A DYNAMIC REGIME-SWITCHING MODEL USING GATED RECURRENT STRAIGHT-THROUGH UNITS

NINO ANTULOV-FANTULIN[a], ALVARO CAUDERAN[a], AND PETTER N. KOLM[c]

Abstract. We introduce a novel approach for regime identification using deep learning, a recurrent neural network architecture termed the *gated recurrent straight-through unit* (GRSTU). The new model can be implemented using commonly available open-source machine learning libraries, enabling automatic differentiation, and trained with the Adam optimizer. Through comprehensive simulation studies, we illustrate that the GRSTU model surpasses statistical jump models, which have shown state-of-the-art performance in regime identification tasks. Specifically, the GRSTU excels in regime classification, particularly on smaller datasets, while demonstrating comparable performance on larger datasets. Finally, in an out-of-sample application, we employ the GRSTU to identify regime changes in the S&P500 index from January 1, 2003 through January 1, 2024. We find that simple regime-switching strategies outperform the index, in terms of lower volatility, CVaR, and drawdown, while maintaining a Sharpe ratio equivalent to or better than that of the baseline.

Keywords: Deep learning; Gated recurrent units; Regime switching; Recurrent neural networks; Statistical jump models; Straight-through estimators; Time series forecasting; Unsupervised learning

(a) Aisot Technologies AG, and ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland. Email: *nino@aisot.ch*

(b) ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland. This work was done while visiting the Courant Institute of Mathematical Sciences, New York. Email: *alvaro.cauderan@alumni.ethz.ch*

(c) Courant Institute of Mathematical Sciences, New York University, 251 Mercer St., NY 10012, USA. Email: *petter.kolm@nyu.edu*

Date: April 29, 2024.

1. Introduction

Analyzing and forecasting time series is essential across a broad range of fields, including healthcare, seismology, economics, and finance. The complex and dynamic nature of time series data necessitates methods capable of capturing continuous trends and abrupt shifts in dynamics, often referred to as "regime changes." Identifying these changes of regimes is crucial as they often indicate fundamental alterations in the systems or processes generating the data, whether it is patient health metrics, seismic waves and noise, economic indicators, or risk-on and risk-off dynamics in financial markets (Hamilton et al., 1994; Stock et al., 1996; Rydén et al., 1998a; Bulla, 2011; Shumway et al., 2017).

Having been applied in various domains, from speech recognition (Gales et al., 2007) to finance (Mamon et al., 2014), many regime-switching models rely on *hidden Markov models* (HMMs). In HMMs, time series observations depend only on hidden (latent) state variables, characterized by a finite-state homogeneous first-order Markov chain, with all observations being independent conditional on their latent state. However, HMMs have limitations, such as lack of robustness to model misspecification and initialization, and require long time series for accurate estimation, frequently making them sample inefficient in practice (see, for example, Nystrup et al. (2021) for a discussion). Recently, to address these challenges, several alternative models have been proposed in the literature, including *spectral clustering HMMs* (Zheng et al., 2021), *statistical jump models* (JMs) (Nystrup et al., 2020b), and *continuous statistical jump models* (CJMs) (Aydinhan et al., 2023).

In this article, we introduce a new model for identifying regimes in univariate time series using deep learning (DL). Our model is inspired by JMs but takes a completely different approach. Rather than clustering temporal features, as typically done in JMs and CJMs, we employ a *recurrent neural network* (RNN) with a custom loss. Our model architecture is simple, built on a GRU-based design with a straight-through unit (Bengio et al., 2013; Goodfellow et al., 2016). The new model, termed the *gated recurrent straight-through unit* (GRSTU), can be implemented using commonly available open-source machine learning libraries. It processes time series observations as input and generates discrete regime assignments for each observation as output. Unlike JMs and CJMs, trained using coordinate descent, we employ automatic differentiation and the Adam optimizer (Kingma et al.,

2014) to train the GRSTU. Adam, a state-of-the-art optimizer incorporating adaptive learning rates and momentum, is known for its effectiveness in solving numerous non-convex optimization problems.

We conduct a comprehensive simulation study comparing the GRSTU and CJM models in two- and three-state settings. While the CJM generally outperforms HMMs, our results demonstrate that the GRSTU excels in state classification, especially on smaller datasets. On larger datasets, the GRSTU exhibits comparable performance to the CJM.

Finally, in an out-of-sample application, we employ the GRSTU to identify regime changes in the S&P500 index from January 1, 2003 through January 1, 2024. We find that simple regime-switching strategies outperform the index, in terms of lower volatility, CVaR, and drawdown, while maintaining a Sharpe ratio equivalent to or better than that of the baseline.

The outline of the article is as follows. Section 2 provides a review of relevant literature. Section 3 details our model's design, training process, and hyperparameter tuning. In Section 4, we compare and assess the performance of the model through an extensive simulation study. Section 5 demonstrates an application of the GRSTU by constructing simple regime-switching strategies for the S&P500 index. Finally, Section 6 concludes.

2. Related Work

Regime-switching and detection models play a crucial role in both academic research and industry applications. We provide an overview of the most commonly used approaches.

