Appendix A

In this appendix, we outline the identification of the ATE with a reflective latent outcome. However, a complete proof under the full set of regularity conditions is beyond the scope of this paper. We outline identification using: a measurement identification stage showing that observed indicators identify the latent distribution, a causal identification stage adapted to latent outcomes, and a demonstration that individual-level functional identification is required for TMLE.

Let the observed data for individual i be $\boldsymbol{o}_i = (\mathbf{x}_i, \mathbf{z}_i, \mathbf{y}_i, W_i, T_i)$, where $\mathbf{y}_i = (Y_{i1}, \ldots, Y_{iJY})$ are J observed indicators for the latent outcome η_i^Y, and $\mathbf{x}_i$, $\mathbf{z}_i$, are indicator vectors for latent covariates η_i^X and η_i^Z. Let W_i be an observed covariate and $T_i \in \{0, 1\}$ the binary treatment. Let $\mathbf{X_i} = (W_i, \eta_i^X)$ be the full covariate vector (observed and latent covariates).

Assumptions

We adopt the following assumptions.

Measurement Assumptions

M1—Local Independence

Conditional on η_i^Y, the observed indicators are mutually independent:

$$Y_{ij} \perp Y_{ij'} \mid \eta_i^Y \quad \text{for all } j \neq j'$$

An analogous condition holds for covariate indicator blocks.

M2—Correct Measurement Model

The confirmatory factor model holds, e.g., when linear:

$$Y_{ij} = \alpha_j^Y + \lambda_j^Y \eta_i^Y + \varepsilon_{ij}^Y,\ \varepsilon_{ij}^Y \sim N\left(0, \sigma_j^Y\right), \eta_i^Y \sim N\left(0, \sigma_{Y_J}^2\right)$$

In the non-parametric VAE extension M2 is replaced by correct specification of the generative architecture.

M3—Scale Normalization

The scale of the latent outcome is fixed by imposing, e.g., $\sigma^2 = 1$

M4—Sufficient Indicators

The number of observed indicators satisfies $J \geq 3$, and the factor loadings λ_j^Y are distinct and non-zero for at least three indicators.

M5—Measurement Ignorability

The indicators for η_i^Y are conditionally independent of treatment and covariates given the latent outcome:

$$\boldsymbol{y}_i \perp (T_i, \boldsymbol{X}_i) \mid \eta_i^Y$$

The treatment and covariates affect the outcome indicators through the latent outcome. An analogous condition holds for latent covariate indicators.

M6—Exclusion Restriction

Treatment affects the observed indicators only through the latent outcome:

$$Y_i(t, \eta_i^Y) = Y_i(t', \eta_i^Y) \quad \text{for all } t, t', \eta_i^Y$$

Causal Assumptions

C1—Stable Unit Treatment Value Assumption (SUTVA)

The latent and manifest potential outcomes depend only on the individual's own treatment assignment:

$$\eta_{\mathrm{i}}^{Y} \;=\; \eta_{\mathrm{i}}^{Y}(T_{\mathrm{i}}), \quad Y_{\mathrm{ij}} \;=\; Y_{\mathrm{ij}}(T_{\mathrm{i}}, \eta_{\mathrm{i}}^{Y}(T_i))$$

C2 —Treatment Ignorability
Conditional on the full covariate vector, the potential latent outcomes are independent of treatment assignment:

$$\left\{\eta_{\mathrm{i}}^{\mathrm{Y}}(0), \eta_{\mathrm{i}}^{\mathrm{Y}}(1)\right\} \;\perp\; T_{\mathrm{i}} \;\mid\; (W_{\mathrm{i}}, \eta_{\mathrm{i}}^{X})$$

C3—Common Latent Support (Positivity)
For all values in the joint support of (W_{i}, η_{i}^{X}):

$$0 \;<\; \mathrm{P}(T_{\mathrm{i}} \;=\; 1 \;\mid\; W_{\mathrm{i}} = \mathrm{w}, \eta_{\mathrm{i}}^{X} = \eta^{x}) \;<\; 1$$

for all covariate values. This ensures meaningful comparisons between treated and control units across the full covariate support.

Identification of the Average Latent Treatment Effect
Under assumptions M1–M6 and C1–C3, the average latent treatment effect

$$\delta_i^{\eta^Y} = \mathrm{E}[\eta_{\mathrm{i}}^{Y}(1) - \eta_{\mathrm{i}}^{Y}(0)]$$

is identified from the observed data distribution $P(\boldsymbol{o}_{\mathrm{i}})$ and equals:

$$\delta_i^{\eta^Y} = \mathrm{E}\left[\mathrm{E}[\eta_{\mathrm{i}}^{Y} \mid T_i \;= 1, W_i, \eta_{\mathrm{i}}^{X}]\right] - \mathrm{E}\left[\mathrm{E}[\eta_{\mathrm{i}}^{Y} \mid T_i = 0, W_i, \eta_{\mathrm{i}}^{X}]\right]$$

Furthermore, the conditional mean function

$$Q(t, w, \eta^x) = \mathrm{E}[\eta_{\mathrm{i}}^{Y} \mid T_{\mathrm{i}} = t, W_{\mathrm{i}} = w, \eta_{\mathrm{i}}^{X} \;=\; \eta^x]$$

and the propensity score

$$g(\mathrm{w}, \eta^x) \;=\; \mathrm{P}(T_{\mathrm{i}} \;=\; 1 \mid W_{\mathrm{i}} \;=\; \mathrm{w}, \eta_{\mathrm{i}}^{X} = \eta^x)$$

are identified as functions of their arguments from observed data. This functional identification is necessary and sufficient for TMLE.

Stage I: Measurement Identification of the Latent Distribution
Under M1–M4, the marginal distribution $P(\eta_{\mathrm{i}}^{Y})$ and the conditional distribution $P(\eta_{\mathrm{i}}^{X} \mid \mathbf{y}_{\mathrm{i}})$ are identified from the observed distribution $P(\mathbf{y}_i)$.
Under M2, the second-order moments of the observed indicators are:

$$\mathrm{Cov}\left(Y_{\mathrm{ij}}, Y_{\mathrm{ij}'}\right) \;=\; \lambda_{\mathrm{j}}^{Y}\lambda_{\mathrm{j}'}^{Y}\sigma_{\eta_{\mathrm{i}}^{Y}}^{2} \quad (j \neq j')$$

$$\mathrm{Var}\left(Y_{\mathrm{ij}}\right) \;=\; \left(\lambda_{\mathrm{j}}^{Y}\right)^{2}\sigma_{\eta_{\mathrm{i}}^{Y}}^{2} \;+\; \sigma_{Y_J}^{2}$$

The off-diagonal covariances are directly identified from observed data. For any triplet (j, j', j'') of distinct indicators:

$$\frac{\mathrm{Cov}\left(Y_{\mathrm{ij}},\; Y_{\mathrm{ij}'}\right)}{\mathrm{Cov}\left(Y_{\mathrm{ij}'},\; Y_{\mathrm{ij}''}\right)} \;=\; \frac{\lambda_{\mathrm{j}}^{Y}}{\lambda_{\mathrm{j}''}^{Y}}$$

which identifies the ratios of all loadings. Combined with the scale constraint M3 ($\sigma_{\eta^Y}^{2}$=1), all loadings are identified. The error variances then follow from:

$$\sigma_{Y_J}^{2} \;=\; \mathrm{Var}\left(Y_{\mathrm{ij}}\right) \;-\; \left(\lambda_{\mathrm{j}}^{Y}\right)^{2}$$

Uniqueness under M4 is guaranteed by the classical identification theorem for single-factor models. Under M2 (Gaussian factor model), the joint distribution is multivariate normal with all parameters identified. By the sufficiency of the covariance structure for Gaussian factor modes, the marginal distribution $P(\eta_{\mathrm{i}}^{Y}) \;= N(0,1)$ is identified. The conditional distribution of the latent outcome given its indicators is:

$$P(\eta_{\mathrm{i}}^{Y} \mid \mathbf{y}_{\mathrm{i}}) \;=\; \mathrm{N}\big(\mu_{\mathrm{post}}(\mathbf{y}_{\mathrm{i}}), \sigma_{\mathrm{post}}^{2}\big)$$

where the posterior mean is:

$$\mu_{\mathrm{post}}(\boldsymbol{y}_{\mathrm{i}}) \;=\; (\boldsymbol{\Lambda}^{\mathrm{Y}\prime}\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}^{\mathrm{Y}} \;+\; \mathbf{I})^{-1}\boldsymbol{\Lambda}^{\mathrm{Y}\prime}\boldsymbol{\Psi}^{-1}(\boldsymbol{y}_{\mathrm{i}} \;-\; \boldsymbol{\alpha}^{\mathrm{Y}})$$

and the posterior variance is:

$$\sigma_{\mathrm{post}}^{2} \;=\; (1 \;+\; \boldsymbol{\Lambda}^{\mathrm{Y}\prime}\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}^{\mathrm{Y}})^{-1}$$

Both are identified because all constituent measurement parameters are now identified. As a result, $P(\eta_{\mathrm{i}}^{Y}|\mathbf{y}_{i})$ is identified from observed data. Further, Bartlett scores

$$\widehat{\eta_{1}^{\mathrm{Y,B}}} \;=\; (\Lambda^{\mathrm{Y}\prime}\Psi^{-1}\Lambda^{\mathrm{Y}})^{-1}\Lambda^{\mathrm{Y}\prime}\Psi^{-1}(y_{\mathrm{i}} \;-\; \alpha^{\mathrm{Y}})$$

satisfy conditional unbiasedness:

$$\mathrm{E}\left[\widehat{\eta_{1}^{\mathrm{Y,B}}} \mid \eta_{\mathrm{i}}^{Y}\right] \;=\; \eta_{\mathrm{i}}^{Y}$$

Under M5, this extends to:

$$\mathrm{E}\left[\widehat{\eta_{1}^{\mathrm{Y,B}}} \mid\; \eta_{\mathrm{i}}^{Y}, T_{\mathrm{i}}, \boldsymbol{X}_{\mathrm{i}}\right] \;=\; \eta_{\mathrm{i}}^{Y}$$

As a result, Bartlett scores provide an unbiased proxy for the latent outcome in the structural model.

Stage II: Identification of the ATE

Under M1–M6 and C1–C3, $E[\eta_{\mathrm{i}}^{Y}(t)]$ is identified from the observed data distribution for $t \in \{0, 1\}$.

Step 1—Law of iterated expectations through the latent covariate distribution:

$$\mathrm{E}[\eta_{\mathrm{i}}^{Y}(t)] = \mathrm{E}_{\boldsymbol{X}}\left[\mathrm{E}[\eta_{\mathrm{i}}^{Y}(\mathrm{t}) \mid \boldsymbol{X}_{\mathrm{i}}]\right]$$

The outer expectation is over the covariates. Because $P(\eta_{\mathrm{i}}^{X})$ is identified from Stage I and $P(W_{i})$ is directly observed, the outer expectation is identified.

Step 2—With treatment ignorability:

$$\mathrm{E}[\eta_{\mathrm{i}}^{Y}(t) \mid \boldsymbol{X}_{\mathrm{i}}] \;=\; \mathrm{E}[\eta_{\mathrm{i}}^{Y}(\mathrm{t}) \mid\; T_{\mathrm{i}} = t, \boldsymbol{X}_{\mathrm{i}}]$$

Step 3—Under SUTVA, for individuals with $T_{i} = t$, $\eta_{\mathrm{i}}^{Y}(\mathrm{t}) = \eta_{\mathrm{i}}^{Y}$:

$$\mathrm{E}[\eta_{\mathrm{i}}^{Y}(t) \mid T_{\mathrm{i}} = \mathrm{t}, \boldsymbol{X}_{\mathrm{i}}] \;=\; \mathrm{E}[\eta_{\mathrm{i}}^{Y} \mid T_{\mathrm{i}} = \mathrm{t}, \boldsymbol{X}_{\mathrm{i}}]$$

Step 4—the conditional mean $\mathrm{E}[\eta_{\mathrm{i}}^{Y} \mid T_{\mathrm{i}} = \mathrm{t}, \boldsymbol{X}_{\mathrm{i}}]$ is identified from the observed data. Because Bartlett factor scores are a deterministic function of the indicators such that $\mathrm{E}\big[\hat{\eta}_{\mathrm{i}}^{Y_B} \mid T_{\mathrm{i}} = \mathrm{t}, \boldsymbol{X}_{\mathrm{i}}\big]$ is identified from the data. By unbiasedness $\mathrm{E}\big[\hat{\eta}_{\mathrm{i}}^{Y_B} \mid T_{\mathrm{i}} = \mathrm{t}, \boldsymbol{X}_{\mathrm{i}}\big] = \eta^{Y}$ and iterated expectation gives $\mathrm{E}\big[\hat{\eta}_{\mathrm{i}}^{Y_B} \mid T_{\mathrm{i}} = \mathrm{t}, \boldsymbol{X}_{\mathrm{i}}\big] = \mathrm{E}[\eta^{Y} \mid T_{\mathrm{i}} = \mathrm{t}, \boldsymbol{X}_{\mathrm{i}}]$. As a result, $\mathrm{E}[\eta^{Y} \mid T_{\mathrm{i}} = \mathrm{t}, \boldsymbol{X}_{\mathrm{i}}]$ is identified from the data as $\mathrm{E}\big[\hat{\eta}_{\mathrm{i}}^{Y_B} \mid T_{\mathrm{i}} = \mathrm{t}, \boldsymbol{X}_{\mathrm{i}}\big]$.

Step 5—When positivity hold the conditional distribution $P(\eta_{i}^{Y} \mid T_{\mathrm{i}} = t, \boldsymbol{X}_{\mathrm{i}})$ is well-defined at all covariate values in the support.

Combining Steps 1–5:

$$E[\eta_{\mathrm{i}}^{Y}(\mathrm{t})] \;=\; E_{W,\eta^{\mathrm{x}}}[Q(\mathrm{t}, W_{\mathrm{i}}, \eta_{\mathrm{i}}^{X})]$$

The ATE is identified as

$$\delta^{\eta^{Y}} \;=\; \mathrm{E}[\eta_{\mathrm{i}}^{Y}(1)] - \mathrm{E}[\eta_{\mathrm{i}}^{Y}(0)] = \mathrm{E}[Q(1, W_{\mathrm{i}}, \eta_{\mathrm{i}}^{X}) - \mathrm{Q}(0, W_{\mathrm{i}}, \eta_{\mathrm{i}}^{X})]$$

Stage III: Functional Identification Required for TMLE

For TMLE, identification in expectation is insufficient; the conditional mean function $Q(t, w, \eta^x)$ and propensity score $g(w, \eta^x)$ must be identified as functions of their arguments. The TMLE plug-in estimator takes the form:

$$\widehat{\delta^{\eta^Y}} = \frac{1}{n}\sum_{i=1}^{n} Q^*\left(1, W_\mathrm{i}, \widehat{\eta_i^X}\right) - Q^*\left(0, W_\mathrm{i}, \widehat{\eta_i^X}\right)$$

where Q* is the targeted update of the initial outcome regression and the clever covariate involves evaluating g at individual-level covariate values. The plug-in structure requires Q and g to be identified as functions—for each covariate value (t, w, η^x) in the support, the corresponding conditional mean and probability must be identified. A weaker identification in expectation (i.e., identification only of $E[Q(t, W, \eta^x)]$ as a scalar quantity) would not suffice because TMLE constructs a fluctuation of Q using the clever covariate at each individual's covariate value and the targeting step solves:

$$\frac{1}{n}\sum_{i} H\left(T_\mathrm{i}, W_\mathrm{i}, \widehat{\eta_\mathrm{i}^X}\right)\left[\widehat{\eta_\mathrm{i}^Y} - Q^*(\cdot)\right] = 0$$

which depends on evaluating $H(\cdot)$ and $Q^*(\cdot)$ at each individual's specific covariate vector. If Q were identified only in aggregate, the individual targeting step would be ill-defined. Functional identification is provided by Stage II: the conditional mean $E[\eta^Y \mid T_\mathrm{i} = t, \boldsymbol{X}_\mathrm{i} = \boldsymbol{x}]$ is identified for every value x in the support (not merely in expectation over x) because the Stage II argument holds for each fixed x. Positivity (C3) ensures this holds everywhere in the support—which permits the efficient influence function to be solved across the entire sample.

The key additional requirement with latent variables is that identification of the ATE requires a measurement identification argument (Stage I). That is, both measurement and causal identification are necessary. For targeted learning, individual-level functional identification is required rather than identification in expectation alone. The clever covariate construction and the targeting step both depend on evaluating Q and g at each individual's specific covariate vector. Positivity plays the crucial role of ensuring this holds everywhere in the covariate support—not merely on average—which is exactly what permits the efficient influence function to be solved across the entire sample.

Appendix B

In this appendix, we outline four related topics: (I) a congeniality framework specifying what the SEM-VAE must provide for downstream targeted learning to be valid, and what neural networks can and cannot provide; (II) the product-of-rates requirement and the degree to which it can be supported in the latent variable context; (III) the double robustness and semiparametric efficiency of the TAG estimator; and (IV) SEM-VAE stability. We note that although we outline conditions, a complete proof is beyond the scope of this paper.

.

I. Congeniality in TAG

Let P_o denote the true data generating distribution. Let $\boldsymbol{\eta}_i = (\eta_i^X, \eta_i^Z, \eta_i^Y)$ be the vector of latent variables. Define the true nuisance functions as:

$$Q_0(t, \eta^X, \eta^Z, W) = \mathrm{E}_{P_0}[\eta^Y \mid T = t, \eta^X, \eta^Z, W]$$
$$g_0(\eta^X, \eta^Z, \mathrm{W}) = P_0(T = 1 \mid \eta^X, \eta^Z, W)$$

The SEM-VAE defines a variational posterior family $\{q_\varphi(\boldsymbol{\eta}|\boldsymbol{o})\}$ and structural networks that induce estimated nuisance functions $\hat{Q}$ and $\hat{g}$ via plug-in latent posteriors. Let $\widehat{\boldsymbol{\eta}}_i$ denote the posterior summary (e.g., mean, Bartlett score) from the SEM-VAE. We frame congeniality in TAG using three conditions.

Congeniality Condition 1—Measurement Congeniality

The variational posterior reproduces the true posterior mean at a rate sufficient for the outcome proxy to be locally unbiased

$$\mathrm{E}_{q_\varphi}[\eta^Y \mid \mathbf{y}_\mathrm{i}] = \mathrm{E}_{P_0}[\eta^Y \mid \mathbf{y}_\mathrm{i}] + o_p\left(n^{-1/2}\right)$$

For the latent covariates the variational posterior satisfies is consistent for the true conditional distribution for an appropriate divergence ℓ (e.g., KL):

$$\int \ell\left(q_\varphi(\eta^X \mid \mathbf{x_i}, C_\mathrm{i}), p_0(\eta^X \mid \boldsymbol{o}_\mathrm{i})\right) \mathrm{d}\eta^X = o_p(1)$$

and the posterior summaries for latent covariates satisfy the rate condition for the product of rates

$$\left\|\widehat{\eta^X} - \eta^X\right\|_{L_2} = o_p\left(n^{-1/4}\right)$$

Congeniality Condition 2—Structural Congeniality

The plug-in nuisance estimates using SEM-VAE posterior summaries converge at rates sufficient for the product-of-rates condition:

$$\left\|\hat{Q}\left(t, \widehat{\eta^X}, \widehat{\eta^Z}, W\right) - Q_0(t, \eta^X, \eta^Z, W)\right\|_{L_2(P_0)} = o_p\left(n^{-1/4}\right)$$
$$\left\|\hat{g}\left(\widehat{\eta^X}, \widehat{\eta^Z}, W\right) - g_0(\eta^X, \eta^Z, W)\right\|_{L_2(P_0)} = o_p\left(n^{-1/4}\right)$$

This is the condition that the SEM-VAE's posterior summaries, when plugged into TMLE's nuisance models, produce estimates that converge at rates sufficient for the product-of-rates condition.

Congeniality Condition 3—Functional Congeniality (Targeting Compatibility)

The SEM-VAE posterior over η^Y produces a proxy $\widehat{\eta_i^Y}$ such that:

$$\mathrm{E}\left[\widehat{\eta_i^Y} - Q_0(T_\mathrm{i}, \eta_i^X, \eta_i^Z, W_\mathrm{i}) \mid T_\mathrm{i}, \eta^X, \eta^Z, W_\mathrm{i}\right] = 0$$

Applying measurement ignorability to the outcome proxy ensures that when TMLE solves the targeting equation, the outcome proxy can be substituted for η^Y in the EIF without introducing systematic bias. Bartlett scores satisfy this condition for linear measurement models (see expressions in the paper).

