Influence Functions for Neural Network Inference: Validation Across Exponential Family Models

Author Name
author@email.com

January 5, 2026

Abstract

We implement and validate the influence function approach of Farrell et al. (2021) for valid inference with neural network estimators. Neural networks estimate structural parameters directly—baseline effects $\alpha(X)$ and treatment effects $\beta(X)$—while influence functions correct for regularization bias to yield valid confidence intervals.

The `deepstats` package implements this framework for nine exponential family models: linear (Gaussian), gamma, Gumbel, Poisson, logit (Bernoulli), Tobit (censored), negative binomial, Weibull, and heteroskedastic linear. K-fold cross-fitting with influence function corrections produces $\sqrt{n}$-consistent and asymptotically normal estimators for the average treatment effect $\mu^* = E[\beta(X)]$.

Monte Carlo simulations validate the approach. On linear and logit models, the influence function method achieves 90–93% coverage of nominal 95% confidence intervals with well-calibrated standard errors (SE ratio near 1.0). In contrast, naive estimation without influence corrections achieves only 3–10% coverage, severely underestimating uncertainty. These results demonstrate that influence function corrections are essential for valid inference with regularized neural network estimators.

1 Introduction

Neural networks offer flexible function approximation for complex relationships in observational data. However, standard neural network training with regularization (weight decay, dropout, early stopping) introduces bias that invalidates traditional inference procedures. This paper implements and validates the influence function approach of Farrell et al. (2021) for constructing valid confidence intervals with neural network estimators.

The Problem. Consider estimating heterogeneous treatment effects from the structural model:

$$Y_i = \alpha(X_i) + \beta(X_i) \cdot T_i + \varepsilon_i$$

where $\alpha(X)$ captures baseline effects, $\beta(X)$ captures the conditional average treatment effect (CATE), and $T \in \{0, 1\}$ is a binary treatment. Neural networks can estimate both functions flexibly, but regularization shrinks predictions toward zero, reducing estimated variance.

Naive standard errors computed as $\text{SE} = \text{std}(\hat{\beta})/\sqrt{n}$ severely underestimate uncertainty. In our Monte Carlo simulations, naive 95% confidence intervals achieve only 3–10% coverage—a catastrophic failure of inference.

The Solution. The influence function framework of Farrell et al. (2021) provides valid inference by correcting for regularization bias. The key insight is that cross-fitting combined with influence function corrections yields $\sqrt{n}$-consistent and asymptotically normal estimators for the average treatment effect $\mu^* = E[\beta(X)]$.

The influence function approach:

1. Uses K-fold cross-fitting to avoid overfitting bias
2. Computes Hessian-based corrections for regularization bias
3. Produces standard errors that achieve the semiparametric efficiency bound

Contributions. This paper makes three contributions:

1. **Implementation.** We implement the FLM influence function framework for nine exponential family models: linear (Gaussian), gamma, Gumbel, Poisson, logit (Bernoulli), Tobit (censored), negative binomial, Weibull, and heteroskedastic linear. Each family specifies the appropriate loss function, residuals, and Hessian weights for influence computation.
2. **Validation.** Monte Carlo simulations on linear and logit models demonstrate that the influence function method achieves 90–93% coverage of nominal 95% confidence intervals, with well-calibrated standard errors (SE ratio near 1.0). Naive estimation without corrections achieves only 3–10% coverage.
3. **Software.** The `deepstats` Python package provides a simple API for practitioners: `get_dgp()`, `get_family()`, and `influence()` functions handle data generation, model specification, and inference respectively. A command-line interface enables reproducible Monte Carlo studies.

Organization. Section 2 presents the structural model and exponential family extensions. Section 3 describes the cross-fitting procedure. Section 4 establishes asymptotic normality. Section 5 reports Monte Carlo validation results. Section 6 describes the software implementation. Section 7 concludes.

2 Model Specification

2.1 Enriched Structural Model

We consider the enriched structural model of Farrell et al. (2021):

$$Y_i = \alpha(X_i) + \beta(X_i) \cdot T_i + \varepsilon_i, \quad E[\varepsilon_i | X_i, T_i] = 0 \tag{1}$$

where $\alpha : \mathcal{X} \to \mathbb{R}$ captures baseline effects, $\beta : \mathcal{X} \to \mathbb{R}$ captures the conditional average treatment effect (CATE), $T_i \in \{0, 1\}$ is the binary treatment indicator, and $X_i \in \mathcal{X} \subseteq \mathbb{R}^d$ are covariates.

