A Multi-Task Deep Learning Model for Inflation Forecasting: Dynamic Phillips Curve Neural Network

Robert Proner*

April 2023

Abstract

Price stability is essential to economic development and high inflation can be harmful to vulnerable segments of the population. However, forecasting inflation has been difficult for central banks in the past due to the abundance of shocks and the complexities of business cycle transitions. To address this, I propose the Dynamic Phillips Curve Neural Network (DPCNN), a multi-task deep learning model which incorporates long-term business cycle dynamics using long short-term memory (LSTM) layers. DPCNN simultaneously forecasts inflation and the unemployment rate, which improves out-of-sample performance by imposing an economic constraint on their joint dynamics and produces a forward looking Phillips Curve. I show DPCNN provides significant advantages over time-series econometric models and off-the-shelf machine learning models.

JEL Classification: E31; E37; C45

Keywords: Inflation forecasting; Deep learning, LSTM, LASSO; Random Forest

*Department of Economics and Finance University of Guelph

1 Introduction

Price stability is vital to the development of an economy as many contracts are determined on a nominal basis. Additionally, high inflation on essential goods can be especially harmful to the most vulnerable segments of the population. As many central banks such as the Bank of Canada and the Federal Reserve practice an inflation targeting monetary policy, accurate forecasts of inflation are essential in designing a monetary (and fiscal) policy which provide appropriate levels of response to expected economic conditions.

In recent years, machine learning (ML) techniques for the forecasting inflation and other macroeconomic time series has been gaining a lot of attention. ML methods are especially well suited for identifying complex linear and non-linear relationships among macroeconomic variables within data rich environments. I propose DPCNN, a multi-task deep learning model, which forecasts inflation and the unemployment rate simultaneously. Multi-task learning (MTL) can improve generalization, increasing out-of-sample performance — often on each task — by facilitating inductive learning, which is a process by which the learning model benefits from learning key features shared between tasks (Ruder, 2017). DPCNN uses inductive learning and imposes an economic constraint on the joint dynamics of inflation and unemployment to improve out-of-sample forecast performance. I show that DPCNN yields significant gains in out-of-sample inflation forecasts accuracy over the usual ML methods explored in the literature, including linear methods such as the LASSO and tree-based methods such as Random Forests. DPCNN outperforms time series models and off-the-shelf ML methods at every forecast horizon, one-, three-, six-, and twelve-months. Moreover, the gains in out-of-sample forecast performance increase with the forecast horizon, even over the LSTM (which has an analogous architecture), suggesting that forecasting inflation and unemployment simultaneously results in gains that are not solely attributable to memory due to the inclusion of LSTM components. Diebold-Mariano (DM) tests show that DPCNN generates lower forecast error on average than the alternative methods. These results are statistically significant at the twelve-month forecasting horizon. Plots of the forecasts gen-

erated by DPCNN, show that DPCNN more accurately captures long-term macroeconomic cycles rather than the Random Forest (RF), especially at longer forecast horizons, where the RF tends to overreact around the tech bubble and financial crisis resulting in large errors even years afterwards. Finally, I provide some insight into the variables that DPCNN finds the most important. I compute variable importances and variable group importances according to the variable groupings of the FRED-MD database. I show that the labour market, specifically average weekly hours worked and payrolls are by far the most important variables when forecasting inflation, followed by spreads on treasuries and corporate bonds.

Advancements in machine learning theory and computational resources have driven encouraging results in macroeconomic time series forecasting, and a large amount of evidence in favour of ML has surfaced. Medeiros et al. (2021) conduct a large study on the effectiveness of ML methods for forecasting inflation in data rich environments. The authors make use of a large database of macroeconomic variables, autoregressive terms, and principle components and a large set of ML models to forecast US CPI inflation at various forecast horizons. They show that ML models, specifically random forests, can consistently outperform traditional benchmarks in forecasting inflation at various forecast horizons. Garcia et al. (2017) use several ML methods to forecast and high-dimensional real-time data to forecast CPI inflation in Brazil and find that the LASSO consistently outperforms benchmarks as well as collections of expert forecasts. Chakraborty and Joseph (2017) also use several macroeconomic variables, though much less than the above studies, and several ML methods to forecast CPI inflation in the UK at the two-year time horizon and find that most ML methods outperform benchmarks, vector autoregressive (VAR) models and autoregressive models (AR). Aras and Lisboa (2022) find advantages to tree-based ensemble methods for forecasting inflation in Turkey.

I contribute to the literature in several ways. First, I propose a multi-task deep learning model, which, by imposing an economic constraint on the joint dynamics of inflation and unemployment and incorporating long-term business cycle dynamics with LSTM lay-

ers, yields significant advantages in performance and economic insight over traditional time series econometric models and out-of-the-box ML models. Using pairwise DM tests, I show that DPCNN produces forecasts with lower error on average than alternative methods, and that these results are statistically significant at the twelve-month forecast horizon. Second, contrary to Atkeson and Ohanian (2001) and some of the early literature on inflation forecasting, I add to the growing evidence that, utilizing state-of-the-art ML techniques and a large amount of covariates, multivariate methods can consistently outperform univariate time series econometric benchmarks such as the random walk model and the autoregressive model. Third, I provide insights into the variables and variable groups that were identified by DPCNN to be the biggest drivers of inflation, and identify average weekly hours worked, payrolls, and spreads on treasuries and corporate bonds as the variables that policy makers should focus on when making policy decisions.

The remainder of this paper proceeds as follows: Section 2 provides a theoretical framework for inflation forecasting and an overview of DPCNN, Section 3 provides a description of the data and transformations used, Section 4 presents the results of DPCNN and other ML methods relative to benchmarks, and finally Section 6 concludes.

2 Methodology

I consider inflation h-periods into the future as a model of the form

$$\pi_{t+h} = f_h(\boldsymbol{x}_t) + \epsilon_h \tag{1}$$

where $\boldsymbol{x}_t = (x_{t1}, x_{t2}, ..., x_{tp}) \in \mathbb{R}^p$ is a vector of p covariates at time t, including π_t and its autoregessive terms, $f_h : \mathbb{R}^p \to \mathbb{R}$ is a function which maps $\boldsymbol{x}_t$ to π_{t+h}, and ϵ_{t+h} is a random error term. Each horizon h has a different mapping f_h.

