Taming data-driven probability distributions *

Jozef Baruník*
Charles University and
Czech Academy of Sciences

Luboš Hanus**
Charles University and
Czech Academy of Sciences

May 29, 2024

Abstract

We propose a deep learning approach to probabilistic forecasting of macroeconomic and financial time series. By allowing complex time series patterns to be learned from a data-rich environment, our approach is useful for decision making that depends on the uncertainty of a large number of economic outcomes. In particular, it is informative for agents facing asymmetric dependence of their loss on the outcomes of possibly non-Gaussian and non-linear variables. We demonstrate the usefulness of the proposed approach on two different datasets where a machine learns patterns from the data. First, we illustrate the gains in predicting stock return distributions that are heavy tailed and asymmetric. Second, we construct macroeconomic fan charts that reflect information from a high-dimensional dataset.

Keywords: Distributional forecasting, machine learning, deep learning, probability, economic time series
JEL: C45, C53, E17, E37

*We are grateful to Wolfgang Hardle, Lukas Vacha, Martin Hronec, Frantisek Cech, and the participants at various conferences and research seminars for many useful comments, suggestions, and discussions. We gratefully acknowledge the support from the Czech Science Foundation under the EXPRO GX19-28231X project. We provide the computational package `DistrNN.jl` in JULIA available at https://github.com/barunik/DistrNN.jl that allows one to obtain our measures on data the researcher desires. The data that support the findings of this study are available from the corresponding author upon reasonable request.

*Corresponding author, Institute of Economic Studies, Charles University, Opletalova 26, 110 00, Prague, CR and Institute of Information Theory and Automation, Czech Academy of Sciences , Pod Vodarenskou Vezi 4, 18200, Prague, Czech Republic. E-mail: barunik@utia.cas.cz Web: barunik.github.io

**Institute of Economic Studies, Charles University, Opletalova 26, 110 00, Prague, CR and Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, Pod Vodarenskou Vezi 4, 18200, Prague, Czech Republic. E-mail: hanusl@utia.cas.cz

1 Introduction

Uncertainty at every stage of decision making is key to understanding for financial operations, central and retail banking, as well as for researchers and practitioners trying to minimize the risk of their decisions, make appropriate plans, and assist in the design and implementation of economic policy. Yet even after decades of research, a conditional mean forecast often serves economists as a convenient tool for measuring the central tendency of a target variable, or simply as a best guess about the future outcomes of a variable. The associated variance forecast often serves as a best estimate of uncertainty and future risk. However, such predictions are not fully informative when a decision maker is faced with asymmetric dependence of her loss on outcomes of possibly non-Gaussian variables. Since uncertainty is a key ingredient in economic decision making, the shift to probabilistic forecasting also shifts our hopes to obtaining better expectations about entire distributions of economic variables. A non-trivial question is how to make such forecasts, especially using available data.

Traditionally, distributional forecasts are made using time series models, surveys, or are collected in real time.[1] With rapid improvements in the accessibility and availability of large datasets, we believe that one can significantly improve the description of uncertainty using methods that focus on learning patterns from data. In line with recent efforts by economists to move away from exclusive reliance on models to machine learning approaches (Athey and Imbens, 2019) where it makes sense to use data and improve our understanding of the problem (Mullainathan and Spiess, 2017), we propose to use deep machine learning to learn the complex patterns in the data and return to the user a prediction of a full distribution.

Our main contribution to the literature is that we propose how to use deep learning techniques as a useful tool for approximating and predicting conditional distributions in data-rich environments. Our distributional neural network takes advantage of deep learning (especially recurrent neural networks), is capable of predicting a distribution of a time series, and allows the use of large numbers of variables. We frame our approach as a multi-output neural network that provides approximate probability functions of the distribution. The novelty of our approach thus lies in learning the conditional distribution using (deep) recurrent networks from large data sets. The proposed network is also able to capture the time variation of distributions, for example when dealing with highly dynamic data, and to recover longer and more complex time dependence structures present in the data. Our framework generalizes binary choice models (Foresi and Peracchi, 1995; Anatolyev

[1] Methods for constructing distributional forecasts are reviewed in a special issue on "Density Forecasting in Economics and Finance" (Timmermann, 2000) and "Probability Forecasting" (Gneiting, 2008) for the collection of papers.

and Baruník, 2019) and, together with state-of-the-art machine learning tools, forms a new toolkit for economists interested in describing future uncertainty of economic variables. An important contribution is also our novel approach to constructing an objective function that adjusts the monotonicity of distributional forecasts by introducing a penalty function for deviating from monotonic behavior.

A key idea of (machine) learning, which can be thought of as inferring plausible models to explain observed data, has recently attracted a number of researchers who document how learning patterns from data can be useful (Mullainathan and Spiess, 2017; Sirignano et al., 2016; Gu et al., 2020; Heaton et al., 2017; Tobek and Hronec, 2020; Bianchi et al., 2021; Israel et al., 2020; Iworiso and Vrontos, 2020; Feng et al., 2018; Goulet Coulombe et al., 2022; Berrisch and Ziel, 2022). A burgeoning literature and an increasing number of applications in economics focus mostly on cross-sectional data and ultimately on point forecasts. While machines can use such models to make predictions about future data, uncertainty plays a fundamental role. At the same time, data, which is a key component of all machine learning systems, is useless on its own unless knowledge or inferences are extracted from it. Shifting the focus from point forecasting to probabilistic forecasting using big data is an essential next step for economists who want to explore what computer science has to offer.

We contribute to this debate by developing a machine learning strategy to predict the probability distributions. We argue that deep learning, in particular recurrent neural networks, provides a useful tool for predicting distributions without the need for model specification, by learning the distributions from the data. While the ability to outperform alternative methods on specific data sets in terms of out-of-sample predictive power is valuable in practice, such performance is rarely explicitly recognized as a goal to be addressed in econometrics. As Mullainathan and Spiess (2017) points out, some substantive problems are naturally cast as prediction problems, and assessing their goodness of fit on a test set may be sufficient for the purposes of the analysis. We believe that the task of predicting distributions in a data-rich environment is one such important problem in economics where machine learning could be helpful to a researcher, policymaker or practitioner.

What are the specific challenges of probabilistic forecasting of economic variables? Time series such as stock returns, electricity prices, traffic data or macroeconomic series have distributions that cannot be captured by a convenient Gaussian distribution and are therefore not fully characterized by means and variances. These distributions have heavy tails, are asymmetric and often violate stationarity. The data also contain irregularities, hard-to-predict peaks and regime shifts. Therefore, complete information about the probability of future outcomes given past information is needed, which can be mapped into different representations to construct prediction intervals or probability distribution functions that reflect the data. Such a fully approximated distribution function provides comprehensive

information about the uncertainty of future observations.

Why should we believe that machine learning can improve probabilistic forecasting? Classical time series econometrics (Box et al., 2015; Hyndman et al., 2008) focuses mainly on predetermined autocorrelation or seasonality structures in data that are parameterized. With large amounts of time series available to researchers, these methods quickly become infeasible and unable to explore more complex data structures. Bearing in mind the famous adage that *"all models are wrong..., but some of them are useful."* (Box et al., 1987), modern machine learning methods can easily overcome these problems. Being a powerful tool for approximating complex and unknown data structures (Kuan and White, 1994), these methods can be useful in a number of application problems where data contain rich information structure and we cannot describe it satisfactorily by a simplifying model. Overcoming the long-standing problem of computational intensity of such data-driven approach with advances in computer science adds to the temptation in using these methods to address new problems such as distribution prediction.

Two different and important economic datasets illustrate how the machine learning approach to probabilistic forecasting can help a decision maker facing uncertainty. First, we study the set of the most liquid US stock returns, which have asymmetric and heavy-tailed, dynamically evolving distributions that are difficult to predict. Second, we use deep learning to construct data-driven macroeconomic fan charts that reflect information contained in a large number of variables. Such data-rich fan charts are the first of their kind to reflect high-dimensional information from 216 relevant variables and are of great importance to policy makers as they reflect the structures in the data and are not influenced by the choice of model. A forecasting model, on the other hand, is learned from the data. Moreover, such data-rich fan charts cannot be obtained using traditional methods.

2 Probabilistic Forecasting via Deep Learning

Consider an economic time series y_t collected over $t = 1 \ldots, T$. The main objective is to approximate the conditional cumulative distribution function $F(y_{t+h}|\mathcal{I}_t)$ as closely as possible and use it for a h-step ahead probabilistic forecast made at time t with information $\mathcal{I}_t$ containing past values of y_t and possibly past values of other exogenous observable variables.

Consider a partition of the support of y_t by $p > 1$ fixed thresholds corresponding to a set of empirical α_j quantiles $\{q^{\alpha_j}\}_{j=1}^{p}$, where $0 < \alpha_1 < \alpha_2 < \ldots < \alpha_p < 1$ are p regularly spaced probability levels on a unit interval $[0, 1]$. These partitions are also time-varying, so in general the elements of the partition are implicitly indexed by t.

The main goal then is to approximate a collection of conditional probabilities corre-

sponding to the empirical quantiles such as

$$\left\{F(q^{\alpha_1}),\ldots,F(q^{\alpha_p})\right\} = \left\{\Pr\left(y_{t+h} \leq q^{\alpha_1}|\mathcal{I}_t\right),\ldots,\Pr\left(y_{t+h} \leq q^{\alpha_p}|\mathcal{I}_t\right)\right\}$$

for the collection of thresholds $1,\ldots,p$. One convenient way of estimating such quantities is distributional regression. Foresi and Peracchi (1995) noted that several binary regressions serve as a good partial description of the conditional distribution. To estimate conditional distribution, one can simply consider distribution regression model

$$\Pr(y_{t+h} \leq q^{\alpha_j}|\mathcal{I}_t) = \Lambda(\beta_j), \tag{1}$$

where $\Lambda : z \to [0,1]$ is a known (monotonically increasing) link function, such as logit, probit, linear, log-log functions[2] and $\beta(.)$ is an unknown function-valued parameter to be determined. In contrast to estimating separate models for separate thresholds, Chernozhukov et al. (2013) considered continuum of binary regressions, and argued it provides a coherent and flexible model for the entire conditional distribution as well as useful alternative to Koenker and Bassett Jr (1978)'s quantile regression. Alternatively, Anatolyev and Baruník (2019) propose to tie the coefficients of predictors in an ordered logit model via smooth dependence on corresponding probability levels. While being able to forecast entire distribution and keeping $0 < F_j < 1$ and $0 < F_1(.) < F_2(.) < \ldots F_p(.) < 1$, the approach still depends on heavy parametrization suited for a specific problem of the time series considered making it an infeasible approach for larger number of variables.

