From Reactive to Proactive Volatility Modeling with Hemisphere Neural Networks

Philippe Goulet Coulombe
Université du Québec à Montréal

Mikael Frenette
Université du Québec à Montréal

Karin Klieber
Oesterreichische Nationalbank

November 6, 2023

Context

- Unlike traditional deep learning strongholds like speech recognition and computer vision, applications in social sciences are typically nowhere near perfect prediction accuracy
- In other words, signal-to-noise ratio is low for most economic applications, and in the vicinity of 0 for finance applications.
- Deep learning methods can sometimes do surprising, yet informative, predictions.
- Thus, it is particularly pertinent to predict heterogeneous prediction uncertainty.
- Econometricians know this as conditional heteroscedasticity, via weighted least squares, or in a time series context, GARCH and Stochastic Volatility (SV).
- Then, there is the whole "at-risk" literature focusing on asymmetry.

Traditional Approaches Are Not Well-Suited for Deep Learning

- The many bells and whistles of gradient descent (like the `Adam` optimizer) can make a sizable difference.
- They are not readily implementable without deviating significantly from the highly optimized software environments that make DNN computations trivial.
  - Current Bayesian offerings often fall short of estimating anything that remotely resembles modern deep learning.
  - SV requires Bayesian computations, which are typically long, even for simple volatility specifications
  - MLE estimation of simple GARCH models is sometimes challenging in itself
- Approaches alternating the fit of the conditional mean and the conditional variance until convergence – à la iterated weighted least squares – are also highly impractical.
  - DNN residuals are often ~ 0 throughout the training sample (Belkin et al., 2019), making them an unusable target in a secondary conditional variance regression.
  - Many DGPs one can think of require simultaneous estimation

Reactive and Proactive Approaches

- SV, being essentially a trend-filtering problem for squared residuals, is unequipped to detect future volatility hikes. Similarly so, GARCH only propagates shocks that already occurred.
- Instead, we can be proactive: Adrian et al. (2019), Adams et al. (2021), Caldara et al. (2021), Delle Monache et al. (2021), Guidolin et al. (2021).
- ML offerings: Clark et al. (2022)'s BARTs and Barunik and Hanus (2022)'s DistrNN
- Can we devise a general-purpose NN that
  1. delivers good MSEs;
  2. provides accurate out-of-the-box uncertainty quantification for its predictions;
  3. will be proactive when it can be, and reactive when need be;
  4. is preferably simple, malleable, and principled.

Reinvigorate MLE

- We replace NN by **H**NN, as introduced in Goulet Coulombe (2022), where in this application, hemisphere 1 (h_m) is the conditional mean and hemisphere 2 (h_v) is the conditional variance

- Those hemispheres get their assigned roles from how they enter the loss function, which is now $\propto$ to a good old log-likelihood. We solve

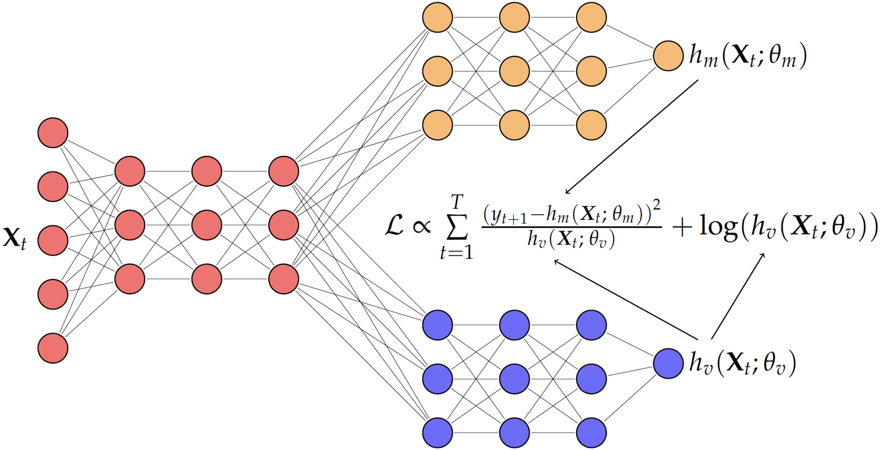
$$\min_{w_m, w_v} \sum_{t=1}^{T} \frac{(y_{t+1} - h_m(\boldsymbol{X}_t; \theta_m))^2}{h_v(\boldsymbol{X}_t; \theta_v)} + \log(h_v(\boldsymbol{X}_t; \theta_v)) \qquad (1)$$

  where θ_m and θ_v are the network parameters. Gradient Descent will gladly minimize (1).

- The model takes a large $\boldsymbol{X}_t$ as input for h_v and h_m, which are both fully nonlinear nonparametric function of inputs

- The architecture for h_v and h_m will allow for both proactive and reactive volatility.

