Bias-regularised Neural-Network Metamodelling of Insurance Portfolio Risk

Wei Luo*, Akib Mashrur†, Antonio Robles-Kelly‡ and Gang Li§
School of Information Technology
Deakin University
Geelong, Australia
Email: *wei.luo@deakin.edu.au, †amashrur@deakin.edu.au, ‡antonio.robles-kelly@deakin.edu.au, §gang.li@deakin.edu.au
ORCID: *0000-0002-4711-7543 †0000-0002-4404-7471 ‡0000-0002-2465-5971 §0000-0003-1583-641X

***Abstract*—Deep learning models have attracted considerable attention in metamodelling of financial risks for large insurance portfolios. Those models, however, are generally trained in disregard of the collective nature of the data in the portfolio under study. Consequently, the training procedure often suffers from slow convergence, and the trained model often has poor accuracy. This is particularly evident in the presence of extreme individual contracts. In this paper, we advocate the view that the training of a meta-model for a portfolio should be guided by portfolio-level metrics. In particular, we propose an intuitive loss regulariser that explicitly accounts for the portfolio-level bias. Further, this training regulariser can be easily implemented with the minibatch stochastic gradient descent commonly used in training deep neural networks. Empirical evaluations on both simulated data and a benchmark dataset show that the regulariser yields more stable training, resulting in faster convergence and more reliable portfolio-level risk estimates.**

***Index Terms*—variable annuity metamodelling, expected bias, percentage error**

I. Introduction

Hedging is a widely used strategy for reducing the investment risk where the individual financial products are put in a *portfolio* so as to diversify risk profiles. This helps offset the overall risk arising from price fluctuations of securities, commodities, assets and liabilities. This is particularly important for the insurance industry, where traders often manage large portfolios of contracts which require constant monitoring and re-balancing based on the real-time estimate of the overall risk exposure.

Although the risk of individual contract can be estimated through Monte Carlo simulation, such kinds of processes cannot scale to the whole portfolio. According to [1], it can take more than 100 hours of CPU run-time to valuate a portfolio of 190,000 variable annuity contracts. As a result, *metamodelling* techniques, such as that in [2] are commonly used. These methods are based upon the assumption that the risk of a contract depends on a set of features or variables of the variable annuity contracts. Thus, a metamodel captures the relationship between the features and the Monte Carlo estimates through an interpolation based on the simulation results on a handful of contracts.

Although kriging is traditionally used to estimate the fair market value [3], recently meta-models based on deep neural networks have shown great promise [4], [5]. This is mainly because deep networks have the advantage of exhibiting a strong interpolation capability and, if properly trained, can achieve very high accuracy. On the other hand, training a deep network often requires a large training set. This is one of the main drawbacks of deep networks for portfolio modelling since a large training set is rarely available in metamodelling due to the high computational cost of the Monte Carlo runs. Moreover, training a deep network can be difficult, especially when outliers exist in the training set.

In this paper, we note that a real need exists to improve the training process so as to make deep learning more applicable for metamodelling of insurance portfolios. Moreover, here we argue that treating metamodelling as a predictive modelling task actually makes the problem harder than it should be. Thus, instead, we advocate that training a meta-model for a portfolio should be guided by portfolio-level metrics. In particular, we propose an intuitive loss regulariser that explicitly accounts for the portfolio-level bias.

At the centre of the proposed method, is the observation that metamodeling can be viewed as a multitask learning problem: a primary task of estimating the portfolio aggregate and an auxiliary task of optimising individual prediction which in general consistent with the primary task.

In this paper, we make the following contributions:

- We point out one major issue with the current practice in metamodelling: the models are trained but ignoring the collective nature of a portfolio, and hence, tend to disregard valuable information that can simplify the solution of the problem in hand.
- We propose a new loss function that directly targets the portfolio-level quantity of interest, be it fair market value (FMV) or the Greek value.
- We evaluated the new loss function on both a simulated data set and a benchmark variable annuity (VA) data set. Our results demonstrate a significant gain in the convergence speed and the accuracy of the estimates yield by the model. To our knowledge, this translates into the first deep-learning meta-model that achieves consistently high performance on a *very small training set* in this benchmark problem.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

II. Background

In this section, we present the background and notations used throughout the paper. Here we also elaborate upon the use of the mean-squared error as a surrogate metric for the loss function and its connection with the prediction bias.