Such probabilistic forecasts are highly dependent on model parametrization and quickly become infeasible as the number of covariates increases. Stationarity of the data at hand is also a requirement that complicates forecasting as it is difficult to achieve in many cases. In sharp contrast to such an approach, we propose a more flexible and general path to distributional regression via deep learning. We propose a novel multiple output neural network, which we refer to as a distribution neural network (DistrNN). Our approach aims to uncover non-linear and mostly complex relationships of time series without specifying a strict parametric structure and without requiring strict assumptions about the data, while focusing on the out-of-sample predictive power of the model.

Deep feedforward networks, often called feedforward neural networks or multilayer perceptrons, are at the heart of deep learning models and are universal approximators that can learn any functional relationship between input and output variables with sufficient data Kuan and White (1994). As a class of supervised learning methods, these approaches are used for classification, recognition and prediction. While increasingly popular in economics for solving specific problems Athey and Imbens (2019); Mullainathan and Spiess

[2]As discussed by Chernozhukov et al. (2013), log-log link nests the Cox model making distribution regression important.

(2017), probabilistic forecasting has not yet been explored in the literature.

This motivates us to reformulate distributional regression into a more general and flexible distributional neural network. The functional form of the new network is driven by the data and we can relax assumptions about the distribution of the data, the parametric model as well as the stationarity of the data. The proposed distributional neural network, as a feed-forward network, is a hierarchical chain of layers representing high-dimensional and/or non-linear input variables with the aim of predicting the target output variable. Importantly, we approximate the conditional distribution function with multiple outputs of the network as a set of joint probabilities.

As a first step, we exchange a known link function from Eq. 1 for an unknown general function $\mathfrak{g}$ that will be approximated by a neural network:

$$\Pr\left(y_{t+h} \leq q^{\alpha_j} | \mathcal{I}_t\right) = \mathfrak{g}_j(\cdot). \tag{2}$$

Next, we consider a set of probabilities corresponding to $0 < \alpha_1 < \alpha_2 < \ldots < \alpha_p < 1$ being p regularly spaced levels that characterize conditional distribution function using set of predictors $z_t = (y_t, x_t^1, ..., x_t^n)^\top$, and model them jointly as

$$\left\{ \Pr\left(y_{t+h} \leq q^{\alpha_1} | z_t\right), \ldots, \Pr\left(y_{t+h} \leq q^{\alpha_p} | z_t\right) \right\} = \mathfrak{g}_{W,b}(z_t), \tag{3}$$

where $\mathfrak{g}_{W,b}$ is a multiple output neural network with L hidden layers that we name as distributional neural network:

$$\mathfrak{g}_{W,b}(z_t) = g^{(L)}_{W^{(L)},b^{(L)}} \circ \ldots \circ g^{(1)}_{W^{(1)},b^{(1)}}(z_t), \tag{4}$$

where $W = \left(W^{(1)}, \ldots, W^{(L)}\right)$ and $b = \left(b^{(1)}, \ldots, b^{(L)}\right)$ are weight matrices and bias vector. Any weight matrix $W^{(\ell)} \in \mathbb{R}^{m \times n}$ contain m neurons as n column vectors $W^{(\ell)} = [w^{(\ell)}_{\cdot,1}, \ldots, w^{(\ell)}_{\cdot,n}]$, and $b^{(\ell)}$ are thresholds or activation levels which contribute to the output of a hidden layer allowing the function to be shifted.

It is important to note that in sharp contrast to the literature, we consider a multiple output (deep) neural network to characterize collection of probabilities. Before discussing the details of estimation that allows us to keep monotonicity of probabilities, we illustrate the framework. Figure 1 illustrates how $l \in 1, ..., L$ hidden layers transform input data into a chain using collection of non-linear activation functions $g^{(1)}, \ldots, g^{(L)}$. A commonly used activation functions, $g^{(\ell)}_{W^{(\ell)},b^{(\ell)}}$, used as

$$g^{(\ell)}_{W^{(\ell)},b^{(\ell)}} := g_\ell\left(W^{(\ell)} z_t + b^{(\ell)}\right) = g_\ell\left(\sum_{i=1}^{m} W_i^{(\ell)} z_t + b_i^{(\ell)}\right)$$

are a sigmoid $g_\ell(u) = \sigma(u) = 1/(1+\exp(-u))$, rectified linear units $g_\ell(u) = \max\{u, 0\}$, or $g_\ell(u) = \tanh(u)$. In case $\mathfrak{g}_{W,b}(z_t)$ is non-linear, neural network complexity grows with increasing number of neurons m, and with increasing number of hidden layers L and we build a deep neural network. We use activation function $g^{(L)}(\cdot) = \sigma(\cdot)$ to transform outputs to probabilities. Note that for $L = 1$, neural network becomes a simple logistic regression.

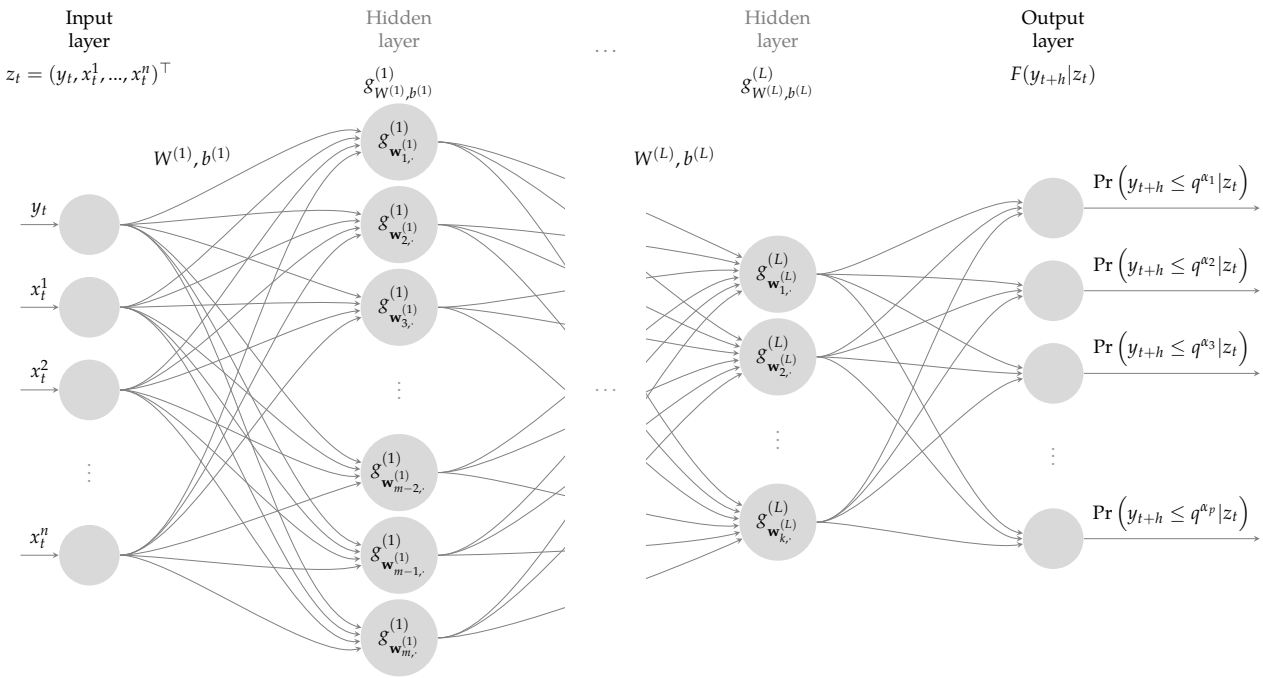

Figure 1. Distributional (Deep) Feed-forward Network.

An illustration of a multiple output (deep) neural network $\mathfrak{g}_{W,b}(z_t)$ to model the collection of conditional probabilities $\left\{ \Pr\left(y_{t+h} \leq q^{\alpha_1}|z_t\right), \ldots, \Pr\left(y_{t+h} \leq q^{\alpha_p}|z_t\right) \right\}$ with set of predictor variables $z_t = (y_t, x_t^1, ..., x_t^n)^\top$. With large number of hidden layers L the network is deep.

2.1 (Deep) Recurrent Neural Networks

Predictors used by economists often evolve over time, and hence traditional neural networks assuming independence of data may not approximate relationships sufficiently well. Instead, a Recurrent Neural Network (RNN) that takes into account time series behavior may help in the prediction task. Taking into account sequential nature of data that evolve over time and possess an auto-correlation structure, RNNs are more suitable for many economic problems. In contrast to plain neural networks, hidden layers in recurrent networks are being updated in a recurrence for every time step of the sequence meaning that the weights of the network are shared over the sequential data, and hidden states remember the time structure.