The target parameter is the average treatment effect:

$$\mu^* = E[\beta(X)] \tag{2}$$

Both $\alpha(\cdot)$ and $\beta(\cdot)$ are estimated using neural networks. The key challenge is that neural network regularization (weight decay, dropout, early stopping) introduces bias that must be corrected for valid inference.

2.2 Exponential Family Extension

The framework extends to exponential family distributions through appropriate link functions. Table 1 summarizes the nine families implemented in the `deepstats` package.

Table 1: Implemented Exponential Family Models

Family	Distribution	Link	Conditional Mean
`linear`	$Y \sim N(\mu, \sigma^2)$	Identity	$\mu = \alpha + \beta T$
`gamma`	$Y \sim \text{Gamma}(k, \mu/k)$	Log	$\mu = \exp(\alpha + \beta T)$
`gumbel`	$Y \sim \text{Gumbel}(\mu, s)$	Identity	$\mu = \alpha + \beta T$
`poisson`	$Y \sim \text{Poisson}(\lambda)$	Log	$\lambda = \exp(\alpha + \beta T)$
`logit`	$Y \sim \text{Bernoulli}(p)$	Logit	$p = \sigma(\alpha + \beta T)$
`tobit`	$Y = \max(0, Y^*)$	Identity	$Y^* = \alpha + \beta T + \varepsilon$
`negbin`	$Y \sim \text{NegBin}(\mu, r)$	Log	$\mu = \exp(\alpha + \beta T)$
`weibull`	$Y \sim \text{Weibull}(k, \lambda)$	Log	$\lambda = \exp(\alpha + \beta T)$
`heterolinear`	$Y \sim N(\mu, \sigma^2(X))$	Identity	$\mu = \alpha + \beta T$

Notes: For logit, $\sigma(\cdot)$ is the sigmoid function. For tobit, $Y^* = \alpha + \beta T + \varepsilon$ is the latent outcome. The heterolinear family estimates heteroskedastic variance $\sigma^2(X)$ as a third parameter.

Each family specifies:

- **Loss function**: Negative log-likelihood for training
- **Residual**: For influence function computation
- **Hessian weight**: For the Λ matrix in influence scores
- **Target functional**: $H(\theta) = \beta$ for most families

2.3 Network Architecture

The `StructuralNet` architecture consists of:

1. A shared multi-layer perceptron (MLP) backbone with ReLU activations
2. Separate linear output heads for $\alpha(X)$ and $\beta(X)$

Default configuration: hidden layers $[64, 32]$, dropout $p = 0.1$, weight decay $\lambda = 10^{-4}$.

For treatment effect estimation, the network is trained by minimizing the negative log-likelihood of the specified family. Regularization through early stopping based on validation loss prevents overfitting while allowing the network to capture complex heterogeneity patterns in $\beta(X)$.

A separate `NuisanceNet` estimates the propensity score $e(X) = P(T = 1|X)$ and treatment variance $\text{Var}(T|X)$ for use in the influence function correction.

3 Estimation

This section details the estimation procedure, including cross-fitting, training, and influence function computation.

3.1 Cross-Fitting Procedure

We use K-fold cross-fitting to ensure out-of-sample predictions. Each fold's model is trained on $(K-1)/K$ of the data, then predictions and influence scores are computed on the held-out $1/K$. With $K = 50$ folds, each model sees 98% of the data, yielding stable estimates while maintaining independence between training and evaluation.

Algorithm 1 Influence Function Estimation with Cross-Fitting

Input: Data $(Y_i, T_i, X_i)_{i=1}^n$, number of folds K
Partition $\{1, \ldots, n\}$ into K folds $\mathcal{I}_1, \ldots, \mathcal{I}_K$
for $k = 1$ to K **do**
  Let $\mathcal{I}_{-k} = \{1, \ldots, n\} \setminus \mathcal{I}_k$ (training set)
  Train `StructuralNet` on $\{(Y_i, T_i, X_i) : i \in \mathcal{I}_{-k}\}$
  Train `NuisanceNet` on $\{(T_i, X_i) : i \in \mathcal{I}_{-k}\}$
  Compute Hessian: $\Lambda_k = \frac{1}{|\mathcal{I}_{-k}|} \sum_{j \in \mathcal{I}_{-k}} W_j \tilde{T}_j^{\otimes 2}$
  for $i \in \mathcal{I}_k$ **do**
    Compute $\hat{\theta}_i = [\hat{\alpha}(X_i), \hat{\beta}(X_i)]$
    Compute influence score: $\psi_i = H(\hat{\theta}_i) - \nabla \ell_i^\top \Lambda_k^{-1} \nabla H$
  end for
end for
Output: $\hat{\mu} = \frac{1}{n} \sum_i \psi_i$, $\text{SE} = \text{std}(\psi)/\sqrt{n}$