To commence, let

$$\mathcal{P} = \{c_i, i = 1, \dots, N\}$$

be a portfolio of insurance contracts. Here, we assume that each contract c_i has a quantity of interest y_i. Throughout the paper, y_i denotes a partial dollar Delta [1], which is a risk measurement calculated making use of Monte Carlo simulation (See [2] for more details).

In hedging, we are mostly interested in the *portfolio delta*

$$\sum_{i=1}^{N} y_i.$$

If a portfolio Delta is close to zero, it is said to be *delta neutral*. This is important since insurance companies monitor and re-balance their product portfolios daily according to the estimated Delta value.

A. Metamodelling and the Mean-squared Error

The calculation of the contract Delta value is computationally expensive, involving complex Monte Carlo simulation often taking long CPU run times. For a large portfolio, calculating all contract Delta y_i's is, hence, computationally infeasible. If, however, each contract can be described by a set of features $\boldsymbol{x}_i$, then we can build a model to infer y_i from $\boldsymbol{x}_i$. This rationale is the essence of metamodelling. In metamodelling, we first obtain a small subset

$$\mathcal{T} = \{c_{i_k}, k = 1, \dots, K\}$$

of contracts in the portfolio $\mathcal{P}$. From $\mathcal{T}$, the corresponding Delta values

$$\{y_{i_k}, k = 1, \dots, K\}$$

are calculated via Monte Carlo simulation. Finally, a regression model $\hat{f}$ is trained using the small training set

$$\{(\boldsymbol{x}_{i_k}, y_{i_k}), k = 1, \dots, K\}$$

and their available Delta values so as to estimate the portfolio Delta

$$\sum_{i=1}^{N} \hat{y}_i,$$

where $\hat{y}_i = \hat{f}(\boldsymbol{x}_i)$.

In metamodelling, since the problem is cast as regression one, the mean squared error (MSE)

$$\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

is the most commonly used loss function in metamodelling [2]. This is the case for methods based on neural networks such as that in [6]. It is important to note that minimising the MSE requires the model to accurately predict the risk of each individual contract at the same time. Although accurate prediction for each individual contract Delta automatically implies accurate prediction of the overall portfolio, the accuracy of the individual predictions is often hard to guarantee in practice.

B. Percentage Error and Prediction Bias

Recall that, for portfolio monitoring and re-balancing, the primary concern is the overall portfolio delta. Hence the most commonly used evaluation metric for meta-models is the *percentage error* given by:

$$\mathrm{PE} = \frac{\sum_{i=1}^{N}\hat{y}_i - \sum_{i=1}^{N} y_i}{\sum_{i=1}^{N} y_i},$$

which measures the aggregated accuracy.

From the regression modelling point of view, the PE can be viewed as a normalised estimate of the model's prediction bias, that is, the expected value $E\left[\hat{y} - y\right]$ for an unseen data point $(\boldsymbol{x}, y)$. It is worth noting that this bias, as yield by the fitted model, is often different from the true expected bias for a family of unfitted models in the well-known bias-variance decomposition of the expected MSE for the test set [7]:

$$\begin{aligned} E\left[(\hat{y} - y)^2\right] =& \text{Variance} + \text{Bias}^2 \\ &+ \text{Intrinsic Prediction Uncertainty}. \end{aligned} \tag{1}$$

Strictly speaking, Bias in Equation (1) is determined by the capacity of the model family. In the context of deep learning, this is computed through marginalisation over the training samples and the optimisation. In portfolio metamodelling, this is mainly induced by the randomness of the contract selection process [8]. In other words, from a single training set, we cannot estimate Bias. However, as mini-batched gradient descent is commonly used in training neural networks, we can view $\hat{y} - y$ computed on mini-batches as a rough estimate of the true test-set bias.

III. Bias-regularised metamodelling

As shown above, the current practice treats metamodelling as a standard regression problem minimising the mean squared error. As mentioned earlier, this is particularly problematic when the metamodel takes the form of deep neural networks, as such models often require a large training data set.

To tackle the problem of the small sample size, we follow Vapnik [9] by noting that the restricted amount of information available from the small subset of contract features and Delta values may be sufficient to solve the problem directly rather than tackling a more general problem as an intermediate step. As Vapnik argues, it is often possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem. This is particularly pertinent towards existing metamodelling practice, whereby these approaches often try to first solve a more general intermediate problem, predicting the fair value or

[1]Here we consider the dollar Delta as the value of the contract so as to hedge one dollar of liability.

Greeks of the individual contracts as a means to the prediction of the portfolio-level value.