Markov switching models (MSMs), one of the earlier methods are widely used for capturing regime-switching dynamics in time series data, rely on a latent Markov chain to govern transitions between multiple states. Hamilton (1989) introduced MSMs to the economics literature, demonstrating their flexibility in capturing changes in model parameters depending on a system's state. Techniques like *expectation-maximization* (EM) (Dempster et al., 1977), *maximum likelihood estimation* (MLE) (Fisher, 1921), and Bayesian estimation typically play a role in training these models to find the best-fitting model parameters. MSMs have found widespread application in fields like finance for modeling stock returns (Schaller et al., 1997), interest rates (Gray, 1996; Ang et al., 2002), and volatility (Dueker, 1997), as well as in macroeconomics for understanding business cycle fluctuations (Hamilton et al., 2013). Extensions of MSMs have evolved to handle multivariate data, varying

transition probabilities, and more intricate state-dependent distributions (Francq et al., 2001). However, critics often highlight the complexity of estimating these models and the challenge of state identification, especially in the presence of noisy data. Despite these challenges, MSMs continue to be a powerful tool for identifying latent structures and regimes.

Building on the idea of Markov-based models, Baum et al. (1967) and Baum et al. (1970) introduced *hidden Markov models* (HMMs). They represent a distinct class of statistical models that are well-suited for analyzing sequential data and are among the most commonly used regime switching methods. HMMs extend the concept of Markov chains by incorporating observable outputs that depend probabilistically on a set of hidden states. Each hidden state generates an observable output according to a certain probability distribution, while transitions between hidden states follow a Markov process. Unlike MSMs, which are designed to handle regime changes in observable time series data, HMMs focus on uncovering hidden structures and patterns within data sequences, where the underlying states are not directly observable but inferred from observed data. HMMs continue to be popular in fields such as natural language processing, bioinformatics, finance, and speech recognition (Krogh et al., 1994; Mamon et al., 2014; Nadkarni et al., 2011), owing to their interpretability, ability to handle missing data, and the availability of algorithms for training and inference.

Advancements in HMMs have led to extensions that allow for continuous states and outputs, as well as the incorporation of more sophisticated structures such as hierarchical and coupled HMMs (Fine et al., 1998; Turin, 2004; Wang et al., 2017). Like MSMs, HMMs also pose their own set of challenges. These include difficulties stemming from the complexity of optimizing the log-likelihood function, sensitivity to model misspecification, and suboptimal initialization (Rydén et al., 1998b; Rydén, 2008; Nystrup et al., 2020b). To address these challenges, Bemporad et al. (2018) and Nystrup et al. (2020a) introduce *statistical jump models* (JM), and Aydinhan et al. (2023) propose *continuous statistical jump models* (CJM). These recent models combine clustering algorithms on temporal features and penalize state transitions to encourage persistence, aiming to provide more accurate representations of the regime-switching dynamics. They exhibit greater robustness, reduced sensitivity to model misspecification, and significantly faster training in comparison to HMMs (see, for example, Aydinhan et al. (2023) for a discussion). However, these models have their limitations as well. In particular, they

employ coordinate descent (Wright, 2015) to optimize a non-convex objective, a process that guarantees only a local optima but does not ensure attainment of the global optima. Therefore, modern optimization techniques such as stochastic gradient decent (Bottou, 2010) that lever automatic differentiation, may be a promising avenue for solving these non-convex optimization problems. While optimization algorithms such as Adam (Kingma et al., 2014) do not guarantee convergence to the global optima of non-convex problems, they have proven to be effective in finding favorable local minima.

Over the last decade, there has been a significant increase in the application of modern *deep learning* (DL) models, such as *recurrent neural networks* (RNNs) like *long short-term memory* (LSTM) networks and *gated recurrent units* (GRUs) (Hochreiter et al., 1997; Goodfellow et al., 2016; Ilhan et al., 2023). One advantage of these DL models is their ability to learn long-term dependencies present in time series data, which is crucial for accurate regime detection. However, DL techniques for regime detection have seen limited adoption, largely due to challenges such as the computational demands of deep learning, the requirement for extensive, high-quality data, and the "black-box" nature of these models which can make them hard to interpret and explain.

3. Methodology

In this section, we present our methodology and discuss our modeling choices. First, we describe the architecture of our GRU-inspired model and explain the benefits of using the *straight through estimator* (Bengio et al., 2013) to obtain discrete labels. Then, we elaborate on the rationale behind our chosen loss function and the tuning of its hyperparameters. We draw parallels between our loss function and other models, such as *Gaussian mixture models* (GMMs) (Dasgupta, 1999) and the *maximum entropy clustering algorithm* (MECA)(Karayiannis, 1994). Lastly, we bring all components together and provide the pseudo-code outlining the complete algorithm of the model.

3.1. **Notation.**

In this article, we represent vectors using bold lowercase letters, such as $\boldsymbol{x}$ and $\boldsymbol{p}_t$, matrices and tensors using bold uppercase letters like $\boldsymbol{O}_t$, and scalars by non-bold letters. The element-wise product between two vectors or matrices of the same size is denoted by $\odot$, the l_1 and l_2 norms of vectors are denoted by $\|\cdot\|_1$ and $\|\cdot\|_2$, respectively.