Formally, RNNs transform a sequence of input variables to another output sequence

with lagged (memory) hidden states

$$h_t = g(W_h h_{t-1} + W_z z_t + b_0). \tag{5}$$

Figure 2 illustrates distinctions of weights where dashed lines correspond to W_h and solid lines to W_z. Intuitively, RNN is a non-linear generalization of an autoregressive process

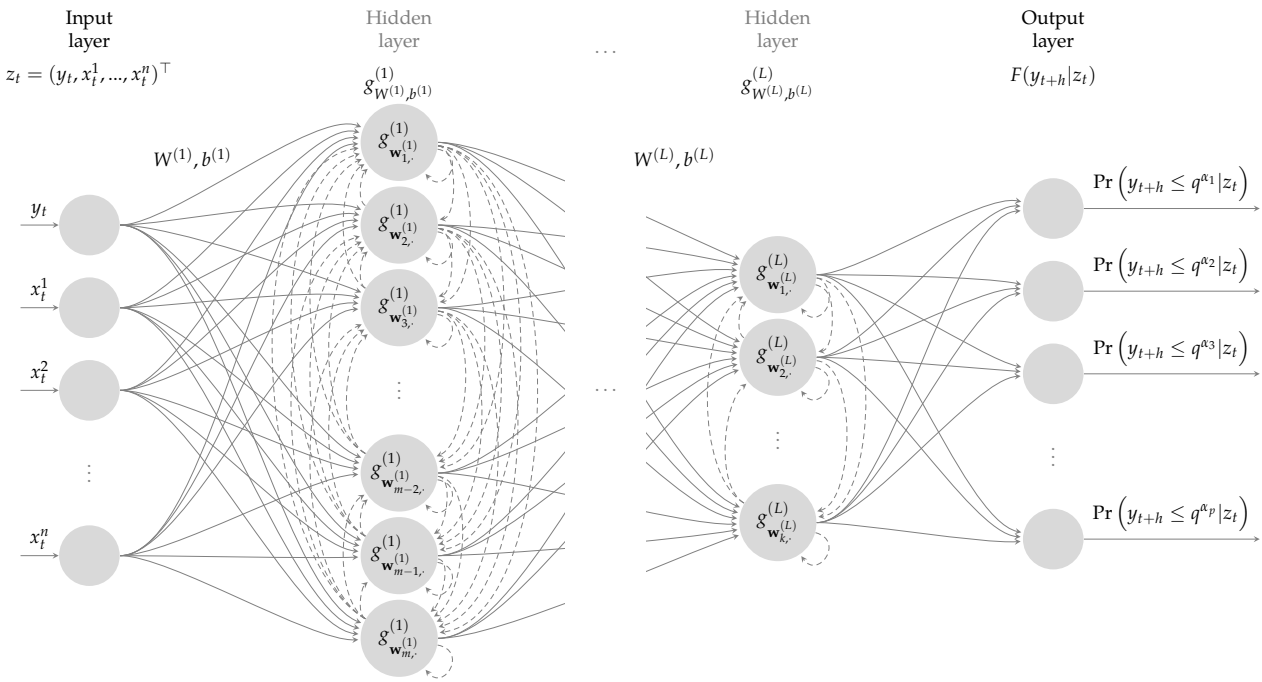

Figure 2. Distributional (Deep) Recurrent Network.

A depiction of a deep recurrent neural network $\mathfrak{g}_{W,b}(z_t)$ capturing the relationship between all nodes (solid connections) and recurrent paths (dashed connection) in the network at time t to model the collection of conditional probabilities $\left\{ \Pr\left(y_{t+h} \leq q^{\alpha_1}|z_t\right), \ldots, \Pr\left(y_{t+h} \leq q^{\alpha_p}|z_t\right) \right\}$ with set of predictor variables $z_t = (y_t, x_t^1, ..., x_t^n)^\top$.

where lagged variables are transformations of the observed variables. Nevertheless, the structure is only useful when the immediate past is relevant. In case the dynamics are driven by events that are further back in the past, the nodes of the network require even more complex structure.

2.1.1 Long Short-term Memory (LSTM)

As a particular form of recurrent networks, an LSTM provides a solution to the short memory problem by incorporating memory units into the structure (Hochreiter and Schmidhuber, 1997) and capture potentially long time dynamics in the time series. Memory units allow the network to learn when to forget previous hidden states and when to update hidden states given new information. Specifically, LSTM unit has five components: an input gate, a hidden state, a memory cell, a forget gate, and output gate. The memory

cell unit combines the previous time step memory cell unit which is modulated by the forget and input modulation gates together with the previous hidden state, modulated by the input gate. These components enable an LSTM to learn very complex long-term and temporal dynamics that a vanilla RNN is not capable of. Additional depth in capturing the complexity of a tie series can be added by stacking LSTM on top of each other.

Formally, at each time step a new memory cell c_t is created taking current input z_t and previous hidden state h_{t-1} and it is then combined with forget gate that controls an amount of information kept in the hidden state as

$$
\begin{aligned}
h_t &= \sigma\left(\underbrace{W_h^{(o)} h_{t-1} + W_z^{(o)} = z_t + b_0^{(o)}}_{\text{output gate}}\right) \circ \tanh(c_t) \\
c_t &= \sigma\left(\underbrace{W_h^{(g)} h_{t-1} + W_z^{(g)} z_t + b_0^{(g)}}_{\text{forget gate}}\right) \circ c_{t-1} + \sigma\left(\underbrace{W_h^{(i)} h_{t-1} + W_z^{(i)} + b_0^{(i)}}_{\text{input gate}}\right) \circ \tanh(k_t).
\end{aligned}
$$

The term $\sigma(\cdot) \circ c_{t-1}$ introduces the long-range dependence, k_t is new information flow to the current cell. The forget gate and input gate states control weights of past memory and new information. In the Figure 2, c_t is the memory pass through multiple hidden states in the recurrent network.

2.2 Loss Function

Since we aim to estimate the cumulative distribution function that is a non-decreasing function bounded on $[0, 1]$, we need to design an objective function that minimizes differences between targets and estimated distribution as well as imposes non-decreasing property of the output. Since the problem is essentially a more complex classification problem closely related to logistic regression, we use a binary cross-entropy loss function. Such a choice of scoring rule dates back to Good (1952) and is widely used in the classification problems. To order the predicted probabilities, we introduce a penalty to the multiple output classification problem.

The loss function is then composed of two parts: traditional binary cross-entropy and a

penalty adjusting monotonicity of predicted output:

$$\begin{aligned}\mathcal{L} \;=\; & \underbrace{-\frac{1}{T}\sum_{t}^{T}\frac{1}{p}\sum_{j}^{p}\left(\mathbb{I}_{\{y_{t+h}\leq q^{\alpha_j}\}}\log\left\{\widehat{\mathfrak{g}}_{W,b,j}(z_t)\right\}+\left(1-\mathbb{I}_{\{y_{t+h}\leq q^{\alpha_j}\}}\right)\log\left\{1-\widehat{\mathfrak{g}}_{W,b,j}(z_t)\right\}\right)}_{\text{binary cross-entropy}} \\ & + \underbrace{\lambda_m\sum_{t}^{T}\sum_{j=1}^{p-1}\left(\widehat{\mathfrak{g}}_{W,b,j}(z_t)-\widehat{\mathfrak{g}}_{W,b,j+1}(z_t)\right)_+}_{\text{monotonicity penalty}} \qquad (6)\end{aligned}$$

where $(u)_+$ is a rectified linear units function, ReLU, $(u)_+ = \max\{u, 0\}$, and $\mathbb{I}_{\{.\}}$ is an indicator function. The monotonicity penalty is controlled by the penalty parameter λ_m and punishes violations of CDF monotonicity condition. The violations are when exist positive differences between two neighboring values, j and $j+1$, of CDF. Furthermore, it should be noted that, in addition to its simplicity, ReLU is employed for convenience reasons in order to facilitate its general use.[3]

2.3 Networks Design and Estimation Steps

The high dimensionality and non-linearity of the problem make estimation of a deep neural network a complex task. The selection of optimal parameters is crucial for estimation, as it ensures the desired performance and avoids potential risks such as overfitting or convergence issues. Moreover, it is necessary to exercise caution and make specific, careful choices in order to minimize the risks associated with each problem and data set. This section presents a comprehensive overview of the model architectures and their respective estimations.

2.3.1 Learning, Regularization and Hyper-parameters

The selection of hyper-parameters in combination with regularisation methods plays a pivotal role in reducing the risk associated with estimation. In particular, the ReLU activation function is employed to introduce non-linearity to the problem, thereby facilitating the convergence of the optimisation algorithm. For the learning process, we use adaptive gradient algorithm, Adam (Kingma and Ba, 2014) and its modification AdamW (Loshchilov and Hutter, 2019) that allows for regularization by decoupling the weight decay from the gradient-based update. The regularization is close to L_2-regularization with improved results.[4]

[3]This option allows the use of the GPU, thereby enhancing the computational capabilities for more complex problems. It is not advisable to utilise in-house or non-optimized functions for the GPU, as $(u)_+$ is a common element in libraries designed for GPUs.

[4]We keep decay of momentum parameters β_1, β_2 constant and at default values throughout all estimations, $\beta_1 = 0.9$, $\beta_2 = 0.999$.

Further, the hyper-optimization algorithms employed utilize a random search over a grid/cube of parameter ranges, which are specific to a given experiment. In some instances, it may be necessary to search over the entire parameter grid, testing all possible combinations. This approach can be costly. The hyper-optimization approach [5] is used to select the learning rate of the optimizer, η, the weight decay parameter of AdamW, λ_W, and a Dropout parameter that regularizes models, as described in Srivastava et al. (2014). This approach represents an efficient method for performing model averaging with neural networks. In particular, the dropout parameter is employed to disable a proportion of nodes in the layer of the network at which it is applied, with this proportion being defined by the parameter ϕ, which is constrained to the interval $(0.0, 1.0)$. In the training phase, the model is presented with a specified number of epochs during which it is expected to learn from the data. The number of epochs is contingent upon the size of the data set and the batch size employed in the estimation process. Nevertheless, we also employ the early stopping technique, which assists in regularization and prevents over-fitting. The early stopping criterion represents the minimum number of epochs that the model is required to complete before it is permitted to stop learning.

2.3.2 Code Implementation

We have estimated our models on 48 core Intel® Xeon® /i7 Gold 6126 CPU@ 2.60GHz, 128 GB of memory, and GeForce 3090 GPU. We implement the models using `Flux.jl` (Innes et al., 2018) package in JULIA 1.6.0. language.

2.3.3 Data Preparation and Information set

To predict the distribution function of a time series y_t with observations $y_1, \ldots, y_T$, we split our time series into several parts. The first partitioning creates, as known in time-series literature, in-sample $[1 : t_0]$ and out-of-sample $[t_0 + 1 : T]$ subsamples. Equivalently, in machine learning jargon, train and test sets. Test subsample is never available to the learning algorithm while training the model. We further divide the train subsample into training and validation sets, which are used to cross-validation of our model and model's parameters selection. The model selection is based on the value of the loss function on the validation subsample(s), mainly the binary cross-entropy loss.

