

**Xintong Li**  
**xintong.li1@uwaterloo.ca**

**Dept. Statistics and Actuarial Science**  
**University of Waterloo**



# **Resilient Machine Learning Approaches for Fast Risk Evaluation and Management of Financial Portfolios and Variable Annuities**

Supervised by Prof. Tony Wirjanto and Prof. Mingbin Feng

Thesis Defense,  
University of Waterloo

Resilient ML for Fast Risk Evaluation of VAs

2025-03-21

Xintong Li  
xintong.li1@uwaterloo.ca

Dept. Statistics and Actuarial Science  
University of Waterloo



Resilient Machine Learning Approaches  
for Fast Risk Evaluation and Management  
of Financial Portfolios and Variable Annuities

Supervised by Prof. Tony Wirjanto and Prof. Mingbin Feng

Thesis Defense,  
University of Waterloo

2025-03-21

## L Outline

- 1 Introduction
- 2 Nested Simulation Procedures in Financial Engineering: A Selected Review
  - Theoretical Results
  - Finite-Sample Analysis
- 3 Cutting Through the Noise: Using Deep Neural Network Metamodels for High-Dimensional Nested Simulation
- 4 Transfer Learning for Rapid Adaptation of DNN Metamodels

## Outline

- 1 Introduction
- 2 Nested Simulation Procedures in Financial Engineering: A Selected Review
  - Theoretical Results
  - Finite-Sample Analysis
- 3 Cutting Through the Noise: Using Deep Neural Network Metamodels for High-Dimensional Nested Simulation
- 4 Transfer Learning for Rapid Adaptation of DNN Metamodels

# Nested Simulation Procedures

Nested simulation procedures are necessary for **complex** financial derivatives and insurance products.

$$\rho(L) = \rho(L(X)), \quad L(X) = \mathbb{E}[Y|X = x]_{x=X}.$$

Involves two levels of Monte Carlo simulations:

- ❖ Outer: underlying risk factors (outer scenarios),  $X_i \sim F_X$
- ❖ Inner: scenario-wise losses (inner replications),  $Y_{ij} \sim F_{Y|X_i}$
- ❖ Total simulation budget:  $\Gamma = M \cdot N$

**Computationally expensive**

## Resilient ML for Fast Risk Evaluation of VAs

### Introduction

### Nested Simulation Procedures

2025-03-21

#### Nested Simulation Procedures

Nested simulation procedures are necessary for **complex** financial derivatives and insurance products.

$$\rho(L) = \rho(L(X)), \quad L(X) = \mathbb{E}[Y|X = x]_{x=X}.$$

Involves two levels of Monte Carlo simulations:

- ❖ Outer: underlying risk factors (outer scenarios),  $X_i \sim F_X$
- ❖ Inner: scenario-wise losses (inner replications),  $Y_{ij} \sim F_{Y|X_i}$
- ❖ Total simulation budget:  $\Gamma = M \cdot N$

Computationally expensive

In the context of quantitative risk management, nested simulation is used for risk evaluation of financial derivatives and insurance products.

The nested structure is due to the fact that the loss random variable is in the form of a conditional expectation of the portfolio loss given the underlying risk factors.

In our problems, the loss random variable  $L$  depends on the risk factors  $X$  and is the conditional expectation of the portfolio loss given  $X$ .

More specifically,  $X$  is  $d$ -dimensional and  $L$  is 1-dimensional.

Estimating a risk measure,  $\rho(L)$ , involves two levels of Monte Carlo simulations: outer level generates underlying risk factors, and inner level generates scenario-wise samples of portfolio losses.

The nested structure makes the estimation computationally expensive.

In this thesis, we focus on examining and improving the efficiency of nested simulation procedures mostly using supervised learning metamodels.

## Common Risk Measures

- Smooth  $h$ , e.g., quadratic tracking error,

$$\rho(L) = \mathbb{E}[(L - b)^2];$$

- hockey-stick  $h$ : mean excess loss,

$$\rho(L) = \mathbb{E}[L \cdot \mathbb{1}_{\{L \geq u\}}];$$

- indicator  $h$ : probability of large loss,

$$\rho(L) = \mathbb{E}[\mathbb{1}_{\{L \geq u\}}];$$

- Value at Risk (VaR),

$$\rho_\alpha(L) = Q_\alpha(L) = \inf\{u : \mathbb{P}(L \leq u) \geq \alpha\};$$

- Conditional Value at Risk (CVaR)<sup>1</sup>,

$$\rho_\alpha(L) = \mathbb{E}[L | L \geq Q_\alpha(L)].$$

<sup>1</sup>Note: If  $Q_\alpha(L)$  falls in a probability mass,  $\rho(L) = \frac{(\beta - \alpha)Q_\alpha(L) + (1 - \beta)\mathbb{E}[L | L \geq Q_\alpha(L)]}{1 - \alpha}$ .

2025-03-21

Smooth  $h$ , e.g., quadratic tracking error,

$$\rho(L) = \mathbb{E}[(L - b)^2];$$

hockey-stick  $h$ : mean excess loss,

$$\rho(L) = \mathbb{E}[L \cdot \mathbb{1}_{\{L \geq u\}}];$$

indicator  $h$ : probability of large loss,

$$\rho(L) = \mathbb{E}[\mathbb{1}_{\{L \geq u\}}];$$

Value at Risk (VaR),

$$\rho_\alpha(L) = Q_\alpha(L) = \inf\{u : \mathbb{P}(L \leq u) \geq \alpha\};$$

Conditional Value at Risk (CVaR)<sup>1</sup>,

$$\rho_\alpha(L) = \mathbb{E}[L | L \geq Q_\alpha(L)].$$

<sup>1</sup>Note: If  $Q_\alpha(L)$  falls in a probability mass,  $\rho(L) = \frac{(\beta - \alpha)Q_\alpha(L) + (1 - \beta)\mathbb{E}[L | L \geq Q_\alpha(L)]}{1 - \alpha}$ .

We consider different types of risk measures in this thesis.

The first three are in expectation form, and the last two are quantile-based risk measures.

The difference between the first three is the choice of the function  $h$ .

$h$  can be a smooth function, a hockey-stick function, or an indicator function.

The examples of risk measures are quadratic tracking error, mean excess loss, and probability of large loss.

The last two are value at risk and conditional value at risk, which are widely used in practice.

Note that the definition of CVaR requires the knowledge of the Value at Risk.

# Standard Nested Simulation

$$\hat{L}_{N,i} = \frac{1}{N} \sum_{j=1}^N Y_{ij}; \quad Y_{ij} \sim F_{Y|X_i}$$

- ❖ Uses inner sample mean to estimate  $L(X_i)$ .
- ❖ Proposed by Gordy and Juneja (2010); finds optimal growth order of  $M$  and  $N$ .
- ❖ Zhang et al. (2021) estimate the optimal  $M$  and  $N$  using a bootstrap method.
- ❖ Computationally expensive and potentially **wasteful** use of budget.

2025-03-21

## Resilient ML for Fast Risk Evaluation of VAs

### └ Nested Simulation Procedures in Financial Engineering: A Selected Review

#### └ Standard Nested Simulation

In our first project, we focus on comparing existing nested simulation procedures for different types of risk measures.

They differ in the way they estimate  $L(X_i)$ , the portfolio loss conditional on the risk factors  $X_i$ .

The standard nested simulation procedure uses the inner sample mean under that scenario to estimate  $L(X_i)$ .

The optimal growth order of  $M$  and  $N$  is found by Gordy and Juneja (2010).

For a growing total simulation budget  $\Gamma$ ,  $M$  should grow in the order of  $\Gamma^{2/3}$ , and  $N$  should grow in the order of  $\Gamma^{1/3}$ .

For a finite-sample case, the bootstrap method is used to estimate the optimal  $M$  and  $N$  by Zhang et al. (2021).

Since the standard procedure only uses the information from the same scenario, it is potentially **wasteful** use of budget.

Subsequent works focus on improving the efficiency of nested simulation by pooling inner replications from different outer scenarios.

$$L_{N,i} = \frac{1}{N} \sum_{j=1}^N Y_{ij}; \quad Y_{ij} \sim F_{Y|X_i}$$

- ❖ Use inner sample mean to estimate  $L(X_i)$ .
- ❖ Proposed by Gordy and Juneja (2010); finds optimal growth order of  $M$  and  $N$ .
- ❖ Zhang et al. (2021) estimate the optimal  $M$  and  $N$  using a bootstrap method.
- ❖ Computationally expensive and potentially **wasteful** use of budget.

# Metamodeling Approach

We focus on procedures that **pool** with **supervised learning metamodels**.

- ❖ Treat inner simulation as a black-box function
- ❖ Approximate  $L(\cdot)$  with  $\hat{L}_{M,N}^{\text{SL}}(\cdot)$
- ❖ Train with feature-label pairs generated by simulation:

$$\{(X_i, \hat{L}_{N,i}) | i = 1, \dots, M, j = 1, \dots, N\}$$

- ❖ Use metamodel predictions to estimate risk measures

There are **computational costs** associated with pooling inner replications.

2025-03-21

## Resilient ML for Fast Risk Evaluation of VAs

### └Nested Simulation Procedures in Financial Engineering: A Selected Review

#### └Metamodeling Approach

#### Metamodeling Approach

We focus on procedures that pool with supervised learning metamodels.

- Treat inner simulation as a black-box function
- Approximate  $L(\cdot)$  with  $\hat{L}_{M,N}^{\text{SL}}(\cdot)$
- Train with feature-label pairs generated by simulation:  
 $\{(X_i, \hat{L}_{N,i}) | i = 1, \dots, M, j = 1, \dots, N\}$
- Use metamodel predictions to estimate risk measures

There are computational costs associated with pooling inner replications.

A supervised learning based nested simulation procedure is implemented following these steps.

First, the user runs the standard nested simulation procedure to generate a set of outer scenarios and inner replications.