Key notation:

- W_i: Hessian weight from family (e.g., $W_i = 1$ for linear, $W_i = p_i(1 - p_i)$ for logit)
- $\tilde{T}_i = [1, T_i - \bar{T}]$: Centered treatment vector
- $H(\theta) = \beta$: Target functional (the treatment effect parameter)
- $\nabla \ell_i$: Gradient of loss with respect to θ at observation i
- $\nabla H = [0, 1]$: Gradient of target with respect to θ

The recommended setting is $K = 50$ folds, which ensures 98% of data is used for training each model while maintaining independence between training and evaluation.

3.2 Training Objective

For exponential family outcomes, the neural network parameters are estimated by minimizing the negative log-likelihood:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i \in \mathcal{I}_{-k}} -\ell(Y_i; \alpha_\theta(X_i), \beta_\theta(X_i), T_i)$$

where ℓ is the family-specific log-likelihood.

Training uses the Adam optimizer with the following regularization:

- **Dropout**: Rate $p = 0.1$ on hidden layers
- **Weight decay**: L2 regularization $\lambda = 10^{-4}$
- **Early stopping**: Patience of 10 epochs based on validation loss

3.3 Influence Function

The influence function for estimating $\mu^* = E[\beta(X)]$ takes the form:

$$\psi_i = H(\hat{\theta}_i) - \nabla \ell_i^\top \Lambda^{-1} \nabla H \tag{3}$$

For the linear family with target $H(\theta) = \beta$:

- $\nabla \ell_i = (Y_i - \hat{\alpha}_i - \hat{\beta}_i T_i) \cdot [1, \tilde{T}_i]$
- $\Lambda = \frac{1}{n} \sum_j \tilde{T}_j^{\otimes 2}$
- $\nabla H = [0, 1]$

The Hessian correction term $\nabla \ell_i^\top \Lambda^{-1} \nabla H$ accounts for the bias introduced by regularization. Without this correction, standard errors are severely underestimated.

3.4 Average Treatment Effect Estimation

The ATE estimator and its standard error are:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} \psi_i \tag{4}$$

$$\widehat{\mathrm{SE}}(\hat{\mu}) = \frac{1}{\sqrt{n}} \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\psi_i - \hat{\mu})^2} \tag{5}$$

The 95% confidence interval is:

$$\mathrm{CI}_{95\%} = \left[\hat{\mu} - 1.96 \cdot \widehat{\mathrm{SE}},\ \hat{\mu} + 1.96 \cdot \widehat{\mathrm{SE}} \right]$$

3.5 Implementation Details

The default configuration in `deepstats`:

- Cross-fitting with $K = 50$ folds
- MLP backbone: hidden layers $[64, 32]$
- Adam optimizer: learning rate 0.01
- Maximum 100 epochs with early stopping (patience 10)
- Batch size 64
- Validation split 10%

A separate `NuisanceNet` estimates $E[T|X]$ for centering the treatment in the Hessian computation: $\tilde{T}_i = T_i - E[T|X_i]$.

4 Asymptotic Theory

This section establishes the asymptotic properties of the cross-fitted neural network estimator following Farrell et al. (2021). The key result is that influence function corrections yield $\sqrt{n}$-consistent and asymptotically normal estimators for the average treatment effect.

4.1 Setup and Assumptions

Consider the enriched structural model:

$$Y_i = a_0(X_i) + b_0(X_i) \cdot T_i + \varepsilon_i, \quad E[\varepsilon_i | X_i, T_i] = 0 \tag{6}$$

where $a_0 : \mathcal{X} \to \mathbb{R}$ is the baseline function, $b_0 : \mathcal{X} \to \mathbb{R}$ is the conditional average treatment effect (CATE), and $\mathcal{X} \subseteq \mathbb{R}^d$ is the covariate space.

The parameter of interest is the average treatment effect:

$$\tau_0 = E[b_0(X)] = E[Y(1) - Y(0)] \tag{7}$$

We estimate a_0 and b_0 using neural networks $\hat{a}$ and $\hat{b}$ with cross-fitting, then form the ATE estimator:

$$\hat{\tau} = \frac{1}{n} \sum_{i=1}^{n} \hat{\psi}_i \tag{8}$$

where $\hat{\psi}_i$ is the influence function evaluated at observation i.