Incongeniality Bias

When the SEM-VAE is incongenial, the bias of the TAG estimator arises as:

$$\widehat{\delta^{\eta^Y}} - \delta^{\eta^Y} = \text{Bias}_{\text{TMLE}}(\hat{Q}, \hat{g}) + \text{Bias}_{\text{LV}}(\hat{\boldsymbol{\eta}}) + \text{Bias}_{\text{proxy}}(\widehat{\eta^Y})$$

The first term is the standard TMLE residual bias. The second term arises from misspecification of latent covariate posteriors and enters through the nuisance functions. The third term arises from an outcome proxy that fails functional congeniality (Condition 3).

Neural Networks

The universal approximation property of neural networks suggests that sufficiently wide and deep feedforward networks can approximate any continuous function on a compact domain to arbitrary precision. The result suggests that the structural networks μ_φ and g_φ in the SEM-VAE can, in principle, approximate Q_0 and g_0 to arbitrary precision. Similarly, the encoder networks can approximate any posterior mean function. In our SEM-VAE, the context encoder $\boldsymbol{C}_i = f_h(\mathbf{x}_i, \mathbf{z}_i, \mathbf{y}_i, W_i, t_i)$ additionally incorporates structural information from all observed variables. When sufficiently expressive, it can encode the sufficient statistics needed for posterior inference over all latent variables simultaneously. That is, for example, $q_\varphi(\eta^x \mid \mathbf{x}_i, \boldsymbol{C}_i)$ can approximate the full conditional posterior $p_0(\eta^x \mid \boldsymbol{o_i})$—a richer target than the block-only posterior $p_0(\eta^x \mid \mathbf{x}_i)$. Deeper/wider networks produce richer function classes, so congeniality can in principle be improved by increasing network capacity.

However, the SEM-VAE minimizes the ELBO, not the true posterior log-likelihood. The variational gap $\text{KL}(q_\varphi \,\|\, p_0)$ need not be zero even with infinite network capacity if the variational family is restricted (e.g., mean-field Gaussian). When the true posterior is non-Gaussian or exhibits complex structure, the Gaussian variational family imposes a structural misspecification that no amount of network width can eliminate. This represents a fundamental limit on measurement congeniality (condition 1 above) under standard VAE training.

Similarly, there are no finite-sample rate guarantees from architecture alone: universal approximation is an asymptotic argument. The convergence rate of $\hat{Q}$ and $\hat{g}$ depends on optimization, the effective complexity of the network, the smoothness of the target functions, and the dimension of the input. For example, for d-dimensional inputs and β-smooth target functions, neural network estimators achieve the minimax rate $n^{-\frac{\beta}{2\beta+d}}$. This rate meets the $n^{-1/4}$ threshold when $\beta \geq d/2$—a non-trivial smoothness condition for each of the networks (structural, encoder, decoder). For example, with the structural network for the outcome, the input dimension of the base condition (e.g., Table 1) is 4 (1 treatment, 2 latent covariates, and an observed covariate). For each latent variable, the input dimension for the encoder is $J_k + d_C$ with J_k as the number of indicators for that latent variable and d_C as the context vector dimension (e.g., 4 in the base simulation condition). For example, in the base simulation condition (e.g., Table 1) the dimensionality of d is $J_k + d_C = 3 + 4 = 7$. Further, even with a universally approximating architecture, the optimized SEM-VAE may not satisfy structural congeniality because the training objective is not aligned with the downstream causal estimand.

Congeniality Assumption

As a result, our approach uses a congeniality assumption—there exist sequences of SEM-VAE architectures indexed by n such that:

(i) the variational posterior satisfies $\text{KL}(q_\varphi(\boldsymbol{\eta} \mid \boldsymbol{o}) \,\|\, p_0(\boldsymbol{\eta} \mid \boldsymbol{o})) = o_p(1)$ uniformly over the support of P_0;

(ii) The posterior summary operator $\boldsymbol{o} \rightarrow \widehat{\boldsymbol{\eta}}(\boldsymbol{o})$ satisfies e.g.,

$$\left\|\widehat{\eta^X} - \eta^X\right\|_{L_2} = o_p\left(n^{-1/4}\right), \text{and analogously for } \widehat{\eta^Z}$$

(iii) The outcome proxy satisfies functional congeniality (Condition 3 above); and
(iv) The structural networks produce nuisance estimates satisfying the product-of-rates condition (see below).

II. Product-of-Rates (POR) in the TAG Context

In the classical TMLE setting with fully observed variables, the key condition for asymptotic linearity is that the nuisance function errors satisfy:

$$\left\|\hat{Q} - Q_0\right\|_{L_2(P_0)} \cdot \|\hat{g} - g_0\|_{L_2(P_0)} = o_p\left(n^{-1/2}\right) \qquad \text{(POR)}$$

A sufficient condition is that both converge at rate $n^{-1/4}$. This condition, combined with sample splitting (cross-fitting), ensures that the second-order remainder in the von Mises expansion of the estimator is asymptotically negligible. The asymptotic linearity of the TMLE relies on the Donsker class condition or cross-fitting to handle the empirical process terms, and on (POR) to control the remainder.

In TAG learning, nuisance functions are evaluated at plug-in latent variable summaries $(e.g., \widehat{\eta^X}, \widehat{\eta^Z})$ rather than true latent variables and that introduces additional terms in the product-of rates- analysis. The estimation error for Q decomposes as:

$$\begin{aligned}&\hat{Q}\left(t, \widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right) - Q_0(t, \eta^X, \eta^Z, \mathrm{W}) \\ &= \hat{Q}\left(t, \widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right) - Q_0\left(t, \widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right) + \\ &\qquad Q_0\left(t, \widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right) - Q_0(t, \eta^X, \eta^Z, \mathrm{W})\end{aligned}$$

The first two terms are the standard nuisance estimation error in Q given plug-in latent variables. With flexible ML and appropriation conditions, cross-fitting addresses the empirical process issue and satisfies the $o_p\left(n^{-1/4}\right)$ requirement. The last two terms are the latent variable substitution error. If we use a first-order Taylor expansion of Q_0 around the true latent values, we have:

$$Q_0\left(t, \widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right) - Q_0(t, \eta^X, \eta^Z, \mathrm{W}) \approx \nabla_{\eta^X} Q_0 \cdot \left(\widehat{\eta^X} - \eta^X\right) + \nabla_{\eta^Z} Q_0 \cdot \left(\widehat{\eta^Z} - \eta^Z\right)$$

The latent variable substitution error is bounded in L_2 when:

$$\|\text{latent variable substitution error}\|_{L_2} \leq \left\|\nabla_\eta Q_0\right\|_\infty \cdot \left\|\widehat{\eta^X} - \eta^X\right\|_{L_2}$$

That is, it is controlled if its steepness is bounded such that Q_0 is Lipschitz in its latent covariate arguments (i.e., $\left\|\nabla_\eta Q_0\right\|_\infty < \infty$) and if the latent variable summary substitutions converge at a rate of $\|\widehat{\boldsymbol{\eta}} - \boldsymbol{\eta}\|_{L_2} = o_p\left(n^{-1/4}\right)$. Similar requirements apply to the propensity score.

Extended Product-of-Rates Conditions (EPOR)

The three EPOR conditions jointly imply the total product-of-rates condition (POR):

$$\left\|\hat{Q}\left(t, \widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right) - Q_0\left(t, \widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right)\right\|_{L_2} \cdot \left\|\hat{g}\left(\widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right) - g_0\left(\widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right)\right\|_{L_2} = o_p\left(n^{-1/2}\right) \text{ (EPOR} - 1)$$

$$\left\|Q_0\left(t, \widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right) - Q_0(t, \eta^X, \eta^Z, \mathrm{W})\right\|_{L_2} \cdot \left\|\hat{g}\left(\widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right) - g_0\left(\widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right)\right\|_{L_2} = o_p\left(n^{-\frac{1}{2}}\right) \text{ (EPOR} - 2)$$

$$\left\|\hat{Q}\left(t, \widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right) - Q_0\left(t, \widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right)\right\|_{L_2} \cdot \left\|g_0\left(\widehat{\eta^X}, \widehat{\eta^Z}, \mathrm{W}\right) - g_0(\eta^X, \eta^Z, \mathrm{W})\right\|_{L_2} = o_p\left(n^{-1/2}\right) \text{ (EPOR} - 3)$$

When the collective EPOR conditions are met they imply $\left\|\hat{Q} - Q_0\right\|_{L_2} \cdot \|\hat{g} - g_0\|_{L_2} = o_p\left(n^{-1/2}\right)$. As a result, there are three primary conditions required. First, the nuisance estimation rates for the outcome and propensity core must be $o_p\left(n^{-1/4}\right)$. Second, the latent

variable consistency rates must be $o_p(r_n(n))$ for some rate $r_n(n) \to 0$. Third, the functions Q_0 and g_0 are Lipschitz in their latent covariate arguments. Under these conditions, the POR condition is satisfied when $r_n(n) = o_p\left(n^{-1/4}\right)$ (the latent variable error is asymptotically negligible relative to the nuisance estimator error). For example, with a linear measurement model for the outcome, Bartlett scores achieve a rate well within this requirement ($\left\|\widehat{\eta^{Y_B}} - \eta^Y\right\|_{L_2} = O_p\left(n^{-1/2}\right)$). For the latent covariates, the posteriors are more complex because they depend on the context vector. The rate with which the approximation converges depends on the smoothness of the network, the approximation error, and estimation error. For instance, for smooth (β-Hölder) nonlinear measurement functions in d-dimensional latent space, SEM-VAE posterior summaries achieve:

$$\left\|\widehat{\eta^X} - \eta^X\right\|_{L_2} = O_p\left(n^{-\frac{\beta}{2\beta+\mathrm{d}}}\right)$$

As noted previously, the $n^{-1/4}$ condition requires $\beta \geq d/2$. In the TAG setting when $d = 1$ (each latent variable is scalar), the condition is $\beta \geq 1/2$—that is, the encoder needs to approximate at least a ½ -Hölder smooth posterior mean function. Given that neural networks with smooth activation functions, this is reasonable for standard continuous latent variable models. In high-dimensional covariate settings (e.g., Table 7), the rate may fall short of $n^{-1/4}$. For example, the high dimensional simulation setting (e.g., Table 7) suggests the larger samples are required to achieve near-zero bias.

These rate conditions should be interpreted as sufficient regularity conditions rather than guaranteed properties of the SEM-VAE. Whether they hold depends on measurement quality, architectural constraints, smoothness of the measurement/structural functions, dimensionality, and optimization stability.

III. Double Robustness and Semiparametric Efficiency of the TAG Estimator

Let $\boldsymbol{o} = (\mathbf{y}_i, \mathbf{x}_i, \mathbf{z}_i, W_i, T_i)$ be the observed data. The target parameter is:

$$\delta^{\eta^Y} = \mathrm{E}_{P_0}[Q_0(1, \eta^X, \eta^Z, W) - Q_0(0, \eta^X, \eta^Z, W)]$$

where $Q_0(t, \eta^X, \eta^Z, W)$ is condition mean of η^Y given $T = t$ and covariates. TAG uses predicted values for the latent variables observed-data proxies $\widehat{\eta^X}$, $\widehat{\eta^Z}$ and $\widehat{\eta^Y}$. Define the working data as $\tilde{\boldsymbol{O}} = (\widehat{\eta^Y}, \widehat{\eta^X}, \widehat{\eta^Z}, W, T)$.

In the fully observed setting, the EIF for the ATE functional at (Q_0, g_0, P_0) is:

$$D_0(O_i) = H_0\left(T_i, \eta_i^X, \eta_i^Z, W_i\right)\left[\eta_i^Y - Q_0\left(T_i, \eta_i^X, \eta_i^Z, W_i\right)\right] + \left[Q_0\left(1, \eta_i^X, \eta_i^Z, W_i\right) - Q_0\left(0, \eta_i^X, \eta_i^Z, W_i\right) - \delta^{\eta^Y}\right]$$

where the clever covariate is:

$$H_0(T, \eta^X, \eta^Z, \mathrm{W}) = \frac{\mathrm{I}(\mathrm{T} = 1)}{g_0(1 \mid \eta^X, \eta^Z, \mathrm{W})} - \frac{\mathrm{I}(\mathrm{T} = 0)}{g_0(0, \eta^X, \eta^Z, \mathrm{W})}$$

Substituting the Bartlett proxy $\hat{\eta}_i^{Y_B} = \eta_i^Y + \varepsilon_i^Y$ (where $\mathrm{E}[\varepsilon_i^Y | \eta_i^X, \eta_i^Z, W_i] = 0$), preserves the conditional mean structure.

Double Robustness of the TAG Estimator

Under assumptions M1–M6, C1–C3, and:

(DR1) Cross-fitting is used with K folds;

(DR2) TAG Congeniality holds;

(DR3) The conditional expectation of the outcome proxy is unbiased;

(DR4) Conditions EPOR-1 through EPOR-4 are satisfied;
the TAG estimator satisfies:

$$\widehat{\delta^{\eta^Y}} - \delta^{\eta^Y} = \frac{1}{n}\sum_{i=1}^{n} \widetilde{D}_0(\tilde{O}_i) + o_p\left(n^{-1/2}\right)$$

and is consistent for δ^{η^Y} if either:
(a) $\hat{Q}(t, \widehat{\eta^X}, \widehat{\eta^Z}, W) \to_p \tilde{Q}_0(t, \widehat{\eta^X}, \widehat{\eta^Z}, W)$ (outcome model consistently specified given latent variable values), or
(b) $\hat{g}(t, \widehat{\eta^X}, \widehat{\eta^Z}, W) \to_p g_0(\widehat{\eta^X}, \widehat{\eta^Z}, W)$ (propensity model consistently specified given latent variable values).
The TAG estimator is

$$\delta^{\eta^Y} = \frac{1}{n}\sum_i [\widehat{Q^*}(1, \widehat{\eta^X}, \widehat{\eta^Z}, W) - \widehat{Q^*}(0, \widehat{\eta^X}, \widehat{\eta^Z}, W)]$$

where $\widehat{Q^*}$ is the targeted fluctuation. In TMLE the targeting step ensures

$$\frac{1}{n}\sum_i \widehat{H_i}[\widehat{\eta^{Y_B}} - \widehat{Q^*}(T_i, \widehat{\eta_i^X}, \widehat{\eta_i^Z}, W_i)] = 0$$

The difference between the estimator and true parameter is a sum of the empirical EIF ($P_n[\widetilde{D}_0]$), the empirical process term ($(P_n - P_0)[\widehat{D} - \widetilde{D}_0]$), and the bias term ($P_0[\widehat{D}]$)

$$\hat{\delta}^{\eta^Y} - \delta^{\eta^Y} = P_n[\widetilde{D}_0] + (P_n - P_0)[\widehat{D} - \widetilde{D}_0] + P_0[\widehat{D}] \qquad (4)$$

The empirical EIF is the leading term and drives asymptotic normality. Under cross-fitting (DR1), the nuisance functions in fold k are trained on data O^{-k} and evaluated on O^k, making them independent of the test fold. By the Cauchy-Schwarz inequality, the empirical process term satisfies:

$$(P_n - P_0)[\widehat{D} - \widetilde{D}_0] = O_p\left(\left\|\hat{Q} - \tilde{Q}_0\right\|_{L_2} + \|\hat{g} - g_0\|_{L_2}\right) \cdot n^{-1/2}$$

Under the product-of-rates conditions this is $o_p(n^{-1/2})$. The bias term ($P_0[\widehat{D}]$) is the expectation of the estimated EIF:

$$P_0[\widehat{D}] = P_0[\widehat{H_i}(\widehat{\eta^{Y_B}} - \hat{Q}\left(T_i, \widehat{\eta_i^X}, \widehat{\eta_i^Z}, W_i\right))] + P_0[\hat{Q}(1, \widehat{\eta^X}, \widehat{\eta^Z}, W) - \hat{Q}(0, \widehat{\eta^X}, \widehat{\eta^Z}, W) - \delta^{\eta^Y}]$$

The first term is zero by construction whereas the second term requires $\hat{Q}$ to be consistent for the true Q. The bias can then be written as

$$P_0[\widehat{D}] = -P_0[(\hat{Q} - \tilde{Q}_0)(\hat{g}^{-1} - g_0^{-1})] + \text{second-order terms}$$

The bias is zero if either $\hat{Q} = \tilde{Q}_0$ or $\hat{g} = g_0$, and is $O_p(r_Q \cdot r_G) = o_p(n^{-1/2})$ under the product-of-rates condition. The latent variable substitution errors enter through an additional remainder of order $O_p(r_\eta^2)$, which is $o_p(n^{-1/2})$ when $r_\eta = o(n^{-1/4})$. Note that double robustness presupposes congeniality—the double robustness is with respect to the nuisance functions $\hat{Q}$ and $\hat{g}$ evaluated at the plug-in latent summaries. It does not provide robustness against misspecification of the measurement model. If e.g. congeniality fails, the EIF targeting equation no longer centers at zero and the estimator is no longer doubly robust.

Semi-parametric Efficiency of the TAG Estimator

Under the conditions of Double Robustness and assuming:
(SE1) Both nuisance models are consistently specified;
(SE2) Conditions EPOR-1 through EPOR-4 hold;
(SE3) $r_\eta(n) = o(n^{-1/4})$;
the TAG estimator satisfies:

$$\sqrt{n}\left(\hat{\delta}^{\eta^Y} - \delta^{\eta^Y}\right) \to^d N\left(0, \sigma^2_{\hat{\delta}^{\eta^Y}}\right)$$

where the asymptotic variance decomposes as:

$$\sigma^2_{\widehat{\delta}^{\eta^Y}} = \mathrm{Var}_{P_0}\left[\widetilde{D}_0(\tilde{O}_i)\right] + \kappa_{\mathrm{meas}}$$

The first term is the semiparametric efficiency bound under the proxy data distribution and the second term is the variance inflation from outcome measurement error:

$$\kappa_{\mathrm{meas}} = \mathrm{E}\left[H_0(T_{\mathrm{i}}, \eta^X, \eta^Z, W_{\mathrm{i}})^2 \cdot \mathrm{Var}\left[\varepsilon^Y_{\mathrm{i}} \mid T_{\mathrm{i}}, \eta^X, \eta^Z, W_{\mathrm{i}}\right]\right]$$

The variance inflation is strictly positive whenever the outcome is measured with error. As a result, the efficiency bounds is not attainable without perfect measurement.

Latent covariate error additionally inflates variance through the outcome and propensity score models:

$$\kappa_{\mathrm{cov}} = \mathrm{E}\left[\left(\boldsymbol{\nabla}_{\boldsymbol{\eta}^X} Q_0(1) - \boldsymbol{\nabla}_{\boldsymbol{\eta}^X} Q_0(0)\right)^2 \cdot \mathrm{Var}\left[\widehat{\boldsymbol{\eta}^X}_{\mathrm{i}} - \boldsymbol{\eta}^X_{\mathrm{i}}\right]\right]$$

When $r_\eta(n) = o(n^{-1/4})$ (Condition SE3), $\kappa_{cov} = o(1)$ is asymptotically negligible, and latent covariate estimation does not contribute to the leading asymptotic variance. In finite samples, however, κ_{cov} is the dominant source of the variance inflation.

IV. SEM-VAE Stability and the Full-Data Implementation

The double robustness outlined in Section III conditions on cross-fitting (DR1): nuisance functions in fold k are estimated on data $\boldsymbol{o}^{(-k)}$ and evaluated on $\boldsymbol{o}^{(k)}$, making them independent of the test fold. In the TAG implementation described in the illustrative example, both the SEM-VAE and the TMLE nuisance functions $\hat{Q}$ and $\hat{g}$ were estimated with K-fold cross-fitting and satisfy DR1. In the TAG implementation described in the simulation section, the TMLE nuisance functions $\hat{Q}$ and $\hat{g}$ are estimated with K-fold cross-fitting and satisfy DR1. However, in our simulations the latent variable summaries entering those nuisance functions were generated by a single SEM-VAE trained on the full sample for computational ease. The full-data SEM-VAE therefore does not necessarily satisfy the independence condition in DR1 with respect to the latent summaries. As a result, the simulations (but not the general use of TAG) test an additional condition.