Then, the outer scenarios and inner sample means can be treated as features and labels. Their relationship is then approximated by a supervised learning metamodel.

Finally, the trained metamodel is used to make predictions for all  $X \in \mathcal{X}$ .

In other words, instead of using the inner sample mean to estimate  $L(X_i)$ , we use the trained metamodel to make predictions of the portfolio loss.

These loss predictions are then used to calculate the risk measures of interest.

For most research in this area, the focus is on the convergence rate of the risk measure estimator in terms of the simulation budget.

However, in this thesis, we note that there are other important aspects to consider, such as the cost of pooling that may come from metamodel training and validation.

# Asymptotic Convergence Rates of Different Procedures

$$\min_{M,N} \mathbb{E} [(\hat{\rho}_{M,N} - \rho)^2]$$

subject to  $M \cdot N = \Gamma$

Procedures	Smooth $h$	Hockey-Stick $h$	Indicator $h$
Standard Procedure	$\mathcal{O}(\Gamma^{-2/3})$	$\mathcal{O}(\Gamma^{-2/3})$	$\mathcal{O}(\Gamma^{-2/3})$
Regression	$\mathcal{O}(\Gamma^{-1})$	$\mathcal{O}(\Gamma^{-1+\delta})$	No Result
Kernel Smoothing	$\mathcal{O}(\Gamma^{-\min(1,4/(d+2))})$		
Kernel Ridge Regression <sup>2</sup>	$\mathcal{O}(\Gamma^{-1})$		
Likelihood Ratio	$\mathcal{O}(\Gamma^{-1})$		

2025-03-21

Resilient ML for Fast Risk Evaluation of VAs

└Nested Simulation Procedures in Financial Engineering: A Selected Review

└Theoretical Results

└Asymptotic Convergence Rates of Different Procedures

Asymptotic Convergence Rates of Different Procedures

$$\min_{M,N} \mathbb{E} [|\hat{\rho}_{M,N} - \rho|^2]$$

subject to  $M \cdot N = \Gamma$

Procedures	Smooth $h$	Hockey-Stick $h$	Indicator $h$
Standard Procedure	$\mathcal{O}(\Gamma^{-2/3})$	$\mathcal{O}(\Gamma^{-2/3})$	$\mathcal{O}(\Gamma^{-2/3})$
Regression	$\mathcal{O}(\Gamma^{-1})$	$\mathcal{O}(\Gamma^{-1+\delta})$	No Result
Kernel Smoothing	$\mathcal{O}(\Gamma^{-\min(1,4/(d+2))})$		
Kernel Ridge Regression <sup>2</sup>	$\mathcal{O}(\Gamma^{-1})$		
Likelihood Ratio	$\mathcal{O}(\Gamma^{-1})$		

We start by comparing the asymptotic convergence rates of different nested simulation procedures.

These are the convergence rates of the MSE of the risk measure estimator when the total simulation budget  $\Gamma$  goes to infinity, and when one follows the optimal growth order of  $M$  and  $N$ .

The standard procedure has the same convergence rate of  $\mathcal{O}(\Gamma^{-2/3})$  for all types of  $h$  considered. The rates in red are not available in the original works but are derived in this thesis.

The regression-based procedure has  $\mathcal{O}(\Gamma^{-1})$  convergence rate for smooth and hockey-stick  $h$ .

It is achieved by only allowing  $M$  to grow in the order of  $\Gamma^1$ . In their experiments,  $N$  is set to equal a value of 1.

The kernel smoothing procedure is the only procedure that depends on the asset dimension  $d$ .

When risk factor  $X$  is high-dimensional, the kernel smoothing procedure may converge slower than the standard procedure.

The kernel ridge regression procedure and likelihood ratio procedure have the same convergence rate as the regression-based procedure.

However, as we will see in the numerical experiments, their additional cost of pooling inner replications may offset their benefit of fast convergence rate.

Additionally for KRR, the authors analyze convergence of absolute error in probabilistic order instead of MSE.

## Key Theoretical Results

**Contribution:** bridging the gap between MSE and absolute error convergence.

❖ Convergence in MSE:

$$\mathbb{E} [(\hat{\rho}_{\Gamma} - \rho)^2] = \mathcal{O}(\Gamma^{-\xi})$$

❖ Convergence in Probabilistic Order:

$$|\hat{\rho}_{\Gamma} - \rho| = \mathcal{O}_{\mathbb{P}}(\Gamma^{-\xi})$$

### Theorem

If  $\hat{\rho}_{\Gamma}$  converges in MSE to  $\rho$  in order  $\xi$ , then  $\hat{\rho}_{\Gamma}$  converges in probabilistic order to  $\rho$  in order  $\frac{\xi}{2}$ .

## Resilient ML for Fast Risk Evaluation of VAs

### Nested Simulation Procedures in Financial Engineering: A Selected Review

#### Theoretical Results

#### Key Theoretical Results

2025-03-21

In most theoretical works, the focus is on the convergence rate of the MSE of the risk measure estimator.

However, for KRR, the authors analyze convergence of absolute error in probabilistic order.

One of the contributions of this thesis is to bridge the gap between MSE and absolute error convergence.

Let's first define the convergence in MSE and the convergence in probabilistic order.

The convergence in MSE is the convergence of the MSE of the risk measure estimator to the true risk measure, that is,  $\hat{\rho}_{\Gamma}$  minus  $\rho$  squared.

The convergence in probabilistic order is the convergence of the absolute error of the risk measure estimator to the true risk measure, that is,  $|\hat{\rho}_{\Gamma} - \rho|$  to 0 in probability.

We show that the convergence rate of the MSE of the risk measure estimator implies the convergence rate of the absolute error in probabilistic order.

We will also show that the converse is not necessarily true.

#### Key Theoretical Results

Contribution: bridging the gap between MSE and absolute error convergence.

❖ Convergence in MSE:

$$\mathbb{E} [(\hat{\rho}_{\Gamma} - \rho)^2] = \mathcal{O}(\Gamma^{-\xi})$$

❖ Convergence in Probabilistic Order:

$$|\hat{\rho}_{\Gamma} - \rho| = \mathcal{O}_{\mathbb{P}}(\Gamma^{-\xi})$$

#### Theorem

If  $\hat{\rho}_{\Gamma}$  converges in MSE to  $\rho$  in order  $\xi$ , then  $\hat{\rho}_{\Gamma}$  converges in probabilistic order to  $\rho$  in order  $\frac{\xi}{2}$ .



## Experiment Design

We compare 5 nested simulation procedures

- ❖ Standard nested simulation
- ❖ Regression-based
- ❖ Kernel smoothing
- ❖ Likelihood ratio
- ❖ Kernel ridge regression

And their empirical convergence across different:

- ❖ Risk measures
- ❖ Option types
- ❖ Asset dimensions
- ❖ Asset models (GBM vs. Heston)
- ❖ Regression bases (only for the regression-based procedure)

Resilient ML for Fast Risk Evaluation of VAs

└ Nested Simulation Procedures in Financial Engineering: A Selected Review

└ Finite-Sample Analysis

└ Experiment Design

2025-03-21

### Experiment Design

- We compare 5 nested simulation procedures
- ❖ Standard nested simulation
  - ❖ Regression-based
  - ❖ Kernel smoothing
  - ❖ Likelihood ratio
  - ❖ Kernel ridge regression
- And their empirical convergence across different:
- ❖ Risk measures
  - ❖ Option types
  - ❖ Asset dimensions
  - ❖ Asset models (GBM vs. Heston)
  - ❖ Regression bases (only for the regression-based procedure)

Previously, we worked on the asymptotic convergence rates of different nested simulation procedures.

In practice, the total simulation budget is finite. When the budget is finite, the theoretical convergence rate may not be the only thing we should consider.

We should also consider the cost of pooling itself.

The additional cost of pooling inner replications is not considered in the theoretical analysis of the previous works.

Neither it is mentioned in their numerical experiments.

More specifically, we have found that different research uses different numerical experiments to show the empirical rate of different procedures.

We aim to provide a fair and comprehensive analysis of the finite-sample performance of different nested simulation procedures.

The following numerical experiments test the empirical convergence rates of different nested simulation procedures across different risk measures, option types, asset dimensions, and asset models.

## Finite-Sample Performance

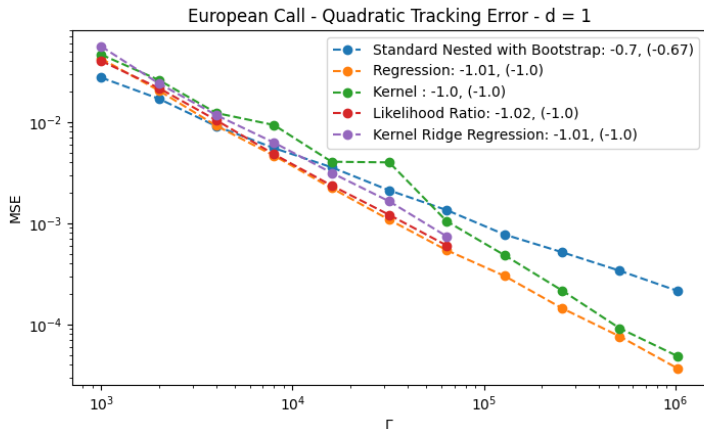


Figure: Empirical convergence rates of different procedures for the base case

## Resilient ML for Fast Risk Evaluation of VAs

### Nested Simulation Procedures in Financial Engineering: A Selected Review

#### Finite-Sample Analysis

#### Finite-Sample Performance

2025-03-21

#### Finite-Sample Performance

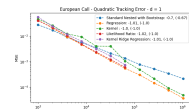


Figure: Empirical convergence rates of different procedures for the base case

This figure shows the performance of all 5 procedures in their estimation of the quadratic tracking error for the portfolio of European call options. It is a log-log plot of the MSE of the estimator versus the total simulation budget  $\Gamma$ . For each procedure, we run 1000 macro replications of Monte Carlo simulations with different total simulation budgets. For each data point on the plot, 1000 independent replications of the procedure are used to estimate the MSE.