One of the crucial parts in estimation of distributional neural networks is the information set. The information set $\mathcal{I}_{t_0}$ is based on the past observation available at time t_0. This is the maximal time-span providing historical information, in our case, up to the last observation of the validation subsample. The importance here lays in finding the empirical quantiles, q^{α}, corresponding to the set of probabilities $\{\alpha_1, \ldots, \alpha_p\}$, which are used to

[5] We refer to Julia package HyperOpt.jl (https://github.com/baggepinnen/Hyperopt.jl)

build the sequence of target values. Given the information about $\{y_t\}_{t=1}^{t_0}$ and the empirical quantiles, we are able to model the distribution conditional on the information set up to time t_0. Given the information set, we face the problem with non-variation or updating the conditional empirical quantiles for the future distributions. Although, we do not assume a shape of the distribution, we assume, to some extent, small level of shift in the distribution. Further, choice of empirical quantiles and probability levels faces the same problem as quantile regression when it comes to small samples of data.

3 Empirical Application: Conditional Distributions of Asset Returns

The vast majority of studies that focus on predicting conditional return distributions characterize the cumulative conditional distribution by a collection of conditional quantiles (Engle and Manganelli, 2004; Žikeš and Baruník, 2016). In contrast, Leorato and Peracchi (2015) argue that a collection of conditional probabilities describing the cumulative distribution function using a set of separate logistic regressions (Foresi and Peracchi, 1995) provides a better approach. The following decades resulted in a few contributions exploring distributional regressions (Chernozhukov et al., 2013; Fortin et al., 2011; Rothe, 2012), including attempts to overcome the problem of monotonicity of predictions (Anatolyev and Baruník, 2019) using ordered logistic parametrization. Another important strand of literature focuses on Bayesian forecasting, where uncertainty is automatically characterized by probabilities (Geweke and Whiteman, 2006; Lahiri et al., 2010). At the same time, the literature in computer science attempts to apply machine learning to the prediction of distributions. These attempts are similar to traditional methods and are usually based on approximating a pre-specified distribution, such as the first two moments.[6] To the best of our knowledge, the literature has not yet moved to fully non-parametric approaches to approximate data structures in the context of distributional forecasting in economics and finance.

[6] Duan et al. (2019) applies the natural gradient boosting algorithm to estimate parameters for conditional probability distribution, while assuming homoskedasticity. Salinas et al. (2020) build an autoregressive recurrent neural network, which learns mean and standard deviation for Gaussian, and mean and shape parameter for Negative binomial. Lim and Gorse (2020) classifies price movements for high-frequency trading via deep probabilistic modelling when optimizing parameters of different families of distribution. Although similarly to Salinas et al. (2020), Chen et al. (2020) proposes to use a deep temporal convolutional neural network to estimate parameters of Gaussian distribution to model probabilistic forecast, and they further propose to use the same architecture for non-parametric estimation of quantile regression. The second approach is distribution-free and can produce more robust results. Another study forecasts distributions via direct quantiles using recurrent neural network, Wen et al. (2017) also perform a multi-horizon predictions. Quantile function represented by spline combined with recurrent neural network proposed by Gasthaus et al. (2019) is a distribution-free approach with objective function based on CRPS score (Gneiting and Raftery, 2007) constructed with respect to monotonicity of quantile function. Hu et al. (2019) build deep neural networks to obtain distribution-free probability distribution where one of the steps in the procedure is to obtain cumulative distribution estimates. Januschowski et al. (2020) provide a detailed discussion about ML methods for forecasting. The text discusses way of distinction between "statistical" and "ML" methods adapted in time.

Stock return data are notorious for their heavy tails and low signal-to-noise ratio (Fama, 1965; Israel et al., 2020). Despite the large literature uncovering these empirical properties, few studies attempt to forecast the distribution of returns.[7]. Among the few, Anatolyev and Baruník (2019) parameterize a simple ordered logit to provide the distribution forecasts.

Here, we aim to develop a machine learning based alternative that is capable of exploring a large number of informative variables. We compare the forecasts with the benchmark Anatolyev and Baruník (2019) (hereafter AB) model to see how well the machine learning approach approximates the parametrized model, but we mainly focus on using more variables that classical models cannot explore due to infeasibility associated with large parameter space.

This application is therefore a good complement to the previous one, where we used machine learning for multi-variable forecasting using big data. Liquid equity returns, on the other hand, are notoriously difficult to predict, so even a small improvement is valuable.

3.1 Data and Estimation

Our dataset includes 29 most liquid U.S. stocks[8] of S&P500. The main reason for this particular choice is comparability of the results with Anatolyev and Baruník (2019). The daily data covers the period from July 1, 2005 to August 31, 2018. We preprocessed the data to eliminate possible problems with liquidity or biases caused by weekend or bank holidays. The final sample period contains 3261 observations.

We start building the models using the same predictors as in Anatolyev and Baruník (2019) to make a direct comparison of the model forecasts. Specifically, they use $\text{Ind}_t = \mathbb{I}_{\{r_t \leq q^{\alpha_j}\}}$ and $\text{LogVol}_t = \ln(1 + |r_t|)$ as a proxy to a volatility measure. We will refer to this first choice as the *AB predictors*. Next, we prepare five realized measures from one-minute intra-day high frequency data obtained from TickData.[9] The realized measures for each of 29 asset returns are realized volatility, skewness, kurtosis, and positive and negative semi-variances labelled as RVol_t, RSkew_t, RKurt_t, RSemiPos_t, and RSemiNeg_t. These are informative about returns distribution and should help the forecast. We will refer to this set of as *RM predictor*

In the third model, we combine both sets of predictors to estimate the conditional distribution of returns, as they are both informative. As the realised measures contain information about higher moments of the return distribution, they could improve the pre-

[7]The literature focusing on value-at-risk forecasting has a special interest in a chosen quantile of the return distribution, mostly the left tail (Engle and Manganelli, 2004)

[8]Assets selected in the sample: AAPL, AMZN, BAC, C, CMCSA, CSCO, CVX, DIS, GE, HD, IBM, INTC, JNJ, JPM, KO, MCD, MRK, MSFT, ORCL, PEP, PFE, PG, QCOM, SLB, T, VZ, WFC, WMT, XOM.

[9]www.tickdata.com

AB predictors	RM predictors	AB+RM predictors
Ind_t	-	Ind_t
LogVol_t	-	LogVol_t
-	RVol_t	RVol_t
-	RSkew_t	RSkew_t
-	RKurt_t	RKurt_t
-	RSemiPos_t	RSemiPos_t
-	RSemiNeg_t	RSemiNeg_t

Table 1. Sets of predictors used in the three models
Note: The indicator Ind_t contains J columns of dummy variables.

diction of the conditional return distribution. At the same time, including these predictors in the original (benchmark) ordered logit model of Anatolyev and Baruník (2019) would result in an over-parameterised model that is not feasible. This is an important caveat, as our approach provides a flexible and more general way of predicting distributions in data-rich settings, while exploring possible non-linearity in the data. Table 1 summarises the predictors used in the three models.

Prior to estimation, we normalize the input data to a suitable range, which makes it easier for the algorithm to find a better optimum. This is a standard procedure in the learning process, as the optimization works on closer ranges while learning in the network structure. Furthermore, we divide the data into training and test parts with a ratio of 0.9 and 0.1, respectively, to 2934 and 327 observations. First, we search for the best set of hyperparameters on the training window, where we perform a fourfold rolling window forward validation scheme. The model is trained and validated during the hyperparameter search on each split composed of 90% and 10% partitions - training and validation. Using a rolling window of size 2934, we predict out-of-sample forecasts one step ahead, H = 1. The window size is equal to the size of the training sample, $t_0 = 2934$. On the first rolling window, the training part, we search the grid for the best parameter set using random and Latin hypercube search algorithms. Table A2 in Appendix B details the parameter ranges for the learning rate, η, the dropout parameter, ϕ, and the weight decay penalty parameter, and λ_W, on which the hyper-optimization algorithm searches for the best model hyper-parameters in the space of 50 combinations. We use the ensemble method for forecasting, i.e. for each rolling window step, the best model is trained three times, given the best model hyperparameters. Three predictions are obtained and an average distribution prediction is made for all t in out-of-sample, $t \in [2935 : 3261]$. In addition, we use the additional regularization technique of early stopping and table A2 in the appendix B also provides the number of epochs allowed to train. This is the number of epochs where the model is patient with the algorithm and

waits for improvement. Finally, we take the model with the best validation loss and use it to predict out-of-sample distributions. We also study an effect of complexity, which specifies the size of the neural networks. We set the number of nodes from a shallow to a deeper DistrNN to [128], [128, 64] and [128, 64, 32]. The output size of the final layer of the network is $p = 10$, which correspond to probability forecasts that approximate the conditional distribution of excess returns given the information set $\mathcal{I}_t$.

3.2 Statistical Evaluation Measures

We evaluate our probabilistic forecasts using several measures. First, we evaluate the accuracy of the forecasts using the mean square prediction error, calculated as

$$MPSE = \frac{1}{T - t_0} \sum_{t=t_0+1}^{T} \frac{1}{p} \sum_{j=1}^{p} \left(\mathbb{I}_{\{y_{t+h} \leq q^{\alpha_j}\}} - \widehat{\mathfrak{g}}_{W,b,j}(z_t) \right)^2, \tag{7}$$

where the out-of-sample predicted outputs $\widehat{\mathfrak{g}}_{W,b,j}(z_t)$ is a matrix keeping dimension of time $[t_0 + 1, \ldots, T]$ and conditional probability levels $\{1, \ldots, p\}$.

To evaluate the compatibility of a cumulative distribution functions with an individual time series observations we use the continuous ranked probability score (CRPS, Matheson and Winkler (1976), Hersbach (2000)):

$$CRPS_t = -\int_{-\infty}^{\infty} \left(\widehat{\mathfrak{g}}_{W,b}(z_t) - \mathbb{I}_{\{y_{t+h} \leq y\}} \right)^2 dy, \tag{8}$$

where the conditional CDF $\widehat{\mathfrak{g}}_{W,b}(z_t)$ is obtained by CDF interpolation (see Appendix A) while the integral is computed numerically using the Gauss-Chebyshev quadrature formulas (Judd (1998), section 7.2) with 300 Chebychev quadrature nodes on $[2y_{min}, 2y_{max}]$. CRPS score is of the highest value when the distributions are equal. We obtain an average CRPS score of the out-of-sample forecast as $CRPS_{OOS} = \frac{1}{T} \sum_{t=t_0+1}^{T} CRPS_t$.