SEM-VAE Stability Assumption

Let $\hat{\theta}^{(full)}$ denote the parameters of the SEM-VAE trained on the full sample, and let $\hat{\theta}^{(-k)}$ denote the parameters of a SEM-VAE trained on all folds except fold k. The SEM-VAE satisfies stability if, for each fold k and each observation $i \in$ fold k, the resulting latent variable summaries are asymptotically equivalent:

$$\left\|\hat{\eta}^{\mathrm{X(full)}}_{\mathrm{i}} - \hat{\eta}^{\mathrm{X(-k)}}_{\mathrm{i}}\right\|_{L_2} = o_p\left(n^{-1/4}\right)$$

$$\left\|\hat{\eta}^{\mathrm{Z(full)}}_{\mathrm{i}} - \hat{\eta}^{\mathrm{Z(-k)}}_{\mathrm{i}}\right\|_{L_2} = o_p\left(n^{-1/4}\right)$$

and the outcome proxy satisfies:

$$\left|\hat{\eta}^{\mathrm{Y(full)}}_{\mathrm{i}} - \hat{\eta}^{\mathrm{Y(-k)}}_{\mathrm{i}}\right| = o_p\left(n^{-1/2}\right)$$

uniformly over $i \in$ fold k for $k = 1, \ldots, K$. Conceptually the assumption requires that the latent summaries produced by the full-data SEM-VAE and those produced by a leave-fold-out SEM-VAE are asymptotically indistinguishable at the rates required by the product-of-rates conditions. When the assumption holds approximately, replacing $\hat{\eta}^{\mathrm{X(-k)}}_{\mathrm{i}}$ with $\hat{\eta}^{\mathrm{X(full)}}_{\mathrm{i}}$ in the TMLE targeting equation introduces only an asymptotically negligible perturbation, and the double robustness and semiparametric efficiency results follow.

TAG Under Stability (without cross-validation on the SEM-VAE)

Under assumptions M1–M6, C1–C3, DR1–DR4 applied to the TMLE step, and stability assumption above, the double robust properties and semiparametric efficiency hold for the full-

data TAG implementation. The key term requiring modification is the empirical process term arising from the dependence of the full-data latent summaries on observation *i*. Under full-data SEM-VAE, the empirical process term contains an additional contribution from the dependence of $\hat{\eta}_{\mathrm{i}}^{\mathrm{X(full)}}$ on observation *i*

$$(P_n - P_0)\left[\widehat{D}^{\mathrm{full}} - \widehat{D}^{-\mathrm{k}}\right] \;\approx\; (P_n - P_0)\left[\nabla_\eta \widehat{D} \cdot \left(\widehat{\eta^{\mathrm{full}}} - \widehat{\eta^{-\mathrm{k}}}\right)\right]$$

Under stability assumption and the Lipschitz condition POR3, each summand is

$$o_p\left(n^{-1/4}\right)$$

so that the additional empirical process contribution is

$$o_p\left(n^{-1/2}\right)$$

and asymptotically negligible.

The stability assumption is not directly verifiable from the data, but is supported by several structural features of the SEM-VAE. First, amortized inference dilutes the influence of any single observation across mini-batches during training, reducing the sensitivity of $\hat{\theta}^{(full)}$ to the inclusion of fold *k* observations. Uniform stability results for stochastic gradient descent, the parameter-level sensitivity satisfies:

$$\left\|\hat{\theta}^{\mathrm{full}} - \hat{\theta}^{-\mathrm{k}}\right\|_{L_2} \;=\; O_p\left(\frac{T_{\mathrm{epochs}} \cdot L_{\mathrm{step}}}{n}\right)$$

where T_{epochs} is the number of training epochs and L_{step} is the per-step gradient Lipschitz constant. For fixed architecture and learning rate, this bound is $O_p(n^{-1})$ which is better than the $o_p\left(n^{-1/4}\right)$ required by the stability assumption. Second, for linear measurement models, the identification constraints imposed by the SEM-VAE architecture—fixed loading or variance constraints, indicator-block routing—ensure that the measurement parameters are identified by global second-order moments of the indicator distribution, making them particularly insensitive to the inclusion or exclusion of any single fold. Similarly, the outcome proxy under a linear measurement model is a linear function of the estimated loadings and error variances, whose stability is directly governed by the parameter-level stability bound above. For nonlinear measurement models this argument does not apply: identification is through higher-order moments and the full joint distribution, so parameter stability must be established through the stochastic gradient descent bound above, and the amplification of parameter perturbations into latent summary perturbations depends additionally on the Lipschitz properties of the nonlinear decoder f_θ.

Although the stability assumption cannot be tested directly without refitting the SEM-VAE on multiple data splits, comparisons between the TAG and NLRAG in the simulations provides an indirect empirical diagnostic. NLRAG uses the identical generative phase as TAG—the same full-data SEM-VAE and the same latent summaries—but replaces the targeted learning step with an oracle structural model. As previously noted, when TAG and NLRAG produce similar estimates, it indicates that the SEM-VAE is generating stable latent summaries that do not materially differ from what a cross-fit SEM-VAE would produce: instability in the latent summaries would manifest as a gap between TAG and NLRAG because NLRAG's oracle structural model is maximally sensitive to the quality of the latent inputs. Across the base and moderate-dimensional simulation conditions (e.g., Tables 1, 2, 4, 5), TAG tracked NLRAG closely at moderate-to-large sample sizes, providing some empirical support for the stability assumption in those settings. In high-dimensional settings (e.g., Tables 6–7), the TAG–NLRAG gap was more sensitive to sample size. The result is consistent with the theoretical prediction that the ratio of SEM-VAE parameter count to sample size increases with dimensionality, relaxing

the parameter-level stability bound above. In general, cross-fitting should be done for both the SEM-VAE and TMLE but future work might assess the stability of results by comparing full data and cross-fit SEM-VAEs in high-dimensional settings.

Appendix C

In this appendix we outline the ELBO training objective for the SEM-VAE. The parameters optimized are: ϕ (encoder networks), φ (structural networks for treatment and outcome), and θ (decoder/measurement networks). The training objective maximizes the sample-average ELBO:

$$\hat{\mathcal{L}}(\phi, \varphi, \theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\phi, \varphi, \theta; \boldsymbol{o}_{\mathrm{i}})$$

where the per-observation ELBO has four reconstruction terms and three KL regularization terms weighted by the hyperparameter β_{kl}.

Reconstruction terms:

$$\mathcal{L}(\phi, \varphi, \theta;\ \boldsymbol{o}_{\mathrm{i}}) = \sum_{j=1}^{J_X} \log p_\theta\left(X_{\mathrm{ij}} \mid \eta_{\mathrm{i}}^X\right) + \sum_{j=1}^{J_Z} \log p_\theta\left(Z_{\mathrm{ij}} \mid \eta_{\mathrm{i}}^Z\right) + \sum_{j=1}^{J_Y} \log p_\theta\left(Y_{\mathrm{ij}} \mid \eta_{\mathrm{i}}^Y\right) + \log p_\varphi(T_{\mathrm{i}} \mid \eta_{\mathrm{i}}^X, \eta_{\mathrm{i}}^Z, W_{\mathrm{i}})$$

KL regularization terms:

$$-\beta_{\mathrm{KL}}\left[KL\left(q_\varphi(\eta_{\mathrm{i}}^X \mid x_{\mathrm{i}}, C_{\mathrm{i}}) \parallel p(\eta_i^X)\right) + KL\left(q_\varphi(\eta_{\mathrm{i}}^Z \mid z_{\mathrm{i}}, C_{\mathrm{i}}) \parallel p(\eta_{\mathrm{i}}^Z)\right) + KL\left(q_\varphi(\eta_{\mathrm{i}}^Y \mid y_{\mathrm{i}}, C_{\mathrm{i}}) \parallel p_\varphi(\eta_{\mathrm{i}}^Y \mid \eta_{\mathrm{i}}^X, \eta_{\mathrm{i}}^Z, W_{\mathrm{i}}, T_{\mathrm{i}})\right)\right]$$

where $\beta_{kl} = 1$. The reconstruction terms are maximized when the model assigns high probability to the observed data; the KL terms prevent the variational posteriors from collapsing away from their respective priors. The expectations in the reconstruction terms are approximated by the reparameterization trick (e.g., $\eta_{\mathrm{i}}^X = \mu_\phi^X + \sigma_\phi^X \odot \varepsilon$, $\varepsilon \sim N(0, \mathrm{I})$).

Each indicator block uses a conditionally Gaussian measurement model. For the j-th indicator of block X:

$$\log p_\theta\left(X_{\mathrm{ij}} \mid \eta_{\mathrm{i}}^X\right) = -\frac{1}{2}\log(2\pi) - \log \sigma_{\mathrm{j}}^X - \frac{\left(X_{\mathrm{ij}} - \mu_{\mathrm{j}}^X(\eta_{\mathrm{i}}^X)\right)^2}{2\left(\sigma_{\mathrm{j}}^X\right)^2}$$

In the linear case, the conditional mean is

$$\mu_{\mathrm{j}}^X(\eta_{\mathrm{i}}^X) = \alpha_{\mathrm{j}}^X + \lambda_{\mathrm{j}}^X \eta_{\mathrm{i}}^X$$

In the nonlinear case, the conditional mean is

$$\mu_{\mathrm{j}}^X(\eta_{\mathrm{i}}^X) = \alpha_{\mathrm{j}}^X + f_{\theta,j}^X(\eta_{\mathrm{i}}^X)$$

with $f_{\theta,j}^X$ as a two-hidden-layer tanh network specific to indicator *j*. The $f_{\theta,j}^X$ share a common nonlinear feature embedding h_θ^X and differ only in their indicator-specific coefficient vector that combines the shared nonlinear features into a single nonlinear curve for indicator *j* (linear readout). The indicator intercepts α_{j}^X and indicator log-residual standard deviations $\log \sigma_{\mathrm{j}}^X$ are learned parameters. The shared feature embedding $h_\theta^X(\eta)$ plus the unit-variance scale anchor identifies η up to the sign because (i) the marginal of the indicator vector **y** given η is determined by the linear readout, the nonlinear feature embedding and the intercepts, (ii) the scale constraint anchors the latent metric, and (iii) with $J \geq 3$ indicators having linearly independent rows (a non-collinearity condition), η is identified up to a global monotone reparameterization, which is then pinned by the unit-variance prior. This is similar in spirit to identifiability results in nonlinear factor models (e.g., Khemakhem et al., 2020).

The full block likelihood in the X block is:

$$\sum_{j=1}^{J_X} \log p_\theta\left(X_{\mathrm{ij}} \mid \eta_{\mathrm{i}}^X\right) = -\frac{J_X}{2}\log(2\pi) - \sum_{j=1}^{J_X} \log \sigma_{\mathrm{j}}^X - \frac{1}{2}\sum_{j=1}^{J_X} \frac{\left(X_{\mathrm{ij}} - \mu_{\mathrm{j}}^X(\eta_{\mathrm{i}}^X)\right)^2}{\left(\sigma_{\mathrm{j}}^X\right)^2}$$

The same form applies to the Z block and the Y block.

The treatment is modelled as Bernoulli with log-odds produced by a three-hidden-layer MLP g_φ:

$$\log p_\varphi(T_{\mathrm{i}} \mid \eta_{\mathrm{i}}^X, \eta_{\mathrm{i}}^Z, W_{\mathrm{i}}) = T_{\mathrm{i}} \log \sigma\left(g_\varphi(\eta_{\mathrm{i}}^X, \eta_{\mathrm{i}}^Z, W_{\mathrm{i}})\right) + (1 - T_{\mathrm{i}}) \log\left(1 - \sigma\left(g_\varphi(\eta_{\mathrm{i}}^X, \eta_{\mathrm{i}}^Z, W_{\mathrm{i}})\right)\right)$$

where $\sigma(\cdot)$ is the logistic sigmoid.

The variational posteriors for the latent covariates are Gaussian, produced by the construct-specific encoders conditioned on local indicators and the context vector:

$$q_\phi(\eta_{\mathrm{i}}^X \mid \mathbf{x}_{\mathrm{i}}, C_{\mathrm{i}}) = N\left(\mu_\phi^X(\mathbf{x}_{\mathrm{i}}, C_{\mathrm{i}}), \left(\sigma_\phi^X(\mathbf{x}_{\mathrm{i}}, C_{\mathrm{i}})\right)^2\right)$$

with prior $p(\eta^x) = N(0, 1)$. The closed-form KL is:

$$KL\left(q_\phi(\eta_{\mathrm{i}}^X \mid \mathbf{x}_{\mathrm{i}}, C_{\mathrm{i}}) \parallel N(0,1)\right) = \frac{1}{2}\left[\left(\sigma_\phi^X\right)^2 + \left(\mu_\phi^X\right)^2 - 1 - \log\left(\sigma_\phi^X\right)^2\right]$$

The same expression applies to details of the Z encoder.

Rather than penalizing $q_\phi(\eta_{\mathrm{i}}^Y \mid \mathbf{y}_{\mathrm{i}}, C_{\mathrm{i}})$ toward a fixed $N(0,1)$ prior, it is penalized toward the structural conditional prior $p_\varphi(\eta_{\mathrm{i}}^Y \mid \eta_{\mathrm{i}}^X, \eta_{\mathrm{i}}^Z, W_{\mathrm{i}}, T_{\mathrm{i}})$, which is the output of the structural outcome network

$$p_\varphi(\eta_{\mathrm{i}}^Y \mid \eta_{\mathrm{i}}^X, \eta_{\mathrm{i}}^Z, W_{\mathrm{i}}, T_{\mathrm{i}}) = N\left(\mu_\varphi^X(\eta_{\mathrm{i}}^X, \eta_{\mathrm{i}}^Z, W_{\mathrm{i}}, T_{\mathrm{i}}), \left(\sigma_\varphi^X(\eta_{\mathrm{i}}^X, \eta_{\mathrm{i}}^Z, W_{\mathrm{i}}, T_{\mathrm{i}})\right)^2\right)$$

The closed-form KL between two Gaussians is

$$KL\left(q_\phi(\eta_{\mathrm{i}}^Y \mid \mathbf{y}_{\mathrm{i}}, C_{\mathrm{i}}) \parallel p_\varphi(\eta_{\mathrm{i}}^Y \mid \eta_{\mathrm{i}}^X, \eta_{\mathrm{i}}^Z, W_{\mathrm{i}}, T_{\mathrm{i}})\right)$$
$$= \frac{1}{2}\left[\log \frac{\left(\sigma_\varphi^Y\right)^2}{\left(\sigma_\phi^Y\right)^2} + \frac{\left(\sigma_\phi^Y\right)^2 + \left(\mu_\phi^Y - \mu_\varphi^Y\right)^2}{\left(\sigma_\varphi^Y\right)^2} - 1\right]$$

The KL is small when the encoder posterior and structural prior agree; it penalises the squared discrepancy relative to the structural prior variance, driving the encoder toward the conditional distribution implied by the structural outcome network. The key difference between the full SEM-VAE ELBO and a standard three-latent-variable VAE ELBO is component structural outcome model. A standard VAE would use:

$$KL\left(q_\varphi(\eta_{\mathrm{i}}^Y \mid \mathbf{y}_{\mathrm{i}}, C_{\mathrm{i}}) \parallel N(0,1)\right)$$

whereas the SEM-VAE uses:

$$KL\left(q_\varphi(\eta_{\mathrm{i}}^Y \mid \mathbf{y}_{\mathrm{i}}, C_{\mathrm{i}}) \parallel p_\varphi(\eta_{\mathrm{i}}^Y \mid \eta_{\mathrm{i}}^X, \eta_{\mathrm{i}}^Z, W_{\mathrm{i}}, T_{\mathrm{i}})\right)$$

This substitution embeds the structural outcome model directly into the training objective. Minimizing the KL forces the encoder posterior to match the structural conditional prior. As a result, the learned latent outcome distribution must simultaneously (a) reconstruct the **y** indicators through the decoder and (b) be consistent with the structural outcome regression. This coupling is what allows the SEM-VAE to jointly optimize measurement and structural components in a single training pass, analogous to full-information estimation in conventional SEM.

Appendix D

In this appendix we outline a heuristic approach for comparing linear and nonlinear measurement models in the TAG learning framework. The TAG learning framework supports both linear and nonlinear measurement models. In applied settings the functional form of the relationship between a latent variable and its indicators is generally unknown. We outline an approach for comparing linear and nonlinear measurement models using the evidence lower bound (ELBO) for CFA-VAEs, supplemented by classical confirmatory factor analysis (CFA) fit indices and residual nonlinearity tests. Because each latent variable has its own indicator block, the comparison is conducted one latent variable at a time to isolate measurement from structural models.

Step 1: Classical CFA Pre-Screen

For each latent variable's indicator block, fit a linear CFA. If the model fits reasonably well (e.g., CFI > 0.95 and RMSEA < 0.06), the linear measurement model is reasonable and the nonlinear comparisons may be used primarily as a robustness check.

Step 2: Residual Nonlinearity Test

For each latent variable, estimate a polynomial regression of each indicator on the factor scores. Using the linear CFA from Step 1, regressing each indicator y_{ij} on the predicted factor scores should. The residuals should produce a flat or uninformative plot with no systematic pattern as a function of the latent variable. If the true relationship is nonlinear, the residuals should show a systematic pattern—e.g., a U-shape for quadratic nonlinearity or an asymmetric pattern for other functional forms. Further, for each indicator *j* of a given latent variable, estimate a polynomial regression using e.g.,

$$Y_{ij} = \gamma_{j0} + \gamma_{j1}\widehat{\eta_i} + \gamma_{j2}\widehat{\eta_i}^2 + \gamma_{j3}\widehat{\eta_i}^3 + \xi_{ij}$$

where $\widehat{\eta_i}$ are the factor scores from the linear CFA. The null hypothesis of linearity is H₀: $\gamma_{j2} = \gamma_{j3} = 0$, tested via an F-test comparing the polynomial model against the linear model. Significant quadratic or cubic terms across indicators suggests the linear measurement model is misspecified.

Step 3: Per-Construct CFA-VAE Comparison

A more direct comparison using the CFA-VAE architecture draws on a held-out ELBO evaluated on validation data. The ELBO evaluated on the same data used for training is optimistic—the nonlinear model has more parameters and will always fit training data at least as well as the linear model, even when the truth is linear. An appropriate comparison criterion is the held-out ELBO evaluated on data not used during training to penalize overfitting. For each latent variable fit both the linear and nonlinear CFA-VAE on an e.g., 80/20 train–validation split. Use identical architecture for both models (same hidden dimension, same β_{kl}, same learning rate, same number of epochs), differing only in the decoder. Let the observed indicators for a single construct be $\mathbf{y} = (y_1, \ldots, y_j)$. For a CFA-VAE with encoder parameters ϕ and decoder parameters θ, the per-observation ELBO is:

$$\mathcal{L}(\phi, \theta;\, y_i) = \mathrm{E}_{q_\phi(\eta \mid y_i)}[\log p_\theta(\mathbf{y}_i \mid \eta)] - KL\left(q_\phi(\eta \mid \mathbf{y}_i) \parallel p(\eta)\right)$$

The held-out ELBO on a validation set *v* is:

$$\hat{\mathcal{L}}_{\text{val}} = \frac{1}{|v|}\sum_{i \in v} \mathcal{L}(\hat{\varphi}, \hat{\theta};\, y_i)$$

Using train-validation split of, for example, 80/20 train, both the linear and nonlinear CFA-VAE are trained on the 80% training set using identical architecture (same hidden layer width and depth, same β_{kl}, same learning rate, same number of epochs), differing only in the decoder. The

held-out ELBO is evaluated on the 20% validation set. The model with the higher held-out ELBO is preferred. Because both models use the same encoder structure, the KL term is comparable across models and the ELBO comparison is primarily driven by the reconstruction quality. A rule of thumb for interpreting the difference is that a per-observation improvement of less than 0.1 natural-logarithm units of information suggests the linear model is adequate; a difference greater than 0.5 natural-logarithm units of information constitutes meaningful evidence for the nonlinear model.