The Architecture



Ingredient 1 – Weight Sharing

A regression model with ARCH(p) errors can be written as

$$\begin{cases} y_t = \mathbf{X}_t\beta + \varepsilon_t \\ \sigma_t^2 = c + \alpha_1\varepsilon_{t-1}^2 + \ldots + \alpha_p\varepsilon_{t-p}^2 \end{cases}$$

In this model, we have

$$\begin{cases} h_m(\boldsymbol{X}_t; \boldsymbol{\beta}) = \mathbf{X}_t\beta \\ h_v(\boldsymbol{X}_t; [\boldsymbol{\alpha} \;\; \boldsymbol{\beta}]) = c + \alpha_1 \left(y_{t-1} - \mathbf{X}_t\beta\right)^2 + \ldots + \alpha_p \left(y_{t-p} - \mathbf{X}_{t-p}\beta\right)^2 \end{cases}$$

As Gouriéroux (1997) puts it:

Even in the simple case, we cannot estimate separately the parameters of the conditional mean and those appearing in the conditional variance.

- This doesn't mean all volatility model need to be estimated jointly, but
- This suggests that successful models of time series volatility often have some **cross-equation restrictions** between h_m and h_v

Ingredient 1 – Weight Sharing (cont'd)

- Here, cross-equation restrictions are likely both unfeasible and undesirable
- However, cross-equation *regularization* (in effect, soft constraints) will help discipline h_m and h_v
- This motivates common layers at the entrance of the network, which can be interpreted as hemispheres sharing weights.
- Moreover, this is intuitive from a "latent variables sharing" perspective. Note that $\varepsilon_{t-1}^2 = (y_{t-1} - h_m(\boldsymbol{X}_{t-1}))^2$, a latent feature in h_v, can be seen as the result from a succession of three layers/operations, from top to bottom:
  1. Squaring;
  2. Differencing between an input (y_{t-1}) and the output of the h_m function on lagged data;
  3. The original h_m layer transforming $\boldsymbol{X}_{t-1}$, a subset of $\boldsymbol{X}_t$.
- **G**ARCH suggests making h_v a recurrent NN (we do so in appendix)
- SV suggests including trends in $\boldsymbol{X}_t$ (we do so).

Ingredient 2 – Volatility Emphasis

- The double descent phenomenon in DNNs. Basically, a mildly deep and large net will give $R^2_{\text{train}} = 1$, even though the true R^2 is nowhere near that, and yet, this model delivers the best R^2_{test}.

- Not a problem for plain prediction, but is certainly one for any in-sample analysis or... uncertainty quantification. Remember: MLE's $\hat{\sigma}^2$ is biased.

- More troubling, HNN can overfit the data in-sample with either $h_m(\boldsymbol{X}_t;\theta_m)$ or $h_v(\boldsymbol{X}_t;\theta_v)$, giving rise to vastly different models.

- Solution: fix $\texttt{mean}(h_v(\boldsymbol{X}_t;\theta_v)) = \nu$ during estimation, and let HNN learn deviations from it. Re-calibrate it ex-post using the realized unconditional volatility of OOB residuals (next slide).

- Ex-ante calibration should preferably be close to the ex-post one. We do $\nu = \texttt{mean}(\hat{\varepsilon}^2_{t,\text{DNN}})$, where $\hat{\varepsilon}$ are blocked OOB residuals.

Ingredient 3 – Blocked OOB Reality Check

- Once dual estimation of h_m and h_v has occurred, the initial ν guess might be suboptimal. We can calibrate h_v back using HNN's OOB residuals.

- To do so, we run

$$\log\left(\hat{\varepsilon}^2_{t,\text{HNN}}\right) = \underbrace{\zeta_0 + \zeta_1 \log\left(\hat{h}_v(\mathbf{X}_t;\theta_v)\right)}_{\hat{\delta}_t} + \xi_t$$

  and then update volatility such that

$$\hat{h}_v(\mathbf{X}_t;\theta_v) \leftarrow \exp(\hat{\delta}_t) \times \mathrm{E}[\exp(\xi_t)].$$

- Mechanically, this provides good nominal coverage in-sample.

Ingredient 4 – Blocked Subsampling

- Obviously, it has been implicit throughout from the use of OOB quantities
- There is no guarantee that a single run of (stochastic) gradient descent initiated randomly will deliver the "true parameters". It does not attempt to win where old-fashioned MLE would likely fail.
- Not a problem: as is commonly done for point prediction itself, we ensemble many runs.
- We do 1000 runs, which is a supreme overkill for the out-of-sample, but just fine for OOB "time series" that utilize on average $(1 - \texttt{subsampling.rate}) \times 1000$ runs at each t.
- Very interestingly, this fits within the framework of Newton and Raftery (1994)'s Weighted Bayesian Bootstrap, particularly Newton et al. (2021)'s extension of it for generic ML losses. (Randomly-weighted optimization of the loss provides an approximate Bayesian posterior.) Thus, it makes statistical sense.