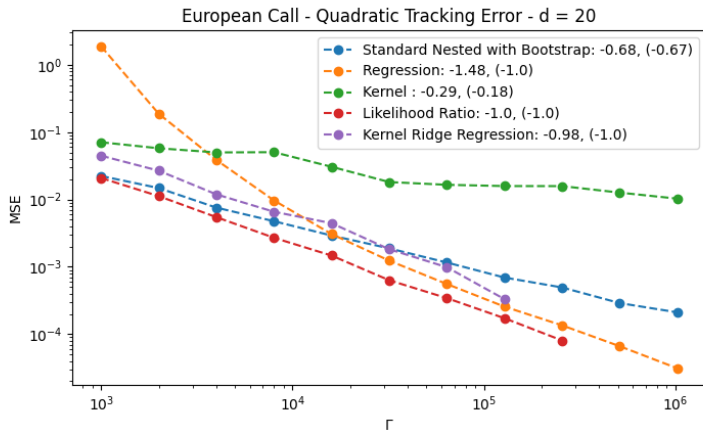
The slopes of the lines estimates the empirical convergence rates of the procedures. In the legend records the slopes of the lines and the corresponding asymptotic convergence rates of the procedures.

In the most basic case, most procedures have their empirical convergence rates match their asymptotic convergence rates.

However, as we will see after, this is not the case for all experiments.

The following slides will show the sensitivity of the procedures to different factors.

## Sensitivity to Asset Dimension



- Standard, KRR, and likelihood ratio procedures are dimension-independent
- Kernel smoothing and regression show sensitivity to dimension, but in different ways

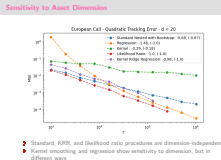
## Resilient ML for Fast Risk Evaluation of VAs

### Nested Simulation Procedures in Financial Engineering: A Selected Review

#### Finite-Sample Analysis

#### Sensitivity to Asset Dimension

2025-03-21



This figure shows the same experiment as the previous figure, but we further increase the asset dimension to 20.

The standard procedure is not affected by the asset dimension.

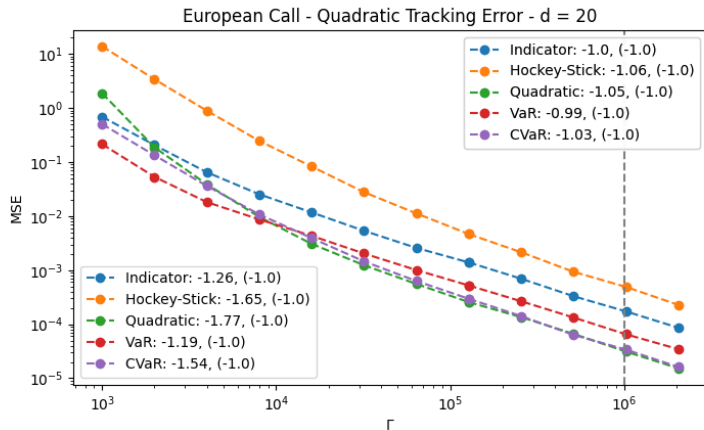
The kernel smoothing procedure actually converges faster than what its asymptotic convergence rate suggests.

The regression-based procedure is the only procedure that actually converges faster when the asset dimension is high. We will do a deeper dive into it in the following experiment.

It is also worth noting that the likelihood ratio and KRR procedures are dimension-independent.

While it is not shown in their empirical rate, due to their additional cost of pooling inner replications, we are not able to run these procedures when  $\Gamma$  is large.

## Fast Convergence of Regression-based Procedure



- Higher initial convergence rate
- Stabilizes to match asymptotic rate at higher budgets
- Consistent across different asset dimensions

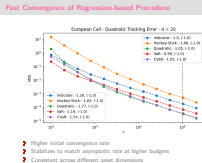
## Resilient ML for Fast Risk Evaluation of VAs

Nested Simulation Procedures in Financial Engineering: A Selected Review

Finite-Sample Analysis

Fast Convergence of Regression-based Procedure

2025-03-21



To investigate the regression-based procedure, we further increase the total simulation budget  $\Gamma$ .

The figure shows the performance of the regression-based procedure in the estimation of the quadratic tracking error for the portfolio of European call options.

We separate different values of  $\Gamma$  into two different parts.

The first part is when  $\Gamma$  is small.

The second part is when  $\Gamma$  is large.

The first part is when the regression-based procedure has a higher convergence rate than its asymptotic convergence rate.

The second part is when the regression-based procedure stabilizes and matches its asymptotic convergence rate.

In our experiments, we find that the regression-based procedure is very poor when  $\Gamma$  is small.

It means that when we don't have enough data points, the regression-based procedure is particularly bad.

As we increase the total simulation budget  $\Gamma$ , the regression-based procedure improves fast, and eventually matches the asymptotic convergence rate.

# Total Computation Time

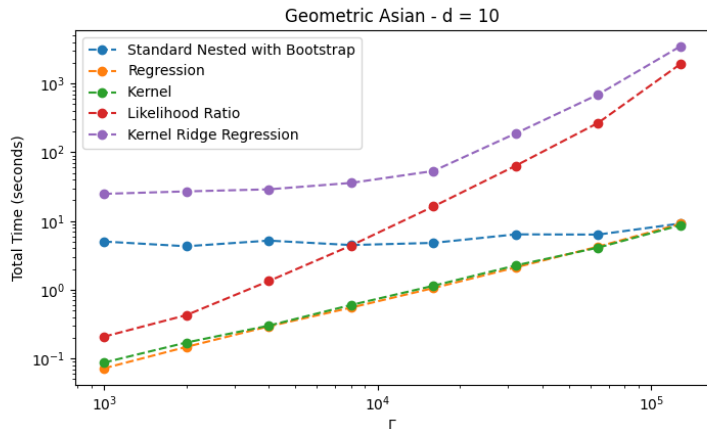


Figure: Total computation time for different procedures

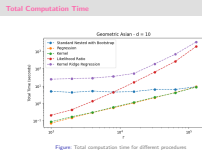
## Resilient ML for Fast Risk Evaluation of VAs

### Nested Simulation Procedures in Financial Engineering: A Selected Review

#### Finite-Sample Analysis

#### Total Computation Time

2025-03-21



This figure shows the total computation time for different procedures.

It includes the time for both the simulation, training, validation, and everything else.

The total computation time of the standard procedure does not increase fast with  $\Gamma$ .

It is because the computation time of estimating the optimal  $M$  and  $N$  does not increase for larger  $\Gamma$ .

The regression procedure and kernel smoothing procedure are the most efficient among metamodel-based procedures.

Our implementation of the kernel smoothing procedure is efficient because we only run the cross-validation once for all macro replications.

The likelihood ratio procedure and KRR procedure are more expensive due to their additional costs. The likelihood ratio weight calculations increase quadratically with  $\Gamma$ .

For KRR, the cost of cross-validation is so high that running it once still takes a very long time.

In the thesis includes a detailed analysis of the computational complexity of different procedures in terms of  $\Gamma$ .

## Conclusion

Regression-based nested simulation procedure:

- Most robust and stable for limited budgets
- Efficient to implement
- Fast empirical convergence for option portfolios

For high-dimensional or complex payoffs:

- Difficult to find a good regression basis
- Neural network-based procedures may be more suitable

Next project: examining performance of metamodel-based simulation procedures for variable annuities

Resilient ML for Fast Risk Evaluation of VAs

└ Nested Simulation Procedures in Financial Engineering: A Selected Review

└ Finite-Sample Analysis

└ Conclusion

2025-03-21

In our numerical experiment, we have shown that the regression-based procedure is the most cost-effective among all the procedures.

It is robust and efficient to implement for limited budget sizes, and has fast empirical convergence for option portfolios.

However, the regression-based procedure is not suitable for extremely high-dimensional risk factors or complex payoffs.

The first reason is that it is hard to find a good regression basis for high-dimensional risk factors. It will involve a lot of feature engineering and expert knowledge.

The second reason is that the regression-based procedure is not able to capture the nonlinearity in the payoff.

In the next project, we will examine the performance of metamodel-based simulation procedures for variable annuities.

This is where a neural network-based procedure can be more suitable as a form of automated feature engineering.

### Conclusion

Regression-based nested simulation procedure:

- Most robust and stable for limited budgets
- Efficient to implement
- Fast empirical convergence for option portfolios

For high-dimensional or complex payoffs:

- Difficult to find a good regression basis
- Neural network-based procedures may be more suitable

Next project: examining performance of metamodel-based simulation procedures for variable annuities

# Nested Simulation for Risk Management of VAs

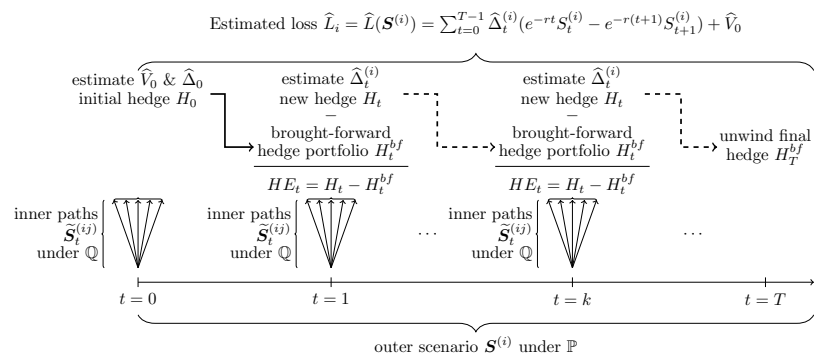


Figure: Illustration of nested simulation that estimates the P&L for one outer scenario

## Resilient ML for Fast Risk Evaluation of VAs

Cutting Through the Noise: Using Deep Neural Network Meta-models for High-Dimensional Nested Simulation

Nested Simulation for Risk Management of VAs



The main difference of the standard nested simulation of VAs can be explained by this illustration.