Direct forecasts of (1) are given by

$$\hat{\pi}_{t+h} = \mathbb{E}[\pi_{t+h}|\boldsymbol{x}_t] = \hat{f}_{h|t}(\boldsymbol{x}_t) \tag{2}$$

where the subscript $t|h$ indicates that f is estimated using all available information at time t.

2.1 Models

2.1.1 Dynamic Phillips Curve Neural Networks (DPCNN)

DPCNN is a MTL approach to inflation forecasting, whereby the Phillips Curve relationship between inflation and unemployment is exploited to yield better generalization and thereby more accurate forecasts out-of-sample. Additionally, DPCNN provides a more comprehensive outlook of inflation by generating forecasts for inflation and unemployment simultaneously using shared parameters and in doing so produces forward looking Phillips Curves, hence the PC in DPCNN. This architecture imposes the economic constraint that inflation and unemployment dynamics are driven jointly by shared economic factors. By imposing the economic constraint on the joint dynamics of inflation and unemployment, I hypothesize that DPCNN outperforms off-the-shelf ML methods and time series econometric methods and provides a more informed picture of future economic conditions which enables policy makers to make better decisions.

Figure 1 plots the simplified architecture of the DPCNN. The hard-sharing LSTM layer imposes an economic constraint on the joint dynamics of inflation and unemployment, which allows the model to learn complex and potentially long-term macroeconomic cycles that drive inflation and unemployment. The network then splits into two sets of task-specific layers. These layers feature an LSTM layer that learns additional macroeconomic dynamics specific to the output at the end of the branch, which then feeds into a small dense layer (a typical feed-forward style layer) which summarizes and refines features from the LSTM layer preceding it.

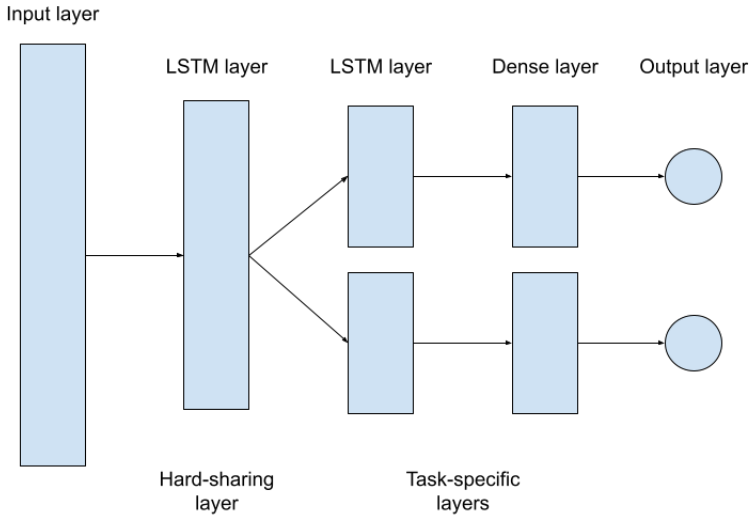


Figure 1: Simplified DPCNN architecture

2.1.2 Off-the-Shelf Methods

As comparative benchmarks I select several ML methods used in Medeiros et al. (2021). These methods include linear methods such as linear regression, ridge regression, LASSO regression, elastic net regression, tree-based methods such as random forests and gradient boosted trees. Additionally, I also include an additional tree-based method called Extremely Randomized Trees and a long short term memory (LSTM) neural network. Off-the-shelf methods are described in the Appendix A.

3 Data

The data used in this study is similar to Medeiros et al. (2021). Data is collected from FRED-MD database and is available on Michael McCraken's webpage. It consists of monthly macroeconomic variables from eight groups: output and income; labour market; housing;

consumption,orders, and inventories; money and credit; interest and exchange rates; prices; and stock market.

The entire sample period of this study ranges from January 1965 to December 2019. The period January 1965 to December 1989 is used for training and validation, while the remainder, January 1990 to December 2019 is reserved for out-of-sample testing. After variables with missing observations are discarded, there are 122 variables remaining and 746 monthly observations over the entire sample. For the off-the-shelf methods excluding the LSTM, I include 13 autoregressive terms of one-month inflation (12 lags plus inflation at time t) for a total of 135 predictor variables. The LSTM is capable of using lagged data and carrying memory and therefore does not require the additional autoregressive terms. In my implementation the LSTM and DPCNN models take as inputs the past 12 months of data for each forecast. Unlike Medeiros et al. (2021), I do not include lags of all predictor variables, or principle components in the final predictor set. Also, following Chen et al. (2021) and Bianchi et al. (2021) I account for the effect of announcement delays in macroeconomic data by lagging all predictor variables by an additional month.

The forecast variable is h-month inflation, defined as $\pi_{(t-h):t} = \log(P_t) - \log(P_{t-h})$, where P_t is the level of the price index at time t. I forecast inflation at one, three, six, and twelve months ($\pi_{t:(t+h)} = \log(P_{t+h}) - \log(P_t)$, $h = 1, 3, 6, 12$). The price index used in this study is CPI on all items (called CPIAUCSL in the FRED-MD data). One-month inflation is forecast for each of one to twelve months ahead (e.g., $\pi_{t+h} = \log(P_{t+h}) - \log(P_{t+h-1})$). Then, h-month inflation forecasts are obtained by summing the individual one-month inflation forecasts for $1, 2, ..., h$. I produce direct forecasts, meaning each inflation horizon has its own model.