We consider univariate time series $\boldsymbol{x} := [x_1, ..., x_T]' \in \mathbb{R}^T$, where T represents the number of historical observations available, and denote by $\boldsymbol{v_t} := [x_{t-L}, ..., x_t]' \in \mathbb{R}^L$ the L past observations available at time $t \leq T$.

We denote the sigmoid activation function by $\sigma(\cdot)$, the hyperbolic tangent activation function by $\tanh(\cdot)$, the rectified linear unit by $\text{RELU}(\cdot)$, batch normalization by $\text{BatchNorm}(\cdot)$, and the softmax function as $\text{softmax}(\cdot)$. The function $\text{argmax}(\cdot)$ returns the index of the maximum value in a vector. As this function is non-differentiable, in the DL model we introduce in the next section, we will use the following *straight-through estimator* (STE)

$$\text{STE}(\boldsymbol{y}) := \text{argmax}(\boldsymbol{y})\,, \text{ and} \tag{1}$$

$$\frac{\partial \text{STE}(\boldsymbol{y})}{\partial \boldsymbol{y}_i} := 1\,, \text{ for all } i\,. \tag{2}$$

Originally introduced by (Bengio et al., 2013), STEs are designed to be "transparent" in the backward pass, allowing gradients to pass through unchanged while behaving normally in the forward pass.

3.2. **A Gated Recurrent Straight-Through Unit for Regime Detection.** Our model, called the *gated recurrent straight-through unit* (GRSTU), trains on a univariate time series, $\boldsymbol{x}$. It receives as input a contiguous window of data $\boldsymbol{v_t} = [x_{t-L}, ..., x_t]' \in \mathbb{R}^L$ and outputs the predicted state $\boldsymbol{s_t} \in \{0,1\}^K$ at time t, where K represents the number of states. The GRSTU is defined as follows

$$\boldsymbol{z_t} = \sigma\big(\boldsymbol{W_u}\ \boldsymbol{v_t} + \boldsymbol{R_u}\ \boldsymbol{h_{t-1}} + \boldsymbol{b_u}\big) \tag{3}$$

$$\boldsymbol{r_t} = \sigma\big(\text{BatchNorm}(\boldsymbol{W_r}\ \boldsymbol{v_t} + \boldsymbol{R_r}\ \boldsymbol{h_{t-1}} + \boldsymbol{b_r})\big) \tag{4}$$

$$\dot{\boldsymbol{h}}_{\boldsymbol{t}} = \tanh\left(\boldsymbol{W_h}\ \boldsymbol{v_t} + \boldsymbol{R_h}(\boldsymbol{r_t} \odot \boldsymbol{h_{t-1}}) + \boldsymbol{b_h}\right) \tag{5}$$

$$\boldsymbol{h_t} = (1 - \boldsymbol{z_t}) \odot \boldsymbol{h_{t-1}} + \boldsymbol{z_t} \odot \dot{\boldsymbol{h}}_{\boldsymbol{t}} \tag{6}$$

$$\boldsymbol{p_t} = \text{softmax}\big(\boldsymbol{W_p}(\text{RELU}\big(\boldsymbol{W_y}\ \boldsymbol{h_t} + \boldsymbol{b_y}\big)) + \boldsymbol{b_p}\big) \tag{7}$$

$$\boldsymbol{s_t} = \text{STE}(\boldsymbol{p_t})\,, \tag{8}$$

where $\boldsymbol{h_t} \in \mathbb{R}^H$ is a hidden state, $\dot{\boldsymbol{h}}_{\boldsymbol{t}} \in \mathbb{R}^H$ is a candidate hidden state, $\boldsymbol{p_t} \in \mathbb{R}^K$ are the state probabilities, $\boldsymbol{z_t} \in \mathbb{R}^H$ is the update gate vector, $\boldsymbol{r_t} \in \mathbb{R}^H$ is the reset gate vector and $\boldsymbol{W_u}, \boldsymbol{W_r}, \boldsymbol{W_h}, \boldsymbol{W_y} \in \mathbb{R}^{H \times L}$, $\boldsymbol{W_p} \in \mathbb{R}^{K \times H}$, $\boldsymbol{R_u}, \boldsymbol{R_r}, \boldsymbol{R_h} \in \mathbb{R}^{H \times H}$, $\boldsymbol{b_u}, \boldsymbol{b_r}, \boldsymbol{b_h}, \boldsymbol{b_y} \in \mathbb{R}^H$, $\boldsymbol{b_p} \in \mathbb{R}^K$ are model weights. H denotes the size of the hidden state of the GRSTU. The model output is a one-hot encoded vector $\boldsymbol{s_t} \in \{0,1\}^K$ which represents what state is chosen at

time t. In our empirical work, the states will represent different economic regimes.