This is how the P&L is estimated for one outer scenario.

It starts at  $t=0$  with a simulation to estimate the initial value of the VA contract and the initial hedge. Here we are using delta-hedging.

This hedge gets carried forward and updated at each time period with another simulation.

The P&L is then estimated by adding up the hedging error at each time period to the initial value of the VA contract.

All the simulation mentioned is a single set of inner simulations for one outer scenario. It is very expensive to run this standard nested simulation for VAs.

# Metamodel-based Nested Simulation

We use deep neural networks (DNNs) as metamodels

- ❖ Use LSTMs for sequential data
- ❖ **Challenge:** lack of transparency and interpretability

## Research Contributions:

1. Propose two generic DNN-based nested simulation procedures
  - ❖ Accurate tail scenario identification
  - ❖ Significant computational savings by **budget concentration**
2. Study noise tolerance of DNNs using simulated data
  - ❖ **Control noise levels** by adjusting simulation parameters
  - ❖ Provide direct evidence on transparency and interpretability

2025-03-21

## Resilient ML for Fast Risk Evaluation of VAs

└ Cutting Through the Noise: Using Deep Neural Network Metamodels for High-Dimensional Nested Simulation

└ Metamodel-based Nested Simulation

After numerical experiments, we find that the deep neural network metamodels are suitable for this problem.

We use LSTM for the sequential data. It provides accurate tail scenario identification and significant computational savings by budget concentration.

However, it is not transparent and interpretable. Many researchers are working on this topic.

This is where we can contribute to the research.

By using nested simulation as a data generating process, we can study the noise tolerance of DNNs.

The noise level in the training labels can be controlled by adjusting the simulation parameters.

By examining the performance of DNNs with different noise levels, we can provide direct evidence on the transparency and interpretability of DNNs.

### Metamodel-based Nested Simulation

We use deep neural networks (DNNs) as metamodels

- ❖ Use LSTMs for sequential data
- ❖ **Challenge:** lack of transparency and interpretability

#### Research Contributions:

1. Propose two generic DNN-based nested simulation procedures
  - ❖ Accurate tail scenario identification
  - ❖ Significant computational savings by **budget concentration**
2. Study noise tolerance of DNNs using simulated data
  - ❖ **Control noise levels** by adjusting simulation parameters
  - ❖ Provide direct evidence on transparency and interpretability



## Two-Stage Metamodel-based Nested Simulation

### Algorithm Two-Stage Metamodel-based Nested Simulation for VAs

- 1: **Generate training data for metamodels**
- 2: **Train metamodels**
- 3: **Estimate  $\alpha$ -CVaR with extensive simulation on predicted tail scenarios**
  - ❖ Concentrate simulation on predicted tails

### Key Findings:

- ❖ Substantial computational savings (70% – 85% reduction)
- ❖ Some DNN metamodels make **accurate loss predictions**

2025-03-21

### Resilient ML for Fast Risk Evaluation of VAs

└ Cutting Through the Noise: Using Deep Neural Network Metamodels for High-Dimensional Nested Simulation

└ Two-Stage Metamodel-based Nested Simulation

#### Two-Stage Metamodel-based Nested Simulation

Algorithm Two-Stage Metamodel-based Nested Simulation for VAs

1. Generate training data for metamodels
2. Train metamodels
3. Estimate  $\alpha$ -CVaR with extensive simulation on **predicted tail scenarios**
  - ❖ Concentrate simulation on predicted tails

#### Key Findings:

- ❖ Substantial computational savings (70% – 85% reduction)
- ❖ Some DNN metamodels make **accurate loss predictions**

The two-stage procedure is a two-step process that is based on the idea of budget concentration.

First, we generate training data for the metamodel.

We use a fraction of the simulation budget to run the standard nested simulation procedure with  $M$  outer scenarios and  $N'$  inner replications.

We then train a metamodel using the feature-label pairs.

We use the trained metamodel to make predictions for all outer scenarios.

We then identify the predicted tail scenario set that contains the  $m$  largest predicted losses.

Finally, we run the standard nested simulation procedure on the predicted tail scenarios.

# Single-Stage Metamodel-based Nested Simulation

## Algorithm Single-Stage Metamodel-based Nested Simulation for VAs

- 1: **Generate training data for metamodels**
- 2: **Train metamodels**
- 3: **Estimate  $\alpha$ -CVaR with **metamodel predictions****
  - ▣ Entirely avoids extensive simulation

### Key advantages:

- ▣ more efficient than a two-stage procedure;
- ▣ avoids specifying a safety margin  $m$ .

2025-03-21

## Resilient ML for Fast Risk Evaluation of VAs

└ Cutting Through the Noise: Using Deep Neural Network Metamodels for High-Dimensional Nested Simulation

└ Single-Stage Metamodel-based Nested Simulation

Algorithm Single-Stage Metamodel-based Nested Simulation for VAs

1. Generate training data for metamodels
2. Train metamodels
3. Estimate  $\alpha$ -CVaR with **metamodel predictions**
  - ▣ Entirely avoids extensive simulation

### Key advantages:

- ▣ more efficient than a two-stage procedure;
- ▣ avoids specifying a safety margin  $m$ .

The single-stage procedure shares the same idea of running the standard nested simulation procedure to generate training data for the metamodel.

After a metamodel is trained, we use it to make predictions for VA contract losses across all outer scenarios.

The risk measure is estimated using the predicted losses of the metamodel.

It is more efficient than the two-stage procedure because it avoids the second stage of the two-stage procedure.

It also avoids specifying a safety margin  $m$ , which is a feature of the two-stage procedure that we will discuss later.

# Experiment Setting

We consider the following metamodel architectures:

Metamodel	Abbreviation	Capacity
Multiple Linear Regression	MLR	241
Quadratic Polynomial Regression	QPR	481
Feedforward Neural Network	FNN	35,009
Recurrent Neural Network	RNN	32,021
Long Short-Term Memory	LSTM	35,729

Table: Metamodel architectures for GMWB inner simulation model

In our experiment:

- 95%-CVaR of loss for a GMWB with a 240-month maturity
- 240-dimensional feature vector and 1-dimensional loss

2025-03-21

## Resilient ML for Fast Risk Evaluation of VAs

### Cutting Through the Noise: Using Deep Neural Network Meta-models for High-Dimensional Nested Simulation

#### Experiment Setting

We consider the following metamodel architectures:  
Multiple Linear Regression, Quadratic Polynomial Regression, Feedforward Neural Network, Recurrent Neural Network, and Long Short-Term Memory Network.  
Capacity is defined as the number of parameters in the metamodel.  
Higher capacity metamodels are more flexible and expressive.  
Lower capacity metamodels are less likely to overfit.  
The deep neural network metamodels are more flexible and expressive, but they are said to be more prone to overfitting.  
We will see whether this is indeed the case in our experiments.

Experiment Setting

We consider the following metamodel architectures:

Metamodel	Abbreviation	Capacity
Multiple Linear Regression	MLR	241
Quadratic Polynomial Regression	QPR	481
Feedforward Neural Network	FNN	35,009
Recurrent Neural Network	RNN	32,021
Long Short-Term Memory	LSTM	35,729

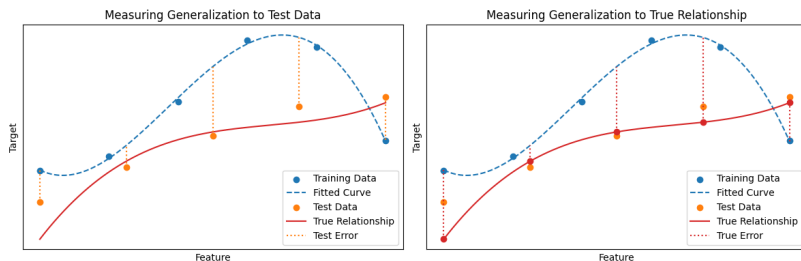
Table: Metamodel architectures for GMWB inner simulation model

In our experiment:

- 95%-CVaR of loss for a GMWB with a 240-month maturity
- 240-dimensional feature vector and 1-dimensional loss

## Research Questions

- What do DNNs learn from noisy data?
- How well do DNNs learn from noisy data?



Three datasets are considered:

- Training:** 90,000 scenarios (100 inner replications).
- Test:** 10,000 scenarios (100 inner replications).
- True:** 100,000 scenarios (100,000 inner replications).

## Resilient ML for Fast Risk Evaluation of VAs

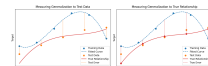
└ Cutting Through the Noise: Using Deep Neural Network Meta-models for High-Dimensional Nested Simulation

└ Research Questions

2025-03-21

### Research Questions

- What do DNNs learn from noisy data?
- How well do DNNs learn from noisy data?



Three datasets are considered:

- Training:** 90,000 scenarios (100 inner replications)
- Test:** 10,000 scenarios (100 inner replications)
- True:** 100,000 scenarios (100,000 inner replications)

To further clarify our research questions, we consider the following scenario:

We have a noisy training data and a noisy test data.

We want to know what the metamodel learns from the noisy data.

We also want to know how well the metamodel can make accurate loss predictions for given scenarios.

In a simulation experiment, we can control the noise level in the training data and the test data.

We also have access to the true feature-label relationship when we increase the number of replications.

We can use this true feature-label relationship to evaluate the performance of deep neural networks, which is usually not possible in practice.