To account for fundamental changes in the data over time, I use an expanding window procedure to train all forecasting methods. The first training window is the initial training set January 1965 to December 1989. The first test window is January 1990 to December 1990. Then, the training window is expanded by one year and the test window shifts by one year. This process continues until forecasts have been produced for the entire test

sample. Throughout this process, the optimization of the deep learning methods and the remaining methods differs. Because of the computational cost associated with training deep learning models, parameters of the neural networks are tuned using validation windows which always consists of the last five years of the training window. I use early stopping to address over-fitting. Additionally, to address poor random initializations and training getting stuck in local optima, neural networks are trained ten times and the arithmetic mean of the predictions is taken, effectively creating an ensemble. Because the validation window is used for early stopping, neural networks are not trained on the last five years of the training window. However, hyperparameters, such as the number of hidden layers and nodes of the networks are chosen based on performance on the validation set and fixed throughout the entire process. In contrast, the remaining methods use a five-fold time series cross-validation for hyperparameter optimization. When optimal hyperparameters are found, the model is refit using the entire training sample. For example, a Random Forest model can optimize over hyperparameters such as the number of estimators and the maximum tree depth for each window. The time series cross-validation combined with the dynamic selection of hyperparameters lends these methods an advantage over the neural networks, as the off-the-shelf methods are trained with more timely information and are more flexible in the sense that these methods are able to alter their hyperparameters from one window to the next. Despite these disadvantages, I show DPCNN consistently outperforms alternative methods.

4 Results

Out-of-sample forecasts are evaluated with root mean squared error (RMSE), which emphasizes large errors and pair-wise DM tests are performed to compare competing forecasts.

Table 1 presents the RMSE relative to the random walk (RW) of all models at all horizons. DPCNN consistenly outperforms all off-the-shelf ML methods and time series models at all

horizons. A particularily important result is that for all methods excluding DPCNN, RMSE tends to increase with horizon length. This is important because policy makers need sufficient lead time in order to design an effective monetary or fiscal policy. This suggests DPCNN may be scalable to even longer horizons, at which it may yield even larger performance gains over alternative methods, which would enable policy makers to consider even longer time horizons when making policy decisions. One possible explanation for this is that the DPCNN is better at learning long-term complex macroeconomic cycles. Furthermore, the superiority of DPCNN compared to LSTM suggests that the gains are not strictly attributable to the capacity for memory due to the LSTM components. Rather, it suggests that imposing the economic constraint on the joint dynamics of inflation and unemployment yields additional gains in out-of-sample performance. Additionally, based on the findings of Stock and Watson (1999), it is possible that swapping unemployment with alternative measures of real economic activity such as housing starts or growth rate of manufacturing may provide additional improvements. Plots of the forecasts from the DPCNN and the RF at each inflation horizon (Figure 2) provide support for the hypothesis the DPCNN is better at predicting long-term macroeconomic cycles. The plots clearly show that as the time horizon increases, the DPCNN captures more of the trend rather than the month-to-month swings, while the RF tends to over-react around recessions, especially the financial crisis and the tech bubble.

Table 1: RMSE relative to the random walk of all forecast methods for one-, three-, six-, and twelve-month forecast horizons. The RMSE ratio of the best model at each horizon is bolded.

	Forecast Horizon			
Model	1m	3m	6m	12m
RW	1.000	1.000	1.000	1.000
LinReg	1.484	1.087	1.869	3.276
Ridge	0.901	0.838	0.922	1.105
LASSO	0.786	0.826	0.942	1.186
EN	0.786	0.831	0.931	1.135
RF	0.743	0.760	0.795	0.905
XT	0.733	0.744	0.788	0.912
GBT	0.778	0.751	0.780	0.875
LSTM	0.746	0.748	0.763	0.771
DPCNN	**0.728**	**0.733**	**0.757**	**0.746**

(a) One-month inflation

(b) Three-month inflation

(c) Six-month inflation

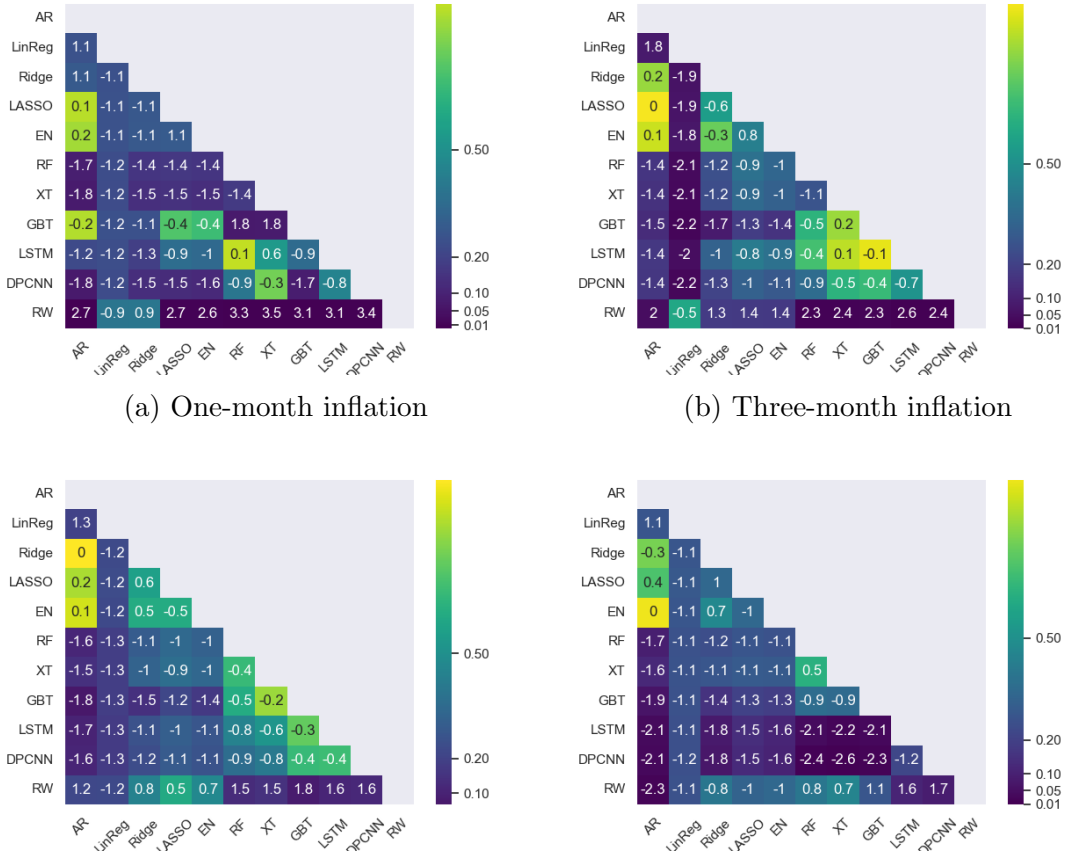
(d) Twelve-month inflation

Figure 2: Plots of forecasts from the RF and the DPCNN at three, six, and twelve months. NBER recessions are shaded in grey.