Deep Dive: Tuning Parameters

- Each layer (common or not) is given `neurons` $= 400$
- The common block has `ws.layers` $= 2$
- Each hemisphere (mean and volatility) has `h.layers` $= 2$
- Activation functions are *ReLu* throughout, for rectified linear unit:

$$\text{ReLU}(x) = \max\{0, x\}$$

- Output activation function for h_v is

$$\text{Softplus}(x) = \log\left(1 + \exp\left(x\right)\right)$$

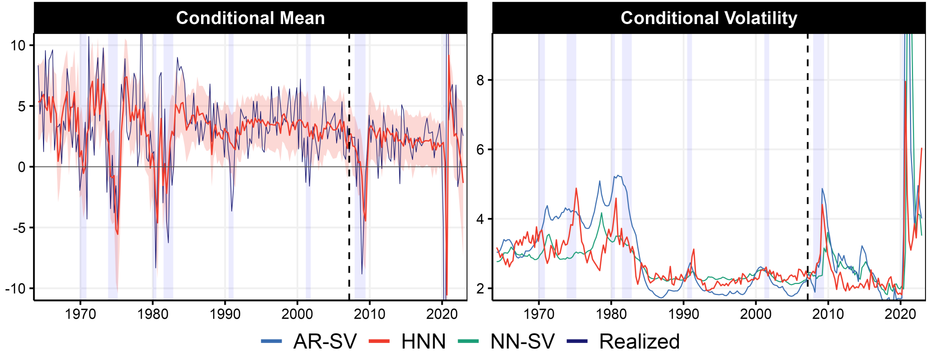
  which is, in effect, a soft *ReLu*. This imposes $\hat{h}_v(\mathbf{X}_t; \theta_v) \geq 0 \;\; \forall t$.
- Network weights w_m and w_v are initialized using $\mathcal{N}\left(0, 3/100\right)$
- `max.epochs` is 100 with a patience of 15 for early stopping
- `learning.rate` is 0.001 with Adam Optimizer
- `dropout.rate` is 0.2
- `sampling.rate` is 0.8, number of bootstraps $B = 1000$
- `block.size` is 8 quarters

The Experiment

We evaluate our proposed model in an empirical application with US data.

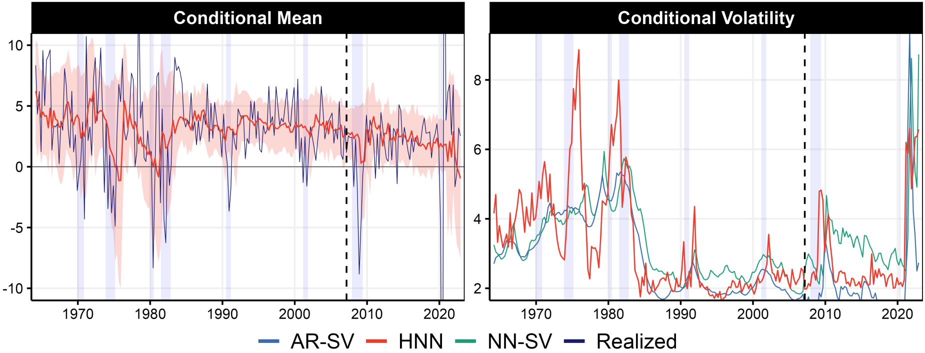
- Quarterly data, predicting $s = 1$ and $s = 4$ steps ahead
- $\mathbf{X}_t \equiv 2$ lags of FRED-QD + 100 linear trends (for exogenous T-V)
- Target variables are GDP Growth, Δ Unemployment Rate, Headline CPI Inflation, Housing Starts Growth, S&P 500 Returns
- Complete out-of-sample is 2007Q1 to 2022Q4
- NN-based models are re-estimated every two years (with expanding window), others every quarter

GDP Growth ($s = 1$)