# Metamodel Performance

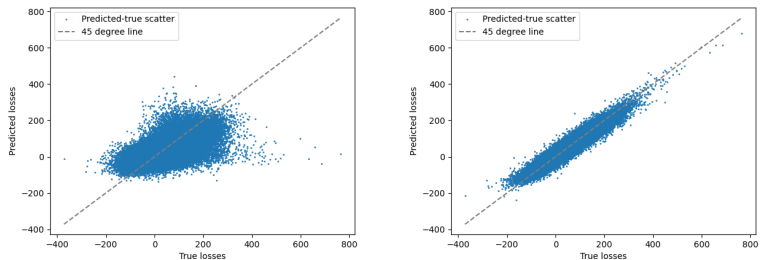


Figure: QQ plots between true and predicted loss labels for QPR and LSTM metamodels

- ❖ QPR make **inaccurate** loss predictions
- ❖ DNN metamodels are more flexible.

## Resilient ML for Fast Risk Evaluation of VAs

└ Cutting Through the Noise: Using Deep Neural Network Meta-models for High-Dimensional Nested Simulation

└ Metamodel Performance

2025-03-21

### Metamodel Performance

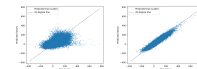


Figure: QQ plots between true and predicted loss labels for QPR and LSTM metamodels

- ❖ QPR make inaccurate loss predictions
- ❖ DNN metamodels are more flexible.

We first consider the traditional regression metamodels.

On the left is the QQ plot between the true loss labels and the predicted loss labels for MLR, and on the right is the QQ plot for QPR.

MLR and QPR metamodels make inaccurate loss predictions.

Feature engineering is hardly feasible for our 240-dimensional  $X$ .

We will see whether deep neural networks can make more accurate loss predictions.

# Metamodel Performance on Different Datasets

Metamodel	Training error	Test error	True error
MLR	0.706 ( $\pm 8.3 \times 10^{-4}$ )	0.713 ( $\pm 2.7 \times 10^{-2}$ )	0.706 ( $\pm 3.4 \times 10^{-4}$ )
QPR	0.543 ( $\pm 8.3 \times 10^{-4}$ )	0.554 ( $\pm 2.7 \times 10^{-2}$ )	0.544 ( $\pm 4.1 \times 10^{-4}$ )
FNN	0.129 ( $\pm 6.0 \times 10^{-3}$ )	0.240 ( $\pm 9.8 \times 10^{-3}$ )	0.132 ( $\pm 5.8 \times 10^{-3}$ )
RNN	0.132 ( $\pm 7.5 \times 10^{-3}$ )	0.137 ( $\pm 7.6 \times 10^{-3}$ )	0.119 ( $\pm 7.5 \times 10^{-3}$ )
LSTM	0.075 ( $\pm 4.5 \times 10^{-3}$ )	0.079 ( $\pm 5.4 \times 10^{-3}$ )	0.063 ( $\pm 4.4 \times 10^{-3}$ )
RNN <sup>*3</sup>	0.109 ( $\pm 5.2 \times 10^{-3}$ )	0.128 ( $\pm 5.2 \times 10^{-3}$ )	0.109 ( $\pm 5.2 \times 10^{-3}$ )

Table: Average MSEs and 95% confidence bands of metamodels for GMWB.

- ❖ RNN-based metamodels have lower true errors than their training errors.
- ❖ DNN metamodels with suitable architectures **cut through the noise** in training labels.

<sup>3</sup>This row summarizes the results of the well-trained RNNs.

2025-03-21

- Resilient ML for Fast Risk Evaluation of VAS
  - Cutting Through the Noise: Using Deep Neural Network Metamodels for High-Dimensional Nested Simulation
  - Metamodel Performance on Different Datasets

Metamodel Performance on Different Datasets			
Metamodel	Training error	Test error	True error
MLR	0.706 ( $\pm 8.3 \times 10^{-4}$ )	0.713 ( $\pm 2.7 \times 10^{-2}$ )	0.706 ( $\pm 3.4 \times 10^{-4}$ )
QPR	0.543 ( $\pm 8.3 \times 10^{-4}$ )	0.554 ( $\pm 2.7 \times 10^{-2}$ )	0.544 ( $\pm 4.1 \times 10^{-4}$ )
FNN	0.129 ( $\pm 6.0 \times 10^{-3}$ )	0.240 ( $\pm 9.8 \times 10^{-3}$ )	0.132 ( $\pm 5.8 \times 10^{-3}$ )
RNN	0.132 ( $\pm 7.5 \times 10^{-3}$ )	0.137 ( $\pm 7.6 \times 10^{-3}$ )	0.119 ( $\pm 7.5 \times 10^{-3}$ )
LSTM	0.075 ( $\pm 4.5 \times 10^{-3}$ )	0.079 ( $\pm 5.4 \times 10^{-3}$ )	0.063 ( $\pm 4.4 \times 10^{-3}$ )
RNN <sup>†</sup>	0.109 ( $\pm 5.2 \times 10^{-3}$ )	0.128 ( $\pm 5.2 \times 10^{-3}$ )	0.109 ( $\pm 5.2 \times 10^{-3}$ )
Table: Average MSEs and 95% confidence bands of metamodels for GMWB.			
❖ RNN-based metamodels have lower true errors than their training errors.			
❖ DNN metamodels with suitable architectures cut through the noise in training labels.			
<sup>†</sup> This row summarizes the results of the well-trained RNNs.			

This table summarizes the results of our experiments. It is the most interesting finding that DNN metamodels with suitable architectures can **cut through the noise** in training labels. The average MSEs of metamodels on the training, test, and true data are reported in the three columns, respectively. Note that the training and test labels are noisy. The simulation noise is what makes them noisy, which is less present in the true data. First, we note that DNN metamodels has lower training errors than the traditional regression metamodels. This is expected because DNNs are more flexible and expressive than traditional regression metamodels. What is more interesting is the comparison between the test and true errors. We find that the true errors of RNN-based metamodels are lower than their test errors. This suggests that the RNN and LSTM metamodels are able to cut through the simulation noise and learn the true feature-label relationship. When we take another look at the table, we find that for LSTM, its true error is even lower than its training error. This is another evidence that LSTM metamodels are able to cut through the noise in training labels and fit to the true relationship. We also find that the LSTM metamodel is able to cut through the noise in training labels and fit to the true relationship. We have another finding for RNN. RNN metamodels suffer from the vanishing gradient

# Safety Margin

Consider estimating the 95% CVaR with 100,000 outer scenarios.

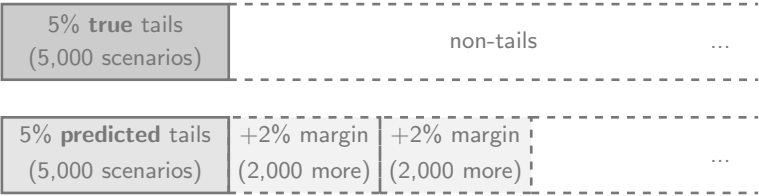


Figure: Illustration: a safety margin of 4% ( $m = 9000$ )

## Trade-off between accuracy and efficiency

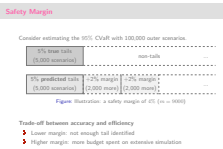
- ❖ Lower margin: not enough tail identified
- ❖ Higher margin: more budget spent on extensive simulation

2025-03-21

# Resilient ML for Fast Risk Evaluation of VAs

## Cutting Through the Noise: Using Deep Neural Network Meta-models for High-Dimensional Nested Simulation

### Safety Margin



For a two-stage procedure, we need to choose a safety margin  $m$  to identify the tail scenarios.

Consider estimating the 95% CVaR with 100,000 outer scenarios.

On the top is the true tail and non-tail scenarios, and on the bottom is the predicted tail and non-tail scenarios.

The 5% predicted tail scenarios are not guaranteed to contain all the true tail scenarios. Therefore, we need to add a safety margin  $m$  to the predicted tail scenarios.

On the bottom, we add a 2% safety margin to the predicted tail scenarios. This adds 2,000 more scenarios to the inner simulations.

We may add another 2% safety margin to be more confident. This introduces a safety margin of 4%. 9000 predicted tail scenarios are used for extensive inner simulations in stage 2.

This is a trade-off between accuracy and efficiency.

For a good metamodel, we only need a small safety margin to be accurate in tail identification.

However, for a bad metamodel, the safety margin needs to be large to identify all the tail scenarios.

This comes at the cost of more inner simulations, which is computationally expensive. So, one of the limitations of the two-stage procedure is the choice of safety margin. It can be partially mitigated by using a good metamodel.

## Metamodel Performance

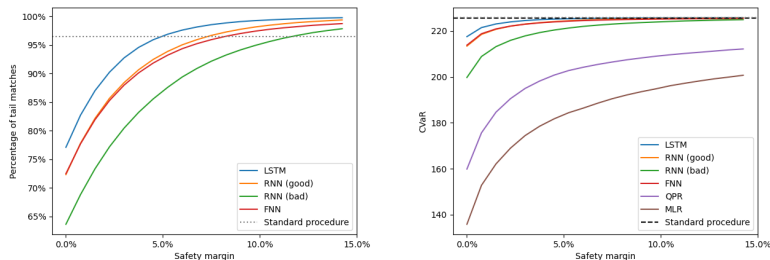


Figure: Tail Matches and CVaR Predictions for DNN Metamodels

- Traditional regression metamodels are **unable** to accurately identify tail scenarios even with high safety margins.
- LSTM metamodels surpasses the standard procedure with 5% safety margin.

## Resilient ML for Fast Risk Evaluation of VAs

Cutting Through the Noise: Using Deep Neural Network Metamodels for High-Dimensional Nested Simulation

Metamodel Performance

2025-03-21

### Metamodel Performance

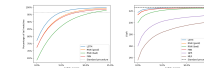


Figure: Tail Matches and CVaR Predictions for DNN Metamodels

- Traditional regression metamodels are unable to accurately identify tail scenarios even with high safety margins.
- LSTM metamodels surpasses the standard procedure with 5% safety margin.

We now consider the tail scenario identification for the two-stage procedure.

On the left is the tail scenario identification for the traditional regression metamodel, and on the right is the tail scenario identification for the LSTM metamodel.