Note that, as percentage error (PE) is the primary metric for metamodelling, it's somewhat natural to attempt to directly optimise the absolute PE or its squared amount as an aim of computation in training of the model. Doing this directly, however, is not a straightforward task. This is mainly because, given a training set, there may be too many solutions that minimise the PE loss (for instance, the empirical risk). This is in contrast with the mean-squared error (MSE), where we can often find a unique global minimum, for example, that given by the least-squares solution for the linear regression.

To show this, we now provide a theoretical foundation in a linear regression setting using the following Theorem:

Theorem 1. *Let $\{(\boldsymbol{x}_i, y_i) : 1 \leq i \leq N\}$ be a training set with a non-constant least-squares solution. Then there is more than one model that minimises the PE squared on the training set:*

$$\left(\frac{\sum_{i=1}^{N} \hat{y}_i - \sum_{i=1}^{N} y_i}{\sum_{i=1}^{N} y_i}\right)^2.$$

Moreover, the empirical PE is 0 for these models.

Proof. This directly follows Lemmas 1 and 2 below. □

Lemma 1. *Let $\{(\boldsymbol{x}_i, y_i) : 1 \leq i \leq N\}$ be a training set. The linear model $y = \boldsymbol{\beta}\cdot\boldsymbol{x}+b$ that minimises $\sum_{i=1}^{N}(\boldsymbol{\beta}\cdot\boldsymbol{x_i}+b-y_i)^2$ also minimises*

$$\left(\sum_{i=1}^{N}(\boldsymbol{\beta}\cdot\boldsymbol{x_i}+b) - \sum_{i=1}^{N} y_i\right)^2.$$

Proof. This follows from the well-known result that the sum of residuals from the least-square regression is always 0. More specifically, as $\boldsymbol{\beta}$ and b minimise $\sum_{i=1}^{N}(\boldsymbol{\beta}\cdot\boldsymbol{x_i}+b-y_i)^2$, we have

$$\frac{\partial \sum_{i=1}^{N}(\boldsymbol{\beta}\cdot\boldsymbol{x_i}+b-y_i)^2}{\partial b} = 2\cdot\sum_{i=1}^{N}(\boldsymbol{\beta}\cdot\boldsymbol{x_i}+b-y_i) = 0.$$

which directly implies

$$\sum_{i=1}^{N}(\boldsymbol{\beta}\cdot\boldsymbol{x_i}+b) - \sum_{i=1}^{N} y_i = 0.$$

□

Lemma 2. *Let $\{(\boldsymbol{x}_i, y_i) : 1 \leq i \leq N\}$ be a training set. The constant model $\hat{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$ minimises*

$$\left(\sum_{i=1}^{N}\hat{y}_i - \sum_{i=1}^{N} y_i\right)^2.$$

Proof. This follows from

$$\sum_{i=1}^{N}\hat{y}_i - \sum_{i=1}^{N} y_i = N\left(\frac{1}{N}\sum_{i=1}^{N} y_i\right) - \sum_{i=1}^{N} y_i = 0.$$

□

As a result, the PE-based loss favours a constant model. This has the consequence that the model fitted using the PE is unlikely to generalise well on *not observed* data, especially when the portfolio Delta, *i.e.* the mean value of the response y, deviates from the data set used for training. Furthermore, this may be even more problematic when stochastic gradient descent (SGD) or minibatch SGD is used in training, which, in turn, may result in a lack of converge or very slow training.

A. Bias-regularised MSE loss

To tackle these drawbacks, we note that metamodeling can be viewed as a multitask learning problem. The primary task is to minimise the PE. An auxiliary task of optimising individual prediction is not identical but is in general consistent with the primary task. This motivates us to propose the following Bias-regularised MSE (**BRMSE**) loss function

$$\mathrm{BRMSE} = \mathrm{MSE} + \lambda\cdot\mathrm{EB}^2, \qquad (2)$$

where λ is a constant that measures the influence of the second term on the left-side of the equation which measures the expected prediction bias and corresponds to the mean square root of the equivalence term in Lemma 2. We call this the *expected bias*:

$$\mathrm{EB} = \frac{1}{N}\left(\sum_{i=1}^{N}\hat{y}_i - \sum_{i=1}^{N} y_i\right).$$

In the loss function above, the use of the mean-squared error MSE ensures that the model fits individual training instances and, hence, has a well-defined minimum and can generalise to other contracts in a portfolio. The additional term EB^2, which we name **EBS**, provides a collective measurement that aims at ensuring the model is optimising the portfolio PE. Although the global minimum for MSE also minimises EBS, as we shall see in Section IV, the inclusion of EB leads to faster model training and a more accurate PE estimate. Moreover, the regularisation term improves training in the presence of extreme outliers, which are very common in a large portfolio of insurance contracts.