Appendix E

In this appendix, we outline the implementation details of TAG learning for the simulations, provide the full simulation results outlined in the paper and provide additional simulations using different data generating processes (DGPs). Our SEM-VAE is implemented in *R* using the torch package as a custom module with amortized encoders for the latent variables and a shared context network. The context network maps the observed data to a four- or ten-dimensional context vector for base or high dimensional conditions using two linear layers with ReLU activations. Each construct encoder is a two-hidden-layer MLP (hidden width of 64) that takes the local indicators with the context vector and outputs Gaussian variational parameters (μ, $\log \sigma^2$) for each latent variable. On the generative side, indicator blocks are reconstructed through (non)linear factor measurement models with learned intercepts, loadings (or decoder parameters), and residuals. Latent variables are identified by fixing their variances to one in the SEM-VAE with linear CFA outcome proxy (second approach) and the SEM-VAE with CFA-VAE outcome proxy (third approach), and in the full SEM-VAE (first approach) by fixing the variances of exogenous and the residual variance of endogenous latent outcome to one. The treatment mechanism is parameterized as logits, modeled as a function of the latent and observed covariates and trained via binary cross-entropy with logits; latent outcomes or Bartlett(-like) scores for the latent outcome are modeled with mean and log variance networks that are functions of the treatment, and latent and observed covariates. Training uses the Adam optimizer with learning rate 0.001 and batch size 128 with unit weights in the ELBO for KL regularization. We then estimated the treatment effect using the posterior summaries of the latent variables and TMLE with ten-fold cross validation and a super learner built on the mean, generalized linear models, generalized additive models, and multivariate adaptive regression splines and added random forests for high dimensional models. In the generative stage, for simulations, we trained the SEM-VAE once on the full sample. The resulting posterior summaries were then treated as fixed covariates entering the ten-fold TMLE. As a result, latent variable summaries potentially contain a within-sample dependence between the TMLE folds. Cross-fitting the SEM-VAE would address this dependence and support cross-fitting conditions underlying the double robustness and semi-parametric efficiency results. We use this full cross-fitting approach in the subsequent illustrative example and in general this would be a prudent approach. However, cross-fitting the SEM-VAE is computationally expensive in simulations. In our simulations, we drew on amortized inference such that the encoder is a shared function across all observations rather than an observation-specific fit. The gradient signal from any single observation is diluted across mini-batches during training, substantially reducing the within-sample overfitting concern relative to non-amortized methods. In addition, we assessed the empirical sensitivity of our results to training the SEM-VAE once on the full sample by comparing TAG learning with an oracle-type comparator (NLRAG) that uses the identical generative phase but bypasses the targeted learning step with a known correct structural model. When TAG and NLRAG produce similar estimates, it indicates that the SEM-VAE is generating stable, low-bias latent summaries that are not materially distorted by having seen each observation during training. Appendix B outlines the stability condition the SEM-VAE must satisfy for the double robustness and semiparametric efficiency results to hold under the full-data implementation. More generally, however, cross-fitting should be used for both the SEM-VAE and TMLE.

Full results from the simulations outlined in the main body of the paper

Table 1 *Summary of simulation results under linear measurement models, structural models that follow expression (24) with 2 latent covariates and 1 observed covariate, and Bartlett scores as an outcome proxy in the SEM-VAE.*

N	R_y	R_x	R_z	Items	Estimator	Bias	SD	RMSE
500	0.63	0.57	0.58	3	NLSEM	0.00	0.17	0.17
					TAG	-0.07	0.14	0.16
					TMLE-FS	-0.17	0.13	0.21
					LSEM	-0.23	0.13	0.26
					LR-FS	-0.22	0.13	0.26
					NLR-FS	-0.13	0.13	0.18
					LRAG	-0.23	0.13	0.26
					NLRAG	-0.04	0.14	0.15
1000	0.8	0.79	0.8	10	NLSEM	0.00	0.08	0.08
					TAG	-0.03	0.08	0.09
					TMLE-FS	-0.10	0.08	0.13
					LSEM	-0.23	0.08	0.25
					LR-FS	-0.23	0.08	0.24
					NLR-FS	-0.07	0.08	0.11
					LRAG	-0.23	0.08	0.25
					NLRAG	-0.01	0.08	0.08
2000	0.66	0.57	0.56	3	NLSEM	0.01	0.07	0.07
					TAG	0.00	0.08	0.08
					TMLE-FS	-0.14	0.06	0.16
					LSEM	-0.23	0.06	0.24
					LR-FS	-0.22	0.06	0.23
					NLR-FS	-0.12	0.06	0.14
					LRAG	-0.23	0.07	0.24
					NLRAG	0.01	0.07	0.07
2000	0.8	0.79	0.8	10	NLSEM	0.00	0.06	0.06
					TAG	0.01	0.06	0.06
					TMLE-FS	-0.09	0.06	0.11
					LSEM	-0.23	0.06	0.24
					LR-FS	-0.23	0.06	0.24
					NLR-FS	-0.07	0.06	0.09
					LRAG	-0.23	0.06	0.24
					NLRAG	0.02	0.06	0.06
2000	0.89	0.88	0.88	20	NLSEM	0.00	0.06	0.06
					TAG	-0.01	0.06	0.06
					TMLE-FS	-0.07	0.06	0.09
					LSEM	-0.23	0.06	0.24
					LR-FS	-0.23	0.06	0.24
					NLR-FS	-0.05	0.06	0.07
					LRAG	-0.23	0.05	0.24

					NLRAG	0.00	0.06	0.06
5000	0.62	0.6	0.59	3	NLSEM	0.00	0.05	0.05
					TAG	0.00	0.05	0.05
					TMLE-FS	-0.14	0.04	0.15
					LSEM	-0.23	0.04	0.23
					LR-FS	-0.22	0.04	0.23
					NLR-FS	-0.12	0.04	0.13
					LRAG	-0.24	0.04	0.24
					NLRAG	0.01	0.05	0.05

Note. Super learner uses GLM, mean, GAM, and MARS

Table 2 *Summary of simulation results under linear measurement models, structural models that follow expression (24) with 2 latent covariates and 1 observed covariate, and CFA-VAE scores as an outcome proxy in the SEM-VAE.*

N	R_y	R_x	R_z	Items	Estimator	Bias	SD	RMSE
1000	0.55	0.60	0.63	3	NLSEM	0.00	0.10	0.10
					TAG	-0.01	0.09	0.09
					TMLE-FS	-0.20	0.05	0.20
					LSEM	-0.18	0.08	0.20
					LR-FS	-0.23	0.05	0.23
					NLR-FS	-0.19	0.05	0.19
					LRAG	-0.19	0.08	0.20
					NLRAG	0.00	0.09	0.09
2000	0.18	0.60	0.63	3	TAG	-0.03	0.10	0.10
					TMLE-FS	-0.25	0.02	0.25
					LR-FS	-0.26	0.02	0.26
					NLR-FS	-0.25	0.02	0.25
					LRAG	-0.17	0.09	0.19
					NLRAG	-0.02	0.10	0.10
2000	0.57	0.61	0.61	3	TAG	0.02	0.07	0.07
					TMLE-FS	-0.16	0.03	0.17
					LR-FS	-0.19	0.03	0.20
					NLR-FS	-0.16	0.03	0.16
					LRAG	-0.15	0.05	0.16
					NLRAG	0.02	0.07	0.07
2000	0.76	0.80	0.80	10	TAG	0.02	0.05	0.06
					TMLE-FS	-0.09	0.04	0.10
					LR-FS	-0.16	0.04	0.17
					NLR-FS	-0.08	0.04	0.09
					LRAG	-0.14	0.05	0.15

					NLRAG	0.03	0.05	0.06
5000	0.19	0.60	0.63	3	TAG	0.03	0.07	0.07
					TMLE-FS	-0.25	0.01	0.25
					LR-FS	-0.26	0.01	0.26
					NLR-FS	-0.25	0.01	0.25
					LRAG	-0.15	0.06	0.17
					NLRAG	0.03	0.07	0.07
5000	0.57	0.62	0.62	3	TAG	0.04	0.04	0.06
					TMLE-FS	-0.16	0.02	0.16
					LR-FS	-0.19	0.02	0.20
					NLR-FS	-0.16	0.02	0.16
					LRAG	-0.15	0.03	0.15
					NLRAG	0.03	0.04	0.05
10000	0.76	0.80	0.80	3	NLSEM	0.00	0.04	0.04
					TAG	0.00	0.05	0.05
					TMLE-FS	-0.19	0.02	0.19
					LSEM	-0.18	0.04	0.18
					LR-FS	-0.23	0.02	0.23
					NLR-FS	-0.18	0.02	0.19
					LRAG	-0.18	0.04	0.19
					NLRAG	0.00	0.04	0.04
10000	0.56	0.60	0.59	3	NLSEM	0.00	0.03	0.03
					TAG	0.00	0.03	0.03
					TMLE-FS	-0.19	0.02	0.19
					LSEM	-0.18	0.03	0.18
					LR-FS	-0.23	0.02	0.23
					NLR-FS	-0.19	0.02	0.19
					LRAG	-0.19	0.03	0.19
					NLRAG	0.00	0.03	0.03

Note. Super learner uses GLM, mean, GAM, and MARS.

Table 3 *Summary of simulation results when measurement models are linear, structural models that follow expression (24) with 2 latent covariates and 1 observed covariate, and the full SEM-VAE is used*

N	R_y	R_x	R_z	Items	Estimator	Bias	SD	RMSE
500	0.48	0.56	0.56	3	NLSEM	0.00	0.15	0.15
					TAG	-0.05	0.22	0.23
					TMLE-FS	-0.13	0.11	0.17
					LSEM	-0.19	0.11	0.22
					LR-FS	-0.18	0.11	0.21
					NLR-FS	-0.11	0.11	0.16
					LRAG	-0.15	0.18	0.24
					NLRAG	-0.02	0.23	0.23
1000	0.60	0.68	0.68	5	NLSEM	0.00	0.09	0.09
					TAG	-0.03	0.14	0.14
					TMLE-FS	-0.10	0.08	0.13
					LSEM	-0.17	0.09	0.19
					LR-FS	-0.17	0.08	0.19
					NLR-FS	-0.08	0.08	0.11
					LRAG	-0.14	0.12	0.19
					NLRAG	0.00	0.14	0.14
2000	0.53	0.60	0.60	3	NLSEM	0.00	0.07	0.07
					TAG	0.01	0.14	0.14
					TMLE-FS	-0.12	0.06	0.13
					LSEM	-0.18	0.06	0.19
					LR-FS	-0.18	0.06	0.19
					NLR-FS	-0.10	0.06	0.12
					LRAG	-0.13	0.11	0.17
					NLRAG	0.03	0.15	0.15
2000	0.7	0.8	0.8	10	NLSEM	0.00	0.05	0.05
					TAG	-0.01	0.12	0.12
					TMLE-FS	-0.07	0.05	0.08
					LSEM	-0.17	0.05	0.18
					LR-FS	-0.17	0.05	0.18
					NLR-FS	-0.05	0.04	0.07
					LRAG	-0.15	0.09	0.17
					NLRAG	0.01	0.13	0.13
5000	0.52	0.58	0.59	3	NLSEM	0.00	0.04	0.04
					TAG	0.03	0.14	0.14
					TMLE-FS	-0.11	0.04	0.11
					LSEM	-0.18	0.04	0.18
					LR-FS	-0.17	0.04	0.18
					NLR-FS	-0.10	0.04	0.10
					LRAG	-0.11	0.10	0.15
					NLRAG	0.04	0.14	0.15
10000	0.55	0.61	0.61	3	NLSEM	-0.01	0.03	0.03

					TAG	0.03	0.11	0.11
					TMLE-FS	-0.11	0.02	0.12
					LSEM	-0.19	0.02	0.19
					LR-FS	-0.18	0.02	0.18
					NLR-FS	-0.10	0.02	0.11
					LRAG	-0.12	0.06	0.14
					NLRAG	0.06	0.11	0.12

Note. Super learner uses GLM, mean, GAM, and MARS.

Table 4 *Summary of simulation results when the outcome measurement model is nonlinear, structural models that follow expression (24) with 2 latent covariates and 1 observed covariate, and CFA-VAE scores as an outcome proxy in the SEM-VAE*

N	R_y	R_x	R_z	Items	Estimator	Bias	SD	RMSE
1000	0.60	0.70	0.70	5	NLSEM	-0.20	0.03	0.20
					TAG	-0.01	0.09	0.09
					TMLE-FS	-0.15	0.05	0.16
					LSEM	-0.25	0.03	0.25
					LR-FS	-0.19	0.05	0.20
					NLR-FS	-0.14	0.05	0.15
					LRAG	-0.15	0.08	0.17
					NLRAG	0.01	0.09	0.09
2000	0.60	0.70	0.70	5	NLSEM	-0.19	0.02	0.19
					TAG	0.02	0.06	0.06
					TMLE-FS	-0.14	0.04	0.15
					LSEM	-0.24	0.02	0.24
					LR-FS	-0.19	0.04	0.19
					NLR-FS	-0.13	0.04	0.14
					LRAG	-0.14	0.06	0.15
					NLRAG	0.02	0.06	0.06
3000	0.41	0.80	0.80	10	NLSEM	-0.19	0.02	0.19
					TAG	-0.01	0.06	0.06
					TMLE-FS	-0.18	0.02	0.18
					LSEM	-0.24	0.02	0.24
					LR-FS	-0.22	0.02	0.22
					NLR-FS	-0.18	0.02	0.18
					LRAG	-0.14	0.05	0.15
					NLRAG	-0.01	0.05	0.05
3000	0.70	0.80	0.80	10	NLSEM	-0.19	0.01	0.19
					TAG	0.00	0.05	0.05
					TMLE-FS	-0.10	0.03	0.10
					LSEM	-0.24	0.01	0.24
					LR-FS	-0.17	0.03	0.17
					NLR-FS	-0.09	0.03	0.09
					LRAG	-0.14	0.04	0.15
					NLRAG	0.01	0.05	0.05
5000	0.60	0.70	0.70	5	NLSEM	-0.19	0.01	0.19
					TAG	0.02	0.05	0.05
					TMLE-FS	-0.14	0.02	0.14

					LSEM	-0.24	0.01	0.24
					LR-FS	-0.19	0.02	0.19
					NLR-FS	-0.13	0.02	0.14
					LRAG	-0.15	0.04	0.15
					NLRAG	0.02	0.05	0.05

Note. Super learner uses GLM, mean, GAM, and MARS.

Table 5 *Summary of simulation results when all measurement models are nonlinear, structural models that follow expression (24) with 2 latent covariates and 1 observed covariate, and CFA-VAE scores as an outcome proxy in the SEM-VAE*.

N	R_y	R_x	R_z	Items	Estimator	Bias	SD	RMSE
500	0.50	0.56	0.56	3	NLSEM	-0.17	0.14	0.22
					TAG	-0.09	0.44	0.45
					TMLE-FS	-0.19	0.07	0.20
					LSEM	-0.24	0.04	0.25
					LR-FS	-0.20	0.07	0.21
					NLR-FS	-0.19	0.07	0.20
					LRAG	-0.13	0.45	0.47
					NLRAG	-0.08	0.47	0.47
500	0.68	0.76	0.76	10	NLSEM	-0.20	0.04	0.20
					TAG	-0.03	0.13	0.13
					TMLE-FS	-0.13	0.08	0.16
					LSEM	-0.24	0.03	0.25
					LR-FS	-0.17	0.08	0.19
					NLR-FS	-0.12	0.08	0.14
					LRAG	-0.15	0.11	0.18
					NLRAG	-0.01	0.13	0.13
500	0.77	0.83	0.84	20	NLSEM	-0.20	0.03	0.20
					TAG	-0.03	0.12	0.12
					TMLE-FS	-0.10	0.08	0.13
					LSEM	-0.24	0.03	0.25
					LR-FS	-0.16	0.09	0.18
					NLR-FS	-0.09	0.08	0.12
					LRAG	-0.15	0.11	0.18
					NLRAG	-0.11	0.12	0.12
1000	0.69	0.77	0.78	10	NLSEM	-0.20	0.03	0.20
					TAG	-0.02	0.09	0.10
					TMLE-FS	-0.14	0.06	0.15
					LSEM	-0.25	0.03	0.25
					LR-FS	-0.17	0.06	0.18
					NLR-FS	-0.13	0.06	0.14
					LRAG	-0.15	0.08	0.17
					NLRAG	-0.01	0.09	0.09
2000	0.50	0.58	0.57	5	NLSEM	-0.19	0.09	0.21
					TAG	-0.06	0.07	0.09
					TMLE-FS	-0.19	0.03	0.19
					LSEM	-0.24	0.02	0.24
					LR-FS	-0.20	0.03	0.20
					NLR-FS	-0.19	0.03	0.19
					LRAG	-0.15	0.06	0.16
					NLRAG	-0.05	0.07	0.09
2000	0.60	0.67	0.67	5	NLSEM	-0.20	0.02	0.20

					TAG	-0.01	0.11	0.11
					TMLE-FS	-0.17	0.04	0.17
					LSEM	-0.24	0.02	0.24
					LR-FS	-0.19	0.04	0.19
					NLR-FS	-0.16	0.04	0.17
					LRAG	-0.14	0.08	0.16
					NLRAG	0.00	0.12	0.12
3000	0.60	0.67	0.67	5	NLSEM	-0.20	0.02	0.20
					TAG	0.02	0.07	0.07
					TMLE-FS	-0.16	0.03	0.16
					LSEM	-0.24	0.01	0.24
					LR-FS	-0.19	0.03	0.19
					NLR-FS	-0.16	0.03	0.16
					LRAG	-0.14	0.04	0.15
					NLRAG	0.03	0.06	0.07
5000	0.60	0.64	0.64	5	NLSEM	-0.20	0.02	0.20
					TAG	0.00	0.05	0.05
					TMLE-FS	-0.17	0.03	0.17
					LSEM	-0.25	0.01	0.25
					LR-FS	-0.19	0.03	0.19
					NLR-FS	-0.17	0.03	0.17
					LRAG	-0.15	0.04	0.16
					NLRAG	0.00	0.05	0.05
5000	0.70	0.77	0.78	10	NLSEM	-0.20	0.01	0.20
					TAG	0.01	0.05	0.05
					TMLE-FS	-0.13	0.02	0.13
					LSEM	-0.24	0.01	0.24
					LR-FS	-0.17	0.02	0.17
					NLR-FS	-0.12	0.02	0.12
					LRAG	-0.15	0.03	0.15
					NLRAG	0.02	0.05	0.05
5000	0.60	0.68	0.68	5	NLSEM	-0.20	0.01	0.20
					TAG	0.01	0.05	0.05
					TMLE-FS	-0.16	0.02	0.17
					LSEM	-0.24	0.01	0.24
					LR-FS	-0.19	0.02	0.19
					NLR-FS	-0.16	0.02	0.16
					LRAG	-0.15	0.04	0.15
					NLRAG	0.02	0.05	0.06

Note. Super learner uses GLM, mean, GAM, and MARS.

Table 6 *Summary of simulation results under linear measurement models, structural models that follow expression (24) with 2 latent covariates and 10 observed covariates, and Bartlett scores as an outcome proxy in the SEM-VAE*

N	R_y	R_x	R_z	Items	Estimator	Bias	SD	RMS
2000	0.92	0.88	0.87	10	NLSEM	0.00	0.05	0.047
					TAG	-0.04	0.05	0.059
					TMLE-FS		0.05	0.087
					LSEM	-0.21	0.05	0.218
					LR-FS	-0.21	0.05	0.218
					NLR-FS	-0.04	0.05	0.06
					LRAG	-0.21	0.05	0.218
					NLRAG	-0.01	0.05	0.047
10000	0.70	0.60	0.60	3	NLSEM	0.01	0.03	0.03
					TAG	0.00	0.03	0.03
					TMLE-FS	-0.13	0.03	0.14
					LSEM	-0.21	0.03	0.21
					LR-FS	-0.21	0.03	0.21
					NLR-FS	-0.11	0.03	0.12
					LRAG	-0.21	0.03	0.22
					NLRAG	0.01	0.03	0.03

Note. Super learner uses GLM, mean, GAM, and MARS

Table 7 *Summary of simulation results under linear measurement models, structural models that follow expression (24) with 5 latent covariates and 10 observed covariates, and Bartlett scores as an outcome proxy in the SEM-VAE*

N	R_y	R_x	R_z	Items	Estimator	Bias	SD	RMSE
2000	0.70	0.62	0.64	4	NLSEM	0.02	0.09	0.09
					TAG	-0.15	0.07	0.17
					TMLE-FS	-0.22	0.07	0.23
					LSEM	-0.35	0.07	0.36
					LR-FS	-0.35	0.07	0.36
					NLR-FS	-0.17	0.07	0.19
					LRAG	-0.35	0.07	0.36
					NLRAG	-0.10	0.08	0.13
2000	0.84	0.78	0.8	10	TAG	-0.08	0.07	0.11
					TMLE-FS	-0.15	0.07	0.17
					LSEM	-0.35	0.07	0.36
					LR-FS	-0.35	0.07	0.36
					NLR-FS	-0.11	0.07	0.12
					LRAG	-0.35	0.07	0.36
					NLRAG	-0.04	0.07	0.08
5000	0.7	0.65	0.65	4	NLSEM	-0.01	0.03	0.03
					TAG	-0.06	0.05	0.08
					TMLE-FS	-0.20	0.05	0.21
					LSEM	-0.35	0.05	0.35
					LR-FS	-0.35	0.05	0.35
					NLR-FS	-0.16	0.05	0.17
					LRAG	-0.35	0.05	0.35
					NLRAG	-0.03	0.05	0.06
5000	0.84	0.78	0.8	10	TAG	-0.04	0.05	0.06
					TMLE-FS	-0.15	0.04	0.15
					LSEM	-0.35	0.04	0.35
					LR-FS	-0.35	0.04	0.35
					NLR-FS	-0.11	0.04	0.11
					LRAG	-0.35	0.04	0.35
					NLRAG	-0.01	0.04	0.04
10000	0.70	0.62	0.64	4	NLSEM	0.01	0.04	0.04
					TAG	-0.03	0.04	0.05
					TMLE-FS	-0.20	0.03	0.20
					LSEM	-0.35	0.03	0.35
					LR-FS	-0.35	0.03	0.35
					NLR-FS	-0.17	0.03	0.17

					LRAG	-0.35	0.03	0.35
					NLRAG	-0.01	0.04	0.04

Note. Super learner uses GLM, mean, GAM, MARS, and random forests.