We compare the two-stage procedure with the standard procedure.

When colored lines cross the grey dashed line, it means that the metamodel is able to identify the tail scenarios as well as the standard procedure.

We find that the LSTM metamodel is able to surpass the standard procedure on tail identification with a less than 5% safety margin.

However, the traditional regression metamodel is unable to identify the tail scenarios even with a high safety margin.

A well-trained RNN is better than a FNN, but the poorly-trained RNN is far worse. This further motivates the use of LSTM metamodels over RNN metamodels.



# Sensitivity Testing for DNNs

Simulation controls **noise level in training labels** and **number of training samples**.

$N'$  varies the noise level.

- ❖ **Low noise labels:**  $N' = 100$
- ❖ **Medium noise labels:**  $N' = 10$
- ❖ **High noise labels:**  $N' = 1$

$M$  varies the number of training samples.

- ❖  $M \in \{10^2, 10^3, 10^4, 10^5\}$

2 LSTMs of **different capacities** are examined based on their MSEs.

## Resilient ML for Fast Risk Evaluation of VAs

└ Cutting Through the Noise: Using Deep Neural Network Metamodels for High-Dimensional Nested Simulation

└ Sensitivity Testing for DNNs

2025-03-21

### Sensitivity Testing for DNNs

Simulation controls noise level in training labels and number of training samples.

$N'$  varies the noise level.

- ❖ Low noise labels:  $N' = 100$
- ❖ Medium noise labels:  $N' = 10$
- ❖ High noise labels:  $N' = 1$

$M$  varies the number of training samples.

- ❖  $M \in \{10^2, 10^3, 10^4, 10^5\}$

2 LSTMs of different capacities are examined based on their MSEs.

We now consider the sensitivity testing for the DNN metamodels.

We are interested in the sensitivity of the DNN metamodels to the noise level in training labels and the number of training samples.

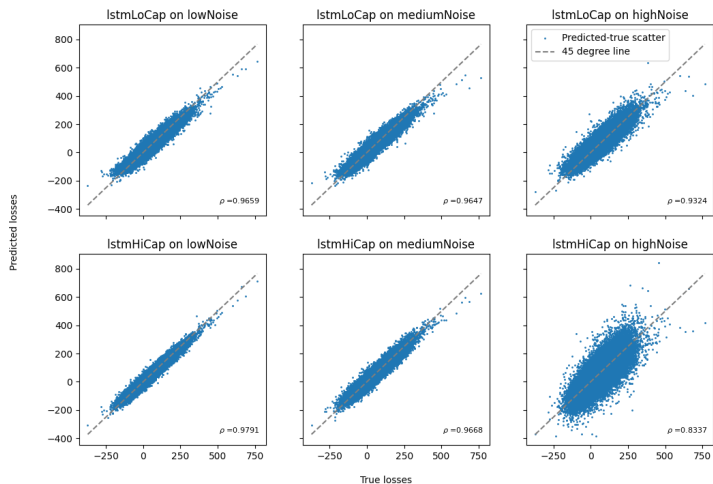
In a simulation experiment, we can vary them to see how the DNN metamodels perform. We consider the noise level in training labels by varying the number of inner replications  $N'$ .

We consider the number of training samples by varying the number of outer scenarios  $M$ .

We consider two LSTM metamodels of different capacities because we want to see if the capacity of the metamodel matters for overfitting.

The high-capacity LSTM metamodel has 10 times more parameters than the regular LSTM metamodel

## Noise Tolerance of DNNs



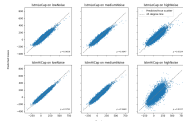
2025-03-21

### Resilient ML for Fast Risk Evaluation of VAs

└ Cutting Through the Noise: Using Deep Neural Network Meta-models for High-Dimensional Nested Simulation

└ Noise Tolerance of DNNs

Noise Tolerance of DNNs



This figure shows the QQ plots of the two LSTM metamodels.

We plot the predicted losses versus the true losses for both LSTMs and different noise levels.

The closer the points are to the line, the better the model is.

The high-capacity LSTM is better at low and medium noise levels, while the regular LSTM is better at high noise levels.

We can see on the bottom right that the high-capacity LSTM has very poor performance.

## Sensitivity of High-capacity LSTM

	$N' = 1$	$N' = 10$	$N' = 100$	$N' = 1000$
$M = 100$	0.764	0.408	0.131	0.087
$M = 1000$	0.878	0.367	0.156	0.087
$M = 10000$	0.351	0.147	0.064	0.063
$M = 100000$	0.149	0.065	0.060	0.038

Table: MSE between high-capacity LSTM’s predicted losses and true losses.

- Same color → same total simulation budget.
- $N' = 10$  is a reasonable budget allocation for LSTM metamodels.

## Resilient ML for Fast Risk Evaluation of VAs

Cutting Through the Noise: Using Deep Neural Network Meta-models for High-Dimensional Nested Simulation

Sensitivity of High-capacity LSTM

### Sensitivity of High-capacity LSTM

	$N' = 1$	$N' = 10$	$N' = 100$	$N' = 1000$
$M = 100$	0.764	0.408	0.131	0.087
$M = 1000$	0.878	0.367	0.156	0.087
$M = 10000$	0.351	0.147	0.064	0.063
$M = 100000$	0.149	0.065	0.060	0.038

Table: MSE between high-capacity LSTM’s predicted losses and true losses.

- Same color → same total simulation budget.
- $N' = 10$  is a reasonable budget allocation for LSTM metamodels.

The findings are similar for the high-capacity LSTM metamodel.

We can find similar patterns as in the previous table.

The model is more sensitive to  $N'$  than  $M$ , unless  $N'$  is very small.

For a practical application, we recommend using the high-capacity LSTM metamodel with  $N' = 10$  for almost all simulation budgets.

## Single-Stage Procedure

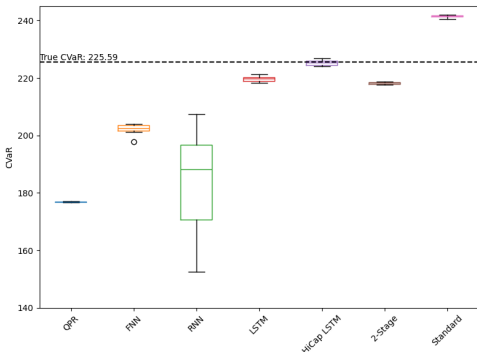


Figure: CVaR estimates of single-stage procedures with  $N' = 10$ .

- ❖ The single-stage procedure outperforms the two-stage procedure.
- ❖  $N' = 10$  is a reasonable budget allocation.

## Resilient ML for Fast Risk Evaluation of VAs

Cutting Through the Noise: Using Deep Neural Network Meta-models for High-Dimensional Nested Simulation

Single-Stage Procedure

2025-03-21

### Single-Stage Procedure

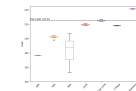


Figure: CVaR estimates of single-stage procedures with  $N' = 10$ .

- ❖ The single-stage procedure outperforms the two-stage procedure.
- ❖  $N' = 10$  is a reasonable budget allocation.

This figure shows the CVaR estimates of the single-stage procedure.

We compare it with the two-stage procedure using regular LSTM metamodel with no safety margin. So we are using the 5% metamodel predicted tails to estimate the 95% CVaR with extensive simulation in stage 2.

The single-stage procedure with LSTM metamodels outperforms the two-stage procedure when noise level is moderate.

Due to its design, the single-stage procedure is more efficient than the two-stage procedure.

Setting  $N' = 10$  is a reasonable budget allocation for the single-stage procedure.

Further decreasing  $N'$  to 1 deteriorates the performance of the single-stage procedure. But it is worth noting that the 2-stage procedure can overcome this issue by increasing the safety margin.

# Convergence Analysis

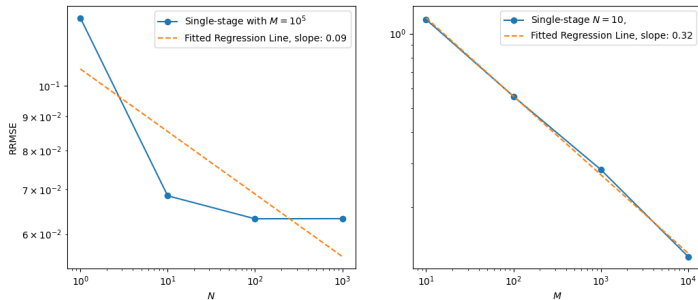


Figure: Empirical convergence of the single-stage procedure with a LSTM metamodel.

- Minimal effect of increasing  $N'$  on CVaR estimation.
- For a given  $\Gamma$ , set  $N'$  constant and allocate budget to outer simulations.

## Resilient ML for Fast Risk Evaluation of VAs

Cutting Through the Noise: Using Deep Neural Network Meta-models for High-Dimensional Nested Simulation

Convergence Analysis

2025-03-21

### Convergence Analysis

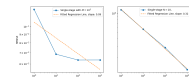


Figure: Empirical convergence of the single-stage procedure with a LSTM metamodel.

- Minimal effect of increasing  $N'$  on CVaR estimation.
- For a given  $\Gamma$ , set  $N'$  constant and allocate budget to outer simulations.

This figure shows the empirical convergence of the single-stage procedure with LSTM metamodels when only  $M$  or  $N'$  is varied.

On the left, we fix  $M = 10^5$  and vary  $N'$ .

There is no significant difference to further increase  $N'$  when  $N' \geq 10$ .

On the right, we fix  $N' = 10$  and vary  $M$ .

We can see that the convergence rate is around  $O(M^{-2/3})$ .

This finding is consistent with our previous analysis on the sensitivity of LSTM meta-models to data quality and data quantity.

Once the quality is good enough, the quantity is more important.

# Conclusion

## Key Findings:

- ❖ LSTMs are **resilient** to moderate levels of noise in training labels.
- ❖ DNNs can learn **true** complex dynamic hedging model.
- ❖ Two-stage procedure addresses regulatory concerns.
- ❖ Single-stage procedure is **efficient**.