Note that EB^2 is likely to be more stable than MSE during the training. Therefore it is likely to regularise learning (e.g, avoiding local minimums), similar to how momentum is used in many learning problems.

IV. Empirical Evaluation

In this section, we evaluate the new loss function defined in Eq (2) on typical metamodelling tasks. We evaluate the effectiveness of the BRMSE loss function first on a simulated data set in Section IV-A. We then turn our attention to a benchmark data set for metamodelling in Section IV-B.

In view of the prohibitive cost of obtaining training examples in real-life applications, the experiments focus primarily on the setting where the training set is very small (fewer than 500 training examples). In such small training sets, existing neural-network meta-models (e.g., [6]) do not easily converge. And when they converge, the performance is often poor. Due to such difficulty in producing consistent baseline results, we

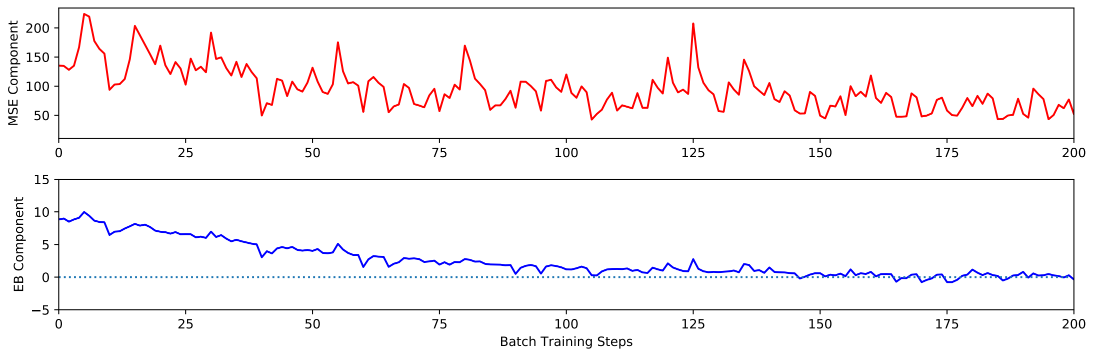

Fig. 1. Comparison of the convergence of Expected Bias Squared (EBS) and MSE in mini-batched gradient descent. The MSE component of the loss has a high variance across different mini-batches. In contrast, the EBS component is smoother and converges faster during training.

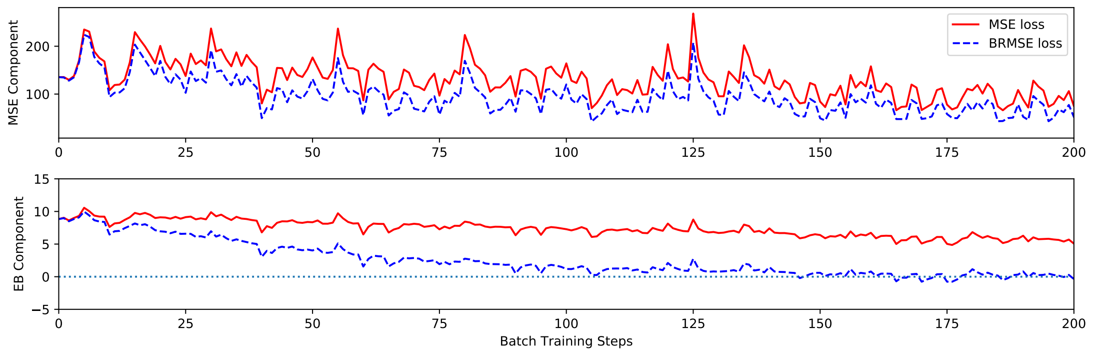

Fig. 2. The effect of adding the EBS loss. Explicitly optimising the expected bias leads to faster reduction in both the EBS and the MSE.

do not compare explicitly with some existing meta-modeling models. Instead, we focus on measuring the performance gain of the new loss function in controlled settings. With that said, the proposed models clearly achieved very competitive performance, even in comparison with some more sophisticated models involving transfer learning [4].

A. Bias Regularisation Evaluation

We first simulate a regression data set $(\boldsymbol{X}, \boldsymbol{y})$ of 10,000 instances so as to better understand how the bias regularisation improves with the training process. To this end, we have used a vector $\boldsymbol{X}$ comprised of five 10-feature isotropic Gaussian blobs generated using the `make_blobs` function from `sklearn` Python library.