Table 8 *Summary of simulation results when true effect is zero under linear measurement models, structural models that follow expression (24) with 2 latent covariates and 1 observed covariate, and CFA-VAE scores as an outcome proxy in the SEM-VAE.*

N	R_y	R_x	R_z	Items	Estimator	Bias	SD	RMSE
500	0.55	0.60	0.63	3	NLSEM	0.02	0.17	0.32
					TAG	-0.06	0.12	0.38
					TMLE-FS	-0.07	0.07	0.38
					LSEM	-0.17	0.12	0.49
					LR-FS	-0.10	0.07	0.40
					NLR-FS	-0.06	0.07	0.36
					LRAG	-0.17	0.11	0.48
					NLRAG	-0.04	0.12	0.36
1000	0.55	0.60	0.63	3	NLSEM	0.00	0.09	0.31
					TAG	-0.01	0.10	0.33
					TMLE-FS	-0.07	0.05	0.37
					LSEM	-0.18	0.08	0.49
					LR-FS	-0.10	0.05	0.40
					NLR-FS	-0.06	0.05	0.36
					LRAG	-0.19	0.08	0.49
					NLRAG	0.00	0.09	0.31
2000	0.57	0.61	0.61	3	NLSEM	0.00	0.07	0.31
					TAG	0.01	0.07	0.30
					TMLE-FS	-0.07	0.04	0.37
					LSEM	-0.18	0.06	0.48
					LR-FS	-0.10	0.03	0.40
					NLR-FS	-0.06	0.03	0.36
					LRAG	-0.19	0.06	0.49
					NLRAG	0.01	0.07	0.30
5000	0.54	0.60	0.60	3	NLSEM	0.00	0.04	0.30
					TAG	0.01	0.04	0.30
					TMLE-FS	-0.06	0.02	0.36
					LSEM	-0.18	0.04	0.48
					LR-FS	-0.10	0.02	0.40
					NLR-FS	-0.06	0.02	0.36

					LRAG	-0.19	0.04	0.49
					NLRAG	0.01	0.04	0.30
10000	0.54	0.60	0.60	3	NLSEM	0.00	0.03	0.30
					TAG	0.00	0.03	0.30
					TMLE-FS	-0.05	0.01	0.35
					LSEM	-0.18	0.03	0.48
					LR-FS	-0.10	0.01	0.40
					NLR-FS	-0.05	0.01	0.35
					LRAG	-0.19	0.03	0.49
					NLRAG	0.03	0.03	0.27

Additional simulation results for different DGPs

DGP 2: Propensity S-Curve; Outcome with Saturating Loss

In addition to the base DGPs noted in the paper, we also considered another set of DGPs (DGP 2) designed to test the TAG estimator under two simultaneous challenges. First, treatment selection follows a sigmoidal propensity in each latent covariate that steepens in the extremes, equivalent to strong positive curvature on the logit scale after a tanh transform. Second, the latent outcome mean follows a concave-saturating function of the latent covariates so that high-X/high-Z regions penalize the treated mean unless nonlinear terms are explicitly modeled. These two features are designed to point in the same direction: estimators that omit the nonlinear and interaction terms will systematically underestimate the true treatment effect, with measurement error in the latent covariates amplifying the bias.

DGP 2 complements the main simulation in which the structural functions are polynomial. Whereas polynomial nonlinearity is symmetric and grows unboundedly in the tails, DGP 2 uses bounded nonlinear functions (tanh for treatment selection; negative softplus for the outcome) that saturate at extreme covariate values. This saturation structure allows effects to diminish at the extremes of a latent construct.

The probability of treatment assignment is modeled via a logistic regression in which each latent covariate enters through a hyperbolic tangent (tanh) transformation, together with a bilinear interaction term and the observed covariate W:

$$\text{logit}\, p(\mathrm{T} = 1 \mid X, Z, W) = \gamma_0 + a \tanh(\alpha_X X) + a \tanh(\alpha_Z Z) + \gamma_{12} XZ + \gamma_W W$$

The conditional mean of the latent outcome given treatment and latent covariates is:

$$\mu_Y = \tau T + b\,[-\log\{1 + \exp(\lambda_X X)\}] + b[-\log\{1 + \exp(\lambda_Z Z)\}] - \beta_{12} XZ + \beta_W W$$

Table 11 reports Monte Carlo simulation results for DGP 2. Three patterns are of particular interest under DGP 2. First, all estimators that use linear structural models exhibit negative bias whose magnitude exceeds that observed under DGP 1 at the same sample sizes. This reflects the double penalization mechanism described above: the S-curve propensity and saturating outcome interact to produce larger and more uniform bias when nonlinear terms are omitted. Second, estimators using latent variable summaries from the SEM-VAE (which employs flexible neural network structural models) show substantially less bias than factor-score approaches, further demonstrating that the super learner in the targeted learning step can adapt to complex nonlinear structures even when the outcome network in the SEM-VAE was trained on a flexible but not oracle structural form. Third, the relationship between sample size and bias

reduction is steeper under DGP 2 than DGP 1, consistent with the theoretical prediction that the nonlinear functions tanh and negative softplus require larger samples for the neural network structural models to achieve the $n^{-1/4}$ convergence rate required by the product-of-rates conditions.

DGP 2 provides a demanding test of the TAG estimator under two simultaneous challenges: an S-shaped propensity that saturates at extreme latent covariate values and a concave-saturating outcome function that penalizes high-X/high-Z regions. The structural alignment between these two mechanisms ensures that misspecification of either the functional form or the latent covariate measurement model produces bias in the same direction, with the two sources compounding rather than cancelling. The TAG estimator, by combining the SEM-VAE's flexible latent covariate summaries with the targeted learning step's adaptive nuisance function estimation, is designed to address both sources of bias simultaneously. The simulation results in Table 11 provide empirical evidence on how well this design succeeds under the specific parameter values and sample sizes considered in the main paper.

Table 9 *Summary of simulation results under DGP 2 with 2 latent covariates and 1 observed covariate, and CFA-VAE scores as an outcome proxy in the SEM-VAE.*

N	R_y	R_x	R_z	Items	Estimator	Bias	SD	RMSE
2000	0.55	0.58	0.58	3	TAG	0.01	0.06	0.06
					TMLE-FS	-0.13	0.03	0.13
5000	0.55	0.58	0.58	3	TAG	0.01	0.04	0.04
					TMLE-FS	-0.13	0.02	0.13

DGP 3: Piecewise (Threshold) Propensity; Outcome with Cubic Flattening that is Locally Concave

We also examined a third DGP (DGP 3) designed to assess the TAG estimator under a qualitatively different form of nonlinearity than the other DGPs. Rather than smooth curvature throughout the covariate space, DGP 3 introduces threshold-driven selection: units with extreme latent covariate values ($|X| > c$ or $|Z| > c$) receive a discrete step up in treatment probability beyond what the linear propensity terms alone would predict. Simultaneously, the outcome function is locally concave near and beyond the threshold c through cubic terms in X and Z, yielding downward curvature similar to a negative quadratic in the region where treated units concentrate. Approaches that fit smooth linear-additive models and ignore measurement error will obtain systematically biased estimates of the treatment effect, with the bias pointing in the same negative direction as the previous DGPs.

DGP 3 is distinctive in two respects. First, the treatment propensity is not smooth: the indicator functions create a discontinuity (a step) in the logit at $|X| = c$ and $|Z| = c$. This means that any smooth nuisance function estimator—including neural networks, GAMs, and GLMs—must approximate a discontinuous propensity surface, creating a harder model selection problem than under DGPs 1 or 2. Second, the cubic outcome function creates a region of increasing outcome for moderate |X| and |Z| and a region of decreasing outcome for large |X| and |Z|, with the turning point determined by the ratio of the linear to cubic coefficients.

The log-odds of treatment assignment is a piecewise linear function of the latent covariates and the observed covariate *W*:

$$\text{logit}\, p(\text{T} = 1 \mid X, Z, W) = \gamma_0 + \gamma_1 X + \gamma_2 Z + \delta(\mathbf{1}\{|X| > c\} + \mathbf{1}\{|Z| > c\}) + \gamma_W W$$

The propensity score can be written explicitly as a piecewise function of the latent covariates. Define the following four exclusive regions of the (X, Z) plane:

Region I (interior): |X| ≤ c and |Z| ≤ c. In this region both indicators are zero and the logit reduces to the smooth linear form.

$$\text{logit}\, p = \gamma_0 + \gamma_1 X + \gamma_2 Z + \gamma_W W \quad \text{(Region I)}$$

Region II (one extreme): |X| > c and |Z| ≤ c, or |X| ≤ c and |Z| > c. Exactly one indicator is active and the logit receives a single step of size δ.

$$\text{logit}\, p = \gamma_0 + \gamma_1 X + \gamma_2 Z + \delta + \gamma_W W \quad \text{(Region II)}$$

Region III (both extremes): |X| > c and |Z| > c. Both indicators are active and the logit receives a double step of size 2δ.

$$\text{logit}\, p = \gamma_0 + \gamma_1 X + \gamma_2 Z + 2\delta + \gamma_W W \quad \text{(Region III)}$$

This piecewise structure creates two discontinuities in the propensity surface: one at |X| = c (as a function of X for fixed Z) and one at |Z| = c (as a function of Z for fixed X). These discontinuities are the defining feature of DGP 3 and the primary source of difficulty for smooth estimators. Any estimator that assumes a smooth propensity surface will necessarily misfit the propensity in the neighborhood of the thresholds, creating systematic weighting errors that bias the ATE estimate.

The conditional mean of the latent outcome is a cubic polynomial in the latent covariates with a negative interaction and a linear W term:

$$\mu_Y = \tau T + \theta_X X - \kappa_X X^3 + \theta_Z Z - \kappa_Z Z^3 - \beta_{12} XZ + \beta_W W$$

The DGP produces a locally concave function that increases for moderate |X| or |Z| and decreases for large |X| or |Z|. The negative interaction $-\beta_{12}XZ$ penalizes units where X and Z have the same sign, which is precisely where treatment concentration is highest under DGP 3.

The result of DGP 3 is that the units receiving elevated treatment probability (those with |X| > c or |Z| > c) are the same units for whom the outcome function is most concave (and most likely to be below its linear approximation). This alignment ensures that misspecified estimators will systematically underestimate the treatment effect. The combination of the mechanisms defining DGP 3 is particularly challenging for factor-score-based estimators.

Table 12 reports Monte Carlo simulation results for DGP 3. Results were in line with other data generating processes.

Table 10 *Summary of simulation results under DGP 3 with 2 latent covariates and 1 observed covariate, and CFA-VAE scores as an outcome proxy in the SEM-VAE.*

N	R_y	R_x	R_z	Items	Estimator	Bias	SD	RMSE
2000	0.55	0.58	0.58	3	TAG	-0.04	0.06	0.07
					TMLE-FS	-0.11	0.03	0.12
5000	0.55	0.58	0.58	3	TAG	-0.03	0.04	0.05
					TMLE-FS	-0.11	0.02	0.11

Appendix F

In this appendix we outline the posterior summary methods we used for latent variables in TAG learning. TAG learning uses posterior summaries of the latent variable distributions produced by the SEM-VAE that are suitable for subsequent targeted learning. The posterior summaries make use of different methods for latent covariates (where posterior means or regression-like scores are useful) and for latent outcomes (where Bartlett-like scores are useful). This appendix outlines the four scoring approaches used in the paper, corresponding to the linear and nonlinear versions of each role.

Under a linear measurement model, factor scores for a latent variable η can be expressed as a linear combination of mean-centered indicators:

$$\widehat{\eta_t^Y} \;=\; \mathbf{w}^{\mathrm{T}}(\mathbf{Y} \;-\; \boldsymbol{\alpha})$$

where $\mathbf{w}^{\mathrm{T}}$ is the vector of scoring weights, Y is the indicator vector, and $\boldsymbol{\alpha}$ is the vector of indicator means. The two classical choices of weight vector are the Bartlett weights and the regression weights:

$$\mathbf{w}^{\mathrm{B}} \;=\; (\boldsymbol{\Lambda}'\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda})^{-1}\boldsymbol{\Lambda}'\boldsymbol{\Psi}^{-1}$$
$$\mathbf{w}^{\mathrm{R}} \;=\; \boldsymbol{\Sigma}\boldsymbol{\Lambda}'(\boldsymbol{\Lambda}\boldsymbol{\Sigma}\boldsymbol{\Lambda}' \;+\; \boldsymbol{\Psi})^{-1}$$

with $\boldsymbol{\Lambda}$ as the factor loadings, $\boldsymbol{\Psi}$ as the residual covariance matrix of the indicators, and $\boldsymbol{\Sigma}$ as the latent variable covariance matrix. The key properties distinguishing the two methods are: Bartlett scores are conditionally unbiased estimators of the latent variable but have larger measurement f(17error variance than regression scores; regression scores are biased toward zero (scaled by the reliability) but have smaller variance.

Linear Measurement Model: Regression-Like Scores for Latent Covariates

Under a linear measurement model, the posterior mean of a latent covariate from a CFA-VAE is numerically equivalent to the regression factor score when the encoder does not condition on the context vector. For a latent covariate η_{i}^X with indicator block $\mathbf{x}$

$$X_{\mathrm{ij}} \;=\; \alpha_{\mathrm{j}}^X \;+\; \lambda_{\mathrm{j}}^X\eta_{\mathrm{i}}^X \;+\; \varepsilon_{\mathrm{ij}}^X$$

the variational posterior is Gaussian with parameters produced by the encoder network:

$$q_\varphi(\eta_{\mathrm{i}}^X \mid \mathbf{x}_{\mathrm{i}}, \boldsymbol{C}_{\mathrm{i}}) \;=\; \mathrm{N}\left(\mu_\varphi(\mathbf{x}_{\mathrm{i}}, \boldsymbol{C}_{\mathrm{i}}), {\sigma_\varphi}^2(\mathbf{x}_{\mathrm{i}}, \boldsymbol{C}_{\mathrm{i}})\right)$$

The posterior mean $\mu_\varphi(\mathbf{x}_{\mathrm{i}}, \boldsymbol{C}_{\mathrm{i}})$ is used as the latent covariate summary. When the encoder does not condition on context, this reduces to the classical regression factor score.

Nonlinear Measurement Model: EKF/Laplace Approximation for Latent Covariates

When the measurement model is nonlinear, e.g.,

$$X_{\mathrm{ij}} \;=\; \alpha_{\mathrm{j}}^X + f_\theta(\eta_{\mathrm{i}}^X) \;+\; \varepsilon_{\mathrm{ij}}^X, \quad \varepsilon_{\mathrm{ij}}^X \;\sim\; \mathrm{N}(0, {\sigma_\theta}^2), \quad \eta_{\mathrm{i}}^X \;\sim\; \mathrm{N}\left(0, {\sigma_{\eta^X}}^2\right)$$

the posterior mean is analytically intractable because f_θ is a nonlinear function. We approximate it using a local linearization approach that combines a first-order Taylor expansion around an initial estimate of the latent variable with a linear-Gaussian Kalman update. At the current iterative estimate $\widehat{\eta_{\mathrm{t}}^X}$, we linearize f_θ such that:

$$f_\theta(\eta_{\mathrm{i}}^X) \;\approx\; f_\theta\left(\widehat{\eta_{\mathrm{t}}^X}\right) \;+\; J_{\mathrm{t}}\left(\eta_{\mathrm{i}}^X - \widehat{\eta_{\mathrm{t}}^X}\right)$$

where the Jacobian (linearized observation matrix) evaluated at the current iteration

$$J_{\mathrm{t}} \;=\; \frac{\partial f_\theta}{\partial \eta}\quad {\scriptstyle \eta \,=\, \widehat{\eta_{\mathrm{t}}^X}}$$

Substituting the linearization into the Gaussian measurement likelihood yields a locally linear-Gaussian model. The Kalman (posterior mean) update step then moves the current estimate toward its conditional expectation given the indicator responses:

$$\widehat{\eta_{t+1}^{X}} = \widehat{\eta_{t}^{X}} + K_{t}\left(\mathbf{x}_{i} - f_{\theta}\left(\widehat{\eta_{t}^{X}}\right)\right)$$

where the Kalman gain is:

$$K_{t} = P_{t}J_{t}'(J_{t}P_{t}J_{t}' + \boldsymbol{\Psi})^{-1}$$

with P_t as the prior (or current posterior) variance of η_{i}^{X}. The result $\widehat{\eta_{t+1}^{X}}$ is a first-order approximation to the posterior mean and constitutes the regression-like score under the nonlinear measurement model.

Linear Measurement Model: Bartlett Scores for the Latent Outcome

Under a linear measurement model for the outcome

$$Y_{ij} = \alpha_{j}^{Y} + \lambda_{j}^{Y}\eta_{i}^{Y} + \varepsilon_{ij}^{Y}$$

we use conventional Bartlett factor scores as the outcome proxy. This approach is best suited when a standard linear factor model is appropriate for the latent outcome. The Bartlett score is:

$$\hat{\eta}_{i}^{Y(B)} = \mathbf{w}^{B,T}(\mathbf{y}_{i} - \boldsymbol{\alpha}^{Y}) = \left(\boldsymbol{\Lambda}^{Y\prime}\boldsymbol{\Psi}^{Y^{-1}}\boldsymbol{\Lambda}^{Y}\right)^{-1}\boldsymbol{\Lambda}^{Y\prime}\boldsymbol{\Psi}^{Y^{-1}}(\mathbf{y}_{i} - \boldsymbol{\alpha}^{Y})$$

As previously noted, The conditional expectation of the Bartlett score given the true latent outcome equals the true latent outcome itself. Substituting the Bartlett weights into the general conditional expectation expression:

$$\mathrm{E}\left[\hat{\eta}_{i}^{Y,B} \mid \eta_{i}^{Y}\right] = (\mathbf{w}^{B,T}\boldsymbol{\Lambda}^{Y})\eta_{i}^{Y} = (\boldsymbol{\Lambda}^{Y\prime}\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda})^{-1}\boldsymbol{\Lambda}^{Y\prime}\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\cdot\eta_{i}^{Y} = \mathrm{I}\cdot\eta_{i}^{Y} = \eta_{i}^{Y}$$

When measurement error is independent of treatment and covariates, this conditional unbiasedness extends to:

$$\mathrm{E}\left[\hat{\eta}_{i}^{Y,B} \mid \eta_{i}^{Y}, \mathrm{T}, \mathbf{X}\right] = \mathrm{E}\left[\hat{\eta}_{i}^{Y,B} \mid \eta_{i}^{Y}\right] = \eta_{i}^{Y}$$

As a result, Bartlett scores can be substituted for the latent outcome in the EIF without introducing bias into the ATE estimate. The Bartlett scores introduce measurement error ε_{i}^{Y} that is additive and mean-zero:

$$\hat{\eta}_{i}^{Y(B)} = \eta_{i}^{Y} + \varepsilon_{i}^{Y}, \quad \mathrm{E}[\varepsilon_{i}^{Y} \mid \mathrm{T}, \mathbf{X}] = 0$$

Because η_{i}^{Y} enters the EIF linearly through the residual, substituting Bartlett scores preserves the target estimand and the double robustness property of TMLE: additive mean-zero outcome error perturbs the EIF but does not shift the estimated parameter. The cost is variance inflation by the factor $(1/\rho_{Y})$, where $\rho_{Y} < 1$ is the reliability of the latent outcome measurement. This variance inflation is captured in the κ_{meas} term in the semiparametric efficiency theorem (Appendix B).