Next, I perform pair-wise forecast comparisons using DM tests, shown in Figure 3. Results of the DM tests suggest DPCNN has a lower squared error on average at one-, three-, six-, and twelve-month horizons, although this is not statistically significant at all horizons. However, at the twelve-month horizon, with p-values of about 0.01 and less, there is very strong evidence that that DPCNN is more accurate than all three tree-based methods. Together with the results of Table 1 and the plots of Figure 2, this provides very strong evidence in favour of DPCNN, especially at longer forecast horizons. Again, it would be interesting to see how the models compare at even longer horizons.

5 Which Covariates Matter?

To get an idea of which variables matter in inflation forecasts in the DPCNN, I take a simple approach in following Gu et al. (2020). I compute the importance of a feature as the average increase in RMSE over the test set from setting the feature to 0 (this is the mean of the standardized feature). To get an idea of how valuable a feature was over a particular forecast horizon, feature importances are averaged over the forecast horizons. Figure 4 plots the top ten features in terms of increase in RMSE for each forecast horizon. Overall, the most important variables are consistent across forecast horizons. The average number of hours works in goods producing and manufacturing are by far the two most important variables according to DPCNN. Additionally, payrolls and housing starts are another set of common variables between forecast horizons. This is not surprising. Labour hours, payrolls, and housing starts are all closely linked and all contribute to the velocity of money. For example, increases in labour hours and increases in payrolls will likely lead to increases in spending, some of which will go towards housing. Housing purchases will require the extension of credit by banks which will have a multiplying effect on money, applying upward pressure on prices.



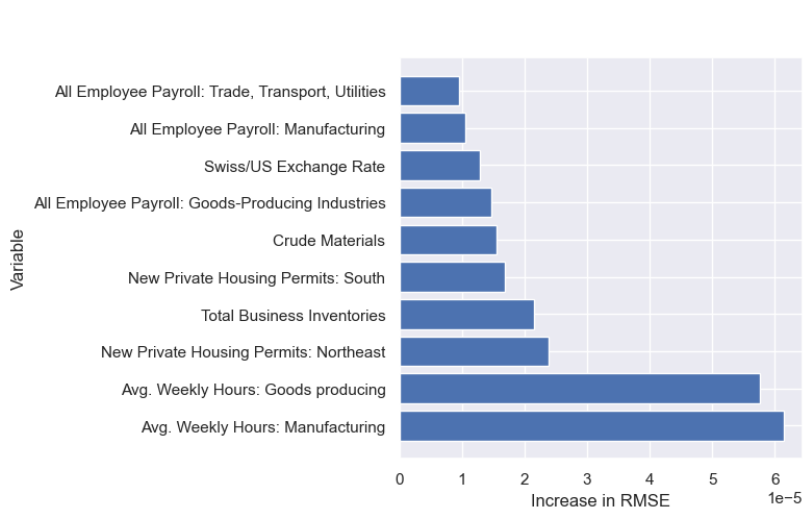
(a) One-month inflation

(b) Three-month inflation

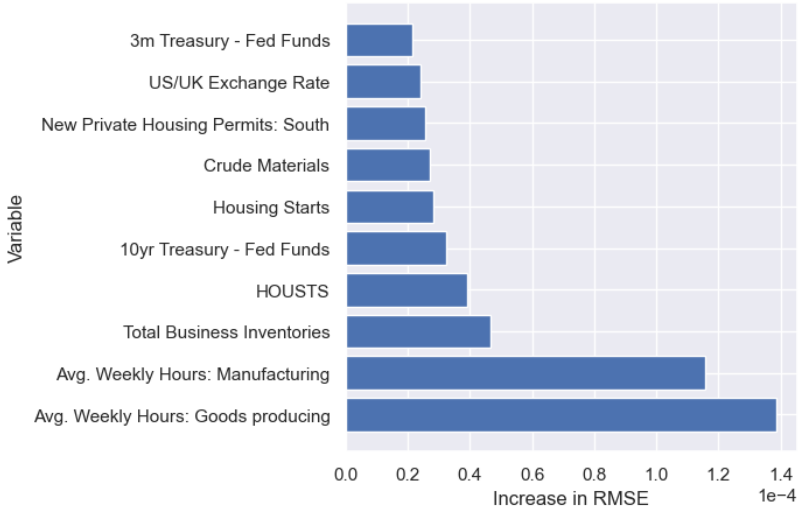
(c) Six-month inflation

(d) Twelve-month inflation

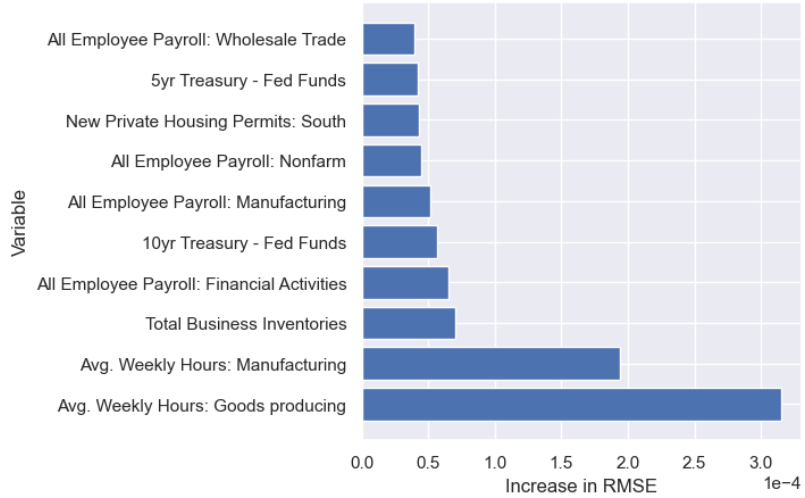
Figure 3: Pair-wise Diebold-Mariano tests for all forecasts. Numeric values within cells represent test statistics and cell color represents the p-value. Tests are performed using Newey-West standard errors. Negative test statistics imply the average error of the model on the x-axis is larger than that of the model on the y-axis.