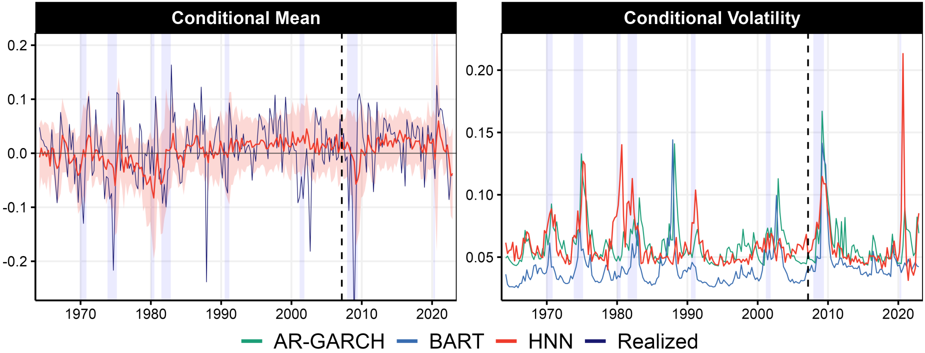
	2007Q1 - 2019Q4							2007Q1 - 2022Q4, Excluding 2020						
	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_{SV}	BLR	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_{SV}	BLR
RMSE	**0.83**	0.93	0.93	0.92	0.86	1.01	0.89	**0.85**	0.96	0.96	0.93	0.92	1.00	0.94
$\mathcal{L}$	**-3.93**	-3.82	-3.80	-3.18	-3.88	-3.75	-3.69	**-3.87**	-3.70	-3.63	-3.23	-3.71	-3.69	-3.63
$R^2_{\|\varepsilon_t\|}$	**0.30**	0.18	0.21	0.04	0.07	-0.23	-1.23	**0.18**	-1.57	-3.82	0.08	-22.20	-0.68	-1.19

GDP Growth ($s = 4$)



	2007Q1 - 2019Q4							2007Q1 - 2022Q4, Excluding 2020						
	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_{SV}	BLR	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_{SV}	BLR
RMSE	0.90	0.88	0.88	1.07	**0.88**	0.99	0.91	0.85	1.46	1.46	0.99	**0.81**	0.98	0.95
$\mathcal{L}$	-3.70	-3.54	-3.52	-2.83	**-3.70**	-3.04	-3.59	**-3.61**	-3.33	-3.36	1.27	-3.55	-3.05	-3.51
$R^2_{\|\varepsilon_t\|}$	**0.28**	0.12	0.27	0.09	-0.03	0.07	-0.67	0.06	0.07	-0.08	-0.03	-9.86	**0.07**	-0.41

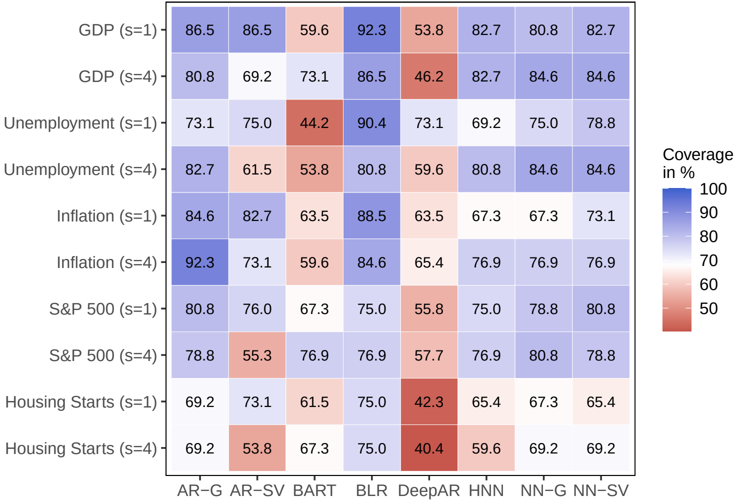
S&P 500 Returns ($s = 1$)



	2007Q1 - 2019Q4							2007Q1 - 2022Q4						
	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_G	BLR	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_G	BLR
RMSE	0.96	1.09	1.09	1.02	**0.92**	0.94	0.98	0.93	1.04	1.04	1.06	**0.89**	0.92	0.96
$\mathcal{L}$	**-1.55**	-1.24	-1.27	-1.34	-1.28	-1.35	-1.25	**-1.52**	-1.30	-1.32	-1.13	-1.34	-1.39	-1.29
$R^2_{\lvert\varepsilon_t\rvert}$	0.26	0.04	0.06	**0.30**	0.11	0.24	-0.13	0.07	0.04	0.05	0.22	0.12	**0.25**	-0.14

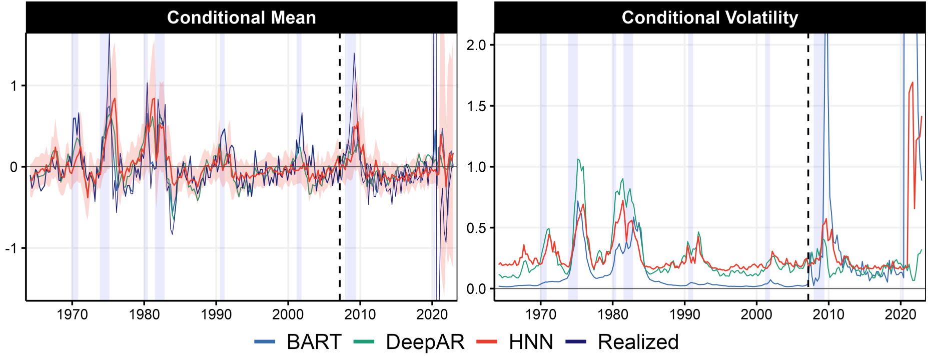
Some Trouble with DeepAR and BART

68% Coverage, 2007Q1-2019Q4



Some Trouble with DeepAR and BART

Δ Unemployment Rate ($s = 4$)