Recall that `make_blobs` generates both the features and the cluster labels. For the purposes of our evaluation, we have only kept the feature vectors. This gives us clustered features. This was done following the notion that, in real-world settings, the insurance portfolios tend to form clusters, reflecting the discrete nature of product and client types.

Thus, making use of the vectors $\boldsymbol{X}$, we generate the target vector $\boldsymbol{y}$ using the following function adapted from [10]:

$$y = 10\sin(\pi x_0 x_1) + 20\,(x_2 - 0.5)^2 + 10x_3 + 5x_4 + 10^4\epsilon,$$

where ϵ is the standard Gaussian noise. Since in metamodelling the training set is often small, we split the 10,000 instances into a training set of 5%, a validation set of 5%, and a testing set of 90%.

Due to the size of the training set and in order to provide a better baseline for comparison of the effects of our regularisation term, we consider only standard neural networks with a relatively small number of dense layers. For the synthetic data, we use a dense network with two hidden layers and ReLU activations. The first hidden layer consists of 50 nodes, followed by another dense hidden layer consisting of 20 nodes. The network was trained using both the standard MSE loss and the proposed BRMSE loss. The training was effected making use of stochastic gradient descent with the minibatch size set to 100.

For purposes of evaluating the effect of the regulariser, we consider two performance measures. These are the MSE and the expected bias (EB). Both of these are computed within the training batch. Figure 2 shows how these training errors change during the course of training as a function of mini-batch index. From the figure we can appreciate the following:

1) The EB (and hence the EBS) is more stable during the training compared to the MSE. This can be seen in both performance measures, with the trace as a function of batch training steps.
2) An improved EB component convergence when the BRMSE loss is used. Moreover, the MSE also converges more quickly, as shown in the comparison of the red line

and the blue dashed line. This is particularly evident in the first 100 training batches.

The additional EBS component not only reduces the bias, but also stabilises the overall loss, and leads to faster training convergence. This is consistent with the prior discussions which suggest that optimising the MSE alone may be inefficient in minimising the bias and, hence, the PE for the whole portfolio.

We now turn our attention to the validation error. To this end, in Figure 3 we show the plots for the validation error, as evaluated on the validation set, as a function of training epochs for the network when both loss functions are used. The traces in the plots have been generated using the 5% validation set where the error is given by the average over the batches at the end of each epoch, yielding smooth curves for both choices of the loss function.

Again, in a consistent trend with respect to that shown in Figure 2, we can see that both the MSE and the EB are reduced at a faster speed on the validation set. Moreover, as expected, the convergence improvement is even more pronounced for the PE.

Sensitivity of the regularisation weight λ : In Equation (2), λ determines how heavy the bias regularisation is applied. Figure 4 shows the effect of λ on the final PE on the simulated dataset. Here a λ value between 0.1 and 0.6 seems to work best. The optimal range of λ may depend on the dataset.

B. Application to variable-annuity portfolio metamodelling

We now turn our attention to the application of our BRMSE loss to the variable annuity VA benchmark dataset provided by Gan [11]. This dataset has been used in several VA metamodelling studies [6], [12]. The dataset consists of 38,000 variable annuity contracts described by 44 features. These features are mainly related to the investment funds and the policyholders. For more details about the dataset, we suggest the interested reader see [2]. The response variable is the partial first-order Greeks given by the Delta. The dataset contains the Delta evaluated monthly for a period of 30 years under 100 market scenarios. Here, we have used the Delta for the first month of the first year, *i.e.* the initial Delta, for one of the fixed market scenarios.

In most metamodelling settings, a very small proportion of the portfolio is used to calculate to different measures of risk, *i.e.* Greeks, via Monte Carlo Simulation. Once these Greeks are in hand, a regression model is fitted to estimate the full-portfolio. Note that, in [13] a sophisticated setup to split the dataset into training, testing and validation was employed using k-means clustering. Here, however, we have opted to randomly use 1% sample for training, 1% sample for validation and the full portfolio for testing. This yields only 380 contracts for training and another 380 for validation.

For our experiments using the benchmark VA dataset, we have added a hidden layer with the ReLU activation to the network presented in the previous section. This makes the network relatively more complex, with 200 nodes in the first hidden layer, 100 nodes in the following layer and 20 nodes in the final hidden layer. Each of the first two hidden layers is penalized with L2 regularization and are followed by dropout layers to avoid over-fitting.