This architecture is based on the GRU cell (Chung et al., 2014; Goodfellow et al., 2016) with some minor changes such as the addition of a layer to compute the state probabilities, $\boldsymbol{p_t}$, and the STE to convert them to discrete states $\boldsymbol{s_t}$. The STE enables us to use the argmax$(\cdot)$ function while maintaining differentiability of the model. This allows us to represent states as discrete rather than continuous values. The decision to base our model on the GRU architecture is primarily driven by its computational efficiency. Despite its simpler architecture compared to models such as the LSTM, the GRU is known to deliver performance results that are comparably robust, making it a suitable choice for sequence-based modeling (Chung et al., 2014).

3.2.1. *Loss Function.* In order to measure how good a state assignment $\boldsymbol{s_t}$ is, we introduce K independent Gaussian distributions with probability densities

$$\mathcal{N}(x; \mu_i, \sigma_i^2) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma_i}\right)^2\right), \tag{9}$$

where μ_i and σ_i^2 are the mean and variance, and each $i = 1, 2, \ldots, K$ is associated with a specific state. As training progresses, the log-likelihood score $\mathcal{N}(x_t; \mu_{k_t}, \sigma_{k_t}^2)$ of each data point x_t is updated, where $k_t := \text{argmax}(\boldsymbol{s_t})$ and $\boldsymbol{s_t} \in \{0, 1\}^K$ is a one-hot encoded state vector. The parameters for each Gaussian density are updated every U epochs with the implied parameters computed from the data points via

$$\mu_i = \frac{1}{N_i} \sum_{t=t_{\min}}^{T} x_t \cdot \mathbb{I}_{\{k_t=i\}}, \tag{10}$$

$$\sigma_i^2 = \frac{1}{N_i - 1} \sum_{t=t_{\min}}^{T} (\mu_i - x_t)^2 \cdot \mathbb{I}_{\{k_t=i\}}, \tag{11}$$

where $\mathbb{I}_{\{\text{condition}\}}$ is the indicator function, which takes a value of one when the condition is met and zero otherwise, and $N_i := |\{t | k_t = i\}|$. Since the model's input comprises historical data, we cannot commence making predictions at $t = 0$. Consequently, we denote by $t_{\min}$ the time at which we start making state predictions.

We emphasize that the parameters for the K Gaussian distributions are not direct parameters of the GRSTU model. In particular, they are not

optimized through automatic differentiation. Rather, they are determined from the state assignments via formulas (10)–(11) above.

To train our model, we use the loss function

$$\mathcal{L}(t,e) := -\frac{1}{J}\sum_{j=0}^{J-1}\log\left(\mathcal{N}(x_{t-j}|\mu_{k_t},\sigma^2_{k_t})\right) \;-\; \beta(e)\cdot H_{\boldsymbol{p_t}} \;+\; \lambda(e)\cdot||\boldsymbol{p_t}-\boldsymbol{p_{t-1}}||_1\,, \tag{12}$$

where e denotes the current epoch and $H_{\boldsymbol{p_t}}$ is the entropy of the discrete probability distribution parameterized by $\boldsymbol{p_t}$, i.e.

$$H_{\boldsymbol{p_t}} := -\sum_{i=1}^{K} p_{t,i}\log p_{t,i}\,. \tag{13}$$

The first term of the loss function (12) is an approximate log-likelihood of a GMM for the previous J data points, penalizing state assignments that deviate too far. This results in variance reduction and gradient smoothing, beneficial for stabilizing the learning process, particularly with noisy data. Furthermore, it also serves a "jump regularizer" (i.e. reduce the frequency the model changes from one state to another), as J becomes larger, the model aims to predict states that better fit a longer sequence of data points and will therefore lead to more persistent state sequences.

Similar, to the MECA, we also include an entropy maximization term at the initial stages of training, which corresponds to the second term in (12). Although not always necessary, there are situations where the model might converge to a sub-optimal solution by, for example, predicting just one state for the whole time series. The entropy term helps mitigate this. We employ an *entropy coefficient* that is dependent on the epoch

$$\beta(e) := \begin{cases} C_1, & e < E_1\,, \\ 0, & e \geq E_1\,, \end{cases} \tag{14}$$

where C_1 denotes the initial value of β and E_1 is the epoch at which the entropy loss coefficient is set to zero. Similarly to MECA, the high entropy at the initial stages encourages a broader exploration, and as per the step function design, the entropy term sharply decreases after a certain point in training, steering the model towards an entropy free solution.

The third term in (12) penalizes jumps (i.e. transitions from one state to another) between states and thereby controls the persistence of state

assignments. We define the *jump penalty* λ as

$$\lambda(e) := \begin{cases} \lambda_1 \cdot \exp\left(\frac{p \cdot e}{E_2} - p\right), & e < E_2\,, \\ \lambda_1, & e \geq E_2\,, \end{cases} \tag{15}$$

where e denotes the epoch, λ_1 is the maximum value of λ, E_2 is the epoch at which we stop increasing λ, and p controls the speed at which λ increases. The orange line in Figure 1 depicts the jump penalty for $p = 7$, $E_2 = 250$ and $\lambda_1 = 3$. In our implementation, we do not update the jump penalty at every epoch. Instead, we update it at regular intervals while also updating the parameters of the K Gaussian densities. Consequently, the behavior of the jump penalty λ resembles a step function, where the value remains constant for U epochs until it is updated, as shown by the blue line in Figure 1. The reason for this form of jump penalty is simple and goes hand in hand with the decision to have an entropy term. If an excessively high jump penalty is introduced in the early stages of training, the model is not able to explore properly. Introducing the jump penalty in a smooth and slow fashion enables the model to explore properly and results in significantly better performance.