	2007Q1 - 2019Q4							2007Q1 - 2022Q4						
	HNN	NN_{SV}	NN_{G}	DeepAR	BART	AR_{TV}	BLR	HNN	NN_{SV}	NN_{G}	DeepAR	BART	AR_{TV}	BLR
RMSE	0.74	**0.69**	**0.69**	0.85	0.75	0.97	0.82	0.70	2.20	2.20	0.73	0.71	0.88	**0.70**
$\mathcal{L}$	**-0.17**	0.04	0.06	0.23	0.65	0.47	0.19	**0.03**	0.62	0.34	2.91	0.82	0.70	0.30
$R^2_{\lvert\varepsilon_t\rvert}$	**0.49**	-0.07	-0.12	0.21	-5.58	0.22	-0.36	-1.23	-0.08	-0.24	**-0.02**	-38.15	-0.24	-1.12

Neural Phillips Curve with Proactive Volatility

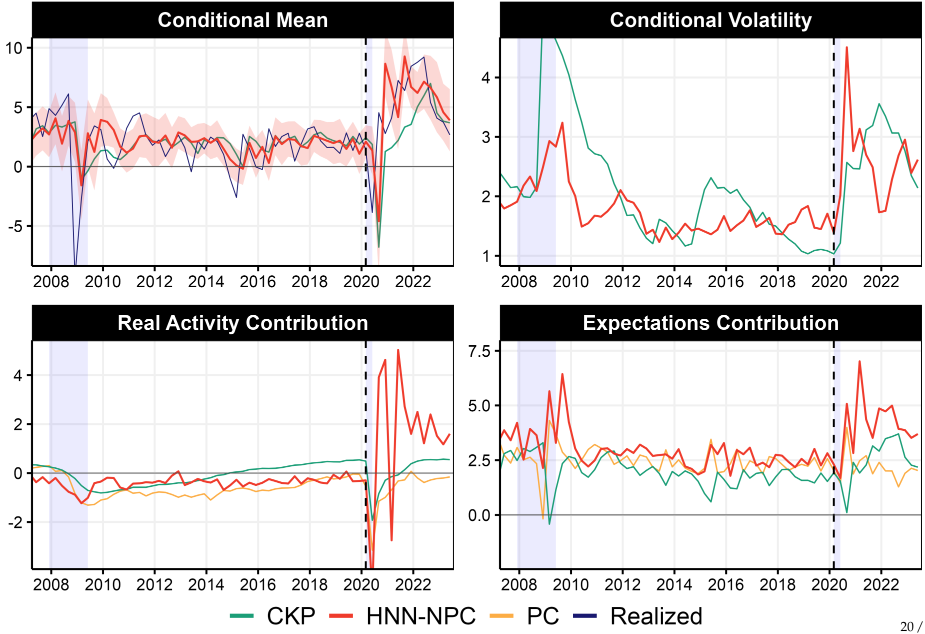
- HNN with 4 additional hemispheres (Goulet Coulombe, 2022):
- long-run, short-run expectations, output gap, commodity prices

$$h_m^{\text{NPC}}(\boldsymbol{X}_t; [\theta_{\mathcal{E}}^{\text{LR}}, \theta_{\mathcal{E}}^{\text{SR}}, \theta_g, \theta_c]) = h_{\mathcal{E}}^{\text{LR}}(\boldsymbol{X}_t^{\mathcal{E}_{\text{LR}}}; \theta_{\mathcal{E}}^{\text{LR}}) + h_{\mathcal{E}}^{\text{SR}}(\boldsymbol{X}_t^{\mathcal{E}_{\text{SR}}}; \theta_{\mathcal{E}}^{\text{SR}}) + h_g(\boldsymbol{X}_t^{g}; \theta_g) + h_c(\boldsymbol{X}_t^{c}; \theta_c)$$

$$h_v^{\text{NPC}}(\boldsymbol{X}_t; [\theta_{\mathcal{E}}^{\text{LR}}, \theta_{\mathcal{E}}^{\text{SR}}, \theta_g, \theta_c, \theta_v, \theta_{\tilde{v}}]) = h_v\left(\left[h_{\mathcal{E}}^{\text{LR}}(\boldsymbol{X}_t^{\mathcal{E}_{\text{LR}}}; \theta_{\mathcal{E}}^{\text{LR}}), h_{\mathcal{E}}^{\text{SR}}(\boldsymbol{X}_t^{\mathcal{E}_{\text{SR}}}; \theta_{\mathcal{E}}^{\text{SR}}), h_g(\boldsymbol{X}_t^{g}; \theta_g), h_c(\boldsymbol{X}_t^{c}; \theta_c), h_{\tilde{v}}(\boldsymbol{X}_t; \theta_{\tilde{v}})\right]; \theta_v\right)$$

