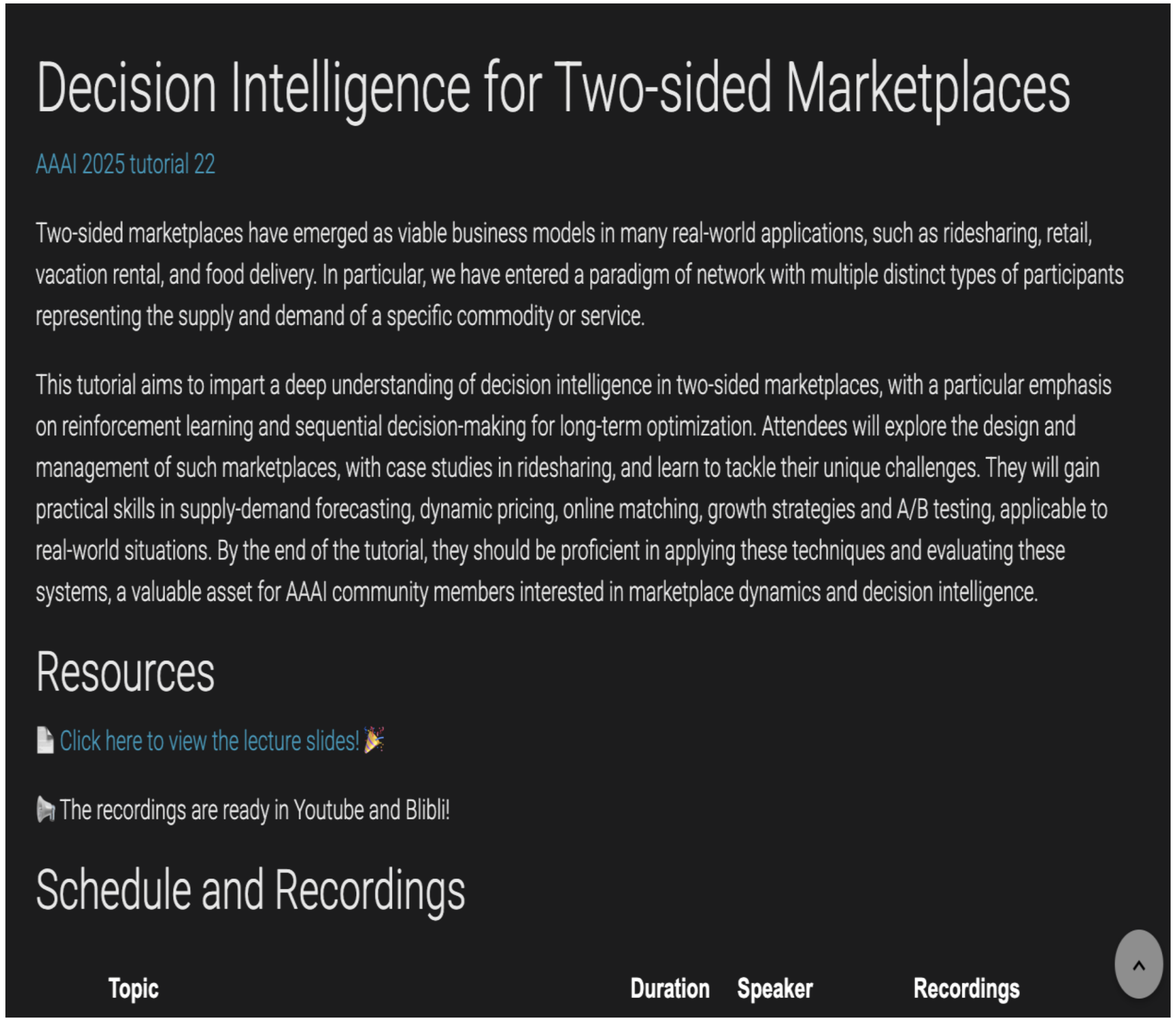
**Fig. 9**. Regression Curves of Cross Entropy Loss vs. Signal-to-Noise Ratio for pretrained modelson a subset of 17,489 ImageNet images.