Another measure for the distributional forecast accuracy is Brier score (Gneiting and Raftery, 2007). At time t, it calculates a squared difference of binary realization and the probability forecast,

$$B_t = -\sum_{j=1}^{p+1} \left(\mathbb{I}_{\{q^{\alpha_{j-1}} < y_t \leq q^{\alpha_j}\}} - \widehat{\Pr}\{q^{\alpha_{j-1}} < y_t \leq q^{\alpha_j}\} \right)^2. \tag{9}$$

We also compute the average value of the Brier score for the out-of-sample period.

We compare proposed models using relative predictive performance of two models, M_1/M_2, where M_i is particular measure (MPSE, CRPS, or Brier) corresponding model i. The model M_1 performs better when the ratio is lower than one.

3.3 Results

Table 2, Figure 3 and Figure A1 in Appendix B provide all out-of-sample results with the horizon $h = 1$ comparing sizes of various machine learning models and different variables used as predictors. The performance for all measures is put relative to the maximum likelihood ordered logit model, an AB benchmark of Anatolyev and Baruník (2019).

Data	Model	**MSPE**	**Bin.CE**	**CRPS**	**Brier**
AB	NN:128	14/29	14/29	20/29	11/29
AB	NN:128x64	13/29	11/29	15/29	11/29
AB	NN:128x64x32	8/29	7/29	8/29	11/29
RM	NN:128	25/29	25/29	27/29	25/29
RM	NN:128x64	25/29	24/29	27/29	25/29
RM	NN:128x64x32	25/29	24/29	26/29	25/29
AB+RM	NN:128	22/29	19/29	27/29	19/29
AB+RM	NN:128x64	22/29	21/29	26/29	21/29
AB+RM	NN:128x64x32	23/29	23/29	27/29	23/29

Table 2. Results according to different scores, assets
The table shows performance of three models with different sizes and of three different input features. All results are benchmarked to Ordered Logit of Anatolyev and Baruník (2019). The number indicates for how many assets given model performs better.

Overall, we document that machine learning is able to predict the conditional distribution of asset returns well and, given informative variables, it delivers improved predictions. In particular, we observe an average out-of-sample improvement of 1% in the MSPE for all assets studied compared to the parametric models. The improvement is even greater in terms of the continuous rank probability score, which assesses the compatibility of the predicted and data distributions.

It is important to note that in financial forecasting, the relationship between statistical and economic gains from predictions is not trivial. Campbell and Thompson (2008); Rapach et al. (2010) Note that a seemingly small statistical improvement can yield large benefits in practice, which has recently been confirmed for expected returns predicted by machine learning (Gu et al., 2020; Babiak and Baruník, 2020). Thus, an average 1 % improvement in the out-of-sample forecasts we document is likely to be of interest to a practitioner who builds portfolios based on our forecasts. While it is tempting to explore such a strategy, it is far beyond the scope and space of this text.

More specifically, Table 2 shows that neural networks of all sizes on average brings improved performance in comparison to parametrized AB model when the same set of predictors is used in approximately half of tested stocks. This result suggests that data does not contain any further non-linearities that are not captured by a parametric AB model,

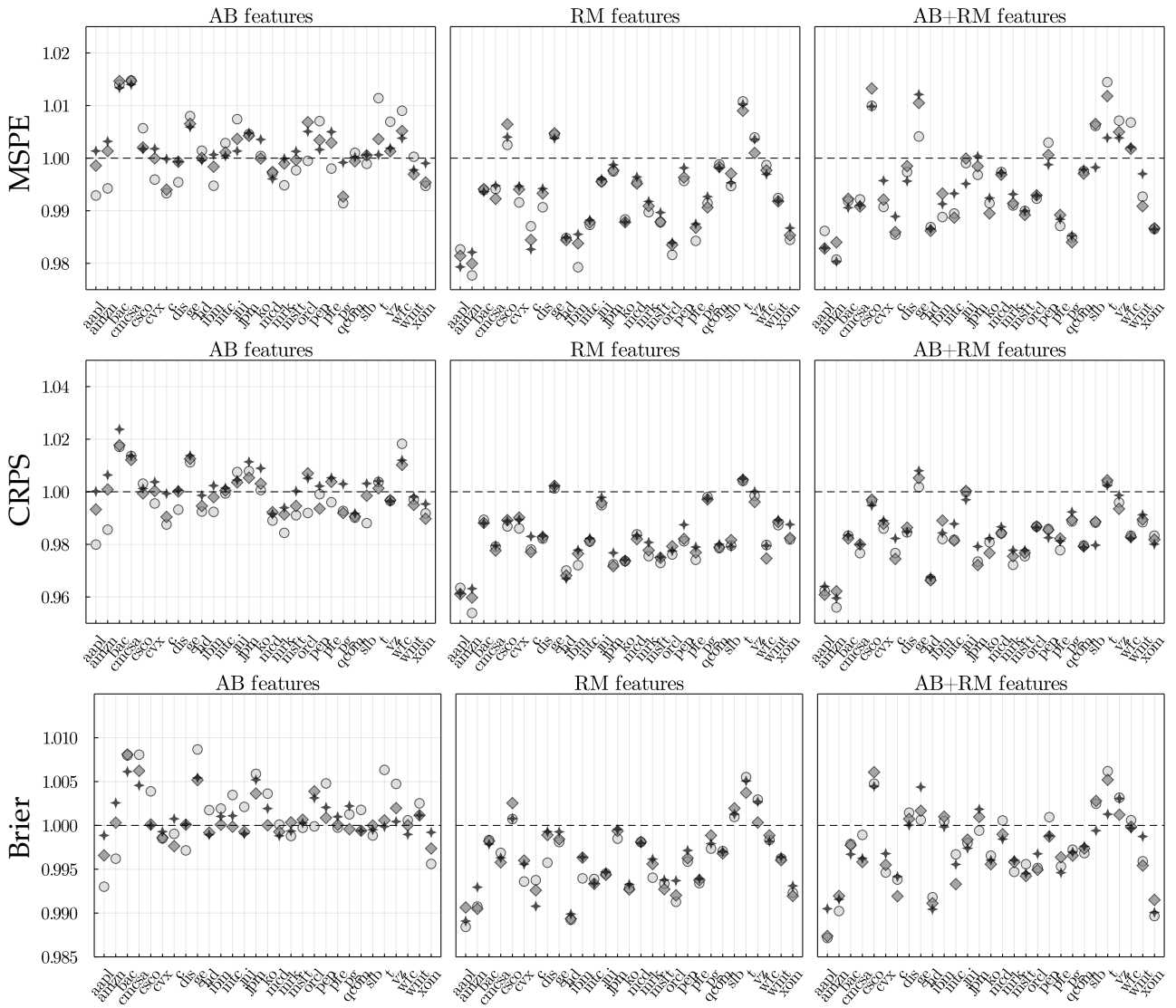

Figure 3. Comparison of the out-of-samples forecasts of 29 U.S. stocks

The three statistical measures, MSPE (top), CRPS (middle), and Brier score (bottom) are used for the three set of predictors and three machine learning models depicted as *star* for 128, *diamond* for 128x64, and *circle* for 128x64x32. Anatolyev and Baruník (2019) ordered logit model is benchmark with value 1. Value lower than 1 show better better performance of a model in comparison to benchmark.

and since machine learning is more flexible in number of parameters to be estimated, it learns and approximates the AB parametrization with a small degree of error.

Situation changes with additional predictors when the AB approach becomes infeasible and machine learning approach offers possibility to explore how informative the predictors are for forecasts. When additional five realized measures (RM) are used as predictors, performance increases with respect to all measures. With respect to depth of networks, the shallow (NN 128) neural network shows best results. This result is similar to Gu et al. (2020) who find that shallow network performs better than deeper structures one on asset returns data.

While the Table 2 provides information on relative counts, Figures A1 and 3 complement it with all measures reported for individual assets in box-plots. The detailed look uncovers that machine learning improves performance of individual stocks such as AAPL, AMZZN, GE, or WMT even more. At the same time, Figure A1 shows that in most cases, deeper networks shows lower variance for most of stocks.

4 Empirical Application: Macroeconomic Fan Charts

Macroeconomic fan charts are a popular tool for communicating uncertainty about economic forecasts. Recognizing the need to communicate uncertainty to the public, the Bank of England began publishing fan charts in 1996 and quickly became a leader in communicating uncertainty. However, the art and science of such an important tool for policy making remains on the shoulders of the methods chosen.

The objective is to construct a data-driven macroeconomic fan chart based on a best approximation model learned from a large number of variables using deep learning. This approach contrasts with the existing literature, which provides uncertainty estimates for macroeconomic variables using so-called predictive fan charts, prediction intervals, (Britton et al., 1998; Stock and Watson, 2017) which are derived from a limited number of variables using a parametrized and structured model that requires a number of assumptions.

4.1 Data

To measure uncertainty, we use a high-dimensional FRED-QD dataset from McCracken and Ng (2020), available from the Federal Reserve Bank of St. Louis and widely used in the literature (Goulet Coulombe et al., 2022). We selected 216 quarterly US macroeconomic and financial indicators observed from 1961Q1 to 2019Q4. Due to the non-stationarity of several variables, we follow the transformation codes used by McCracken and Ng (2020). We construct a data-rich fan chart for real GDP growth (GDPC1), inflation (CPIAUCSL) and unemployment rate (UNRATE), reflecting information from 216 relevant variables. In comparison to the data-driven fan charts, we use a state-of-the-art macroeconomic model based on Bayesian vector autoregression that incorporates data-driven factors to incorporate the big data information (McCracken and Ng, 2020).