Nonlinear Measurement Model: Gauss–Newton Bartlett-Like Scores for the Latent Outcome

When the outcome measurement model is nonlinear

$$Y_{ij} = \alpha_{j}^{Y} + f_{\theta}(\eta_{i}^{Y}) + \varepsilon_{ij}^{Y}, \quad \varepsilon_{ij}^{Y} \sim \mathrm{N}(0, {\sigma_{\theta}}^{2}), \quad \eta_{i}^{Y} \sim \mathrm{N}\left(0, {\sigma_{\eta^{Y}}}^{2}\right)$$

the conventional Bartlett weights require the linear loading matrix Λ, which does not exist when f_θ is nonlinear. We instead use a locally unbiased, minimum-measurement-error-variance estimator that generalizes Bartlett scores to the nonlinear case using a Gauss-Newton linearization of f_θ.

Linearization and Gauss–Newton Bartlett Weights

At the current iterate $\widehat{\eta_t^Y}$ (initialized, for example, by the posterior mean from the encoder), linearize f_θ at $\widehat{\eta_t^Y}$:

$$f_\theta(\eta_i^Y) \approx f_\theta(\widehat{\eta_t^Y}) + J_t(\eta_i^Y - \widehat{\eta_t^Y})$$

where the outcome Jacobian at the current iterate is:

$$J_t = \frac{\partial f_\theta}{\partial \eta^Y} \quad \eta^Y = \widehat{\eta_t^Y}$$

Replacing the linear loading matrix Λ^y with the Jacobian J_t in the Bartlett weight formula yields the nonlinear Bartlett weights:

$$\mathbf{w}_t^{NL,B} = \left(J_t' \mathbf{\Psi}^{Y^{-1}} J_t\right)^{-1} J_t' \mathbf{\Psi}^{Y^{-1}}$$

The nonlinear Bartlett-like score at iteration t is then:

$$\hat{\eta}_{t+1}^{Y,B} = \widehat{\eta_t^Y} + \mathbf{w}_t^{NL,B}\left(\mathbf{y}_i - f_\theta(\widehat{\eta_t^Y})\right)$$

Local Unbiasedness Property

Under the linear approximation, the conditional expectation of the nonlinear Bartlett-like score is locally unbiased:

$$E\left[\hat{\eta}_{t+1}^{Y(B)} \mid \eta_i^Y\right] \approx \eta_i^Y \quad \text{(to first order in the linearization error)}$$

This is established by substituting the linearization into the conditional expectation and applying the Gauss-Newton weight formula. Deviations from exact unbiasedness arise from the second and higher-order terms in the Taylor expansion of f_θ, and these are small when $\widehat{\eta_t^Y}$ is close to the true $\widehat{\eta_i^Y}$. In our implementation, the $\widehat{\eta_t^Y}$ is initialized with the posterior mean from the CFA-VAE encoder and updated once or twice via the Gauss-Newton step. For smooth, moderately nonlinear f_θ, one or two iterations are typically sufficient for the local approximation to be accurate. The resulting scores satisfy the functional congeniality condition (Appendix B) approximately, with approximation error of order $O(\|\widehat{\eta_i^Y} - \widehat{\eta_t^Y}\|_2)$, which is small when the encoder posterior mean is a good initial estimate of the true latent value.

Appendix G: Details of the Illustrative Example

In this appendix, we outline the illustrative example. We used data collected through an open-access online personality assessment administered through the Open Source Psychometrics Project (https://openpsychometrics.org). Participants completed the 50-item IPIP representation of the Big Five personality dimensions and a supplemental survey including the 12-item Grit-S Scale (Duckworth et al., 2007) and demographic questions. In our analysis we used only complete cases and the first three indicators for each of the latent variables: Grit indicators GS1, GS2, GS3; Agreeableness indicators A1, A2, A3; and Conscientiousness indicators C1, C2, C3. We recoded the a four-point education scale (1 = less than high school, 2 = high school, 3 = university degree, 4 = graduate degree) to create an indicator of whether the individual completed a college degree and/or graduate degree (42%).

The Grit-S scale contains two subscales: Perseverance of Effort (6 items) and Consistency of Interest (6 items). For parsimony in this illustration, we used three items from the full scale (GS1, GS2, GS3), reverse-scoring items identified as negatively keyed (GS1). The resulting three-item grit parcel does not distinguish between the two facets and represents a simplified overall Grit dimension. Similarly, three agreeableness items (A1, A2, A3) and three Conscientiousness items (C1, C2, C3) were selected, with reverse-coding applied to the negatively keyed items from each scale (A2 and C2). We used linear measurement models with unit-variance identification for each latent variable.

We trained the SEM-VAE using Adam optimization (learning rate 0.001, batch size 128) with a maximum of 200 epochs and early stopping based on a 10% within-training-fold validation split; training was halted when the relative improvement in the validation ELBO fell below 0.1% for five consecutive validation checks evaluated every 10 epochs. The SEM-VAE architecture used 64-unit hidden layers, a context output dimension of 4, and a KL regularization weight of β_{KL}=1. We used 5 folds to cross fit the SEM-VAE and the super learner ensemble consisted of GLMs, GAMs, mean and MARS. The bootstrap uncertainty quantification used 100 case-resampled replicates.

We examined the impact of college graduation on grit while controlling for two latent covariates—agreeableness and conscientiousness. We emphasize that the example simply demonstrates a convenient and accessible methodological illustration rather than a substantive investigation. The resulting estimates should not be interpreted as evidence of an effect of educational attainment on grit for numerous reasons. For example, all measures were collected at a single time point in a cross-sectional online survey, so there is no temporal ordering that would support treating personality traits as antecedents of a grit level that is subsequently shaped by educational experience. If anything, the causal order could run in the reverse direction—grittier individuals may be more likely to pursue and complete higher education—making the treatment-outcome relationship bidirectional by design. Second, educational attainment is heavily confounded with a wide array of personal and background characteristics that are not measured in these data or not included in the model (e.g., age, socioeconomic background, cognitive ability, early family environment, and baseline grit prior to any educational exposure). The absence of these confounders from the model means the estimated average treatment effect cannot be interpreted as the effect of education net of selection, and the illustration should not be taken as a careful analysis of the causal impact of education on grit. Third, the latent constructs and indicators used potentially contain more complicated considerations and structures (e.g., multidimensionality, cross-loadings. The illustration is therefore best understood as a

demonstration of TAG learning's computational and inferential machinery rather than as a substantive investigation.