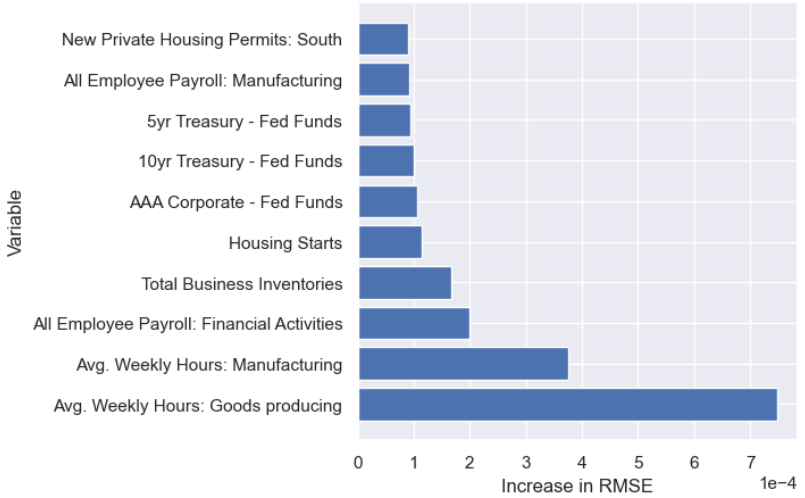
(a) 1-month forecast horizon



(b) 3-month forecast horizon



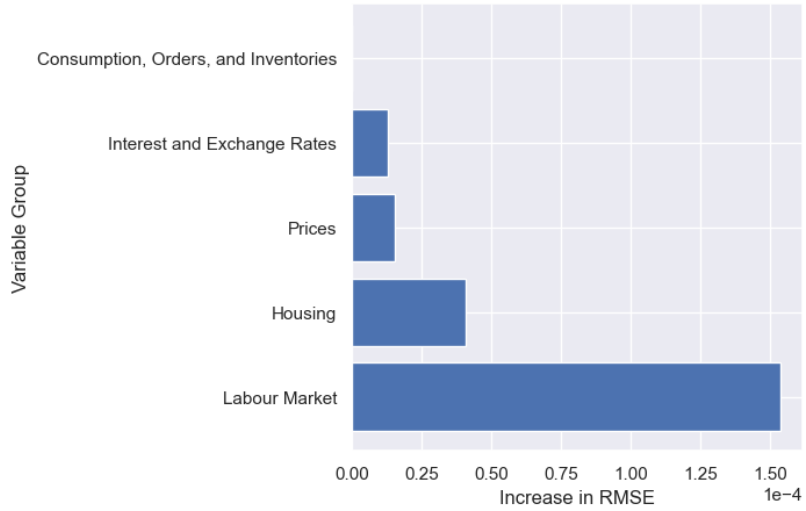
(c) 6-month forecast horizon



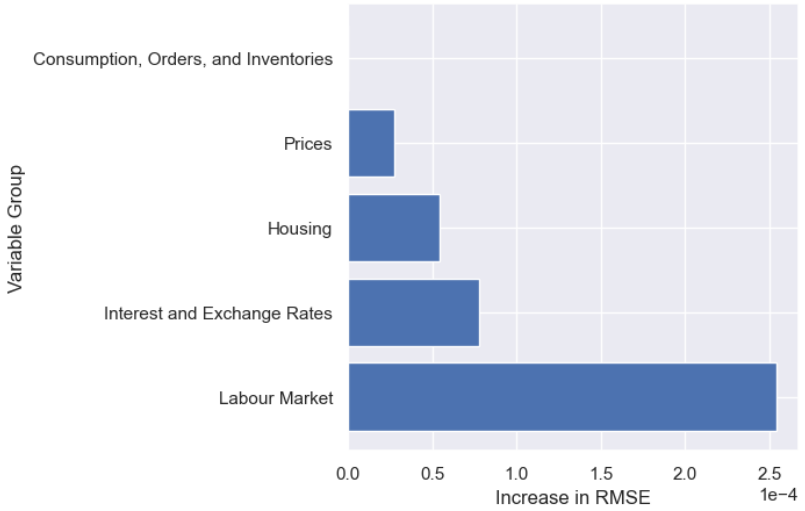
(d) 12-month forecast horizon

Figure 4: The ten most important features at each forecast horizon based on increase in RMSE when the feature is set to 0 (the mean of the standardized data).

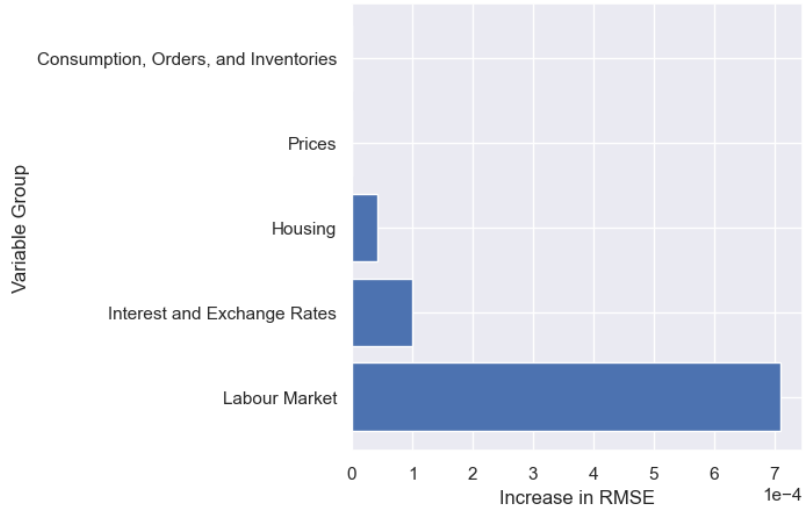
To summarize the results above I attain the variable importance of each variable group by summing the variable importance of features within each group: Output and Income; Labour Market; Housing; Consumption, Orders, and Inventories; Money and Credit; Interest and Exchange Rates; Prices; and Stock Market. The group importances are plotted in Figure 5. Out of the eight groups, only five were found to be useful according to the DPCNN, namely, Labour Market, Interest and Exchange Rates, Housing, Prices, and Consumption, Orders, and Inventories, in that order. Interestingly Prices and Consumption seem to be relatively unimportant when Labour Market and Interest and Exchange Rate variables are present in the model, on aggregate. Labour Market is by far the most important variable group at all horizons. Housing is the most important after Labour Market, at one-month, but falls behind Interest and Exchange Rates at the longer forecast horizons. This suggests that to the DPCNN, labour market conditions and credit spreads, specifically spreads on treasuries and corporate bonds are the most important drivers of future inflation. This implies that policy makers should focus on these variables to guide policy decisions.