4.2 Deep-learning Based Fan Charts

We approach the exercise via direct forecasting scheme in order to obtain a h-step ahead probabilistic forecast which can be then depicted as a fan chart. For every h-step ahead forecast we learn distributional networks given the set of inputs z_t

$$\widehat{\mathfrak{g}}^{(1)}_{W,b}(z_t), \ldots, \widehat{\mathfrak{g}}^{(h)}_{W,b}(z_t). \tag{10}$$

We obtain the entire h step-ahead conditional distribution from $\widehat{\mathfrak{g}}^{(h)}_{W,b}(z_t)$ by interpolating the cumulative distribution function, while preserving the monotonicity of the result. To interpolate, we apply the Fritsch-Carlson monotonic cubic interpolation (Fritsch and Carlson, 1980), see Appendix A for details, and use the predicted cumulative distribution function $\widehat{F}_{t+h}(\alpha|\mathcal{I}_t)$ to form $k-size$ prediction intervals as

$$PI^{k}_{t+h} = \left[\widehat{F}^{-1}_{t+h}(\alpha_l|\mathcal{I}_t), \widehat{F}^{-1}_{t+h}(\alpha_u|\mathcal{I}_t)\right], \tag{11}$$

with the size of the interval $k = \alpha_u - \alpha_l$.

To show usefulness of our approach is, we compare the predictions obtained from the distributional network with the Bayesian vector autoregression (BVAR) with inputs including four factors formed from the data of McCracken and Ng (2020). BVAR is a state of the art approach in macroeconomics and it uses the information, via factors, from the whole dataset, so the forecasts are comparable. To obtain the fan chart (forecast intervals), we use the best performing recursive (iterative) scheme $\widehat{y}_{t+h} = f(y_{t+h-1}|\mathcal{I}_t)$, where the forecast intervals are based on the distribution of the residuals. Formally, if we assume a normal distribution, we obtain the h-step ahead α prediction interval as $[\widehat{y}_{t+h} - \phi(1-\alpha/2)\widehat{\sigma}_h,\ \widehat{y}_{t+h} + \phi(1-\alpha/2)\widehat{\sigma}_h]$, where $\phi(1-\alpha/2)$ is the corresponding quantile of the standard normal distribution and, for example, for a naive forecast we have $\hat{\sigma}_h = \hat{\sigma}\sqrt{h}$ and $\hat{\sigma}_h$ is the residual standard deviation. [10]

In this example, we use the quantile or tick loss function (Clements et al., 2008) to evaluate the h-step-ahead forecasts given by

$$L^{h}_{\alpha,m} = E\left[(\alpha - \mathbb{I}\{e_{t+h,m} < 0\})e_{t+h,m}\right], \tag{12}$$

where m is a model and α-quantile, where $e_{t+h,m} = y_{t+h} - \widehat{F}^{-1}_{t+h,m}(\alpha|\mathcal{I}_t)$ is the difference between the true values and α-quantile forecast given the information set, $\mathcal{I}_t$. To assess the predictive accuracy of the models statistically, we use the Diebold-Mariano test (Diebold and Mariano, 1995) with Newey-West variance for $h > 1$ cases and test the null hypothesis $H_0 : L_{\alpha,m_1} > L_{\alpha,m_2}$ against the alternative that m_2 is less accurate than m_1.

4.3 Setup

In order to compute the $h = 1,\ldots,6$ horizon forecasts for each quarter of the out-of-sample period, we utilize quarterly data. This period begins in 2012:Q3 and concludes in 2019:Q4. The conditional distribution is approximated using $j = 1,\ldots,19$ empirical $\alpha_j = (0.01,...,0.99)$ probability levels. The learning process explores 36 combinations of hyperparameters to identify the optimal approximating model for each h-step ahead fore-

[10] Alternatively, prediction intervals or fan charts can be obtained using bootstrap methods, Britton et al. (1998).

cast. The hyperparameters space is optimized once on the training data set, and the training procedure employs a growing-window forward-validation scheme on the training data set using three folds. The training data set is split while training each fold of validation on the training and test parts by a ratio of 0.93. The present study presents predictions for deep recurrent neural networks with two hidden LSTM layers, each comprising a different number of neurons, which were chosen during the hyper-optimization process.

Table A1 in the Appendix provides a summary of all parameters and details used in the estimation process. To facilitate comparison of the deep-learning-based fan charts, we perform the standard estimation procedure for the Bayesian Vector Autoregression (BVAR) model with factor components, as described in McCracken and Ng (2020). The information criteria, namely AIC and BIC, were employed to select the model with four lags. Furthermore, the prediction intervals for GDP growth, inflation, and the unemployment rate were determined. The data for both procedures were transformed in accordance with the McCracken and Ng (2020) codes and standardized to a normal distribution with a mean of zero and a standard deviation of one.

4.4 Results

The discussion begins by presenting the qualitative results of the GDP growth, inflation and unemployment forecasts in the form of fan charts. Figure 4 shows prediction intervals of 50%, 68%, 80% and 90% as fan charts over four different periods, produced by both recurrent distribution neural network and Bayesian vector autoregression approaches. It highlights the benefits of deep learning based forecasting.

Prediction intervals derived from distributional neural networks exhibit asymmetry, in contrast to the smoothness observed in traditional time series forecasting, here represented by BVAR. As uncertainty in the future increases, deep learning continues to extract some structure from the data, resulting in probability intervals that are less similar to the shape of "fans". The intervals are narrower than those produced by BVAR. In the case of the unemployment rate, the BVAR model is less effective in reducing uncertainty about future observations. This is probably due to the presence of pronounced peaks and seasonal factors. In contrast, our DistrNN model with LSTM units is able to capture uncertainty effectively.

Figure 4 is intended to provide an illustrative overview; however, it only shows a limited number of periods. To provide a more comprehensive assessment of the benefits of the deep learning approach, we have quantified the prediction differences for the entire out-of-sample period. Table 3 presents a quantitative comparison of the predictions from both models. A comparison of the forecasts at horizons $h = 1, \ldots, 6$ is presented using the tick loss metric (Eq. 12) for selected quantile levels, $\alpha = \{0.05, 0.1, 0.16, 0.25, 0.5, 0.75, 0.84, 0.9, 0.95\}$.

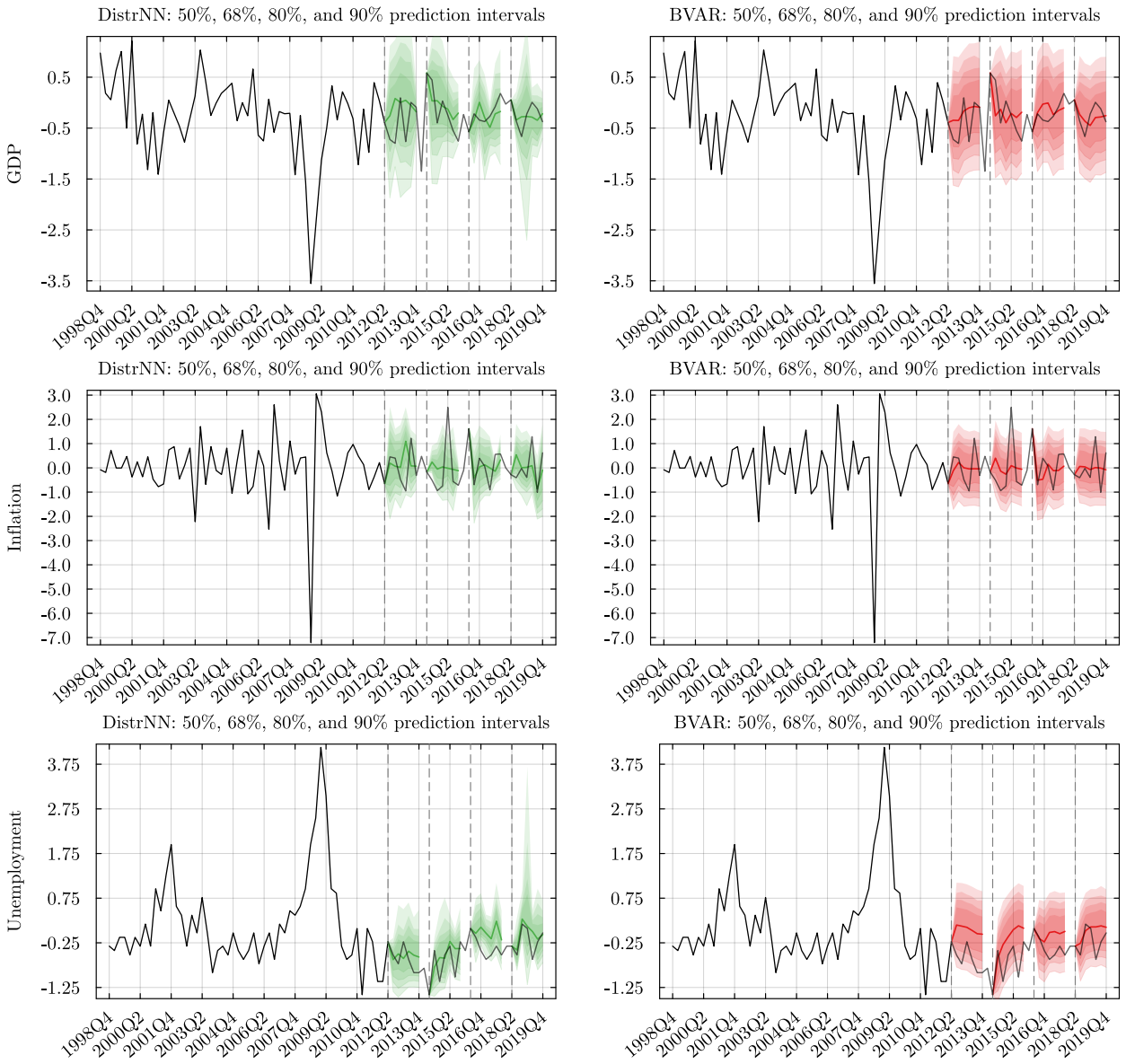

Figure 4. Deep-learning based (green) and Bayesian VAR (red) fan charts
6-step-ahead quarterly 50%, 68%, 80% and 90% prediction intervals of GDP growth (top), Inflation (middle), and Unemployment rate (bottom) as obtained by the distributional network (green) and a factor Bayesian VAR (red). Forecasts are made at the end of the 2012:Q2, 2014:Q2, 2016:Q2 and 2018:Q2 depicted by dashed vertical lines. Train and test data are plotted by black solid line.