3.2.2. *Hyperparameter tuning.* There are several hyperparameters that we tune in order to obtain the best performance possible. We discuss the important ones below.

Based on our experience, some parameters have a greater impact on the results, including C_1 and E_1 that determine β; p, λ_1, and E_2 that determine λ; and J. When it comes to the parameters for β and λ, they matter for several reasons. The entropy coefficient β is the only term that keeps the probabilities from becoming either 0 or 1, as soon as it is removed, probabilities become binary. This is specially important at the initial stages of training, considering the jump penalty is defined as $||\boldsymbol{p_t} - \boldsymbol{p_{t-1}}||_1$, binary probabilities would lead to very sparse gradients. We settle on a starting value for the entropy coefficient of 0.1, and set it to zero at epoch 500; in other words, $E_1 = 500$. As for the jump penalty, it needs to be low in the beginning to encourage exploration. We find that in our empirical work, using λ with $p = 7, \lambda_1 = 3$, and $E_2 = 250$ performs well.

3.2.3. *Training.* To train our model, we employ *truncated backpropagation through time* (TBTT) (Elman, 1990), a variation of the standard *backpropagation through time* (BPTT) algorithm, designed for computational efficiency. Unlike BPTT, which unfolds the entire data sequence to compute gradients,

TBTT breaks it into smaller, manageable chunks. Then it backpropagates the error only for a fixed number of steps R at a time, effectively "forgetting" the activations that are more than R steps behind the current time step. We find that truncation speeds up the training of the model without having an impact on the model's predictive performance. We use the Adam optimizer to minimize our loss function (Kingma et al., 2014), employing mini-batches of size 64, as this allows for more efficient parallelization across CPU or GPU cores.

Finally, given that this is an unsupervised learning task, label permutation during predictions can be a challenge, potentially slowing down the overall learning process. This is because any such permutations could alter the parameters of the Gaussian distributions associated with specific states. To address this issue, we employ a Kullback-Leibler (KL) divergence-based approach during the update phase of the parameters of the Gaussian densities. Specifically, we compute the new distribution parameters $\tilde{\mu}_i$ and $\tilde{\sigma}_i^2$, and then update the labels such as to minimize the KL divergence between the old and new distributions

$$j^* = \underset{j}{\operatorname{argmin}}\, D_{\mathrm{KL}}\big(\mathcal{N}(\mu_j, \sigma_j^2)||\mathcal{N}(\tilde{\mu}_i, \tilde{\sigma}_i^2)\big)\,. \tag{16}$$

Typically, $j^* \equiv i$ as label permutation is rare and only happens in early stages of training. Once a distribution has been assigned, it is removed from the pool of distributions as to not use the same distribution more than once. We provide the pseudo-code outlining the complete algorithm of the model in Algorithm 1.

4. A Simulation Study

We conduct a simulation study to compare the performance of the CJM and GRSTU, generating data from two- and three-state Gaussian HMMs where the true hidden state sequences and model parameters are known. This allows us to accurately evaluate the performance of the models in recovering the model parameters and original state sequence.

4.1. **Two-State HMM.**

Following Nystrup et al. (2020a), we simulate data from the two-state HMM

$$y_t|k_t \sim \mathcal{N}(\mu_{k_t}, \sigma_{k_t}^2)\,, \tag{17}$$

where the state sequence $\{k_t\}$ follows a first-order Markov chain, and the state-conditional parameters and transition matrix are given by

$$\begin{aligned} \mu_1 &= 0.0006, \quad \mu_2 = -0.0008, \\ \sigma_1 &= 0.0078, \quad \sigma_2 = 0.0174, \\ \boldsymbol{P} &= \begin{bmatrix} 0.9979 & 0.0021 \\ 0.0120 & 0.9880 \end{bmatrix}. \end{aligned} \tag{18}$$

These parameters are the daily equivalent of a model Hardy (2001), estimated from monthly returns on a stock market index, with the two states representing bullish and bearish market regimes.