In training, 1000 epochs were run. To measure the convergence, we recorded the number of epochs before reaching the target validation-set PE of 0.1. The final PE is evaluated on the whole portfolio, using the best performing model within the 1000 epochs.

To demonstrate the generalisability of the BRMSE loss, we examine various combinations of the optimisers and learning rate schedules. For the optimisers, we have tried several alternatives. These are *vanilla* Stochastic Gradient Descent (SGD), AdaGrad, RMSProp and Adam. For learning rate schedules, we have considered both a fixed learning rate and the Cyclic Learning Rate (CLR) schedule described in [14]. For the CLR, the learning rate loops from 0.001 to 0.005 in 100 steps. In addition to implementing CLR with the *vanilla* stochastic gradient descent optimiser, we have also tested the suitability of our loss function under an experimental setting combining both CLR and the adaptive gradient optimiser (Adagrad). The Cyclic Learning Rate was not used with RMSProp due to the incompatibility described by the authors in [14].

For our experiments, we considered two of the mainstream error metrics used in the literature for the evaluation of the trained meta-model. First and foremost, we calculated the portfolio percentage error:

$$\mathrm{PE} = \frac{\sum_{c_i \in \mathcal{P}} \hat{y}_i - \sum_{c_i \in \mathcal{P}} y_i}{\sum_{c_i \in \mathcal{P}} y_i}.$$

Additionally, we also calculate the regression R^2:

$$R^2 = 1 - \frac{\sum_{c_i \in \mathcal{P}} \left(\hat{y}_i - y_i\right)^2}{\sum_{c_i \in \mathcal{P}} \left(y_i - \mu\right)^2},$$

where

$$\mu = \frac{1}{n} \sum_{c_i \in \mathcal{P}} y_i.$$

Note that both PE and R^2, are evaluated on the full portfolio, and, to gauge the convergence speed, we also track the number of training epochs until the target absolute PE is reached on the validation data. Also, we have fixed the learning rate for SGD and, in the case of the adaptive-learning-rate optimisers, the global learning rate. We have done this so as to reduce the effect of a changing learning-rate schedule in the experiments shown here.

In Table I, we show the results for SGD and Adagrad. In the table, we show the PE, R^2 and the number of epochs to reach a validation PE of less than or equal to 0.1. For all the optimisers and evaluation metrics, we have compared our BRMSE loss against the MSE loss. Note that our loss consistently requires fewer training epochs to reach the target validation PE of 0.1. Moreover, our BRMSE loss provides a margin of advantage in the final PE and R^2 as tested on the full portfolio as compared to that based on the MSE.

Finally, we focus on the results yielded by our loss and the alternative when the Cyclic Learning Rate (CLR) is used on all compatible optimisers. Recall that the CLR is

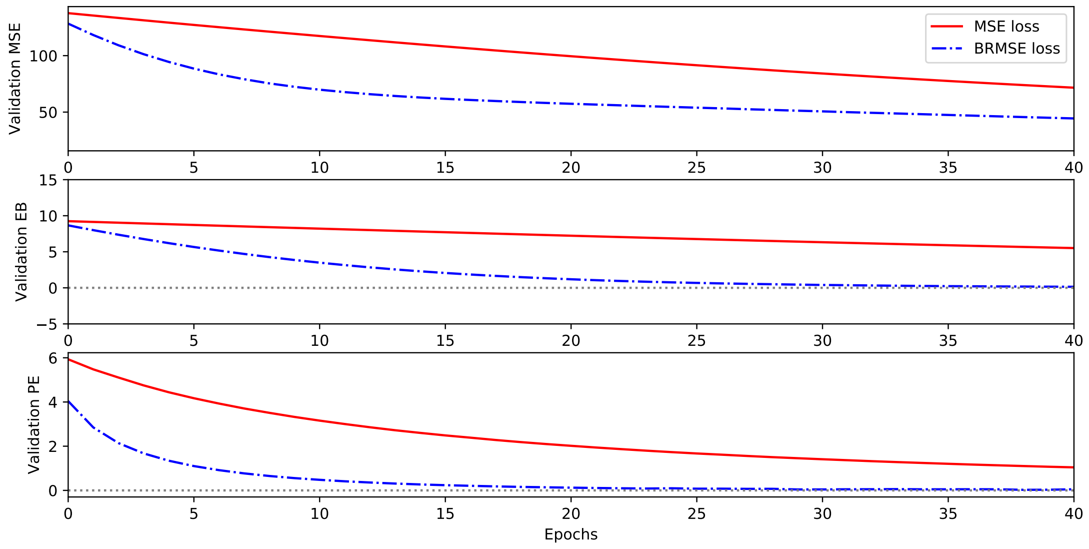

Fig. 3. The proposed loss converges both squared error and expected bias faster than MSE loss, which results in faster and more stable PE optimization