R Code implementing TAG

```
# ============================================================================
# TAG Learning
#
# Phase 0 : Measurement model selection
# Phase 1 : K-fold SEM-VAE cross-fitting + TMLE  (main estimate)
# Phase 2 : Bootstrap confidence intervals       (optional)
# ============================================================================

# ----------------------------------------------------------------------------
# LIBRARIES
# ----------------------------------------------------------------------------
library(lavaan)
library(psych)
library(psychTools)
library(semTools)
library(SuperLearner)
library(tmle)
library(torch)

#specify model using lavaan like expressions
#specify both the outcome and the treatment models
model_spec <- "
            G=~NA*GS1+GS2+GS3
            G~~1*G
            A=~NA*A1+A2+A3
            A~~1*A
            C=~NA*C1+C2+C3
            C~~1*C
            G~college+ A+C
            college~  A+C
"

# ── Data preparation ──────────────────────────────────────────────────────────
#setwd()
#example using GRIT data from https://openpsychometrics.org

{
dat <- read.table(
  "data.csv",
  header = TRUE,
  sep = "\t",
  stringsAsFactors = FALSE,
  quote = "",
  fill = TRUE
)
```

```
rev_items <- list(
  # Big Five
  E = c("E2", "E4", "E6", "E8","E10"),
  N = c("N2", "N4"),
  A = c("A2", "A4", "A6", "A8", "A9", "A10"),
  C = c("C2", "C4", "C6", "C8"),
  O = c("O2", "O4", "O6"),
  GS = c("GS1", "GS4", "GS6", "GS9", "GS10", "GS12")
)

reverse_code <- function(df, items) {
  df[items] <- lapply(df[items], function(x) ifelse(x == 0, 0, -x))
  df
}

for (grp in names(rev_items)) {
  dat <- reverse_code(dat, rev_items[[grp]])
}


datna<-na.omit(dat)

#education          "How much education have you completed?" 1=Less than high school, 2=High school,
3=University degree, 4=Graduate degree
datna$college <- 0
datna$college[which(datna$education>2)] <- 1

dat <- datna

}

# ── Identification method ──────────────────────────────────────────────────
# "auto"          : inferred from ~~ 1*LV in model_spec
# "unit_variance"  : force unit variance for all latent vars
# "marker_loading" : force marker-loading ID for all
id_method_override <- "auto"

# ── MEASUREMENT MODEL MODE
──────────────────────────────────────────────────────────────
# Controls how the measurement model is chosen for each latent variable.
#
#   "auto"      : run Appendix D selection protocol (Phase 0) — fits both
#                 linear and nonlinear CFA-VAE for each block and selects
#                 based on held-out ELBO, CFI/RMSEA, and residual F-tests.
#   "linear"    : use linear measurement for ALL latent variables; skip Phase 0.
#   "nonlinear" : use nonlinear (tanh MLP) measurement for ALL latent
#                 variables; skip Phase 0.
#
measurement_mode <- "linear" #"auto"   # "auto" | "linear" | "nonlinear"

# ── BOOTSTRAP CONFIDENCE INTERVALS
──────────────────────────────────────────────────────
# NOTE: each bootstrap replicate may take a long time
use_bootstrap    <- TRUE  # TRUE to add bootstrap CIs; FALSE to skip
n_bootstrap      <- 100     # number of bootstrap replicates (default 100)
bootstrap_seed   <- runif(1,0,1000)# seed
```

```
ci_level        <- 0.95    # coverage for percentile CI
boot_epoch_scale <- 1.0     # fraction of main epochs used for bootstrap fits

# ── K-fold and repetitions ──────────────────────────────────────────────────────

K      <- 5   # folds for SEM-VAE cross-fitting AND TMLE (must match)
n_reps <- 1   # repetitions to test for stability of optimization



# ── SEM-VAE hyperparameters
─────────────────────────────────────────────────────────────────

N_neurons       <- 64
batchsize       <- 128
nepochs         <- 500 # more complex models require more epochs
context_dim_out <- 4
betakl          <- 1.0
val_prop_inner  <- 0.10
min_rel_improve <- 0.001
patience        <- 5
validate_every  <- 10

# ── CFA-VAE hyperparameters
─────────────────────────────────────────────────────────────────

epochs_cfavae         <- 500
hidden_cfavae         <- 64
beta_kl_cfavae        <- 1.0
val_prop_es           <- 0.10
patience_cfavae       <- 5
validate_every_cfavae <- 10

# ── measurement model selection thresholds (used only when measurement_mode="auto")
val_prop_selection <- 0.20
cfi_threshold      <- 0.90
rmsea_threshold    <- 0.10
alpha_nonlin       <- 0.05
elbo_strong        <- 0.50
elbo_weak          <- 0.20

# ── Gauss-Newton scoring steps ──────────────────────────────────────────────────
n_steps_nonlinear <- 2

# ── TMLE super learner
──────────────────────────────────────────────────────────────────

SL.lib <- c("SL.glm", "SL.gam", "SL.mean", "SL.earth")

# ============================================================================
# MODEL PARSER — for model specificaiton
# ============================================================================

parse_tag_model <- function(model_str) {
  lines <- unlist(strsplit(model_str, "[;\n\r]"))
  lines <- trimws(lines); lines <- lines[nchar(lines) > 0]
  measurement <- list(); unit_var_lvs <- character(0); structural <- list()

  for (line in lines) {
```

```
  if (grepl("=~", line, fixed=TRUE)) {
    p   <- strsplit(line, "=~", fixed=TRUE)[[1]]
    lv  <- trimws(p[1])
    rhs <- strsplit(trimws(p[2]), "\\+")[[1]]
    rhs <- trimws(sub("^(NA|[0-9]+\\.?[0-9]*)\\s*\\*\\s*", "", trimws(rhs)))
    rhs <- rhs[nchar(rhs) > 0]
    measurement[[lv]] <- rhs
  } else if (grepl("~~", line, fixed=TRUE)) {
    p  <- strsplit(line, "~~", fixed=TRUE)[[1]]
    lv <- trimws(p[1])
    if (grepl("^\\s*1\\s*\\*", trimws(p[2]))) unit_var_lvs <- c(unit_var_lvs, lv)
  } else if (grepl("~", line, fixed=TRUE)) {
    p   <- strsplit(line, "~", fixed=TRUE)[[1]]
    lhs <- trimws(p[1])
    rhs <- trimws(strsplit(trimws(p[2]), "\\+")[[1]])
    rhs <- rhs[nchar(rhs) > 0]
    structural[[lhs]] <- rhs
  }
}

latent_vars       <- names(measurement)
outcome_candidates <- intersect(names(structural), latent_vars)
if (length(outcome_candidates) != 1)
  stop("Expected exactly one latent outcome. Found: ",
       paste(outcome_candidates, collapse=", "))
outcome_lv <- outcome_candidates

treatment_candidates <- setdiff(names(structural), latent_vars)
if (length(treatment_candidates) != 1)
  stop("Expected exactly one treatment variable. Found: ",
       paste(treatment_candidates, collapse=", "))
treatment_var <- treatment_candidates

latent_covariates <- setdiff(latent_vars, outcome_lv)
all_predictors    <- unique(c(structural[[outcome_lv]], structural[[treatment_var]]))
observed_covariates <- setdiff(all_predictors, c(latent_vars, treatment_var))

if (id_method_override == "auto") {
  id_method <- setNames(
    ifelse(latent_vars %in% unit_var_lvs, "unit_variance", "marker_loading"),
    latent_vars)
} else {
  id_method <- setNames(rep(id_method_override, length(latent_vars)), latent_vars)
}

list(latent_vars=latent_vars, latent_covariates=latent_covariates,
     outcome_lv=outcome_lv, treatment_var=treatment_var,
     observed_covariates=observed_covariates,
     measurement=measurement, id_method=id_method)
}

parsed      <- parse_tag_model(model_spec)
lv_covs     <- parsed$latent_covariates
lv_out      <- parsed$outcome_lv
t_var       <- parsed$treatment_var
obs_covs    <- parsed$observed_covariates
```

```
meas       <- parsed$measurement
id_meth    <- parsed$id_method

N          <- nrow(dat)
n_obs_cov  <- length(obs_covs)
n_lv_cov   <- length(lv_covs)
n_items_cov <- setNames(sapply(lv_covs, function(lv) length(meas[[lv]])), lv_covs)
n_items_out <- length(meas[[lv_out]])
obs_all_dim <- sum(n_items_cov) + 1 + n_obs_cov + 1

cat("═══════════════════════════════════════════════════════════════════════════════
\n")
cat("TAG Learning — Parsed Model\n")
cat(sprintf("  Latent covariates : %s\n", paste(lv_covs, collapse=", ")))
cat(sprintf("  Latent outcome    : %s  (%d items)\n", lv_out, n_items_out))
cat(sprintf("  Treatment         : %s\n", t_var))
cat(sprintf("  Observed cov(s)   : %s\n",
          if (n_obs_cov > 0) paste(obs_covs, collapse=", ") else "(none)"))
cat(sprintf("  Measurement mode  : %s\n", toupper(measurement_mode)))
cat(sprintf("  Bootstrap CI      : %s%s\n",
          if (use_bootstrap) "YES" else "NO",
          if (use_bootstrap) sprintf(" (%d reps, %.0f%% CI)", n_bootstrap, ci_level*100) else ""))
cat(sprintf("  N = %d  |  Treatment prevalence = %.3f\n", N, mean(dat[[t_var]])))
cat("═══════════════════════════════════════════════════════════════════════════════
\n\n")

# Column validation
for (.lv in c(lv_covs, lv_out)) {
  .miss <- setdiff(meas[[.lv]], names(dat))
  if (length(.miss) > 0)
    stop(sprintf("Indicator(s) for '%s' not found in dat: %s", .lv,
                 paste(.miss, collapse=", ")))
}
if (n_obs_cov > 0) {
  .miss <- setdiff(obs_covs, names(dat))
  if (length(.miss) > 0)
    stop(sprintf("Observed covariate(s) not found in dat: %s", paste(.miss, collapse=", ")))
}
if (!t_var %in% names(dat))
  stop(sprintf("Treatment variable '%s' not found in dat", t_var))
cat("  Column validation passed.\n\n")

device <- if (cuda_is_available()) torch_device("cuda") else torch_device("cpu")

# ============================================================================
# HELPER FUNCTIONS
# ============================================================================
reparam <- function(mu, logvar) mu + torch_exp(0.5*logvar) * torch_randn_like(mu)

log_normal <- function(x, mu, log_sd)
  torch_sum(-0.5*log(2*pi) - log_sd - 0.5*((x-mu)/torch_exp(log_sd))^2, dim=-1L)

kl_std_normal <- function(mu, logvar)
  0.5 * torch_sum(torch_exp(logvar) + mu^2 - 1 - logvar, dim=-1L)

kl_q_to_scaled_normal <- function(mu_q, logvar_q, log_sigma_p) {
```

```
  var_p <- torch_exp(2*log_sigma_p)
  0.5 * torch_sum((2*log_sigma_p - logvar_q) +
                  (torch_exp(logvar_q) + mu_q^2)/var_p - 1, dim=-1L)
}

run_tmle <- function(Y, A, W, SL.lib, K=10)
  tmle(Y=Y, A=A, W=W, family="gaussian",
       Q.SL.library=SL.lib, g.SL.library=SL.lib,
       cvQinit=TRUE, V.Q=K, V.g=K)

# ============================================================================
# CFA-VAE MODULE
# ============================================================================
CFAVAE_Y <- nn_module("CFAVAE_Y",
  initialize = function(n_items=5, hidden=64, beta_kl=1.0,
                        id_method="unit_variance", measurement_model="linear",
                        dec_hidden=64) {
    self$n_items          <- as.integer(n_items)
    self$beta_kl          <- beta_kl
    self$id_method        <- id_method
    self$measurement_model <- measurement_model
    self$has_sigma_z      <- (id_method == "marker_loading")

    self$enc    <- nn_sequential(nn_linear(self$n_items, hidden), nn_relu(),
                                 nn_linear(hidden, hidden), nn_relu())
    self$enc_mu <- nn_linear(hidden, 1L)
    self$enc_lv <- nn_linear(hidden, 1L)
    self$iy     <- nn_parameter(torch_zeros(self$n_items))
    self$log_sdy <- nn_parameter(torch_zeros(self$n_items))

    if (measurement_model == "linear") {
      if (id_method == "marker_loading") {
        self$ly_free    <- nn_parameter(torch_zeros(self$n_items - 1L))
        self$log_sigma_z <- nn_parameter(torch_zeros(1))
      } else {
        self$ly <- nn_parameter(
          torch_tensor(c(1, rep(0.7, self$n_items - 1L)), dtype=torch_float()))
      }
    } else {
      self$dec <- nn_sequential(nn_linear(1L, dec_hidden), nn_tanh(),
                                nn_linear(dec_hidden, dec_hidden), nn_tanh(),
                                nn_linear(dec_hidden, self$n_items))
      if (id_method == "marker_loading")
        self$log_sigma_z <- nn_parameter(torch_zeros(1))
    }
  },
  forward = function(Y_items) {
    # Ensure Y_items is 2D [batch, n_items] — protects against single-row batches
    if (Y_items$dim() == 1L) Y_items <- Y_items$unsqueeze(1L)$t()
    h  <- self$enc(Y_items)
    mu <- self$enc_mu(h)$reshape(c(-1L, 1L))   # always [batch, 1]
    lv <- self$enc_lv(h)$reshape(c(-1L, 1L))   # always [batch, 1]
    z  <- reparam(mu, lv)
    mu_items <- if (self$measurement_model == "linear") {
      ly <- if (self$id_method == "marker_loading")
        torch_cat(list(torch_ones(1, device=z$device), self$ly_free), dim=1)
```

```
    else self$ly
    self$iy$unsqueeze(1) + z * ly$unsqueeze(1)
   } else { self$iy$unsqueeze(1) + self$dec(z) }
   list(mu_y=mu, lv_y=lv, z=z, mu_items=mu_items, log_sdy=self$log_sdy)
  },
  loss = function(out, Y_items) {
   ll <- log_normal(Y_items, out$mu_items,
                    out$log_sdy$unsqueeze(1)$expand_as(out$mu_items))
   kl <- if (self$has_sigma_z)
     kl_q_to_scaled_normal(out$mu_y, out$lv_y, self$log_sigma_z)
   else kl_std_normal(out$mu_y, out$lv_y)
   elbo <- ll - self$beta_kl * kl
   list(loss=-torch_mean(elbo), elbo=torch_mean(elbo)$item(),
        ll=torch_mean(ll)$item(), kl=torch_mean(kl)$item())
  }
)

# ============================================================================
# CFA-VAE TRAINING WITH EARLY STOPPING
# ============================================================================
fit_cfa_vae <- function(Y_mat, id_method="unit_variance",
                        measurement_model="linear",
                        epochs=500, batchsize=128, hidden=64,
                        beta_kl=1.0, lr=1e-3, val_prop=0.10,
                        patience=10, validate_every=5, verbose=FALSE) {
  Nmod <- nrow(Y_mat); p <- ncol(Y_mat)
  val_i <- sample(Nmod, max(1L, floor(Nmod * val_prop)))
  trn_i <- setdiff(seq_len(Nmod), val_i)
  Y_d   <- torch_tensor(Y_mat, dtype=torch_float())$to(device=device)
  mod   <- CFAVAE_Y(n_items=p, hidden=hidden, beta_kl=beta_kl,
                    id_method=id_method, measurement_model=measurement_model,
                    dec_hidden=hidden)$to(device=device)
  opt   <- optim_adam(mod$parameters, lr=lr)
  best_e <- -Inf; best_st <- NULL; pat <- 0L; prev <- NA_real_

  for (ep in seq_len(epochs)) {
    mod$train(); idx <- sample(trn_i, length(trn_i))
    for (s in seq(1, length(idx), by=batchsize)) {
      b <- idx[s:min(s+batchsize-1L, length(idx))]
      # Force 2D batch even when b has length 1
      Yb <- Y_d[b, , drop=FALSE]
      if (Yb$dim() == 1L) Yb <- Yb$unsqueeze(1L)$t()
      o <- mod(Yb); L <- mod$loss(o, Yb)
      opt$zero_grad(); L$loss$backward(); opt$step()
    }
    if (ep %% validate_every == 0) {
      mod$eval()
      Yv <- Y_d[val_i, , drop=FALSE]
      with_no_grad({ ov <- mod(Yv); Lv <- mod$loss(ov, Yv) })
      ve  <- as.numeric(Lv$elbo)
      if (ve > best_e) { best_e <- ve; best_st <- mod$state_dict() }
      rel <- if (!is.na(prev)) (ve-prev)/max(1e-8,abs(prev)) else NA_real_
      pat <- if (!is.na(rel) && rel < 0.001) pat+1L else 0L; prev <- ve
      if (verbose && ep %% 100 == 0)
        cat(sprintf("    ep %4d | val ELBO %.4f | pat %d/%d\n",ep,ve,pat,patience))
      if (pat >= patience) { if(verbose) cat("    Early stop.\n"); break }
```

```
    }
  }
  if (!is.null(best_st)) mod$load_state_dict(best_st); mod$eval(); mod
}

compute_heldout_elbo <- function(model, Y_mat) {
  Y_t <- torch_tensor(Y_mat, dtype=torch_float())$to(device=device)
  model$eval()
  with_no_grad({ o <- model(Y_t); L <- model$loss(o, Y_t) })
  L$elbo
}

# ============================================================================
# PARAMETER EXTRACTION AND SCORING
# ============================================================================
extract_cfavae_params <- function(model) {
  iy    <- as.numeric(torch::as_array(model$iy$detach()$cpu()))
  theta <- as.numeric(exp(torch::as_array(model$log_sdy$detach()$cpu())))^2
  if (model$measurement_model == "linear") {
    lam <- if (model$id_method == "marker_loading")
      c(1, as.numeric(torch::as_array(model$ly_free$detach()$cpu())))
    else as.numeric(torch::as_array(model$ly$detach()$cpu()))
    psi <- if (model$has_sigma_z)
      as.numeric(exp(2*torch::as_array(model$log_sigma_z$detach()$cpu()))) else 1.0
    list(measurement_model="linear", lambda=lam, intercepts=iy, theta=theta, psi=psi)
  } else {
    psi <- if (model$has_sigma_z)
      as.numeric(exp(2*torch::as_array(model$log_sigma_z$detach()$cpu()))) else 1.0
    list(measurement_model="nonlinear", intercepts=iy, theta=theta, psi=psi)
  }
}

decode_no_intercept <- function(model, z_batch) {
  if (model$measurement_model == "linear") {
    ly <- if (model$id_method == "marker_loading")
      torch_cat(list(torch_ones(1, device=z_batch$device), model$ly_free), dim=1)
    else model$ly
    z_batch * ly$unsqueeze(1)
  } else { model$dec(z_batch) }
}

gauss_newton_step <- function(model, Y_mat, eta_init, ThetaInv, Psi,
                              use_bartlett=TRUE, n_steps=1) {
  pars <- extract_cfavae_params(model)
  Yc   <- sweep(Y_mat, 2, pars$intercepts, "-")
  N_gn <- nrow(Yc); p <- ncol(Yc)
  eta_vals <- as.numeric(torch::as_array(eta_init$detach()$cpu()))

  for (step in seq_len(n_steps)) {
    eta_new <- eta_vals
    for (i in seq_len(N_gn)) {
      zi    <- torch_tensor(matrix(eta_vals[i], 1L, 1L), dtype=torch_float(),
                            requires_grad=TRUE)$to(device=device)
      fi_t  <- decode_no_intercept(model, zi); fi_row <- fi_t[1,]
      J <- numeric(p)
      for (j in seq_len(p)) {
```

```
      zi$grad <- NULL
      fi_row[j]$backward(retain_graph=(j < p))
      J[j] <- if (!is.null(zi$grad))
        as.numeric(torch::as_array(zi$grad$detach()$cpu())) else 0.0
    }
    Ji <- matrix(J, ncol=1L)
    ri <- Yc[i,] - as.numeric(torch::as_array(fi_row$detach()$cpu()))
    G  <- as.numeric(t(Ji) %*% ThetaInv %*% Ji)
    K_row <- if (use_bartlett)
      as.numeric(t(Ji) %*% ThetaInv) / max(G, 1e-8)
    else
      as.numeric(Psi) * as.numeric(t(Ji) %*% ThetaInv) / max(1+Psi*G, 1e-8)
    eta_new[i] <- eta_vals[i] + sum(K_row * ri)
   }
   eta_vals <- eta_new
  }
  eta_vals
}

compute_bartlett_scores <- function(model, Y_mat, n_steps=1) {
  pars <- extract_cfavae_params(model)
  Yc   <- sweep(Y_mat, 2, pars$intercepts, "-")
  TI   <- diag(1/pars$theta, nrow=ncol(Yc))
  if (pars$measurement_model == "linear") {
    lam <- matrix(pars$lambda, ncol=1)
    A   <- solve(t(lam) %*% TI %*% lam) %*% t(lam) %*% TI
    return(as.vector(A %*% t(Yc)))
  }
  cat("  Computing nonlinear Bartlett scores...\n")
  model$eval()
  with_no_grad({
    eta0 <- model$enc_mu(model$enc(
      torch_tensor(Y_mat, dtype=torch_float())$to(device=device)))$detach()
  })
  gauss_newton_step(model, Y_mat, eta0, TI, pars$psi, use_bartlett=TRUE, n_steps=n_steps)
}

# ============================================================================
# MEASUREMENT MODEL SELECTION
# ============================================================================
select_measurement_model <- function(Y_mat, block_name, item_names,
                                     id_method="unit_variance",
                                     cfi_thr=0.95, rmsea_thr=0.06,
                                     alpha_nl=0.05, elbo_str=0.5, elbo_wk=0.1,
                                     val_comp=0.20, val_es=0.10,
                                     epochs=500, hidden=64, beta_kl=1.0,
                                     patience=10, validate_every=5) {
  Nmod <- nrow(Y_mat); p <- ncol(Y_mat)
  cat(sprintf("\n=== Selection: %s (%d items, N=%d) ===\n", block_name, p, Nmod))

  df_tmp  <- as.data.frame(Y_mat); colnames(df_tmp) <- item_names
  cfa_str <- paste0("lat =~ ", paste(item_names, collapse=" + "))
  fit_cfa <- tryCatch(cfa(cfa_str, data=df_tmp, std.lv=TRUE), error=function(e) NULL)
  cfi <- NA_real_; rmsea <- NA_real_; lin_ok <- TRUE; n_sig <- 0L

  if (!is.null(fit_cfa)) {
```

```
  fm   <- fitMeasures(fit_cfa, c("cfi","rmsea"))
  cfi  <- as.numeric(fm["cfi"]); rmsea <- as.numeric(fm["rmsea"])
  lin_ok <- cfi > cfi_thr && rmsea < rmsea_thr
  cat(sprintf("  Step 1 | CFI=%.3f  RMSEA=%.3f  -> linear %s\n",
              cfi, rmsea, if (lin_ok) "ADEQUATE" else "POOR"))
  eta_hat <- as.numeric(lavPredict(fit_cfa, method="Bartlett")[,1])
  pvals <- sapply(seq_len(p), function(j) {
    yj <- Y_mat[,j]
    tryCatch(anova(lm(yj ~ eta_hat),
                   lm(yj ~ eta_hat + I(eta_hat^2) + I(eta_hat^3)))$`Pr(>F)`[2],
             error=function(e) NA_real_)
  })
  n_sig <- sum(pvals < alpha_nl, na.rm=TRUE)
  cat(sprintf("  Step 2 | %d/%d items: significant nonlinearity\n", n_sig, p))
 } else { cat("  Step 1/2 | lavaan CFA failed.\n") }

 set.seed(42)
 val_i <- sample(Nmod, max(5L, floor(Nmod * val_comp)))
 trn_i <- setdiff(seq_len(Nmod), val_i)
 Y_trn <- Y_mat[trn_i,,drop=FALSE]; Y_val <- Y_mat[val_i,,drop=FALSE]
 cat(sprintf("  Step 3 | train n=%d  held-out n=%d\n", nrow(Y_trn), nrow(Y_val)))

 m_lin <- fit_cfa_vae(Y_trn, id_method=id_method, measurement_model="linear",
                      epochs=epochs, hidden=hidden, beta_kl=beta_kl,
                      val_prop=val_es, patience=patience, validate_every=validate_every)
 e_lin <- compute_heldout_elbo(m_lin, Y_val)

 m_nl  <- fit_cfa_vae(Y_trn, id_method=id_method, measurement_model="nonlinear",
                      epochs=epochs, hidden=hidden, beta_kl=beta_kl,
                      val_prop=val_es, patience=patience, validate_every=validate_every)
 e_nl  <- compute_heldout_elbo(m_nl, Y_val)

 diff  <- e_nl - e_lin
 cat(sprintf("  Step 3 | ELBO: lin=%.4f  nl=%.4f  diff=%.4f nats\n", e_lin, e_nl, diff))

 selected <- if (diff > elbo_str) {
   cat("  Decision: NONLINEAR (ELBO diff > strong threshold)\n"); "nonlinear"
 } else if (diff > elbo_wk && (!lin_ok || n_sig >= 2L)) {
   cat("  Decision: NONLINEAR (moderate ELBO + corroborating evidence)\n"); "nonlinear"
 } else {
   cat("  Decision: LINEAR\n"); "linear"
 }

 rm(m_lin, m_nl); if (cuda_is_available()) gc()
 list(selected=selected, block=block_name, cfi=cfi, rmsea=rmsea,
      lin_ok=lin_ok, n_sig_nl=n_sig,
      elbo_lin=e_lin, elbo_nl=e_nl, elbo_diff=diff)
}

# ============================================================================
# SEM-VAE
# ============================================================================
SEMVAE_general <- nn_module("SEMVAE_general",
  initialize = function(cov_names, n_items_cov, meas_cov,
                        n_obs_cov, obs_all_dim,
                        context_dim_out=4, hidden=64, beta_kl=1.0,
```

```
                  init_loadings=NULL) {
  self$cov_names  <- cov_names
  self$n_lv       <- length(cov_names)
  self$n_obs_cov  <- as.integer(n_obs_cov)
  self$beta_kl    <- beta_kl
  self$meas_cov   <- meas_cov

  self$context_net <- nn_sequential(
    nn_linear(obs_all_dim, hidden), nn_relu(),
    nn_linear(hidden, context_dim_out), nn_relu())

  for (lv in cov_names) {
    ni <- as.integer(n_items_cov[[lv]])
    self[[paste0("enc_",lv)]] <- nn_sequential(
      nn_linear(ni + context_dim_out, hidden), nn_relu(),
      nn_linear(hidden, hidden), nn_relu())
    self[[paste0("enc_",lv,"_mu")]] <- nn_linear(hidden, 1L)
    self[[paste0("enc_",lv,"_lv")]] <- nn_linear(hidden, 1L)
    self[[paste0("i_",lv)]]     <- nn_parameter(torch_zeros(ni))
    self[[paste0("logsd_",lv)]] <- nn_parameter(torch_zeros(ni))
    if (meas_cov[[lv]] == "linear") {
      init_l <- if (!is.null(init_loadings) && !is.null(init_loadings[[lv]]))
        init_loadings[[lv]] else rep(0.7, ni)
      self[[paste0("l_",lv)]] <- nn_parameter(
        torch_tensor(init_l, dtype=torch_float()))
    } else {
      self[[paste0("dec_",lv)]] <- nn_sequential(
        nn_linear(1L, hidden), nn_tanh(),
