Discussion of "Enhanced Bayesian Neural Networks for Macroeconomics and Finance"

by Hauzenberger, N., Huber, F., Klieber, K., and Marcellino, M.

Carlos Montes-Galdón

European Central Bank, DG-Economics, Forecasting and Policy Modelling
Prepared for the 12th ECB Conference on Forecasting Techniques

Disclaimer: This discussion reflects my own views and not those of the ECB or the Eurosystem

I would like to thank **Chat-GPT 4** for providing excellent research assistance

This paper

- Build an algorithm to estimate models with generic and unknown nonlinearities and time variation for (possibly large sets of) macro and financial variables, with SV - Bayesian Neural Network
- Apply the algorithm to several time series models - the algorithm seems to perform very well in out-of-sample forecasting

This paper

- Build an algorithm to estimate models with generic and unknown nonlinearities and time variation for (possibly large sets of) macro and financial variables, with SV - Bayesian Neural Network
- Apply the algorithm to several time series models - the algorithm seems to perform very well in out-of-sample forecasting
- **Overall, I like this paper a lot - very well developed algorithm , good applications, and very interesting results...**

This paper

- Build an algorithm to estimate models with generic and unknown nonlinearities and time variation for (possibly large sets of) macro and financial variables, with SV - Bayesian Neural Network
- Apply the algorithm to several time series models - the algorithm seems to perform very well in out-of-sample forecasting
- **Overall, I like this paper a lot - very well developed algorithm , good applications, and very interesting results...**
- *... but I have some doubts about the performance compared to standard Bayesian Neural Network algorithms and the differences*
  - The authors claim that their algorithm outperforms quite dramatically standard BNN estimated by backpropagation - but I think the estimations are not comparable

A simple neural network

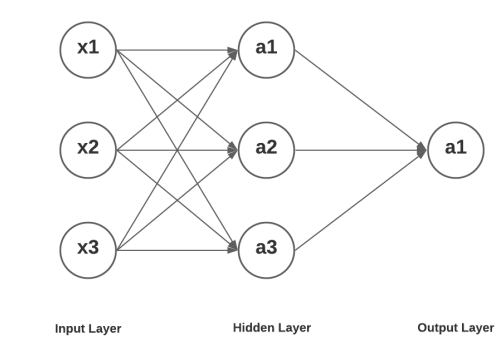
- Hidden layer

$$a_1 = a(b_1 + w_{11}x_1 + ... + w_{13}x_3)$$
$$a_2 = a(b_2 + w_{21}x_1 + ... + w_{23}x_3)$$
$$a_3 = a(b_3 + w_{31}x_1 + ... + w_{33}x_3)$$

- Output layer

$$a_1 = a(b_o + w_{o1}a_1 + ... + w_{o3}a_3)$$

- If there is no Hidden Layer and $a(...) = linear$, then we are back to a linear regression
- The goal is to estimate the *weights* (w_{ij}) and the *biases* (b_i)
  - Minimize a loss function - usually MSE, for example $(y - a_1(x))^2$



Bayesian Neural Networks

"Standard" BNN

- Select a prior over $\theta = w, b$
- Pick a variational approximation to the posterior distribution, $q_\lambda(\theta)$ where λ are the parameters that define the variational approximation
- minimize the Kullback-Leibler divergence between $q_\lambda(\theta)$ and $p(\theta|y)$, which is not tractable - that is, find the parameters of the variational approximation

This paper

- Select a prior over $\theta = w, b$
- As in standard Bayesian analysis, find the posterior distribution, $p(\theta|y)$ without relying on variational approximations
- As the posterior distribution is not tractable, they develop a MCMC sampler to find the posterior distribution as they can find tractable conditional posterior distributions

A simulation example

- Let me simulate highly non-linear data and use a standard BNN to learn the non-linearities,