Table 3 shows the relative performance of quantile losses between the two methods, namely Distributional Recurrent Neural Network (DistrNN) and Bayesian Vector Autoregression (BVAR) forecasts. The results cover forecasts of GDP growth, inflation and the unemployment rate. The deep learning approach provides forecasts with lower error (in bold) at most of the probability levels and horizons considered. While deep learning provides more accurate forecasts in the majority of cases, a significant number of these cases show lower losses for the DistrNN approach than for the traditional BVAR approach, which are also statistically significant. Notable and significant gains are observed in the right

Table 3. Relative out-of-sample performance of DistrNN and BVAR

α	0.05	0.1	0.16	0.25	0.5	0.75	0.84	0.9	0.95
GDP									
h=1	1.031	1.002	**0.952**	**0.949**	**0.948**	**0.784** *	**0.763** **	**0.747** **	**0.755** ***
h=2	1.271	**0.990**	1.026	1.006	1.044	**0.779** **	**0.793** ***	**0.816** ***	**0.873** **
h=3	1.603	1.186	1.142	1.042	**0.911**	**0.909**	**0.900**	**0.906**	**0.890** *
h=4	**0.931**	**0.947**	**0.905**	1.014	**0.958**	**0.765** **	**0.700** ***	**0.688** ***	**0.738** ***
h=5	**0.688** ***	**0.844** *	**0.884**	**0.875** *	**0.900**	**0.603** ***	**0.597** ***	**0.590** ***	**0.587** ***
h=6	**0.897**	1.008	1.029	1.086	**0.928**	**0.623** ***	**0.644** ***	**0.699** ***	**0.740** ***
Inflation									
h=1	1.863 ‡	1.624 ‡	1.451 ‡	1.199	**0.951**	1.122	1.210	1.253	1.350
h=2	**0.954**	1.151	1.300 ‡	1.278 ‡	1.018	**0.922**	**0.918**	**0.881***	**0.935**
h=3	**0.826*****	**0.786*****	**0.918**	1.044	1.037	1.114 ‡	1.167	1.200	1.268
h=4	**0.920**	1.132	1.324	1.311 ‡	1.131 ‡	1.037	1.067	1.090	1.084
h=5	1.016	1.153	1.189	1.153	**0.991**	**0.900**	**0.892**	**0.941**	1.005
h=6	1.231	1.311	1.379 ‡	1.294 ‡	1.046	**0.941**	**0.919**	**0.954**	**0.967**
Unemployment									
h=1	**0.746**	**0.987**	**0.986**	**0.823**	**0.704** **	**0.688** ***	**0.650** ***	**0.691** ***	**0.784** ***
h=2	1.410	1.439	1.149	**0.926**	**0.744** ***	**0.691** ***	**0.680** ***	**0.685** ***	**0.725** ***
h=3	1.005	1.114	1.227	1.062	**0.895**	**0.854** *	**0.855** *	**0.966**	1.207
h=4	2.195 ‡	1.531	1.177	**0.925**	**0.733** **	**0.783** *	**0.792** *	**0.881**	1.024
h=5	1.070	**0.869**	**0.830**	**0.894**	**0.787** *	**0.660** ***	**0.625** ***	**0.665** ***	**0.755** **
h=6	1.195	1.103	**0.942**	**0.867**	**0.650** ***	**0.640** ***	**0.652** ***	**0.676** ***	**0.819** *

Note: The values depict relative performance between the two methods, lower than one means DistrNN is better than BVAR. Based on the Diebold-Mariano test statistics, with the null hypothesis $H_0 : L_{\alpha,DistrNN} > L_{\alpha,BVAR}$ against the alternative, we show when Bayesian VAR is less accurate than the Distributional Recurrent Neural Network. Stars indicate statistical significance of of the test that ***, **, * correspond to 1%, 5%, 10% levels, accordingly. Alternatively, ‡ indicates when BVAR loss is statistically lower than the DistrNN loss. We report the out-of-sample forecasts for 25 quarters for the variables GDP growth (GDP), Inflation, and Unemployment rate, at quantiles $\alpha = \{0.05, 0.1, 0.16, 0.25, 0.5, 0.75, 0.84, 0.9, 0.95\}$, horizons $h = 1, \ldots, 6$, begining at Q3/2013 and concluding at Q4/2019.

tail of the distributions, where the DistrNN approach significantly outperforms the BVAR method. It is worth noting that the DistrNN also reduces losses for inflation forecasts, although this is not statistically significant. There is an exception for inflation forecasts, where the distributional network also reduces losses, unfortunately without statistical significance. We note that the results depend on the use of 25 out-of-sample observations, a size that may be limiting for the test.

5 Conclusion

In this paper, we present a novel approach to modelling probability distributions of economic variables using state-of-the-art machine learning methods. The distributional neural network we propose is flexible and allows the exploration of large datasets containing variables with non-Gaussian, non-linear and asymmetric structures. The approach is

shown to be useful in an economic and financial context. At the same time, the approach is generalisable and can be applied to any other dataset.

Specifically, we show that our distributional neural network improves out-of-sample distributional accuracy for US stock returns. The distributional NN model learns and approximates distributions in a data-poor environment such as macroeconomic variables. We go one step further with the recurrent distributional neural network and show how deep learning can be used to improve probabilistic forecasting of data that is notoriously difficult to predict.

References

Anatolyev, S. and J. Baruník (2019). Forecasting dynamic return distributions based on ordered binary choice. *International Journal of Forecasting 35*(3), 823–835.

Athey, S. and G. W. Imbens (2019). Machine learning methods that economists should know about. *Annual Review of Economics 11*, 685–725.

Babiak, M. and J. Baruník (2020). Deep learning, predictability, and optimal portfolio returns. *arXiv preprint arXiv:2009.03394*.

Berrisch, J. and F. Ziel (2022). Distributional modeling and forecasting of natural gas prices. *Journal of Forecasting 41*(6), 1065–1086.

Bianchi, D., M. Büchner, and A. Tamoni (2021). Bond risk premiums with machine learning. *The Review of Financial Studies 34*(2), 1046–1089.

Box, G. E., N. R. Draper, et al. (1987). *Empirical model-building and response surfaces*, Volume 424. Wiley New York.

Box, G. E., G. M. Jenkins, G. C. Reinsel, and G. M. Ljung (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Britton, E., P. Fisher, and J. Whitley (1998). The inflation report projections: understanding the fan chart. *Chart 8*(10).

Campbell, J. Y. and S. B. Thompson (2008). Predicting excess stock returns out of sample: Can anything beat the historical average? *The Review of Financial Studies 21*(4), 1509–1531.

Chen, Y., Y. Kang, Y. Chen, and Z. Wang (2020). Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing*.

Chernozhukov, V., I. Fernández-Val, and B. Melly (2013). Inference on counterfactual distributions. *Econometrica 81*(6), 2205–2268.

Clements, M. P., A. B. Galvão, and J. H. Kim (2008). Quantile forecasts of daily exchange rate returns from forecasts of realized volatility. *Journal of Empirical Finance 15*(4), 729–750.

Diebold, F. X. and R. S. Mariano (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics 13*(3).

Duan, T., A. Avati, D. Y. Ding, S. Basu, A. Y. Ng, and A. Schuler (2019). Ngboost: Natural gradient boosting for probabilistic prediction. *arXiv preprint arXiv:1910.03225*.

Engle, R. F. and S. Manganelli (2004). Caviar: Conditional autoregressive value at risk by regression quantiles. *Journal of Business & Economic Statistics 22*(4), 367–381.

Fama, E. F. (1965). Portfolio analysis in a stable paretian market. *Management science 11*(3), 404–419.

Feng, G., J. He, and N. G. Polson (2018). Deep learning for predicting asset returns. *arXiv preprint arXiv:1804.09314*.

Foresi, S. and F. Peracchi (1995). The conditional distribution of excess returns: An empirical analysis. *Journal of the American Statistical Association 90*(430), 451–466.

Fortin, N., T. Lemieux, and S. Firpo (2011). Decomposition methods in economics. In *Handbook of labor economics*, Volume 4, pp. 1–102. Elsevier.

Fritsch, F. and R. Carlson (1980). Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis 17*(2).

Gasthaus, J., K. Benidis, Y. Wang, S. S. Rangapuram, D. Salinas, V. Flunkert, and T. Januschowski (2019). Probabilistic forecasting with spline quantile function rnns. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1901–1910.

Geweke, J. and C. Whiteman (2006). Bayesian forecasting. *Handbook of economic forecasting 1*, 3–80.

Gneiting, T. (2008). Probabilistic forecasting. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, 319–321.

Gneiting, T. and A. E. Raftery (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association 102*(477), 359–378.

Good, I. J. (1952). Rational decisions. *Journal of the Royal Statistical Society: Series B (Methodological) 14*(1), 107–114.

Goulet Coulombe, P., M. Leroux, D. Stevanovic, and S. Surprenant (2022). How is machine learning useful for macroeconomic forecasting? *Journal of Applied Econometrics 37*(5), 920–964.

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

Heaton, J. B., N. G. Polson, and J. H. Witte (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry 33*(1), 3–12.

Hersbach, H. (2000). Decomposition of the continuous ranked probability score for ensemble prediction systems. *Weather and Forecasting 15*(5), 559–570.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

Hu, T., Q. Guo, Z. Li, X. Shen, and H. Sun (2019). Distribution-free probability density forecast through deep neural networks. *IEEE Transactions on Neural Networks and Learning*

Systems 31(2), 612–625.

Hyndman, R., A. B. Koehler, J. K. Ord, and R. D. Snyder (2008). *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media.

Innes, M., E. Saba, K. Fischer, D. Gandhi, M. C. Rudilosso, N. M. Joy, T. Karmali, A. Pal, and V. Shah (2018). Fashionable modelling with flux. *CoRR abs/1811.01457*.

Israel, R., B. T. Kelly, and T. J. Moskowitz (2020). Can machines' learn'finance? *Available at SSRN 3624052*.

Iworiso, J. and S. Vrontos (2020). On the directional predictability of equity premium using machine learning techniques. *Journal of Forecasting 39*(3), 449–469.

Januschowski, T., J. Gasthaus, Y. Wang, D. Salinas, V. Flunkert, M. Bohlke-Schneider, and L. Callot (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting 36*(1), 167–177.