4.2. **Three-State HMM.** To assess the performance as the number of states increases, we extend the two-state HMM to include a third and intermediate state, using the following state-conditional parameters and transition matrix

$$\begin{aligned} &\mu_1 = 0.0006, \quad \mu_2 = 0.0000, \quad \mu_3 = -0.0008, \\ &\sigma_1 = 0.0078, \quad \sigma_2 = 0.0112, \quad \sigma_3 = 0.0174, \\ &\boldsymbol{P} = \begin{bmatrix} 0.9981 & 0.0009 & 0.0010 \\ 0.0032 & 0.9931 & 0.0037 \\ 0.0058 & 0.0062 & 0.9880 \end{bmatrix}. \end{aligned} \tag{19}$$

4.3. **Results.** Due to the irregular frequency of states in the training time series, we encounter an imbalanced classification problem. Therefore, solely relying on accuracy as a performance metric would be misleading. To mitigate this, we use as our primary metric of interest the *balanced accuracy* (BAC), defined as

$$\text{BAC} := \frac{1}{2}\left(\frac{\text{TP}}{\text{TP}+\text{FN}} + \frac{\text{TN}}{\text{TN}+\text{FP}}\right), \tag{20}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

Additionally, we present the parameters estimated by our model, along with their respective standard errors, based on the outcomes from one thousand simulations.

Throughout the training phase, we evaluate our model whenever its probability distributions are updated, retaining only the model (and its associated prediction) with the lowest validation loss (i.e. same as training loss). It is important to note that to accurately compare the estimated parameters

across multiple time series, we employ a permutation-invariant evaluation method. We determine the permuted labels that yield the highest accuracy (not BAC) compared to the ground truth. Subsequently, we compute our metrics and implied parameters based on these labels.

Establishing baselines for comparison is crucial to evaluate the performance of our model. An obvious baseline is comparing the recovered parameters with the true ones – the parameters and state sequence characterizing the HMMs. In addition, we compare the GRSTU to the CJM proposed by Aydinhan et al. (2023), as it has demonstrated superior performance compared to all previously evaluated models for this task, including Gaussian HMMs.

4.3.1. *Two-State Results.* Table 2 presents the mean and standard errors (in parenthesis) of parameter estimates and BAC scores of 1000 simulations of lengths $T = 250, 500, 1000, 2000$ from the daily two-state HMM estimated with the CJM and GRSTU models. Upon examining the mean (μ_1, μ_2), standard deviations (σ_1, σ_2), and transition probabilities $(\gamma_{12}, \gamma_{21})$ parameters, it is apparent that all models obtain comparable estimates across all series lengths. However, our model obtains better estimates with lower standard deviations for shorter time series (i.e. 250 and 500), whereas the CJM and GRSTU have similar performance for longer time series (i.e. 1000 and 2000). Finally, when evaluating BAC, clearly the GRSTU also obtains better performance across shorter time series. This is because the CJM struggles to accurately identify the state sequence of time series characterized by a single state. Shorter time series tend to have a higher occurrence of single-state scenarios, which explains the significant difference in BAC in this setting. The standard errors of the estimates (in brackets) decrease as the time series get longer, consistent with the notion that more data leads to more precise parameter estimates.

4.3.2. *Three-State Results.* Table 3 presents the mean and standard errors (in parenthesis) of parameter estimates and BAC scores of 1000 simulations of lengths $T = 500, 1000, 2000$ from the daily three-state HMM estimated with the CJM and GRSTU models. In this setting, we refrain from conducting experiments with time series of length 250, as they are too short to obtain meaningful fits. As above, we evaluate the accuracy of each model in its ability to recover parameters and state sequences of the Gaussian HMMs, including means (μ_1, μ_2, μ_3), standard deviations $(\sigma_1, \sigma_2, \sigma_3)$, transition probabilities $(\gamma_{12}, \gamma_{13}, \gamma_{21}, \gamma_{23}, \gamma_{31}, \gamma_{32})$, and BAC. We observe that the results for the

three-state HMM are qualitatively similar to those of the two-state HMM. Specifically, the GRSTU demonstrates superior performance in terms of BAC, with this difference being more pronounced in shorter-length time series. Additionally, the GRSTU yields similar results to the CJM in recovering the underlying model parameters.

5. An Application: Identifying Regimes of the S&P 500 Index

In this section, we apply the GRSTU to identify the regimes of the S&P500 index. S&P500 is a market-cap weighted index comprising approximately five hundred of the largest companies listed on U.S. exchanges. We use a time series of daily log-returns of this index spanning from January 1, 1996 through January 1, 2024.

We examine the ability of the GRSTU to identify meaningful economic regimes (risk-on vs. risk-off) through the lens of simple trading strategies, and compare them to a buy and hold strategy. The strategies go long the market whenever the model predicts a bull regime, and fully or partially decrease the long position in the market when the model predicts a bear regime. As opposed to the previous experiments performed on synthetic time series, this is done completely out of sample in order to make it as realistic as possible. An issue we face when applying our model out of sample is differentiating between the bear and bull regimes, as our model only provides the state number without associating an economic interpretation with them. To address this, we first train our model on seven years of data, spanning from 1996 to 2003. We then evaluate the results and manually label each state as either a "bull" or "bear" regime. After the initial training and labeling, we predict the regimes for the subsequent three years out-of-sample manner. Following this, we retrain our model to predict the next three years, by integrating previously available data to our training set in a causal manner. This iterative process continues until we have obtained regime predictions for the entire period under study, the mapping from state to regime stays unchanged.