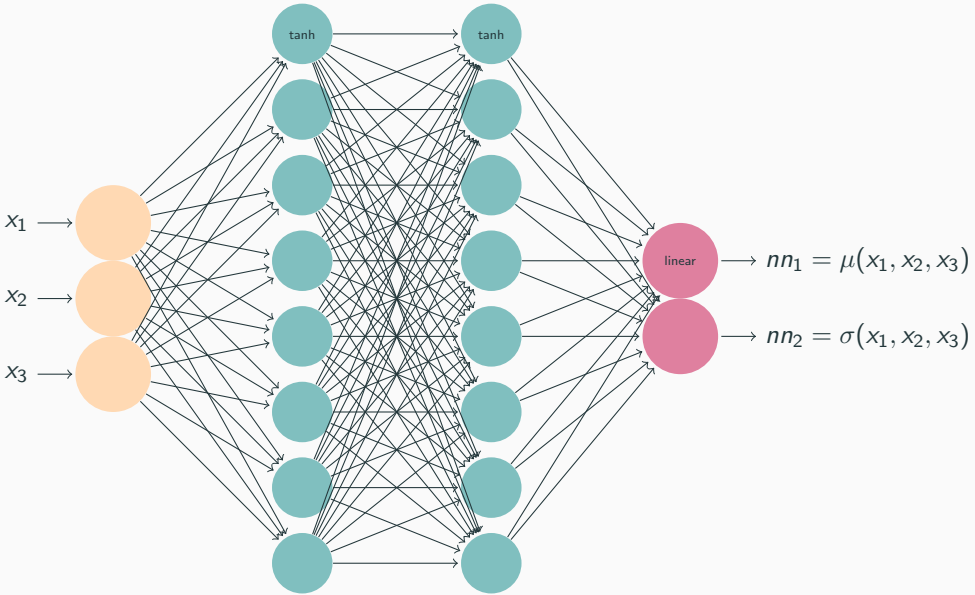
$$y_t = f(x_t) + \sigma(x_t)\epsilon_t$$

- As prior for the weights and biases use $\mathcal{N}(0, 1)$ and as variational posterior $\mathcal{N}(\lambda_{i,1}, \lambda_{i,2})$.
- The neural network (nn) has 2 outputs, which are going to be the mean of the model and the standard deviation, so that I minimize the KL divergence between the variational posterior and the (exact) posterior,

$$\lambda^* = \text{argmin}\{KL[q_\lambda(\theta)||p(\theta)] - E_{\theta\sim q_\lambda}[\log(p(y|\theta)]\}$$

  in my case $p(y|\theta) \equiv \mathcal{N}(nn_1, nn_2)$
- In order to simulate the posterior densities, I use both *epistemic* uncertainty (coming from uncertainty in the posterior distribution of the weights) and *aleatoric* uncertainty, coming from the residuals, ϵ

Architecture for the neural network



Estimated results

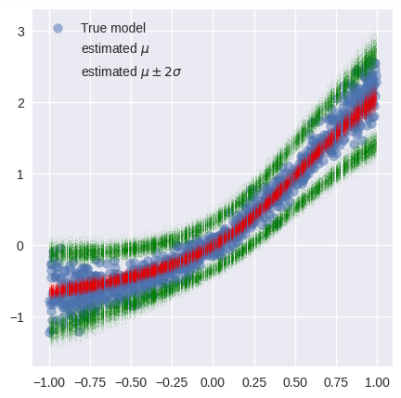


Figure 1: x_1

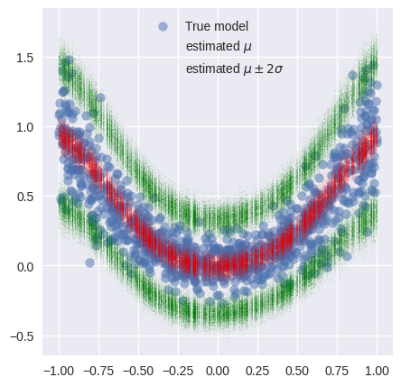


Figure 2: x_2

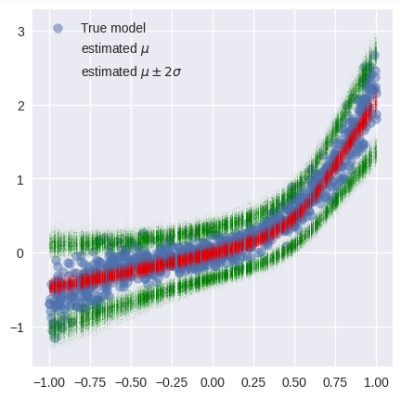


Figure 3: x_3

Comparison with BNN in the paper

- In the paper, when comparing LPL, a standard BNN performs much worse than the proposed algorithm - "the dismal performance of BNN-BP is driven by too narrow predictive bounds"...

Comparison with BNN in the paper

- In the paper, when comparing LPL, a standard BNN performs much worse than the proposed algorithm - "the dismal performance of BNN-BP is driven by too narrow predictive bounds"...
  - ... but the authors use a MSE loss function to estimate the BNN-BP - no aleatoric uncertainty, which they have in their algorithm, together with SV.
  - In fact, when looking at table B.3 which compares the forecast performance according to the RMSE, the performance is not so different

Table B.3: Macro A. Forecast performance across 252 hold-out observations (one-month-ahead).