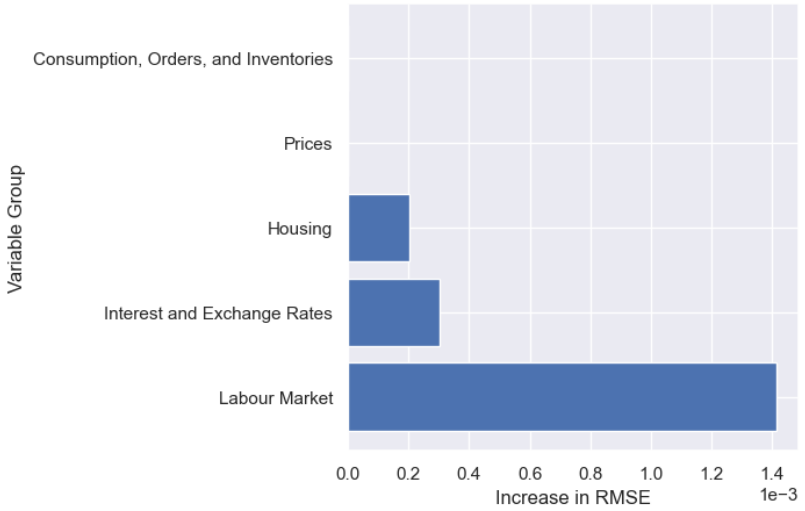
(a) 1-month forecast horizon



(b) 3-month forecast horizon



(c) 6-month forecast horizon



(d) 12-month forecast horizon

Figure 5: The ten most important groups at each forecast horizon based on increase in RMSE when the feature is set to 0 (the mean of the standardized data).

6 Conclusion

In this paper I propose DPCNN, a multi-task deep learning model which forecasts inflation and unemployment simultaneously, producing a forward-looking Phillips Curve. DPCNN imposes the economic constraint that inflation and unemployment dynamics are determined jointly by related long-term macroeconomic cycles. I show that DPCNN provides significant gains in out-of-sample forecast accuracy relative to time series models and off-the-shelf ML methods, especially as the forecast horizon increases. Using DM tests I show these gains are statistically significant at the twelve-month forecast horizon. Finally, I compute feature importances for the DPCNN and find that between 1990 and 2019, past labour market conditions and interest rates, specifically credit spreads on treasuries and corporate bonds, are by far the most important group of variables, followed closely by housing.

References

Aras, S. and Lisboa, P. J. G. (2022). Explainable inflation forecasts by machine learning models. *Expert systems with applications*, 207:117982–.

Atkeson, A. E. and Ohanian, L. (2001). Are phillips curves useful for forecasting inflation? *Quarterly review - Federal Reserve Bank of Minneapolis*, 25(1):2–.

Bianchi, D., BÃ¼chner, M., and Tamoni, A. (2021). Bond risk premiums with machine learning. *The Review of financial studies*, 34(2):1046–1089.

Chakraborty, C. and Joseph, A. (2017). Machine learning at central banks. Bank of England working papers 674, Bank of England.

Chen, L., Pelger, M., and Zhu, J. (2021). Deep learning in asset pricing. *arXiv.org*.

Chollet, F. (2018). *Deep learning with Python*. Manning Publications Co., Shelter Island, New York, 1st edition edition.

Garcia, M. G., Medeiros, M. C., and Vasconcelos, G. F. (2017). Real-time inflation forecasting with high-dimensional models: The case of brazil. *International journal of forecasting*, 33(3):679–693.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of financial studies*, 33(5):2223–2273.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning data mining, inference, and prediction*. Springer series in statistics. Springer, New York, 2nd ed. edition.

Medeiros, M. C., Vasconcelos, G. F. R., Veiga, Ã., and Zilberman, E. (2021). Forecasting inflation in a data-rich environment: The benefits of machine learning methods. *Journal of business & economic statistics*, 39(1):98–119.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks.

Stock, J. H. and Watson, M. W. (1999). Forecasting inflation. *Journal of monetary economics*, 44(2):293–335.

A Off-the-Shelf Methods

A.1 Benchmark Models

Random walk I define the random walk model for forecasting h-period inflation as the last observed h-period inflation. For example, when forecasting twelve-month inflation, I use inflation over the last twelve months as the forecast.

Autoregressive model of order p—AR(p) The AR(p) model in my context is a linear model of $p+1$ autoregressive terms, p lags plus the current month's inflation.

$$\pi_{t+h} = \alpha_0 + \alpha_1 \pi_t + \alpha_2 \pi_{t-1} + ... + \alpha_{p+1} \pi_{t-p} + \epsilon_{t+h}$$

where $\epsilon_{t+h} \sim WN(0, \sigma^2)$. At each training window I select the number of lags p according to the AIC criteria and the model is fit with OLS.

A.2 Linear models

Linear regression Linear regression assumes a linear functional form

$$\pi_{t+h} = \boldsymbol{x}_t' \boldsymbol{\beta} + \epsilon_{t+h} = \beta_{h0} + \beta_{h1} x_{t1} + ... \beta_{hp} x_{tp} + \epsilon_{t+h}$$

where $(x_{t1}, ..., x_{tp})$ includes autoregressive terms of pi_t. It can be fit by minimizing the sum of squared residuals (RSS), which can be done with gradient descent, but a fast closed form solution exists: $\boldsymbol{\beta}_h = (X'X)^{-1} X' \boldsymbol{\pi}_h$, where $\boldsymbol{\beta}_h$ is the vector of regression coefficients, X is the $n \times (p+1)$ matrix of features plus a columns of ones.