## Future Directions:

- ❖ Apply DNNs to other risk management tasks.
- ❖ Fast adaptation to new contracts/market conditions.

2025-03-21

## Resilient ML for Fast Risk Evaluation of VAs

└─Cutting Through the Noise: Using Deep Neural Network Meta-models for High-Dimensional Nested Simulation

└─Conclusion

In this project, we present our extensive research into LSTM metamodels in a simulation setting. Our findings demonstrate that LSTMs can effectively learn complex hedging strategies even with noisy training data. The single-stage procedure offers an excellent balance of efficiency and accuracy for practical applications. We've shown that allocating computational budget to increase outer scenarios rather than inner simulations yields better results. These insights provide valuable guidance for implementing machine learning in financial risk management. For the research directions. In the future, we can apply deep neural network metamodels to other actuarial applications. In the future, we can also apply transfer learning to speed up the training process for deep neural network metamodels.

Conclusion

Key Findings:

- ❖ LSTMs are resilient to moderate levels of noise in training labels.
- ❖ DNNs can learn true complex dynamic hedging model.
- ❖ Two-stage procedure addresses regulatory concerns.
- ❖ Single-stage procedure is efficient.

Future Directions:

- ❖ Apply DNNs to other risk management tasks.
- ❖ Fast adaptation to new contracts/market conditions.

# Transfer Learning for Rapid Adaptation of DNN Metamodels

**Problem:** Updating models for new conditions is expensive.

- ❖ Full retraining takes too much time.
- ❖ New VA contracts need quick model updates.
- ❖ Need balance between speed and accuracy.

**Solution:** Transfer learning for faster model adaptation.

- ❖ Train first on existing contract data.
- ❖ Update with small amount of new contract data.
- ❖ Reuse knowledge between similar contracts.
- ❖ Benefits:
  - ❖ Faster training.
  - ❖ Less data needed.

## Resilient ML for Fast Risk Evaluation of VAs

### Transfer Learning for Rapid Adaptation of DNN Metamodels

### Transfer Learning for Rapid Adaptation of DNN Metamodels

2025-03-21

In this project, we propose a transfer learning framework for rapid adaptation of deep neural network metamodels in VA dynamic hedging.

The problem is the same as the previous project. We want to use deep neural network metamodels to approximate the complex simulation model that involves dynamic hedging.

The difference is that we want to use transfer learning to speed up the training process. This can be beneficial for the practical applications.

Firstly, new VA contracts are continuously issued.

Secondly, market conditions are changing.

Thirdly, the computational budget is limited.

Transfer learning can help us to adapt to the new market conditions quickly and efficiently.

We can pre-train the deep neural network metamodel on a large dataset of VA contracts with abundant simulation data.

Then, we can fine-tune the metamodel on a smaller dataset of new VA contracts with limited data.

The data is often limited for new VA contracts as simulation is expensive to run.

The shared features between VA contracts can be leveraged to speed up the training process.

We can also transfer the knowledge to other contract types.

When a new VA contract is issued, we can use transfer learning to adapt the metamodel to the new contract quickly.

Problem: Updating models for new conditions is expensive.

- ❖ Full retraining takes too much time.
- ❖ New VA contracts need quick model updates.
- ❖ Need balance between speed and accuracy.

Solution: Transfer learning for faster model adaptation.

- ❖ Train first on existing contract data.
- ❖ Update with small amount of new contract data.
- ❖ Reuse knowledge between similar contracts.
- ❖ Benefits:
  - ❖ Faster training.
  - ❖ Less data needed.

# Transfer Learning Framework

## Key Components:

- **Domain  $\mathcal{D}$ :** feature space  $\mathcal{X}$  + probability distribution  $F$
- **Task  $\mathcal{T}$ :** label space  $\mathcal{Y}$  + predictive function  $f : \mathcal{X} \rightarrow \mathcal{Y}$

## Source vs. Target:

- **Source:**  $\mathcal{D}_{\text{So}} = \{\mathcal{X}_{\text{So}}, F_{\text{So}}(X)\}$
- **Target:**  $\mathcal{D}_{\text{Ta}} = \{\mathcal{X}_{\text{Ta}}, F_{\text{Ta}}(X)\}$

## Our Goal:

- **Input features  $X$ :** risk factors from outer simulation
- **Output labels  $L$ :** contract losses
- **Source and target:** from VAs with abundant simulation data to new VAs with limited data
- **Goal:** improve  $f_{\text{Ta}}(\cdot)$  using knowledge from  $\mathcal{D}_{\text{So}}$  and  $f_{\text{So}}(\cdot)$

2025-03-21

# Resilient ML for Fast Risk Evaluation of VAs

- └ Transfer Learning for Rapid Adaptation of DNN Metamodels
  - └ Transfer Learning Framework

Transfer Learning Framework

Key Components:

- Domain  $\mathcal{D}$ : feature space  $\mathcal{X}$  + probability distribution  $F$
- Task  $\mathcal{T}$ : label space  $\mathcal{Y}$  + predictive function  $f : \mathcal{X} \rightarrow \mathcal{Y}$

Source vs. Target:

- Source:  $\mathcal{D}_{\text{So}} = \{\mathcal{X}_{\text{So}}, F_{\text{So}}(X)\}$
- Target:  $\mathcal{D}_{\text{Ta}} = \{\mathcal{X}_{\text{Ta}}, F_{\text{Ta}}(X)\}$

Our Goal:

- Input features  $X$ : risk factors from outer simulation
- Output labels  $L$ : contract losses
- Source and target: from VAs with abundant simulation data to new VAs with limited data
- Goal: improve  $f_{\text{Ta}}(\cdot)$  using knowledge from  $\mathcal{D}_{\text{So}}$  and  $f_{\text{So}}(\cdot)$

Transfer learning provides a formal framework for adapting models from one domain to another.

The domain consists of a feature space and a probability distribution over that space. In our context, the feature space includes risk factors from the outer simulation. It is equipped with a probability distribution that describes the joint distribution of the risk factors.

We don't know the exact joint distribution of the risk factors, but we can simulate outer scenarios from it using monte carlo simulation.

The task consists of a label space and a predictive function mapping features to labels. For VA contracts, our labels are the contract losses under different scenarios.

The source domain typically has abundant data - this could be existing VA contracts with extensive simulation data.

The target domain has limited data - new VA contracts or existing contracts under new market conditions.

Our goal is to leverage knowledge from the source domain to improve prediction in the target domain.

This is particularly valuable when simulation data is expensive to generate for new contracts.

The transfer learning approach allows us to maintain accuracy while significantly reducing computational requirements.



## Transfer Learning Techniques

### Common Techniques:

- **Fine-tuning:** a model pre-trained on a source task is used as a starting point for a target task.
- **Layer freezing:** only part of the model is fine-tuned.
- **Multi-task learning:** perform training on multiple tasks simultaneously.

### Key considerations:

- Similarity between source and target tasks
- Appropriate learning rate

## Resilient ML for Fast Risk Evaluation of VAs

### Transfer Learning for Rapid Adaptation of DNN Metamodels

### Transfer Learning Techniques

2025-03-21

Fine-tuning involves using a pre-trained model as a starting point for a new task.

This approach is effective when the source and target tasks are similar.

For VA contracts, we can fine-tune models between different contract types or market conditions.

Layer freezing is a technique where we keep some layers of the pre-trained model fixed. Typically, early layers capture general features while later layers are more task-specific. By freezing early layers, we preserve general knowledge while adapting task-specific layers.

Multi-task learning trains a single model to perform multiple related tasks simultaneously.

This approach can improve generalization by leveraging the domain-specific information across tasks.

For VA contracts, we can train a model to handle multiple contract types or features.

The learning rate is a critical hyperparameter in transfer learning.

Typically, we use a smaller learning rate for fine-tuning compared to training from scratch.

The similarity between source and target tasks greatly impacts transfer learning effectiveness.

More similar tasks generally allow for more effective knowledge transfer.

#### Common Techniques:

- **Fine-tuning:** a model pre-trained on a source task is used as a starting point for a target task.
- **Layer freezing:** only part of the model is fine-tuned.
- **Multi-task learning:** perform training on multiple tasks simultaneously.

#### Key considerations:

- Similarity between source and target tasks
- Appropriate learning rate

# Multi-task Learning



- ❖ LSTM layers shared across multiple tasks
- ❖ Task-specific fully connected layers
- ❖ Objective: Minimize sum of loss functions across all tasks

2025-03-21

## Resilient ML for Fast Risk Evaluation of VAs

### Transfer Learning for Rapid Adaptation of DNN Metamodels

#### Multi-task Learning



Multi-task learning offers an alternative approach to transfer learning.

Instead of sequential knowledge transfer, we train on multiple contract types simultaneously.

The shared LSTM layers capture common features across different VA contracts.

Task-specific layers then specialize for each contract type's unique characteristics.

We will examine whether this approach can be more efficient than training separate models for each contract type.

## Experiment Setup

Contract	Asset Model	Lapse	$M_{So}$	$M_{Ta}$
GMMB	GBM	No lapse	50000	N/A
GMMB	RS-GBM	No lapse	50000	2000
GMMB	RS-GBM	Static lapse	50000	2000
GMMB	RS-GBM	Dynamic lapse	50000	2000
GMWB	RS-GBM	Dynamic lapse	N/A	2000

Table: VA Contracts for Transfer Learning Experiments

We aim to examine the performance of TL techniques

- ❖ learning the lapse features,
- ❖ learning the dynamic lapse, and
- ❖ transferring to other contract types.