	2007Q1 - 2019Q4						2007Q1 - 2022Q4					
	RMSE	$\mathcal{L}$	$R^2_{\lvert\varepsilon_t\rvert}$	CRPS	Cov68	PIT-pv	RMSE	$\mathcal{L}$	$R^2_{\lvert\varepsilon_t\rvert}$	CRPS	Cov68	PIT-pv
Inflation ($s = 1$)												
HNN	0.93	-3.63	-0.06	0.96	**67.3**	0.41	1.12	-3.41	**0.17**	1.05	62.5	0.61
HNN-NPC	**0.88**	**-3.87**	**0.09**	**0.91**	69.2	**0.74**	**1.02**	**-3.60**	0.13	**0.98**	**64.1**	**0.69**

Visualizing the Neural Phillips Curve and its Volatility



Conclusion

- We devised a new general-purpose deep learning model for joint mean/variance prediction that exhibits enviable performance for many series at various horizons.
- In the paper, we extend the analysis to a monthly application (FRED-MD), euro area application, LSTM version of the model and show variable importance by hemisphere.
- More generally, this suggest that many macro time series models can be estimated with deep learning techniques via reinvigorated MLE.

Q: What's the *real* use for DNNs in Macro Forecasting?

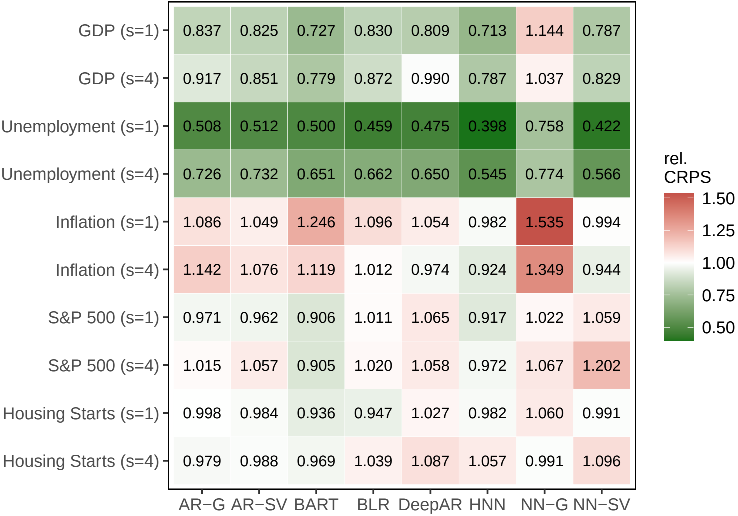
- Besides handling non-traditional data, we can use them to put basic structure and move beyond extraction of the conditional mean.
- It is not excluded that structure also improves point forecasts themselves. Tailor-made architectures made deep learning work in the first place. CNNs made image recognition work, Attention drives ChatGPT.
- Plain DNNs are intricate nonlinear ridge regressions where regularization comes from early stopping, with its "neural" structure being a prior on how best to capture nonlinear relationships.
- Unclear, *based purely on statistical grounds*, whether that structure is more appropriate than trees or anything econometricians have conceived.
- The core innovation is the ease of optimization through the recursive structure of the NN function (i.e., backprop, learning rates, etc.)
- HNN for volatility prediction does exactly that by tackling a likelihood which Gauss-Newton methods would struggle with

HNN vs HNN-LSTM

	2007Q1 - 2019Q4				2007Q1 - 2022Q4			
	RMSE (s_1)	$\mathcal{L}$ (s_1)	RMSE (s_4)	$\mathcal{L}$ (s_4)	RMSE (s_1)	$\mathcal{L}$ (s_1)	RMSE (s_4)	$\mathcal{L}$ (s_4)
GDP								
HNN	**0.83**	-3.93	0.90	**-3.70**	**0.85**	-3.87	**0.85**	**-3.61**
HNN-LSTM	0.85	**-3.95**	**0.89**	-3.70	0.88	**-3.88**	0.94	-3.61
Unemployment Rate								
HNN	**0.73**	**-0.37**	**0.74**	**-0.17**	0.82	**-0.24**	**0.70**	**0.03**
HNN-LSTM	0.76	-0.31	0.78	-0.08	**0.67**	-0.17	0.90	0.10
Inflation								
HNN	0.94	**-3.63**	0.93	-3.48	1.14	**-3.41**	0.94	**-3.30**
HNN-LSTM	**0.93**	-3.51	**0.88**	**-3.49**	**0.93**	-3.36	**0.92**	-3.28
S&P 500								
HNN	**0.96**	**-1.55**	**1.00**	-1.27	**0.93**	**-1.52**	1.00	**-1.27**
HNN-LSTM	1.02	-1.49	1.00	**-1.28**	1.01	-1.43	**1.00**	-1.27
Housing Starts								
HNN	**0.99**	**-1.14**	1.03	**-0.88**	**0.86**	**-1.07**	1.01	**-0.66**
HNN-LSTM	1.05	-1.04	**1.00**	-0.80	0.94	-0.95	**0.99**	-0.41