Covariates	Model				
	BAICT	BNN	BNN-BP	BNN-NS	Linear model
			Inflation		
AR(1)	1.11***	1.05	**1.03**	1.05	1.05
	(-0.22***)	(-0.01)	(-0.19***)	**(-0.01)**	(-0.03)
Large	0.97	0.94***	1.03	**0.94*** **	0.94***
	(-0.03)	**(0.09***)**	(-0.12***)	(0.08***)	(0.08***)
Medium	1.04	0.97	1.04**	**0.97***	0.98
	(-0.10*)	**(0.06***)**	(-0.12***)	(0.04*)	(0.06***)
PCA	1.07**	**1.00**	1.04*	1.01	1.16
	(-0.20***)	**(0.00)**	(-0.11***)	(-0.04***)	(-1.45)
			Industrial production		
AR(1)	**0.89**	0.91	0.92	0.91	0.91
	(-0.12)	**(0.09*)**	(-0.81)	(0.06)	(0.03)
Large	**0.88**	0.97***	0.93	1.02**	0.98***
	(0.08)	(0.14***)	(-0.41*)	**(0.17***)**	(0.12***)
Medium	**0.88**	0.97*	0.93	0.97**	0.99**
	(0.03)	(0.10***)	(-0.41*)	**(0.14***)**	(0.06***)
PCA	**0.89**	1.00	0.94	1.01	1.75
	(-0.01)	(0.05**)	(-0.46)	**(0.07*)**	(-1.33)
			Employment		
AR(1)	1.14**	**1.01**	1.01	1.02	1.02
	(-0.67)	**(0.11)**	(-0.96)	(0.11)	(-0.07*)
Large	1.04	1.00	1.01	**1.00**	1.01
	(0.11)	**(0.14***)**	(-0.87)	(0.14)	(0.10*)
Medium	1.05	**0.99**	1.01	0.99	1.00
	(-0.28)	**(0.14)**	(-0.97)	(0.00)	(0.04)
PCA	1.03	**0.99**	1.01	1.00	3.50
	(-0.19*)	**(0.07)**	(-0.59)	(-0.01)	(-1.88)

Note: The table shows root mean squared errors (RMSEs), and average log predictive likelihoods (LPLs) in parentheses, relative to the linear benchmark. In bold we mark the best performing model for each case. The grey shaded area gives the actual RMSE and LPL scores of our benchmark (linear model). Asterisks indicate statistical significance by means of the Diebold and Mariano (1995) test for each model relative to the benchmark at the 1% (***), 5% (**) and 10% (*) significance levels. Results are averaged across the hold-out.