Assumption 1 (Unconfoundedness and Overlap)**.** *(i) (Unconfoundedness)* $(Y(0), Y(1)) \perp T \mid X$

(ii) (Overlap) There exists $\delta > 0$ *such that* $\delta < e_0(X) < 1 - \delta$ *almost surely, where* $e_0(X) = P(T = 1|X)$

Assumption 2 (Boundedness)**.** *(i) The covariate space* $\mathcal{X}$ *is compact*

(ii) $|a_0(x)|, |b_0(x)| \leq M$ *for all* $x \in \mathcal{X}$ *and some* $M < \infty$

(iii) $E[\varepsilon^4|X] \leq M$ *almost surely*

Assumption 3 (Smoothness)**.** *The functions* a_0, b_0 *belong to a Hölder class* $\mathcal{H}^s(\mathcal{X})$ *with smoothness* $s > 0$*:*

$$\mathcal{H}^s = \left\{ f : \sum_{|\alpha| \leq \lfloor s \rfloor} \|D^\alpha f\|_\infty + \sum_{|\alpha| = \lfloor s \rfloor} \sup_{x \neq y} \frac{|D^\alpha f(x) - D^\alpha f(y)|}{|x - y|^{s - \lfloor s \rfloor}} \leq B \right\}$$

Assumption 4 (Neural Network Rate)**.** *The neural network estimators* $\hat{a}, \hat{b}$ *satisfy, for some* $\gamma > 1/4$*:*

$$\|\hat{a} - a_0\|_{L_2} = O_p(n^{-\gamma}), \quad \|\hat{b} - b_0\|_{L_2} = O_p(n^{-\gamma})$$

Assumption 4 is satisfied when the neural network architecture is appropriately chosen for the smoothness class. By Farrell et al. (2021), networks with depth $L = O(\log n)$ and width $W = O(n^{d/(2s+d)})$ achieve $\gamma = s/(2s + d)$, which exceeds $1/4$ when $s > d/2$.

4.2 Main Result

Theorem 1 (Asymptotic Normality of Cross-Fitted ATE Estimator)**.** *Under Assumptions 1–4, the cross-fitted estimator with* $K \geq 2$ *folds satisfies:*

$$\sqrt{n}(\hat{\tau} - \tau_0) \xrightarrow{d} N(0, V) \tag{9}$$

where

$$V = E[\psi_0(W)^2], \quad \psi_0(W) = b_0(X) + \frac{T - e_0(X)}{e_0(X)(1 - e_0(X))} (Y - a_0(X) - b_0(X)T) - \tau_0 \tag{10}$$

is the efficient influence function.

Moreover, the variance estimator

$$\hat{V} = \frac{1}{n}\sum_{i=1}^{n}\hat{\psi}_i^2, \quad \hat{\psi}_i = \hat{b}(X_i) + \frac{T_i - \hat{e}(X_i)}{\hat{e}(X_i)(1-\hat{e}(X_i))}\Big(Y_i - \hat{a}(X_i) - \hat{b}(X_i)T_i\Big) - \hat{\tau} \tag{11}$$

is consistent: $\hat{V} \xrightarrow{p} V$.

The variance V equals the semiparametric efficiency bound for estimating the ATE under model (6). Thus, the cross-fitted estimator is *asymptotically efficient*.

4.3 Proof Sketch

The proof follows Farrell et al. (2021).

Step 1: Influence Function Representation. Define the oracle influence function:

$$\psi_0(W) = b_0(X) - \tau_0 + \frac{T - e_0(X)}{e_0(X)(1-e_0(X))}\left(Y - a_0(X) - b_0(X)T\right)$$

By construction, $E[\psi_0(W)] = 0$ and $E[\psi_0(W)^2] = V$.

Step 2: Neyman Orthogonality. The score ψ satisfies Neyman orthogonality:

$$\frac{\partial}{\partial r}E[\psi(W; \eta_0 + r(\eta - \eta_0), \tau_0)]\Big|_{r=0} = 0$$

for all directions $\eta - \eta_0$ in the nuisance parameter space $\eta = (a, b, e)$. This orthogonality ensures that first-order errors in nuisance estimation have no first-order effect on the estimator.

Step 3: Cross-Fitting Decomposition. Let $\mathcal{I}_k$ denote fold k and $\hat{\eta}^{(-k)} = (\hat{a}^{(-k)}, \hat{b}^{(-k)}, \hat{e}^{(-k)})$ denote nuisance estimates from observations not in fold k. The cross-fitted estimator admits the decomposition:

$$\sqrt{n}(\hat{\tau} - \tau_0) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\psi_0(W_i) + R_n$$

where the remainder R_n satisfies $R_n = o_p(1)$.