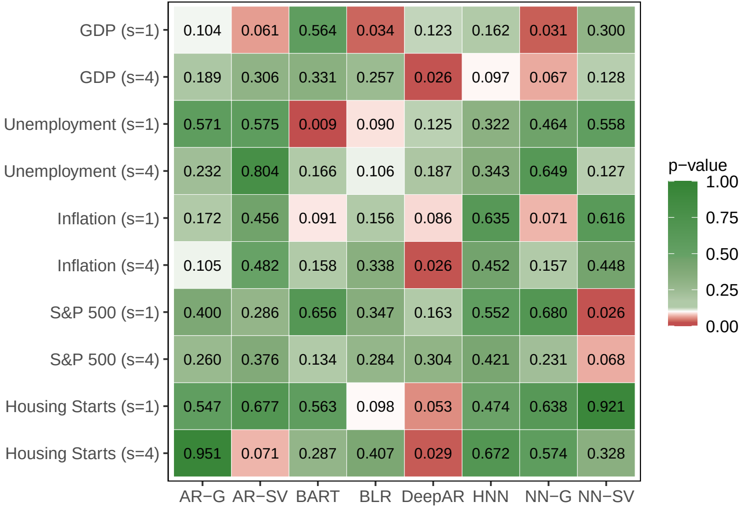
Additional Metrics

CRPS, 2007Q1-2019Q4

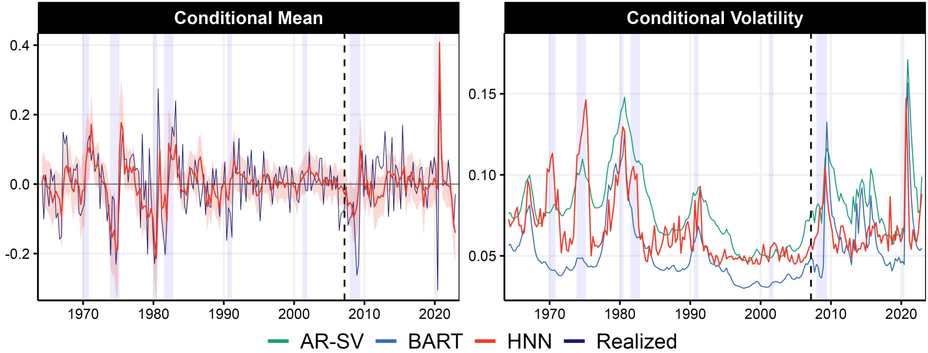


Additional Metrics

PIT test, 2007Q1-2019Q4

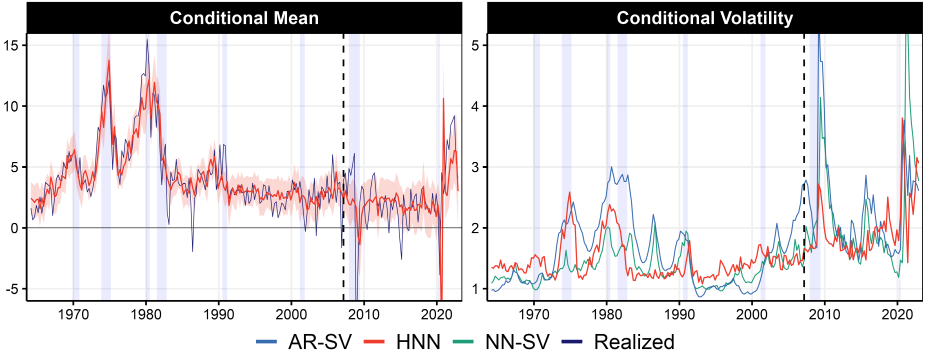


The interesting case of Housing Starts Growth ($s = 1$)