Estimated results with MSE loss

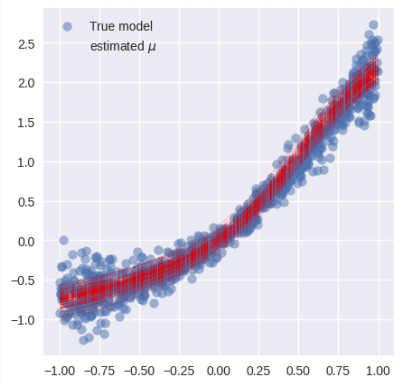


Figure 4: x_1

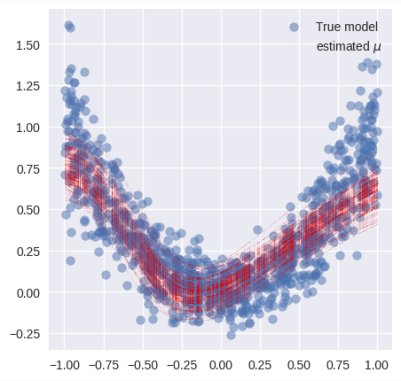


Figure 5: x_2

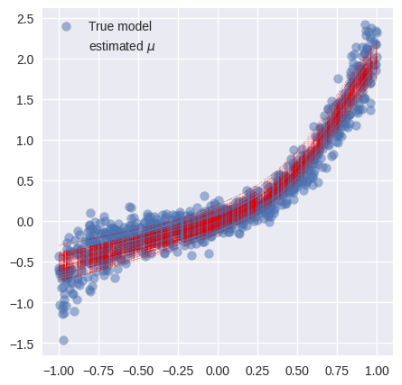
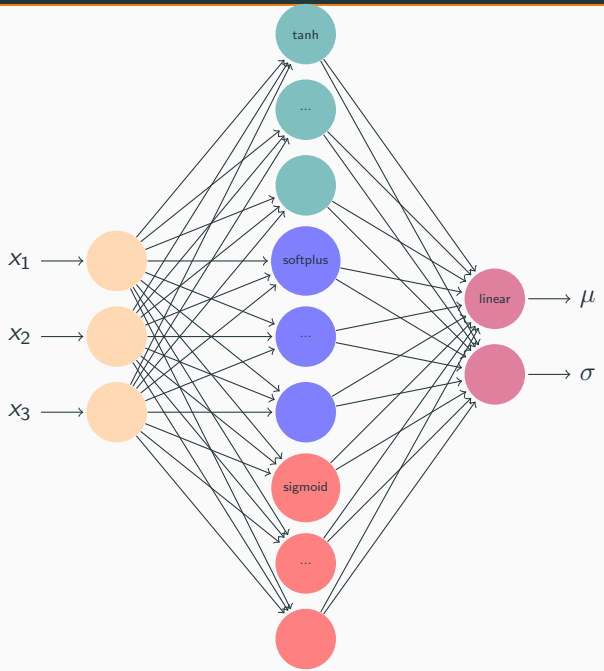


Figure 6: x_3

Alternative architecture similar to the paper



- Use one layer, with independent activation functions, that then are combined to produce the two outputs
- Use prior shrinkage - Laplace distribution

Estimated BNN with alternative architecture

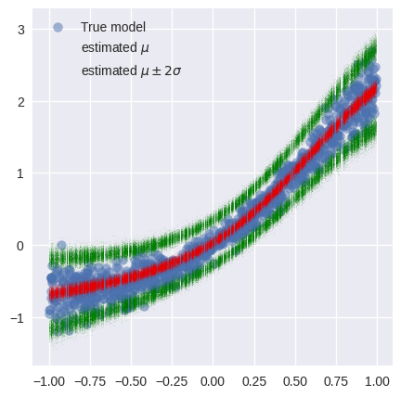


Figure 7: x_1

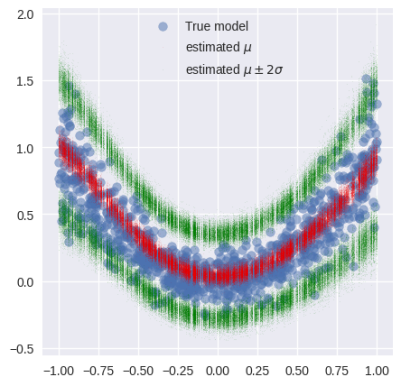


Figure 8: x_2

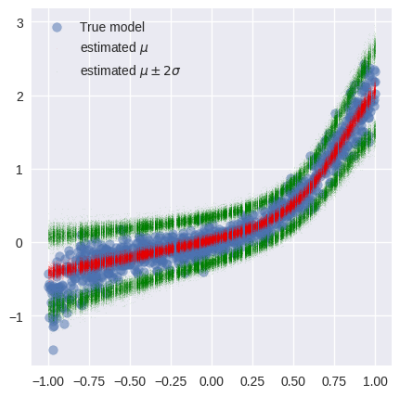


Figure 9: x_3

Conclusion

- Overall, a very good paper, but I am still not convinced that the algorithm is better than standard BNN, which are "easily" implementable (with a steep learning curve), with appropriate shrinkage priors
- The paper will also improve with a more through discussion of what are the variables driving the nonlinearities, in case they are present
- In the simulated data section, I would prefer to see how the algorithm learns the nonlinear impact of each regressor on the outcome variable - I struggle to understand why the linear model with SV performs as well as the BNN with the same activation function for each neuron

THANK YOU!

Model used to simulate data

I use the following model to simulate artificial data,

$$
\begin{aligned}
Y_t &= f_1(x_{1t}) + f_2(x_{2t}) + f_3(x_{3t}) + \sigma_t(x_t)\epsilon_t \\
f_1(x_{1t}) &= (b_1 + tanh(x_{1t}))x_{1t} \\
f_2(x_{2t}) &= x_{2t}^2 \\
f_3(x_{3t}) &= b_{31}x_{3t} + b_{32}x_{3t}^2 + b_{33}x_{3t}^3 \\
\sigma_t(x_t) &= s_0 + s_{11}x_{1t}^2 + s_{12}x_{2t}^2 + s_{13}x_{3t}^2
\end{aligned}
$$