Step 4: Remainder Bound. The remainder decomposes as:

$$R_n = \underbrace{\frac{1}{\sqrt{n}}\sum_{k}\sum_{i\in\mathcal{I}_k}\Big[\psi(W_i; \hat{\eta}^{(-k)}, \tau_0) - \psi_0(W_i)\Big]}_{R_{n,1}} + \underbrace{\sqrt{n}(\hat{\tau} - \tau_0 - \bar{\psi})}_{R_{n,2}}$$

By Neyman orthogonality and the product rate condition $\|\hat{a} - a_0\| \cdot \|\hat{b} - b_0\| = O_p(n^{-2\gamma}) = o_p(n^{-1/2})$, we have $R_{n,1} = o_p(1)$.

By the $\sqrt{n}$-rate for the sample average, $R_{n,2} = o_p(1)$.

Step 5: CLT Application. By the central limit theorem:

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} \psi_0(W_i) \xrightarrow{d} N(0, V)$$

Combined with $R_n = o_p(1)$, this yields the result. □

4.4 Confidence Intervals

Theorem 1 immediately yields asymptotically valid confidence intervals:

$$\text{CI}_{1-\alpha} = \left[\hat{\tau} - z_{\alpha/2} \frac{\sqrt{\hat{V}}}{\sqrt{n}},\ \hat{\tau} + z_{\alpha/2} \frac{\sqrt{\hat{V}}}{\sqrt{n}} \right] \tag{12}$$

where $z_{\alpha/2}$ is the $(1 - \alpha/2)$ quantile of the standard normal.

For quantile treatment effects $Q_q = \text{quantile}_q(\{b_0(X_i)\})$, the influence function approach does not directly apply. We recommend bootstrap inference:

1. Draw B bootstrap samples of size n with replacement
2. For each bootstrap sample b, compute $\hat{Q}_q^{(b)}$ from the cross-fitted ITE estimates
3. Construct percentile intervals: $[\hat{Q}_q^{(\alpha/2)}, \hat{Q}_q^{(1-\alpha/2)}]$

5 Monte Carlo Validation

We validate the influence function approach on linear and logit families.

Setup. $M = 30$ simulations, $N = 10{,}000$ observations, $K = 50$ folds. Covariates $X \sim \text{Uniform}(-1, 1)^{10}$ with nonlinear $\alpha^*(X)$ and heterogeneous $\beta^*(X)$. True $\mu^* = E[\beta^*(X)] \approx -0.168$.

Methods.

- **Naive**: Train on full data, SE = $\text{std}(\hat{\beta})/\sqrt{n}$
- **Influence**: K-fold cross-fitting with influence scores, SE = $\text{std}(\psi)/\sqrt{n}$

Table 2: Monte Carlo Results ($M = 30$, $N = 10{,}000$, $K = 50$)

Family	Method	Coverage	SE Ratio	Grade
Linear	Influence	93.3%	1.03	PASS
Linear	Naive	10.0%	0.09	FAIL
Logit	Influence	90.0%	0.93	WARNING
Logit	Naive	3.3%	0.03	FAIL

Key Finding. The influence function method achieves 90–93% coverage with SE ratios near 1.0. Naive estimation achieves only 3–10% coverage, underestimating standard errors by 91–97%. This confirms that the influence function correction is essential for valid inference with regularized neural networks.

6 Software

The `deepstats` Python package implements the FLM influence function framework.

6.1 Installation

```
pip install deepstats
```

6.2 Basic Usage

```
from deepstats import get_dgp, get_family, influence, Config

# Generate data
dgp = get_dgp("linear")
data = dgp.generate(n=1000)

# Run inference
family = get_family("linear")
config = Config(epochs=100, n_folds=50)
result = influence(data.X, data.T, data.Y, family, config)

# Results
print(f"mu = {result.mu_hat:.4f} +/- {1.96*result.se:.4f}")
```

6.3 Monte Carlo CLI

```
python -m deepstats.run_mc --M 30 --N 10000 --n-folds 50 \
    --models linear logit --methods naive influence
```

Available at: `https://github.com/rawatpranjal/deepstats`

7 Conclusion

We implement and validate the FLM influence function approach for neural network inference. Monte Carlo simulations confirm that influence function corrections achieve 90–93% coverage, while naive estimation fails catastrophically (3–10% coverage). The `deepstats` package provides this framework for nine exponential family models.

References

Farrell, M. H., Liang, T., and Misra, S. (2021). Deep neural networks for estimation and inference. *Econometrica*, 89(1):181–213.