## References

1. Allen-Zhu, Z., Li, Y., and Song, Z. (2019). A convergence theory for deep learning via over-parameterization. In International conference on machine learning, pages 242–252. PMLR.
2. Andrianov, D., Eremeev, S., and Kuptsov, K. (2015). The review of spatial objects recognition models and algorithms. Procedia Engineering, 129:374–379.
3. Jiao, Y., Shen, G., Lin, Y., and Huang, J. (2023). Deep nonparametric regression on approximate manifolds: Nonasymptotic error bounds with polynomial prefactors. The Annals of Statistics, 51(2):691–716.
4. Langer, S. and Schmidt-Hieber, J. (2022). A statistical analysis of an image classification problem. arXiv preprint arXiv:2206.02151.
5. Lin, S. and Zhang, J. (2019). Generalization bounds for convolutional neural networks. arXiv preprint arXiv:1910.01487.
6. Lempitsky, V. and Zisserman, A. (2010). Learning to count objects in images. Advances in Neural Information Processing Systems, 23.
7. Petersen, P. and Voigtlaender, F. (2020). Equivalence of approximation by convolutional neural networks and fully-connected networks. Proceedings of the American Mathematical Society, 148(4):1567–1581.
8. Schauerte, B. and Stiefelhagen, R. (2013). How the distribution of salient objects in images influences salient object detection. 2013 IEEE International Conference on Image Processing, 74–78.
9. Schmidt-Hieber, J. (2019). Deep ReLU network approximation of functions on a manifold. arXiv preprint arXiv:1908.00695.
10. Shen, G. (2024). Exploring the complexity of deep neural networks through functional equivalence. In Forty-first International Conference on Machine Learning.
11. Zhou, D.-X. (2020a). Theory of deep convolutional neural networks: Downsampling. Neural Networks,481124:319–327.
12. Zhou, T.-Y. and Huo, X. (2024). Classification of data generated by gaussian mixture models using deep ReLU networks. Journal of Machine Learning Research, 25(190):1–54.
13. Zhou, P. and Feng, J. (2018). Understanding generalization and optimization performance of deep CNNs. International Conference on Machine Learning, 5960–5969.

# Course Websites

https://twosidedm.github.io/



Decision Intelligence for Two-sided Marketplaces

AAAI 2025 tutorial 22

Two-sided marketplaces have emerged as viable business models in many real-world applications, such as ridesharing, retail, vacation rental, and food delivery. In particular, we have entered a paradigm of network with multiple distinct types of participants representing the supply and demand of a specific commodity or service.

This tutorial aims to impart a deep understanding of decision intelligence in two-sided marketplaces, with a particular emphasis on reinforcement learning and sequential decision-making for long-term optimization. Attendees will explore the design and management of such marketplaces, with case studies in ridesharing, and learn to tackle their unique challenges. They will gain practical skills in supply-demand forecasting, dynamic pricing, online matching, growth strategies and A/B testing, applicable to real-world situations. By the end of the tutorial, they should be proficient in applying these techniques and evaluating these systems, a valuable asset for AAAI community members interested in marketplace dynamics and decision intelligence.