Judd, K. (1998). *Numerical Methods in Economics*. Cambridge, MA: MIT Press.

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koenker, R. and G. Bassett Jr (1978). Regression quantiles. *Econometrica*, 33–50.

Kuan, C.-M. and H. White (1994). Artificial neural networks: An econometric perspective. *Econometric Reviews 13*(1), 1–91.

Lahiri, K., G. Martin, et al. (2010). Bayesian forecasting in economics. *International Journal of Forecasting 26*(2), 211–215.

Leorato, S. and F. Peracchi (2015). Comparing distribution and quantile regression.

Lim, Y.-S. and D. Gorse (2020). Deep probabilistic modelling of price movements for high-frequency trading. *arXiv preprint arXiv:2004.01498*.

Loshchilov, I. and F. Hutter (2019). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Matheson, J. E. and R. L. Winkler (1976). Scoring rules for continuous probability distributions. *Management science 22*(10), 1087–1096.

McCracken, M. and S. Ng (2020). Fred-qd: A quarterly database for macroeconomic research. Technical report, National Bureau of Economic Research.

Mullainathan, S. and J. Spiess (2017). Machine learning: an applied econometric approach. *Journal of Economic Perspectives 31*(2), 87–106.

Rapach, D., J. Strauss, and G. Zhou (2010). Out-of-sample equity premium prediction: Combination forecasts and links to the real economy. *The Review of Financial Studies 23*(2), 821–862.

Rothe, C. (2012). Partial distributional policy effects. *Econometrica 80*(5), 2269–2301.

Salinas, D., V. Flunkert, J. Gasthaus, and T. Januschowski (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting 36*(3), 1181–1191.

Sirignano, J., A. Sadhwani, and K. Giesecke (2016). Deep learning for mortgage risk. *arXiv preprint arXiv:1607.02470*.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research 15*(1), 1929–1958.

Stock, J. H. and M. W. Watson (2017). Twenty years of time series econometrics in ten pictures. *Journal of Economic Perspectives 31*(2), 59–86.

Timmermann, A. (2000). Density forecasting in economics and finance. *Journal of Forecasting 19*(4), 231.

Tobek, O. and M. Hronec (2020). Does it pay to follow anomalies research? machine learning approach with international evidence. *Journal of Financial Markets*, 100588.

Wen, R., K. Torkkola, B. Narayanaswamy, and D. Madeka (2017). A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*.

Žikeš, F. and J. Baruník (2016). Semi-parametric conditional quantile models for financial returns and realized volatility. *Journal of Financial Econometrics 14*(1), 185–226.

Appendix for

"Learning Probability Distributions of Economic Variables"

A CDF interpolation

The Fritsch–Carlson monotonic cubic interpolation (Fritsch and Carlson, 1980) provides a monotonically increasing CDF with range $[0, 1]$ when applied to CDF estimates on a finite grid.

Suppose we have CDF $F(y)$ defined at points $(y_k, F(y_k))$ for $k = 1, \ldots, K$, where $F(y_0) = 0$ and $F(y_K) = 1$. We presume that $y_k < y_{k+1}$ and $F(y_k) < F(y_{k+1})$ for all $k = 0, \ldots, K-1$, which is warranted by continuity of returns and construction of the estimated distribution. First, we compute slopes of the secant lines as $\Delta_k = (F(y_{k+1}) - F(y_k)))/(y_{k+1} - y_k)$ for $k = 1, \ldots, K-1$, and then the tangents at every data point as $m_1 = \Delta_1$, $m_k = \frac{1}{2}(\Delta_{k-1} + \Delta_k)$ for $k = 2, \ldots, K-1$, and $m_K = \Delta_{K-1}$. Let $\alpha_k = m_k/\Delta_k$ and $\beta_k = m_{k+1}/\Delta_k$ for $k = 1, \ldots, K-1$. If $\alpha_k^2 + \beta_k^2 > 9$ for some $k = 1, \ldots, K-1$, then we set $m_k = \tau_k \alpha_k \Delta_k$ and $m_{k+1} = \tau_k \beta_k \Delta_k$, with $\tau_k = 3(\alpha_k^2 + \beta_k^2)^{-1/2}$. Finally, the cubic Hermite spline is applied: for any $y \in [y_k, y_{k+1}]$ for some $k = 0, \ldots, K-1$, we evaluate $F(y)$ as

$$F(y) = (2t^3 - 3t^2 + 1)F(y_k) + (t^3 - 2t^2 + t)hy_k + (-2t^3 + 3t^2)F(y_{k+1}) + (t^3 - t^2)hm_{k+1},$$

where $h = y_{k+1} - y_k$ and $t = (y - y_k)/h$.

B Additional tables and figures

Hyper parameters	**Values**
Learning rate, η	0.0001, 0.001, 0.005
Dropout rate, ϕ	0.2, 0.4
L_2-decay regularization rate, λ_W	0.00001, 0.00005
Nodes dimensions	32x32, 64x64, 60x50
Fixed parameters	Value
Number of layers	2
Mini batch size	8
Epochs	350
Monotonicity parameter, λ_m	5.0
Cross-validation, k-folds	3
Train/test ratio	0.93
Ensembles	1

Table A1. Fan chart recurrent DistrNN parameters space for the empirical application, Sec. 4
The hyperoptimization algorithm searches through the whole hyperparameter space and tries all sets/combinations of hyperparameters to evaluate the model.

Hyper parameters	**Minimum value**	**Maximum value**
Learning rate, η	0.0001	0.02
Dropout rate, ϕ	0.0	0.5
L_2-decay regularization rate, λ_W	0.0	0.0018
Fixed parameters	Value	
Epochs	250	
Early stopping patience	25	
Monotonicity parameter, λ_m	0.2	
Mini batch size	32	
Ensembles	3	
Number of layers	1, 2, 3	
Nodes dimensions	128, 128x64, 128x64x32	

Table A2. DistrNN parameters space for the emprical application, Sec. 3
The hyperoptimization algorithm searches through the hyperparameter space and randomly tries sets of parameters to evaluate the model.

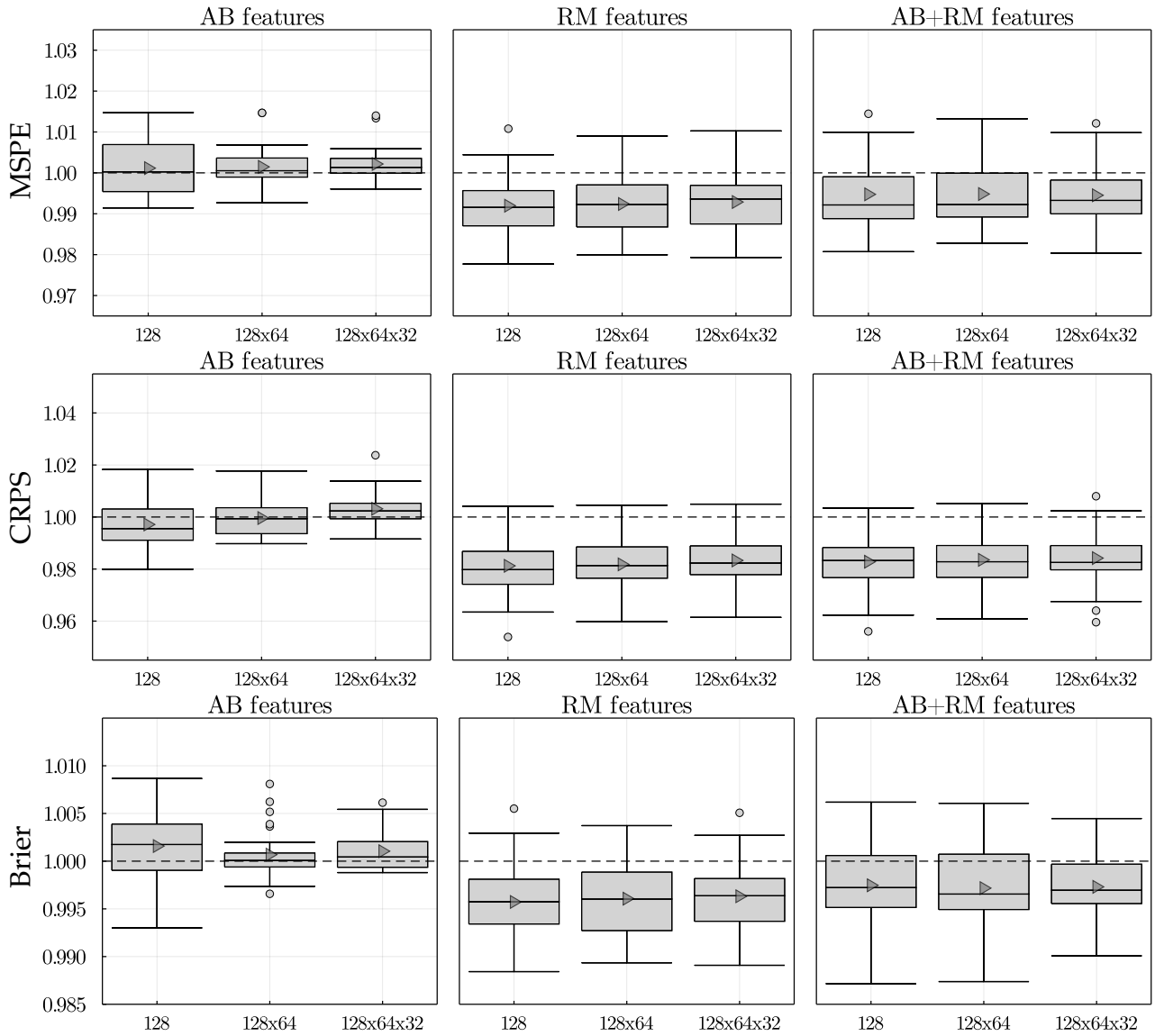

Figure A1. Comparison of the out-of-sample forecasts.

The three statistical measures: MSPE (top), CRPS (middle), and Brier (bottom). Each box-plot depicts benchmark values of 29 assets of given NN model size. Anatolyev and Baruník (2019) ordered logit model is benchmark=1. Value lower than 1 states that the purposed model is better that benchmark.