We do not expect the model to perfectly time the market. For example, exiting the market entirely during a "bear" regime may result in missed gains. Therefore, we examine the impact of hedging market exposure during the "bear" regime by 100%, 80%, 60%, 40%, and 20%, while retaining any remaining capital invested in the index. Adopting a conservative approach, we assume that any capital not allocated to the index earns zero return.

For the S&P500, we use the same hyperparameters as in our synthetic experiments with the exception of the length of the lookback window L, the number of past data points fed to the model (3), and the number of data points J used to compute the log-likelihood (12). Since these time series are longer than those in our simulation study, we decide to increase these hyperparameters to 50 and 30, respectively.

Figure 2 and Figure 3 depict the regimes predicted by the model and the cumulative log returns of the strategies compared to a buy and hold strategy for the S&P500 index. Our model accurately identifies many of the bear periods, such as those in 2008 and 2020. However, it has a few "misses," in particular, misclassifying some bullish periods as bearish. One possible explanation for this is that bear regimes are characterized by their high volatility, an attribute that is exploited by the model. However, some bullish periods may also demonstrate higher-than-normal volatility, posing challenges for accurate identification by the GRSTU model. To address this issue, future research could explore integrating additional exogenous data sources and features into the GRSTU model, similar to JMs (Nystrup et al., 2020a; Nystrup et al., 2021), to enhance the accuracy of regime identification.

Figure 4 presents performance metrics comparing the buy and hold baseline with the strategies. Our strategies demonstrate a significant reduction in volatility, *conditional value-at-risk* (CVaR) and *maximum drawdown* (MDD) while preserving a Sharpe ratio akin to that of the baseline.

6. Conclusion

In this article, we introduced a novel approach for regime identification using deep learning, presenting a recurrent neural network architecture termed the *gated recurrent straight-through unit* (GRSTU). The new model can be implemented using commonly available open-source machine learning libraries, enabling automatic differentiation, and trained with the Adam optimizer. Through comprehensive simulation studies, we illustrated that the GRSTU model surpasses statistical jump models, which have shown state-of-the-art performance in regime identification tasks. Specifically, we showed the GRSTU excels in regime classification, particularly on smaller datasets, while demonstrating comparable performance on larger datasets. Finally, in an out-of-sample application, we employed the GRSTU to identify regime changes in the S&P500 index from January 1, 2003 through January 1, 2024. We found that simple regime-switching strategies outperform the index in terms

of lower volatility, CVaR, and drawdown, while maintaining a Sharpe ratio equivalent to or better than that of the baseline.

There are several directions for future work. Extending the model to multivariate time series would broaden its applicability, especially in financial contexts. Exploring alternative loss functions and incorporating exogenous data sources could lead to performance improvements. An intriguing approach is to formulate the model within a variational Bayesian framework.

	H	$grad\_clip$	B	R	L	J	η	U	C_1	E_1	E_2	$epochs$	p	λ_1
2 states	32	0.001	64	20	30	10	0.005	30	0.1	500	250	750	7	3
3 states	32	0.001	64	80	30	10	0.005	30	0.1	400	200	750	7	3

TABLE 1. Hyperparameters for the two- and three-state simulation studies. H is the GRSTU hidden state size, $grad\_clip$ is the gradient clipping threshold, and B is the batch size. R and L define the length of truncated backpropagation and the window of past observations fed to the model, respectively. J is the parameter for the likelihood loss, η is the learning rate and U is the frequency at which the distributions are updated. C_1, E_1, E_2, p and λ_1 are parameters for λ and β. $epochs$ represents the number of epochs for which a model is trained.

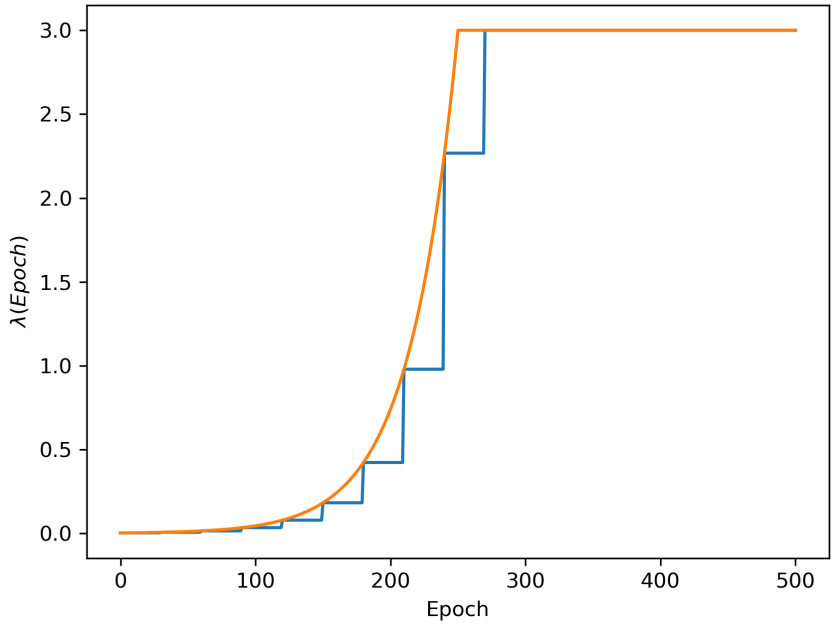

FIGURE 1. The jump penalty λ. The λ function (in orange) with $p = 7, \lambda_1 = 3$, and $E_2 = 250$, and its approximation (in blue).