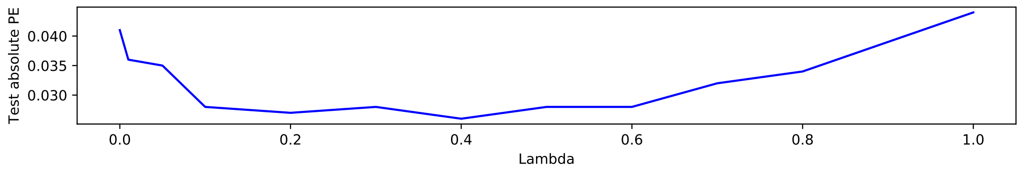

Fig. 4. The effect of λ on the final portfolio PE (evaluated on the whole dataset) for the simulated regression data. In training, the Cyclic Learning Rate was used. In the figure, $\lambda = 0$ indicates the standard MSE loss. Clearly, the absolute PE dropped once λ becomes nonzero. After passing a relatively stable region from 0.1 to 0.6, over-regularisation slowly took effect, causing poorer fit and bigger absolute PE values.

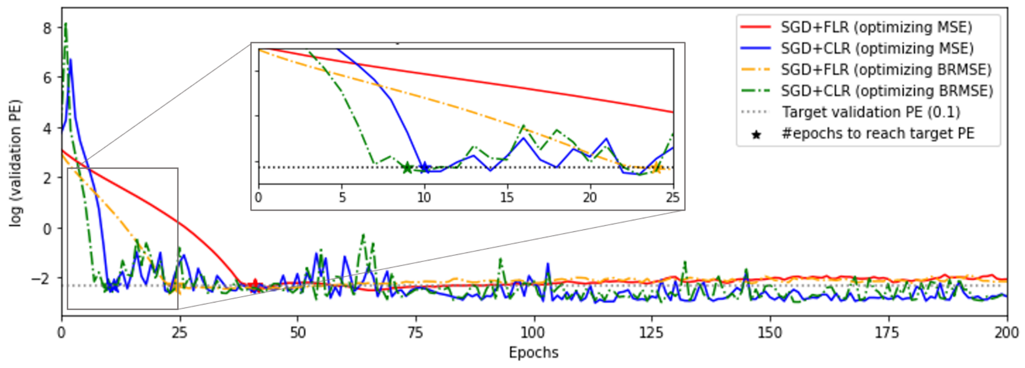

Fig. 5. Validation PE convergence trained with MSE and BRMSE, with both a fixed learning rate (FLR) and a cyclic learning rate (CLR) schedule. The stars (in both the original plot and the inset) indicate the first time the target absolute PE of 0.1 was reached.

generally believed to be incompatible with several optimisers with adaptive learning rate, including RMSProp and Adam. Nonetheless, as shown in [14], for the compatible optimisers, applying CLR can lead to much faster convergence and a more accurate final model. For the CLR, we have used a standard triangular schedule with learning rate fluctuating over a full cycle every 100 training steps. This allows for the training process to prevent falling into saddle point plateaus and improves overall performance. As shown in figure 5, the proposed BRMSE loss function leads to faster convergence and a more accurate estimate, with or without CLR.

In Table II we show the results yielded by CLR in a manner akin to that in Table I. Moreover, as compared with the results in Table I, we can see that the PE results are greatly improved when CLR is used. This is somewhat expected and consistent across both loss functions. That said, again, the proposed BRMSE leads to a better estimation of the portfolio PE.

TABLE I
PE PERFORMANCE ON VA DATASET WITH A FIXED GLOBAL LEARNING RATE.

Optimizer	**SGD**			**Adagrad**		
	PE[†]	R^2	#epochs	PE	R^2	#epochs[*]
training loss function						
MSE	0.231	0.799	42	0.184	0.815	5
BRMSE	**0.211**	**0.811**	**25**	**0.182**	0.815	**4**
Optimizer	**RMSprop**			**Adam**		
	PE	R^2	#epochs	PE	R^2	#epochs
training loss function						
MSE	0.283	0.746	5	0.234	0.787	5
BRMSE	**0.203**	**0.759**	**2**	**0.190**	**0.798**	**4**

[†] PE evaluated on the whole portfolio.
[*] The number of epochs before the target validation $|\text{PE}| < 0.1$ is reached.