Ridge Regression Ridge regression imposes the ridge or ℓ_2 penalty, which shrinks the β coefficients towards zero. This helps reduce overfitting by reducing the contribution of irrelevant or less important predictors to the output, which often occurs in high-dimensional

feature spaces as features are likely to be collinear. With the ridge penalty the loss function becomes

$$J(\hat{\boldsymbol{\beta}}) = \sum_{t=1}^{T}(\pi_{t+h} - \hat{\pi}_{t+h})^2 + \lambda \sum_{j=1}^{p} \hat{\beta}_p^2$$

where λ is a hyperparameter which can be optimized with cross-validation.

LASSO The Least Absolute Shrinkage and Selection Operator (LASSO) imposes the LASSO or ℓ_1 penalty on the RSS so that the new loss function becomes

$$J(\hat{\boldsymbol{\beta}}) = \sum_{t=1}^{T}(\pi_{t+h} - \hat{\pi}_{t+h})^2 + \lambda \sum_{j=1}^{p} |\hat{\beta}_p|$$

The ℓ_1 penalty shrinks coefficients towards — and perhaps sets them equal to — zero, which allows it to also perform feature selection.

Elastic Net Regression Elastic Net Regression imposes the elastic net penalty on the RSS, which is a combination of ℓ_1 and ℓ_2 penalties. The amount of each penalty is controlled by the hyperparameter α

$$J(\hat{\boldsymbol{\beta}}) = \sum_{t=1}^{T}(\pi_{t+h} - \hat{\pi}_{t+h})^2 + \lambda \sum_{j=1}^{p} \frac{1-\alpha}{2}\hat{\beta}_p^2 + \alpha|\hat{\beta}_p|$$

For $\alpha = 1$ the Elastic Net is equivalent to LASSO and for $\alpha = 0$ it is equivalent to Ridge Regression.

Ridge Regression, LASSO, and Elastic Net Regression are solved using numerical optimization methods (e.g., gradient descent).

A.3 Tree-based methods

Decision Trees Decision trees are simple recursive rule-based procedures that are capable of capturing non-linear relationships and can be used for regression or classification

tasks. Decision trees are commonly constructed using the Classification and Regression Trees (CART) algorithm, which proceeds as follows:

1. Let the data at node m be denoted by Q_m. For each split candidate $\theta = (X_j, s)$ where X_j is a feature and s is some threshold, split the data into two subsets

$$Q_m^{left} = \{(\boldsymbol{x}_t, \pi_{t+h}) : x_{tj} \leq s\} \text{ and } Q_m^{right} = \{\boldsymbol{x}_t, \pi_{t+h} : x_{tj} > s\}$$

   where the prediction $\hat{\pi}_t + h$ in each region is given by the mean of the targets in that region.

2. Evaluate quality of split according to

$$J(\theta) = \frac{T_{left}}{T} J(Q_m^{left}) + \frac{T_{right}}{T} J(Q_m^{right})$$

3. select the parameters θ^* that minimize $J(\theta)$.

4. Repeat for each of Q_m^{left} and Q_m^{left} until a desired tree depth is reached, until the number of samples in each leaf (end node) reaches some set minimum, or until there is only one sample remaining in each leaf.

The tree depth or the minimum samples per leaf are hyperparameters than can be optimized via cross-validation. Mathematically, the decision tree forecast can be written as

$$\hat{\pi}_{t+h} = \sum_{m=1}^{M} \bar{\pi}_{t+h,m} I(\boldsymbol{x}_t \in R_m)$$

where m denotes the region, M is the total number of regions, and $\bar{\pi}_{t+h,m}$ is the mean inflation of region R_m.

Bagging Bagging involves building B decision trees, one for each of B bootstrap samples of the data. The output of the Bagging Regressor is given by averaging the output of each

decision tree

$$\hat{\pi}_{t+h} = \bar{\pi}_{t+h,b} = \frac{1}{B}\sum_{b=1}^{B}\hat{\pi}_{t+h,b}$$

Random Forests Random Forests is similar to bagging except at each split, only a subset of the features are considered as candidates, typically, $\sqrt{p}$ or $p^{1/3}$.

Extremely Randomized Trees Extremely Randomized Trees are like Random Forests except the threshold for each split candidate is also chosen randomly.

Gradient Boosted Trees Loosely, building a Gradient Boosted Regression Tree model involves constructing a number of decision trees sequentially, each tree being fitted on the residuals of the preceding tree. The first tree is initialized to simply predict the unconditional mean. Subsequent trees are multiplied by a learning rate λ, which slows training and reduces overfitting. The final model is the sum of all trees (Trees. Algorithm 1 from Hastie et al. (2009) outlines this process in more detail

Algorithm 1 Gradient Boosting Regression Trees

1: Initialize $\hat{f}_0(x) = \arg\min_\gamma \sum_{t=1}^{T} L(\pi_{t+h}, \gamma)$
2: for $b = 1, 2, ..., B$:

$$r_{t+h,b} = -\frac{\partial L(\pi_{t+h}, f(x_t))}{\partial f(x_t)} = \pi_{t+h} - f(x_t)$$

3: Fit a classification tree on $r_{t+h,b}$ giving terminal regions $R_{j,b}$, $j = 1, 2, ..., J_b$
4: for $j = 1, 2, ..., J_b$ compute

$$\gamma_{j,b} = \arg\min_{\gamma} \sum_{x_t \in R_{j,b}} L(\pi_{t+h}, f_{b-1}(x_t) + \gamma)$$

5: update $f_b(x) = f_{b-1}(x) + \sum_{j=1}^{J_b} \gamma_{j,b} I(x \in R_{j,b})$
6: Output $\hat{f}(x) = f_B(x)$