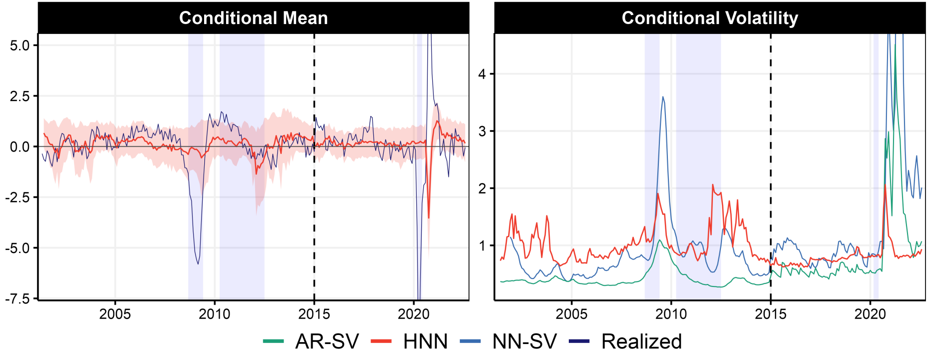
	2007Q1 - 2019Q4							2007Q1 - 2022Q4						
	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_{SV}	BLR	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_{SV}	BLR
RMSE	0.99	1.01	1.01	1.06	0.96	0.99	**0.96**	**0.86**	0.87	0.87	0.97	0.93	1.00	0.99
$\mathcal{L}$	-1.14	-1.08	-1.08	0.07	-0.98	-1.15	**-1.16**	-1.07	-0.97	-0.93	-0.05	-0.67	**-1.15**	-0.92
$R^2_{\|\varepsilon_t\|}$	0.14	-0.06	-0.01	-0.14	-0.03	**0.36**	-0.27	0.09	-0.09	-0.12	0.02	0.03	**0.15**	-0.03

Inflation?



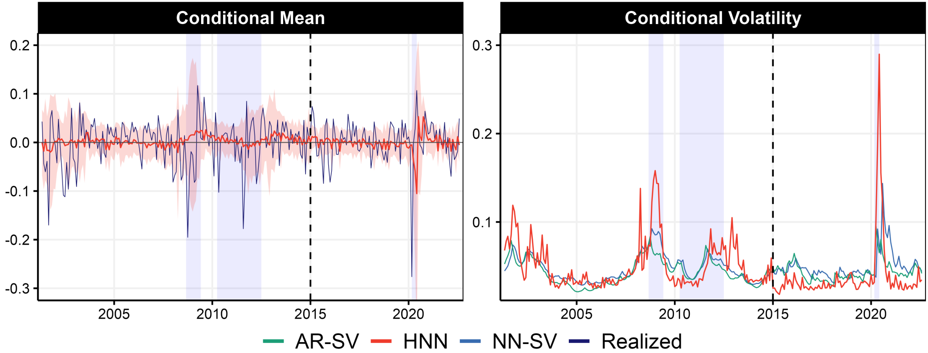
	2007Q1 - 2019Q4							2007Q1 - 2022Q4						
	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_{SV}	BLR	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_{SV}	BLR
RMSE	**0.94**	0.95	0.95	1.02	1.07	1.11	1.05	1.14	1.17	1.17	**0.93**	0.96	1.00	1.23
$\mathcal{L}$	-3.63	-3.74	**-3.82**	-3.57	-2.91	-3.26	-3.60	-3.41	-2.72	-3.47	**-3.52**	-1.30	-3.32	-3.33
$R^2_{\lvert\varepsilon_t\rvert}$	-0.06	-0.02	-0.03	**0.15**	-0.48	0.04	-0.32	**0.17**	-0.02	0.08	0.15	-0.41	0.06	-0.02

Industrial Production in the Euro Area ($s = 6$)



	2007Q1 - 2019Q4							2007Q1 - 2022Q4						
	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_G	BLR	HNN	NN_{SV}	NN_G	DeepAR	BART	AR_G	BLR
RMSE	**0.80**	0.81	0.81	0.84	0.87	0.94	0.94	**0.67**	0.69	0.69	0.78	0.78	0.95	1.12
$\mathcal{L}$	**-4.41**	-4.31	-4.37	-3.44	>10	-4.31	-4.06	**-4.31**	-4.04	-4.25	0.55	>10	-4.12	-3.89
$R^2_{\lvert\varepsilon_t\rvert}$	0.65	0.43	0.59	0.17	-0.84	**0.70**	-2.70	0.59	-4.96	0.34	-0.31	<-10	**0.71**	-0.39

Stock Market in the Euro Area ($s = 1$)



	2007Q1 - 2019Q4							2007Q1 - 2022Q4						
	HNN	NN_{SV}	NN_{G}	DeepAR	BART	AR_{G}	BLR	HNN	NN_{SV}	NN_{G}	DeepAR	BART	AR_{G}	BLR
RMSE	1.05	1.11	1.11	1.12	1.09	**1.02**	1.11	1.07	1.31	1.31	0.99	1.03	**0.99**	1.16
$\mathcal{L}$	-1.95	-1.88	-1.89	-1.76	-1.91	**-1.99**	-1.84	-1.53	-1.45	-1.38	-1.61	-1.23	**-1.67**	-1.50
$R^2_{\lvert\varepsilon_t\rvert}$	**0.56**	0.28	0.36	0.05	-0.05	0.35	-0.81	**0.10**	0.03	-0.08	0.00	-0.17	0.06	-0.13