TABLE II
PE PERFORMANCE ON VA DATASET FOR DIFFERENT OPTIMIZERS WITH THE CYCLICAL LEARNING RATE.

Optimizer	**SGD**			**Adagrad**		
	PE	R^2	#epochs	PE	R^2	#epochs
training loss function						
MSE	0.071	0.797	11	0.052	0.743	20
BRMSE	**0.051**	**0.803**	**10**	**0.029**	**0.811**	**6**

V. CONCLUSIONS

In this paper, we have proposed a new approach to train a neural network for metamodelling of a large insurance portfolio. Our method is based upon a bias-regularised loss function which employs the mean-squared error and a regulariser given by the expected bias. In this context, the mean-squared error enforces a good fit towards individual contracts whereas the expected bias provides a measure regarding the portfolio percentage error. We have evaluated our loss function on a simulated evaluation dataset and a variable annuity benchmark. The simulated data has been used to assess the effect of the regulariser on both the training convergence and performance of the trained network. We have used the benchmark to evaluate our loss across different optimisers and learning rate strategies, including the cyclical learning rate. We have also compared our approach to that often used elsewhere based upon the mean-squared error alone. In our experiments, our method yielded a margin of improvement in both convergence and performance. Additional experiments have been planned to further validate our results with multiple independent runs and additional baseline models such as tree-based machine learning models [15].

To our knowledge, this is the first work applying collective loss on portfolio metamodelling. Unlike previous research, our experiments focus on the setting of very small training sets (fewer than 500 training examples), showing that highly accurate PE estimate can be achieved even with a small training set. A natural next step is to better understand the loss landscape of this new type of loss functions, probably through recently proposed visualisation tools (e.g., [16]). Another line of future work is to integrate the new loss function in the transfer-learning metamodelling approach (e.g., [4], [5]).

REFERENCES

[1] G. Gan and E. A. Valdez, "Valuation of large variable annuity portfolios: Monte carlo simulation and synthetic datasets," *Dependence Modeling*, vol. 5, pp. 354–374, 2017.
[2] ——, *Metamodeling for Variable Annuities*. CRC, 2019.
[3] G. Gan, "Application of data clustering and machine learning in variable annuity valuation," *Insurance: Mathematics and Economics*, vol. 53, no. 3, pp. 795–801, 2013.
[4] X. Cheng, W. Luo, G. Gan, and G. Li, "Deep neighbor embedding for evaluation of large portfolios of variable annuities," in *KSEM 2019*, 2019, pp. 472–480.
[5] ——, "Fast valuation of large portfolios of variable annuities via transfer learning," in *PRICAI 2019*, 2019, pp. 716–728.
[6] S. A. Hejazi and K. R. Jackson, "A neural network approach to efficient valuation of large portfolios of variable annuities," *Insurance: Math. and Eco.*, vol. 70, pp. 169–181, 2016.
[7] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, no. 1, pp. 1–58, Jan. 1992.
[8] B. Neal *et al.*, "A modern take on the bias-variance tradeoff in neural networks," *arXiv preprint arXiv:1810.08591*, 2018.
[9] V. N. Vapnik, *Statistical learning theory*. Wiley, 1998.
[10] J. Friedman *et al.*, "Multivariate adaptive regression splines," *Ann. Stat.*, vol. 19, pp. 1–67, 1991.
[11] G. Gan and E. A. Valdez, "Nested stochastic valuation of large variable annuity portfolios: Monte carlo simulation and synthetic datasets," *Data*, vol. 3, no. 3, p. 31, 2018.
[12] W. Xu *et al.*, "Moment matching machine learning methods for risk management of large variable annuity portfolios," *JEDC*, vol. 87, pp. 1–20, 2018.
[13] G. Gan and J. Huang, "A data mining framework for valuing large portfolios of variable annuities," in *SIGKDD*, 2017, pp. 1467–1475.
[14] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE Winter Conference on Applications of Computer Vision*. IEEE, 2017, pp. 464–472.

[15] Z. Quan, G. Gan, and E. A. Valdez, “Tree-based models for the efficient valuation of large variable annuity portfolios,” *Available at SSRN 3247100*, 2018.

[16] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, “Visualizing the loss landscape of neural nets,” in *Neural Information Processing Systems*, 2018, pp. 6389–6399.