A.4 Deep learning models

LSTM and DPCNN LSTM cells are similar to feed-forward neural network cells, except that they have the capacity for memory, which is carried from one time step to the next by two mechanisms, the hidden state h_t and the carry c_t. This is depicted well in a figure obtained from Chollet (2018):

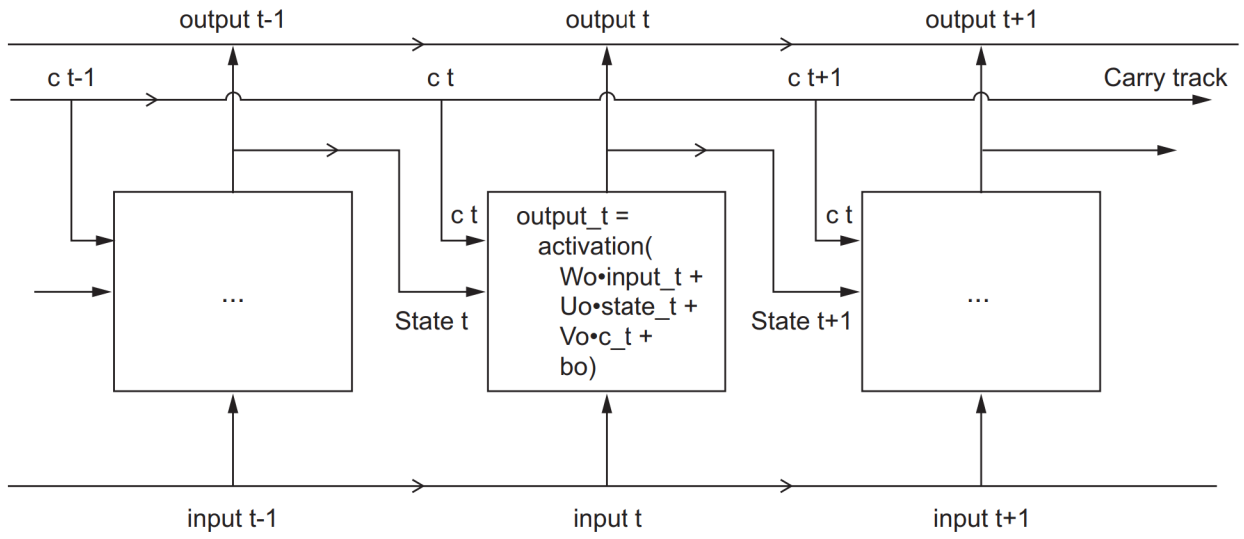


Figure 6: An example of an LSTM cell unrolled through time.

6 shows a singular LSTM cell unrolled through time. LSTM cells and other recurrent neural network (RNN) style cells process data sequentially, rather than in one shot using matrix multiplication.

Mathematically, the output of an LSTM cell at time t is given by

$$o_t = g(W_o \cdot \boldsymbol{x}_t + U_o \cdot h_t + V_o \cdot c_t) \tag{3}$$

where W_o, U_o, V_o are matrices of weights, $\boldsymbol{x}_t$ is the input at time t, h_t is the state at time t (the previous time steps output) , and c_t is the carry, which is similar to the hidden state but is capable of remembering information from past time steps, and g is an activation function.

At each time step a new carry is computed as

$$c_{t+1} = i_t \times k_t + c_t \times f_t$$

where

$$i_t = g(U_i \cdot h_t + W_i \cdot \boldsymbol{x}_t + b_i)$$
$$k_t = g(U_k \cdot h_t + W_k \cdot \boldsymbol{x}_t + b_k)$$
$$f_t = g(U_f \cdot h_t + W_f \cdot \boldsymbol{x}_t + b_f)$$

While we have no way of knowing exactly what each of these components are doing during the training process, we can think of multiplying c_t and f_t as forgetting past information and multiplying i_t and k_t as adding new information to the carry, which will be used down the line.

The architecture of the DPCNN reported in this study is shown in 7. It consists of three hard-sharing LSTM layers with 32 hidden units each. These layers learn shared long-term macroeconomic cycles. Then, the network splits into task-specific layers. These consist of another LSTM layer with 32 hidden units, which learn macroeconomic cycles specific to the task at the end of the branch, inflation or unemployment. Finally, two dense layers, each with 8 hidden units, capture additional nonlinear relationships from the preceeding layers and connect to the output layer, which produces a forecast for inflation or unemployment at time $t + h$. To make the comparison as accurate as possible, the LSTM has the exact same architecture except it lacks the branch for forecasting unemployment.

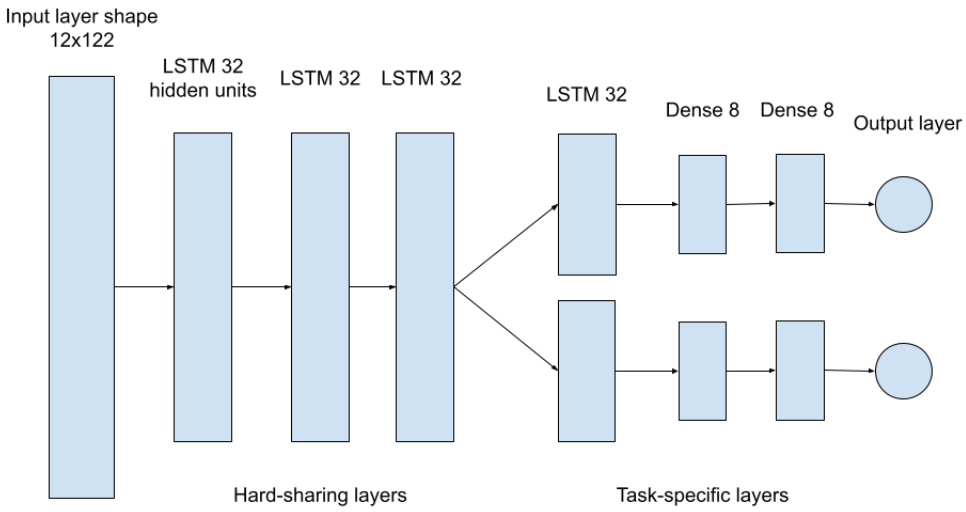


Figure 7: Architecture of DPCNN used in this paper.