Resources

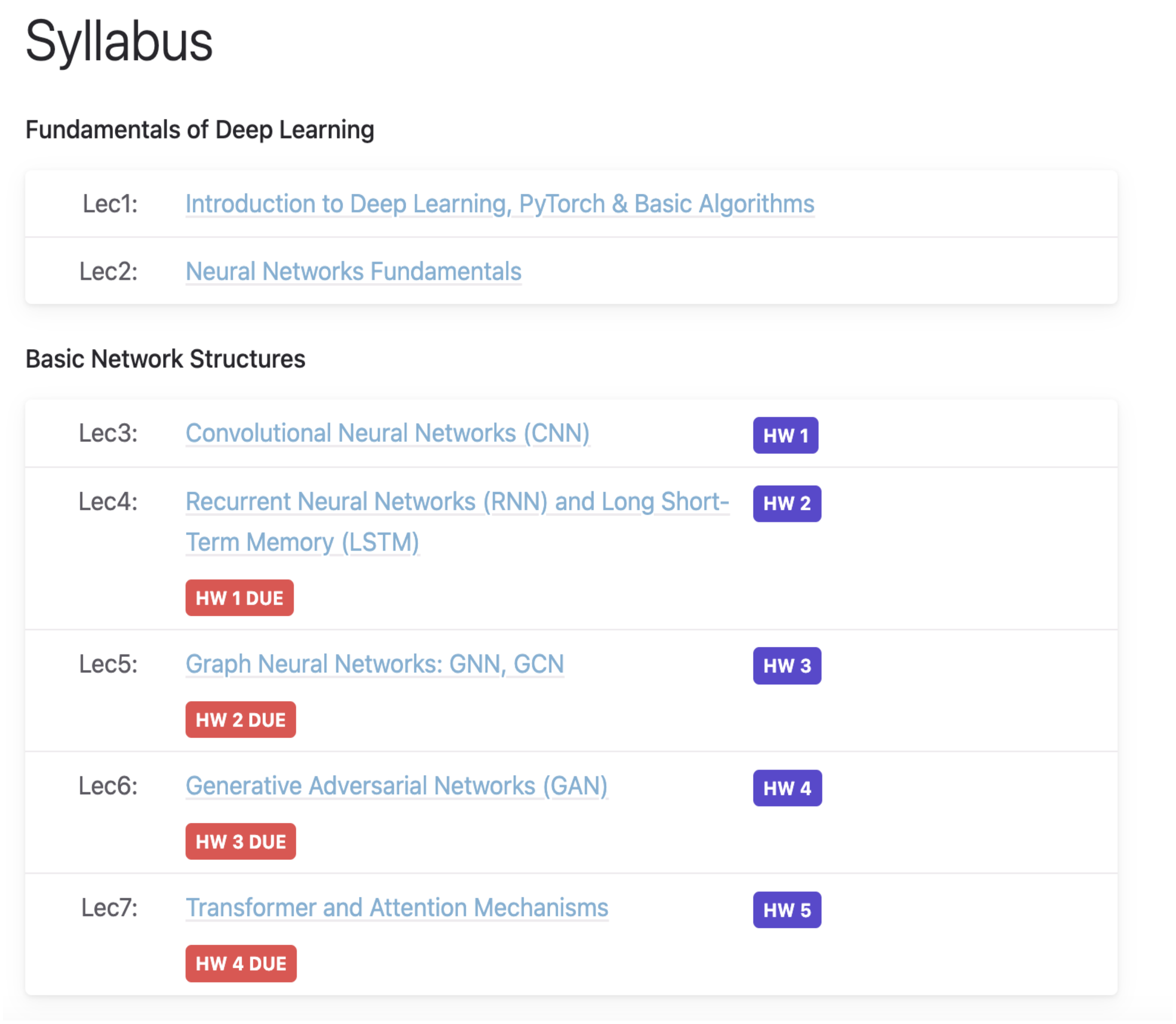
Click here to view the lecture slides!

The recordings are ready in Youtube and Blibli!

Schedule and Recordings

| Topic | Duration | Speaker | Recordings |
|---|---|---|---|

https://bios740.github.io/



Syllabus

Fundamentals of Deep Learning

| | | |
|---|---|---|
| Lec1: | Introduction to Deep Learning, PyTorch & Basic Algorithms | |
| Lec2: | Neural Networks Fundamentals | |

Basic Network Structures

| | | |
|---|---|---|
| Lec3: | Convolutional Neural Networks (CNN) | HW 1 |
| Lec4: | Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) HW 1 DUE | HW 2 |
| Lec5: | Graph Neural Networks: GNN, GCN HW 2 DUE | HW 3 |
| Lec6: | Generative Adversarial Networks (GAN) HW 3 DUE | HW 4 |
| Lec7: | Transformer and Attention Mechanisms HW 4 DUE | HW 5 |

## Acknowledgement

UNC | GILLINGS SCHOOL OF GLOBAL PUBLIC HEALTH

**Funding:** U.S. NIH Grants R01AG082938, 1R01AG085581, 1R01MH136055, and R01AR082684.