2025-03-21

Resilient ML for Fast Risk Evaluation of VAs

Transfer Learning for Rapid Adaptation of DNN Metamodels

Experiment Setup

Experiment Setup

Contract	Asset Model	Lapse	$M_{So}$	$M_{Ta}$
GMMB	GBM	No lapse	50000	N/A
GMMB	RS-GBM	No lapse	50000	2000
GMMB	RS-GBM	Static lapse	50000	2000
GMMB	RS-GBM	Dynamic lapse	50000	2000
GMWB	RS-GBM	Dynamic lapse	N/A	2000

Table: VA Contracts for Transfer Learning Experiments

We aim to examine the performance of TL techniques

- ❖ learning the lapse features,
- ❖ learning the dynamic lapse, and
- ❖ transferring to other contract types.

We use a variety of VA contracts with different features to test transfer learning. The source model ( $M_{So}$ ) is trained with 50,000 samples for comprehensive learning. The target model ( $M_{Ta}$ ) uses only 2,000 samples to simulate limited data scenarios. We progress from simpler to more complex models (No lapse  $\rightarrow$  Static  $\rightarrow$  Dynamic). We examine both within-contract transfer (GMMB to GMMB with different features) and cross-contract transfer (GMMB to GMWB). The regime-switching GBM (RS-GBM) model captures more realistic market dynamics. All training data is generated using 100 inner replications for consistency. We compare fine-tuning and layer freezing as two primary transfer learning approaches. The experiments are designed to evaluate how similarity between source and target affects transfer efficiency. This is another attempt to use simulation models as data generating processes to examine deep neural networks. By examining the transfer learning performance, we can gain insights into which parts of the neural network are transferable, or more generally, which layers of the neural network learn which features of the VA contracts.

## Results: Accuracy Comparison

Method and Setting	MSE
Extensive Training (Dynamic)	<b>0.0587</b>
Fine-tuning (Static)	0.0794
Layer Freezing (Static)	0.0763
Layer Freezing (No Lapse)	0.3361
Fine-tuning (No Lapse)	0.4894
Without TL (Dynamic)	0.2950

Table: Comparison of different TL methods on GMMB contracts (best MSE values)

2025-03-21

- Resilient ML for Fast Risk Evaluation of VAs
  - Transfer Learning for Rapid Adaptation of DNN Metamodels
    - Results: Accuracy Comparison

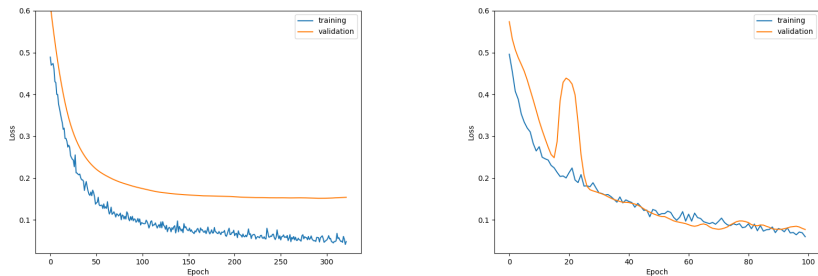
Results: Accuracy Comparison

Method and Setting	MSE
Extensive Training (Dynamic)	<b>0.0587</b>
Fine-tuning (Static)	0.0794
Layer Freezing (Static)	0.0763
Layer Freezing (No Lapse)	0.3361
Fine-tuning (No Lapse)	0.4894
Without TL (Dynamic)	0.2950

Table: Comparison of different TL methods on GMMB contracts (best MSE values)

This table summarizes the MSE values for different training methods. Extensive training with full data achieves the best performance (0.0587 MSE). Without transfer learning, training from scratch on dynamic lapse data yields mediocre results (0.2950 MSE). Transfer learning from static lapse models (both fine-tuning and layer freezing) performs well. Transfer from no lapse models performs poorly, demonstrating that negative transfer can occur. Negative transfer happens when source and target tasks are substantially different. In such cases, knowledge from the source model can actually hinder learning on the target task. This highlights the importance of selecting appropriate source models for transfer learning.

## Learning Curves: Effect of Similarity



*The effect of similarity between source and target on the convergence speed*

❖ Left: from No Lapse; right: from Static Lapse

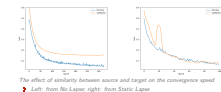
Resilient ML for Fast Risk Evaluation of VAs

└ Transfer Learning for Rapid Adaptation of DNN Metamodels

└ Learning Curves: Effect of Similarity

2025-03-21

Learning Curves: Effect of Similarity



The effect of similarity between source and target on the convergence speed  
 ❖ Left: from No Lapse; right: from Static Lapse

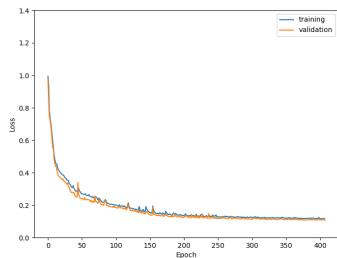
These figures further illustrate how source-target similarity affects transfer learning performance.

The left graph shows transfer from no lapse models, which has slower convergence.

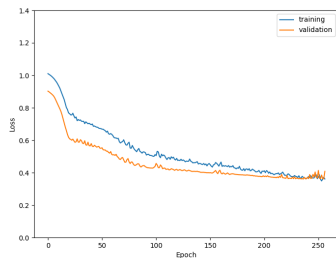
The right graph shows transfer from static lapse models, with faster convergence.

This reinforces our finding that greater similarity between source and target domains leads to more effective knowledge transfer.

## Learning Curves: Transfer Knowledge to other Contract Types



(a) 50000 samples



(b) 2000 samples

Figure: Learning curves of GMWB contracts

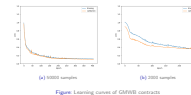
2025-03-21

Resilient ML for Fast Risk Evaluation of VAs

└ Transfer Learning for Rapid Adaptation of DNN Metamodels

└ Learning Curves: Transfer Knowledge to other Contract Types

Learning Curves: Transfer Knowledge to other Contract Types



These graphs show learning curves for GMWB contracts with different training sample sizes.

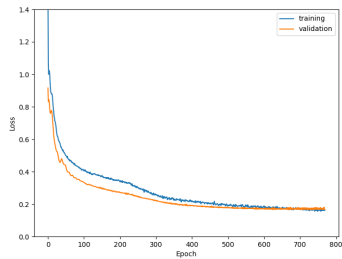
Left figure (50,000 samples) shows that with abundant data, all methods eventually converge well.

Right figure (2,000 samples) demonstrates the advantage of transfer learning in data-limited scenarios.

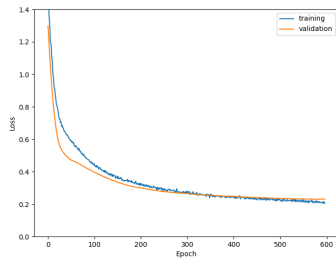
With limited data, transfer learning methods significantly outperform training from scratch.

This is particularly relevant for new products or market conditions where historical data is scarce.

## Learning Curves: Effect of Similarity



(a) Fine-tuning



(b) Layer Freezing

Figure: Comparison of different TL methods on GMWB contracts

## Resilient ML for Fast Risk Evaluation of VAs

### Transfer Learning for Rapid Adaptation of DNN Metamodels

#### Learning Curves: Effect of Similarity

Learning Curves: Effect of Similarity

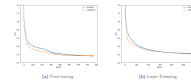


Figure: Comparison of different TL methods on GMWB contracts

These figures compare fine-tuning vs. layer freezing approaches for GMWB contracts. Fine-tuning (left) allows all network parameters to be updated during training. Layer freezing (right) keeps early layers fixed and only updates later layers. For dissimilar contracts like GMMB to GMWB, fine-tuning generally performs better. This suggests that when transferring between different contract types, more flexibility in adaptation is beneficial.

# Conclusions

- ❖ Transfer learning significantly improves metamodeling for VA contracts:
  - ❖ Faster training convergence
  - ❖ Better prediction accuracy
  - ❖ Reduced computational requirements
- ❖ Enables more frequent risk assessments and faster decision-making

## Future Work:

- ❖ Incorporating domain knowledge into transfer process
- ❖ Extension to other insurance and financial products
- ❖ Multi-task learning with more than two tasks

2025-03-21

Resilient ML for Fast Risk Evaluation of VAs

└ Transfer Learning for Rapid Adaptation of DNN Metamodels

└ Conclusions

Our research demonstrates that transfer learning significantly enhances VA contract metamodeling. The benefits include faster convergence, better accuracy with limited data, and reduced computational needs. These improvements enable more frequent and timely risk assessments. This is particularly valuable in volatile markets where rapid decision-making is essential. The approach bridges the gap between traditional actuarial methods and modern machine learning techniques. Several promising directions exist for extending this research. Incorporating actuarial domain knowledge could further improve transfer learning effectiveness. The approach could be extended to other insurance products and financial derivatives. Multi-task learning could be expanded to handle more than two contract types simultaneously. Integration with existing risk management systems would facilitate practical adoption. Ultimately, these techniques could transform how financial institutions manage complex product portfolios.

Conclusions

- Transfer learning significantly improves metamodeling for VA contracts:
  - Faster training convergence
  - Better prediction accuracy
  - Reduced computational requirements
- Enables more frequent risk assessments and faster decision-making

Future Work:

- Incorporating domain knowledge into transfer process
- Extension to other insurance and financial products
- Multi-task learning with more than two tasks



# References

Gordy, M. B. and Juneja, S. (2010). Nested simulation in portfolio risk measurement. *Management Science*, 56(10):1833–1848.

Zhang, K., Liu, G., and Wang, S. (2021). Bootstrap-based budget allocation for nested simulation. *Operations Research*.

2025-03-21

Resilient ML for Fast Risk Evaluation of VAs

└─ Transfer Learning for Rapid Adaptation of DNN Metamodels

└─ References

Gordy, M. B. and Juneja, S. (2010). Nested simulation in portfolio risk measurement. *Management Science*, 56(10):1833–1848.

Zhang, K., Liu, G., and Wang, S. (2021). Bootstrap-based budget allocation for nested simulation. *Operations Research*.