        nn_linear(hidden, hidden), nn_tanh(),
        nn_linear(hidden, ni))
    }
  }

  struct_dim <- self$n_lv + self$n_obs_cov
  self$net_t       <- nn_sequential(nn_linear(struct_dim, hidden), nn_relu(),
                                    nn_linear(hidden, hidden), nn_relu(),
                                    nn_linear(hidden, 1L))
  self$net_y_mu    <- nn_sequential(nn_linear(struct_dim+1L, hidden), nn_relu(),
                                    nn_linear(hidden, hidden), nn_relu(),
                                    nn_linear(hidden, 1L))
  self$net_y_logsd <- nn_sequential(nn_linear(struct_dim+1L, hidden), nn_relu(),
                                    nn_linear(hidden, 1L))
},
forward = function(items_list, W_mat, T_t, OBS_all) {
  ctx <- self$context_net(OBS_all)
  z_tensors <- list(); mu_list <- list(); lv_list <- list()
  mu_items_list <- list(); logsd_list <- list()

  for (lv in self$cov_names) {
    h   <- self[[paste0("enc_",lv)]](torch_cat(list(items_list[[lv]], ctx), dim=-1L))
    mu  <- self[[paste0("enc_",lv,"_mu")]](h)
    lvt <- self[[paste0("enc_",lv,"_lv")]](h)
    z   <- reparam(mu, lvt)
    mu_list[[lv]] <- mu; lv_list[[lv]] <- lvt; z_tensors[[lv]] <- z
    iy  <- self[[paste0("i_",lv)]]$unsqueeze(1)
    mu_items_list[[lv]] <- if (self$meas_cov[[lv]] == "linear")
```

```
      iy + z * self[[paste0("l_",lv)]]$unsqueeze(1)
    else iy + self[[paste0("dec_",lv)]](z)
    logsd_list[[lv]] <- self[[paste0("logsd_",lv)]]
  }

  z_stack   <- torch_cat(lapply(z_tensors, identity), dim=-1L)
  struct_in <- if (self$n_obs_cov > 0)
    torch_cat(list(z_stack, W_mat), dim=-1L) else z_stack

  t_logits <- self$net_t(struct_in)
  y_in     <- torch_cat(list(struct_in, T_t), dim=-1L)
  mu_y     <- self$net_y_mu(y_in)
  logsd_y  <- torch_clamp(self$net_y_logsd(y_in), -5, 5)

  list(mu_list=mu_list, lv_list=lv_list, z_list=z_tensors,
       mu_items_list=mu_items_list, logsd_list=logsd_list,
       t_logits=t_logits, mu_y=mu_y, logsd_y=logsd_y)
 },
 loss = function(out, items_list, Y_obs, W_mat, T_t) {
  ll_cov <- NULL; kl_cov <- NULL
  for (lv in self$cov_names) {
    ls    <- out$logsd_list[[lv]]$unsqueeze(1)$expand_as(out$mu_items_list[[lv]])
    ll_lv <- log_normal(items_list[[lv]], out$mu_items_list[[lv]], ls)
    kl_lv <- kl_std_normal(out$mu_list[[lv]], out$lv_list[[lv]])
    ll_cov <- if (is.null(ll_cov)) ll_lv else ll_cov + ll_lv
    kl_cov <- if (is.null(kl_cov)) kl_lv else kl_cov + kl_lv
  }
  ll_y <- log_normal(Y_obs, out$mu_y, out$logsd_y)
  ll_t <- -torch_sum(nnf_binary_cross_entropy_with_logits(
    out$t_logits, T_t, reduction="none"), dim=-1L)
  elbo <- (ll_cov + ll_y + ll_t) - self$beta_kl * kl_cov
  list(loss=-torch_mean(elbo), elbo=torch_mean(elbo)$item())
 }
)

# ============================================================================
# TAG with K-fold SEM-VAE cross-fitting + TMLE so it can be called
#
# Arguments:
#   dat_sub       : data.frame (full data OR one bootstrap resample)
#   yb_sub        : numeric vector of Bartlett scores for dat_sub
#   meas_cov_sel  : named list/vector: lv -> "linear"|"nonlinear"
#   init_loadings : named list: lv -> numeric starting loadings
#   fold_seed     : integer seed for the stratified K-fold partition
#   verbose       : logical
#
# Returns a named list with ATE, SE, and the out-of-sample posterior means.
# ============================================================================
run_tag_pipeline <- function(dat_sub, yb_sub, meas_cov_sel,
                             init_loadings, fold_seed, verbose=TRUE) {
 N_sub <- nrow(dat_sub)

 # Stratified K-fold partition
 set.seed(fold_seed)
 fold_ids <- integer(N_sub)
 for (tv in c(0L, 1L)) {
```

```
  idx_t <- which(dat_sub[[t_var]] == tv)
  if (length(idx_t) == 0) next
  fold_ids[idx_t] <- sample(rep(seq_len(K), length.out=length(idx_t)),
                            length(idx_t), replace=FALSE)
}

mu_oos <- setNames(lapply(lv_covs, function(lv) numeric(N_sub)), lv_covs)

# Build torch tensors for this subsample
items_full <- lapply(lv_covs, function(lv)
  torch_tensor(as.matrix(dat_sub[, meas[[lv]]]),
               dtype=torch_float())$to(device=device))
names(items_full) <- lv_covs

W_full <- if (n_obs_cov > 0)
  torch_tensor(as.matrix(dat_sub[, obs_covs, drop=FALSE]),
               dtype=torch_float())$to(device=device)
else
  torch_tensor(matrix(0, N_sub, 1), dtype=torch_float())$to(device=device)

T_full  <- torch_tensor(matrix(dat_sub[[t_var]], ncol=1),
                        dtype=torch_float())$to(device=device)
Y_full  <- torch_tensor(matrix(yb_sub, ncol=1),
                        dtype=torch_float())$to(device=device)

obs_tlist <- c(lapply(items_full, identity), list(Y_full))
if (n_obs_cov > 0) obs_tlist <- c(obs_tlist, list(W_full))
obs_tlist <- c(obs_tlist, list(T_full))
OBS_full  <- torch_cat(obs_tlist, dim=-1L)

for (k in seq_len(K)) {
  if (verbose) cat(sprintf("  Fold %d / %d\n", k, K))
  test_idx  <- which(fold_ids == k)
  train_idx <- which(fold_ids != k)

  val_inner <- integer(0)
  for (tv in c(0L, 1L)) {
    idx_t_tr  <- train_idx[dat_sub[[t_var]][train_idx] == tv]
    if (length(idx_t_tr) == 0) next
    val_inner <- c(val_inner,
                   sample(idx_t_tr,
                          max(1L, floor(length(idx_t_tr)*val_prop_inner)),
                          replace=FALSE))
  }
  train_inner <- setdiff(train_idx, val_inner)

  model_k <- SEMVAE_general(
    cov_names       = lv_covs,
    n_items_cov     = as.list(n_items_cov),
    meas_cov        = as.list(meas_cov_sel),
    n_obs_cov       = if (n_obs_cov > 0) n_obs_cov else 0L,
    obs_all_dim     = obs_all_dim,
    context_dim_out = context_dim_out,
    hidden          = N_neurons,
    beta_kl         = betakl,
    init_loadings   = init_loadings
```

```
  )$to(device=device)

  opt_k    <- optim_adam(model_k$parameters, lr=1e-3)
  best_val <- -Inf; best_st <- NULL; pat_c <- 0L; prev_v <- NA_real_

  for (ep in seq_len(nepochs)) {
    model_k$train()
    shuf <- sample(train_inner, length(train_inner))
    for (s in seq(1, length(shuf), by=batchsize)) {
      b  <- shuf[s:min(s+batchsize-1L, length(shuf))]
      ib <- lapply(items_full, function(X) X[b,])
      ob <- model_k(ib, W_full[b,], T_full[b,], OBS_full[b,])
      Lb <- model_k$loss(ob, ib, Y_full[b,], W_full[b,], T_full[b,])
      opt_k$zero_grad(); Lb$loss$backward(); opt_k$step()
    }
    if (ep %% validate_every == 0) {
      model_k$eval()
      with_no_grad({
        iv <- lapply(items_full, function(X) X[val_inner,])
        ov <- model_k(iv, W_full[val_inner,], T_full[val_inner,], OBS_full[val_inner,])
        Lv <- model_k$loss(ov, iv, Y_full[val_inner,], W_full[val_inner,], T_full[val_inner,])
      })
      ve    <- as.numeric(Lv$elbo)
      if (ve > best_val) { best_val <- ve; best_st <- model_k$state_dict() }
      rel   <- if (!is.na(prev_v)) (ve-prev_v)/max(1e-8,abs(prev_v)) else NA_real_
      pat_c <- if (!is.na(rel) && rel < min_rel_improve) pat_c+1L else 0L
      prev_v <- ve
      if (verbose && ep %% 50 == 0)
        cat(sprintf("    ep %4d | val ELBO %.4f | pat %d\n", ep, ve, pat_c))
      if (pat_c >= patience) {
        if (verbose) cat(sprintf("    Early stop ep %d\n", ep)); break }
    }
  }

  if (!is.null(best_st)) model_k$load_state_dict(best_st); model_k$eval()

  with_no_grad({
    it <- lapply(items_full, function(X) X[test_idx,])
    ot <- model_k(it, W_full[test_idx,], T_full[test_idx,], OBS_full[test_idx,])
  })
  for (lv in lv_covs)
    mu_oos[[lv]][test_idx] <-
      as.numeric(torch::as_array(ot$z_list[[lv]]$detach()$cpu()$squeeze()))

  rm(model_k, opt_k, best_st); if (cuda_is_available()) gc()
}

W_oos <- as.data.frame(
  c(setNames(mu_oos, paste0("mu_", lv_covs)),
    if (n_obs_cov > 0) as.list(dat_sub[, obs_covs, drop=FALSE]) else list()))

fit_r <- tryCatch(
  run_tmle(Y=yb_sub, A=dat_sub[[t_var]], W=W_oos, SL.lib=SL.lib, K=K),
  error=function(e) { warning(e$message); NULL })

if (is.null(fit_r)) return(list(ate=NA_real_, se=NA_real_, mu_oos=mu_oos))
```

```
  list(ate = fit_r$estimates$ATE$psi,
       se  = sqrt(fit_r$estimates$ATE$var.psi),
       mu_oos = mu_oos)
}

# ============================================================================
# PHASE 0 — Measurement Model Selection
# Skipped when measurement_mode is "linear" or "nonlinear"
# ============================================================================

if (measurement_mode == "auto") {

cat("\n╔═══════════════════════════════════════════════════════════════════════════
═══╗\n")
  cat("║  PHASE 0: Measurement Model Selection (Appendix D)           ║\n")

cat("╚═══════════════════════════════════════════════════════════════════════════
═══╝\n")

  sel_results <- list()
  for (lv in lv_covs) {
    sel_results[[lv]] <- select_measurement_model(
      Y_mat      = as.matrix(dat[, meas[[lv]]]),
      block_name = sprintf("%s (covariate)", lv),
      item_names = meas[[lv]],
      id_method  = id_meth[[lv]],
      cfi_thr=cfi_threshold, rmsea_thr=rmsea_threshold,
      alpha_nl=alpha_nonlin, elbo_str=elbo_strong, elbo_wk=elbo_weak,
      val_comp=val_prop_selection, val_es=val_prop_es,
      epochs=epochs_cfavae, hidden=hidden_cfavae, beta_kl=beta_kl_cfavae,
      patience=patience_cfavae, validate_every=validate_every_cfavae)
  }
  sel_results[[lv_out]] <- select_measurement_model(
    Y_mat      = as.matrix(dat[, meas[[lv_out]]]),
    block_name = sprintf("%s (outcome)", lv_out),
    item_names = meas[[lv_out]],
    id_method  = id_meth[[lv_out]],
    cfi_thr=cfi_threshold, rmsea_thr=rmsea_threshold,
    alpha_nl=alpha_nonlin, elbo_str=elbo_strong, elbo_wk=elbo_weak,
    val_comp=val_prop_selection, val_es=val_prop_es,
    epochs=epochs_cfavae, hidden=hidden_cfavae, beta_kl=beta_kl_cfavae,
    patience=patience_cfavae, validate_every=validate_every_cfavae)

  meas_cov_sel <- setNames(sapply(lv_covs, function(lv) sel_results[[lv]]$selected), lv_covs)
  meas_out_sel <- sel_results[[lv_out]]$selected

  cat("\n——— Selection Summary ———————————————————————————————————————\n")
  for (lv in c(lv_covs, lv_out)) {
    s <- sel_results[[lv]]
    cat(sprintf("  %-22s : %-10s  (CFI=%.3f RMSEA=%.3f NL-items=%d ELBO-diff=%.4f)\n",
                s$block, toupper(s$selected),
                ifelse(is.na(s$cfi),0,s$cfi), ifelse(is.na(s$rmsea),0,s$rmsea),
                s$n_sig_nl, s$elbo_diff))
  }
```

```
cat("──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
\n")

} else {
  # measurement_mode is "linear" or "nonlinear" — set all blocks uniformly
  meas_cov_sel <- setNames(rep(measurement_mode, length(lv_covs)), lv_covs)
  meas_out_sel <- measurement_mode
  sel_results  <- NULL
  cat(sprintf("\nMeasurement mode fixed by user: %s (Phase 0 skipped)\n",
              toupper(measurement_mode)))
}

# ============================================================================
# HELPER: prepare_stage1
# Produces Bartlett scores for the latent outcome and lavaan starting loadings
# for the SEM-VAE covariate encoders. Called once for the main estimate and
# once per bootstrap replicate so every pipeline stage uses parameters
# estimated from the same (possibly resampled) dataset.
#
#
# Arguments:
#   dat_sub       : data.frame (full data or bootstrap resample)
#   meas_out_sel   : "linear" or "nonlinear"
#   epoch_scale    : fraction of epochs_cfavae to use for CFA-VAE (nonlinear only)
#   verbose       : logical
#
# Returns named list: yb (Bartlett scores, length N), init_loadings (named list)
# ============================================================================
prepare_stage1 <- function(dat_sub, meas_out_sel,
                           epoch_scale = 1.0, verbose = FALSE) {

  # ── Resolve global model metadata inside the function ──────────────────────
  .lv_out      <- get("lv_out",      envir = parent.env(environment()))
  .lv_covs     <- get("lv_covs",     envir = parent.env(environment()))
  .meas        <- get("meas",        envir = parent.env(environment()))
  .id_meth     <- get("id_meth",     envir = parent.env(environment()))

  # Fallback: if parent.env lookup fails (e.g. called from a different scope),
  # try the global environment directly.
  if (!exists(".lv_out") || is.null(.lv_out)) {
    .lv_out  <- get("lv_out",  envir = globalenv())
    .lv_covs <- get("lv_covs", envir = globalenv())
    .meas    <- get("meas",    envir = globalenv())
    .id_meth <- get("id_meth", envir = globalenv())
  }

  out_items <- .meas[[.lv_out]]

  # ── column check ──────────────────────────────────────────────────────────
  missing_cols <- setdiff(out_items, names(dat_sub))
  if (length(missing_cols) > 0)
    stop(sprintf(
      paste0("prepare_stage1: outcome indicator column(s) not found in the data.\n",
             "  Expected columns : %s\n",
             "  Missing          : %s\n",
```

```
        "  Available columns (first 30): %s\n",
        "  Check that the indicator names in model_spec match the column names in dat."),
    paste(out_items, collapse=", "),
    paste(missing_cols, collapse=", "),
    paste(head(names(dat_sub), 30), collapse=", ")))

Y_mat_out <- as.matrix(dat_sub[, out_items, drop=FALSE])

# ── Outcome Bartlett scores ──────────────────────────────────────────────────
if (meas_out_sel == "linear") {

  # Linear measurement: use lavaan CFA + classical Bartlett scores.
  if (verbose)
    cat(sprintf("  Fitting lavaan CFA for outcome %s [linear — using lavaan]\n",
                .lv_out))

  # Build the lavaan model string matching the user's identification choice.
  # We always use unit-variance ID (std.lv=TRUE) for simplicity; the
  # resulting Bartlett scores are on a standardised scale.
  cfa_str <- paste0(
    .lv_out, " =~ NA*", out_items[1],
    if (length(out_items) > 1)
      paste0(" + ", paste(out_items[-1], collapse=" + "))
    else "",
    "\n",
    .lv_out, " ~~ 1*", .lv_out)

  df_out          <- as.data.frame(Y_mat_out)
  colnames(df_out) <- out_items

  fit_out <- tryCatch(
    cfa(cfa_str, data=df_out, std.lv=TRUE),
    error = function(e) {
      warning(sprintf("lavaan CFA for %s failed: %s — falling back to standardised first indicator.",
                      .lv_out, e$message))
      NULL
    })

  yb <- if (!is.null(fit_out)) {
    as.numeric(lavPredict(fit_out, method="Bartlett")[, 1])
  } else {
    warning(sprintf("Outcome CFA failed; using standardised %s as proxy.", out_items[1]))
    as.numeric(scale(Y_mat_out[, 1]))
  }

} else {

  # Nonlinear measurement: CFA-VAE with tanh decoder.
  ep_y <- max(50L, floor(epochs_cfavae * epoch_scale))
  if (verbose)
    cat(sprintf("  Fitting CFA-VAE for outcome %s [nonlinear, epochs=%d]\n",
                .lv_out, ep_y))

  model_y <- fit_cfa_vae(
    Y_mat            = Y_mat_out,
    id_method        = .id_meth[[.lv_out]],
```

```
    measurement_model = "nonlinear",
    epochs           = ep_y,
    batchsize        = batchsize,
    hidden           = hidden_cfavae,
    beta_kl          = beta_kl_cfavae,
    val_prop         = val_prop_es,
    patience         = patience_cfavae,
    validate_every   = validate_every_cfavae,
    verbose          = verbose)

  yb <- compute_bartlett_scores(model_y, Y_mat_out,
                                n_steps = n_steps_nonlinear)

  rm(model_y); if (cuda_is_available()) gc()
 }

 if (verbose)
  cat(sprintf("  Bartlett scores: mean=%.3f  sd=%.3f\n", mean(yb), sd(yb)))

 # ── Lavaan starting loadings for SEM-VAE covariate encoders ──────────────

 init_l <- lapply(.lv_covs, function(lv) {
  lv_items <- .meas[[lv]]
  miss_lv  <- setdiff(lv_items, names(dat_sub))
  if (length(miss_lv) > 0) {
   warning(sprintf("Covariate %s: indicator(s) not found — using default loadings 0.7", lv))
   return(rep(0.7, length(lv_items)))
  }
  f <- tryCatch(
   cfa(paste0("L=~", paste(lv_items, collapse="+")),
       data    = dat_sub[, lv_items, drop=FALSE],
       std.lv  = TRUE),
   error = function(e) NULL)
  if (is.null(f)) return(rep(0.7, length(lv_items)))
  pe <- parameterEstimates(f)
  co <- pe$est[pe$op == "=~" & pe$lhs == "L"]
  co[is.na(co)] <- 0.7; co
 })
 names(init_l) <- .lv_covs

 list(yb = yb, init_loadings = init_l)
}

# ============================================================================
# PHASE 1 — TAG Learning Pipeline (main estimate)
# ============================================================================
cat("\n╔═══════════════════════════════════════════════════════════════════════╗\n")
cat("║  PHASE 1: TAG Learning Pipeline (main estimate)                       ║\n")
cat("╚═══════════════════════════════════════════════════════════════════════╝\n")

# Step 1: fit outcome CFA-VAE + Bartlett scores + covariate init loadings
cat(sprintf("\nStep 1: Outcome measurement for %s [%s]\n", lv_out, meas_out_sel))
stage1_main   <- prepare_stage1(dat, meas_out_sel, epoch_scale=1.0, verbose=TRUE)
yb_std        <- stage1_main$yb
```

```
init_loadings <- stage1_main$init_loadings

# K-fold cross-fitting repetitions (main estimate)
cat(sprintf("\nStep 2: K=%d fold SEM-VAE cross-fitting (%d rep(s))\n", K, n_reps))
cat(sprintf("  Covariates: %s\n",
          paste(paste0(lv_covs,"[",meas_cov_sel,"]"), collapse=", ")))

tag_ate_reps <- numeric(n_reps); tag_se_reps <- numeric(n_reps)

for (irep in seq_len(n_reps)) {
  cat(sprintf("\n--- Main Rep %d / %d ---\n", irep, n_reps))
  res <- run_tag_pipeline(dat, yb_std, meas_cov_sel,
                          init_loadings, fold_seed=irep*1000, verbose=TRUE)
  tag_ate_reps[irep] <- res$ate; tag_se_reps[irep] <- res$se
  cat(sprintf("  TAG ATE=%.4f  (EIF SE=%.4f)\n", res$ate, res$se))
}

# TMLE-FS comparator
cat("\nTMLE-FS comparator (conventional factor scores)...\n")
W_fs <- as.data.frame(
  c(setNames(lapply(lv_covs, function(lv) {
    f <- cfa(paste0("L=~", paste(meas[[lv]], collapse="+")),
             data=dat[, meas[[lv]], drop=FALSE], std.lv=TRUE)
    as.numeric(lavPredict(f, method="regression")[,1])
  }), paste0("r_", lv_covs)),
  if (n_obs_cov > 0) as.list(dat[, obs_covs, drop=FALSE]) else list()))
tmle_fs <- run_tmle(Y=yb_std, A=dat[[t_var]], W=W_fs, SL.lib=SL.lib, K=K)

# ============================================================================
# PHASE 2 — Bootstrap Confidence Intervals (optional)
# ============================================================================
boot_ates <- NULL
boot_ci   <- NULL

if (use_bootstrap) {

cat("\n╔═══════════════════════════════════════════════════════════════════════╗\n")
  cat("║  PHASE 2: Bootstrap Confidence Intervals                              ║\n")

cat("╚═══════════════════════════════════════════════════════════════════════╝\n")
  ep_scale <- boot_epoch_scale
  cat(sprintf("  Replicates: %d  |  CI level: %.0f%%  |  Epoch scale: %.2f\n",
              n_bootstrap, ci_level*100, ep_scale))
  cat("  All measurement models (outcome CFA-VAE + covariate SEM-VAE) are\n")
  cat("  re-estimated from scratch on each bootstrap resample.\n")
  cat("  This is a fully principled bootstrap that captures uncertainty from:\n")
  cat("    (a) Outcome measurement model parameter estimation\n")
  cat("    (b) Latent covariate posterior estimation\n")
  cat("    (c) TMLE nuisance function estimation and targeting\n\n")

  set.seed(bootstrap_seed)
  boot_ates <- numeric(n_bootstrap)

  for (b in seq_len(n_bootstrap)) {
```

```
  cat(sprintf("\n——— Bootstrap replicate %d / %d ———\n", b, n_bootstrap))

  # ── Case-based resample with replacement, stratified by treatment ──────────
  boot_rows <- unlist(lapply(c(0L, 1L), function(tv) {
    idx_tv <- which(dat[[t_var]] == tv)
    sample(idx_tv, length(idx_tv), replace = TRUE)
  }))
  dat_b <- dat[boot_rows, , drop = FALSE]
  rownames(dat_b) <- NULL

  # ── Re-estimate outcome measurement + Bartlett scores + init loadings ──────
  stage1_b <- tryCatch(
    prepare_stage1(dat_b, meas_out_sel,
                   epoch_scale = ep_scale, verbose = FALSE),
    error = function(e) {
      warning(sprintf("Boot %d Stage-1 error: %s", b, e$message)); NULL })

  if (is.null(stage1_b)) {
    boot_ates[b] <- NA_real_
    cat(sprintf("  Boot %d SKIPPED (Stage-1 failed)\n", b))
    next
  }

  yb_b       <- stage1_b$yb
  init_l_b   <- stage1_b$init_loadings

  # ── Re-estimate SEM-VAE + TMLE on bootstrap sample ─────────────────────────
  res_b <- tryCatch(
    run_tag_pipeline(dat_b, yb_b, meas_cov_sel, init_l_b,
                     fold_seed = bootstrap_seed + b, verbose = FALSE),
    error = function(e) {
      warning(sprintf("Boot %d Pipeline error: %s", b, e$message)); NULL })

  boot_ates[b] <- if (!is.null(res_b)) res_b$ate else NA_real_
  cat(sprintf("  Boot %d ATE = %.4f\n", b, boot_ates[b]))
 }

 alpha_ci <- 1 - ci_level
 boot_ci  <- quantile(boot_ates, c(alpha_ci/2, 1-alpha_ci/2), na.rm=TRUE)
 cat(sprintf("\n  Bootstrap SE   = %.4f  (%d valid reps)\n",
             sd(boot_ates, na.rm=TRUE), sum(!is.na(boot_ates))))
 cat(sprintf("  %.0f%% Percentile CI = [%.4f, %.4f]\n",
             ci_level*100, boot_ci[1], boot_ci[2]))
}

# ============================================================================
# RESULTS SUMMARY
# ============================================================================
cat("\n═══════════════════════════════════════════════════════════════════════
═══\n")
cat("RESULTS SUMMARY\n")
cat("─────────────────────────────────────────────────────────────────────────
\n")

if (measurement_mode == "auto" && !is.null(sel_results)) {
  cat("Phase 0 — Measurement Model Selection\n")
```

```
  for (lv in c(lv_covs, lv_out)) {
    s <- sel_results[[lv]]
    cat(sprintf("  %-22s : %s\n", s$block, toupper(s$selected)))
  }
  cat("\n")
} else {
  cat(sprintf("Measurement mode: %s (fixed, Phase 0 skipped)\n", toupper(measurement_mode)))
cat(sprintf("  Outcome (%s): %s\n",
            lv_out,
            if (meas_out_sel=="linear") "lavaan CFA + Bartlett scores"
            else "nonlinear CFA-VAE + GN Bartlett scores"))
cat("\n")
}

cat("Phase 1 — Main TAG Estimate\n")
cat(sprintf("  Mean ATE (EIF)     : %.4f\n",  mean(tag_ate_reps, na.rm=TRUE)))
if (n_reps > 1)
  cat(sprintf("  SD  ATE (reps)     : %.4f  (%d reps)\n",
              sd(tag_ate_reps, na.rm=TRUE), n_reps))
cat(sprintf("  Mean EIF SE        : %.4f\n",  mean(tag_se_reps, na.rm=TRUE)))

if (use_bootstrap && !is.null(boot_ci)) {
  cat("\nPhase 2 — Bootstrap CI\n")
  cat("  (all measurement models re-estimated per replicate)\n")
  cat(sprintf("  Valid reps         : %d / %d\n",
              sum(!is.na(boot_ates)), n_bootstrap))
  cat(sprintf("  Bootstrap SE       : %.4f\n",
              sd(boot_ates, na.rm=TRUE)))
  cat(sprintf("  %.0f%% Percentile CI  : [%.4f, %.4f]\n",
              ci_level*100, boot_ci[1], boot_ci[2]))
}

cat("\nTMLE-FS Comparator (conventional factor scores)\n")
cat(sprintf("  ATE                : %.4f  (EIF SE=%.4f)\n",
            tmle_fs$estimates$ATE$psi,
            sqrt(tmle_fs$estimates$ATE$var.psi)))
cat("═══════════════════════════════════════════════════════════════════════════════
\n")


#compare to LSEM

model_spec <- "
              G=~NA*GS1+GS2+GS3
              G~~1*G
              A=~NA*A1+A2+A3
              A~~1*A
              C=~NA*C1+C2+C3
              C~~1*C
              G~college+ A+C
              college~  A+C
"
summary(sem(model_spec,dat))

#compare to LR-FS
summary(lm(yb_std~dat$college+W_fs$r_A+W_fs$r_C))
```

```
mean(tag_ate_reps)
mean(tag_se_reps)
sd(boot_ates, na.rm=TRUE)
quantile(boot_ates, c(.025, .975), na.rm=TRUE)
```