Algorithm 1 GRSTU Pseudo Code

Input: Time series $\{x_t\}_{t=1}^{T}$
Initialize distributions: For all $i \in [1, 2, \ldots, K]$, initialize state conditional parameters, $\mu_i \leftarrow \mu, \sigma_i^2 \leftarrow \sigma^2$ where $\mu = \frac{1}{T}\sum_{t=1}^{T} x_t$ and $\sigma^2 = \frac{1}{T-1}\sum_{t=1}^{T}(\mu - x_t)^2$
for $e = 1$ **to** E **do**
  for $t = t_{\min}$ **to** T **do**
    $\boldsymbol{h_t} \leftarrow$ ForwardPass($\boldsymbol{v_t}, \boldsymbol{h_{t-1}}$), given by eqns. (3)–(6)
    $\boldsymbol{p_t} \leftarrow$ ComputeStateProbabilities($\boldsymbol{h_t}$), using eqn. (7)
    $\boldsymbol{s_t} \leftarrow$ STE($\boldsymbol{p_t}$)
    $k_t \leftarrow$ argmax($\boldsymbol{s_t}$)
    Perform backpropagation on $\mathcal{L}(t, e)$ given by formula (12)
  end for
  if e mod $U = 0$ **then**
    For all $i \in [1, 2, \ldots, K]$, update μ_i and σ_i^2 using formulas (10), (11) and (16)
    In evaluation mode, generate state prediction and compute evaluation loss
  end if
end for
Return state prediction with lowest evaluation loss

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	BAC
True	0.0006	-0.0008	0.0078	0.0174	0.0021	0.012	-
250							
CJM	**0.0006**	0.0001	**0.008**	0.0121	0.0042	**0.0169**	0.8552
	(0.0008)	(0.0024)	(0.0016)	(0.0046)	(0.0049)	(0.0273)	(0.1593)
GRSTU	**0.0006**	**-0.0008**	0.0081	**0.0160**	**0.0023**	0.0171	**0.9153**
	(0.0007)	(0.0024)	(0.0009)	(0.0025)	(0.0044)	(0.0178)	(0.1561)
500							
CJM	**0.0006**	-0.0002	**0.008**	0.0137	0.0027	**0.0153**	0.8854
	(0.0005)	(0.0023)	(0.0005)	(0.0044)	(0.0021)	(0.0206)	(0.1371)
GRSTU	**0.0006**	**-0.0007**	**0.008**	**0.0163**	**0.0023**	0.0171	**0.9065**
	(0.0004)	(0.0020)	(0.0006)	(0.0019)	(0.0031)	(0.0161)	(0.1536)
1,000							
CJM	**0.0006**	**-0.0006**	**0.0079**	0.0157	**0.0021**	**0.0141**	**0.9199**
	(0.0003)	(0.0019)	(0.0003)	(0.0033)	(0.0014)	(0.0120)	(0.1037)
GRSTU	**0.0006**	**-0.0006**	0.0080	**0.0162**	0.0024	0.0175	0.9102
	(0.0003)	(0.0016)	(0.0004)	(0.0017)	(0.0020)	(0.0155)	(0.1312)
2,000							
CJM	**0.0006**	**-0.0007**	**0.0079**	**0.0168**	**0.0020**	**0.0141**	**0.9366**
	(0.0002)	(0.0014)	(0.0002)	(0.0019)	(0.0011)	(0.0095)	(0.0685)
GRSTU	**0.0006**	-0.0006	0.0080	0.0164	0.0025	0.0166	0.9225
	(0.0002)	(0.0012)	(0.0003)	(0.0014)	(0.0015)	(0.0194)	(0.1065)

TABLE 2. Parameter estimates based on 1,000 simulated series of different lengths from the two-state HMM. The presented values are the averages and their standard errors (in parenthesis). Bold entries represent the best results across models. $^{(*)}$The values for the CJM are from Aydinhan et al. (2023).

	μ_1	μ_2	μ_3	σ_1	σ_2	σ_3	γ_{12}	γ_{13}	γ_{21}	γ_{23}	γ_{31}	γ_{32}	BAC
True	0.0006	0.0000	-0.0008	0.0078	0.0112	0.0174	0.0009	0.001	0.0032	0.0037	0.0058	0.0062	-
500													
CJM	**0.0006**	**0.0000**	**-0.0007**	0.0084	0.0117	0.0148	**0.0015**	**0.0015**	0.0084	0.0060	**0.0075**	0.0072	0.7427
	(0.0012)	(0.0023)	(0.0024)	(0.0021)	(0.0030)	(0.0038)	(0.0052)	(0.0058)	(0.0458)	(0.0440)	(0.0171)	(0.0182)	(0.1497)
GRSTU	**0.0006**	**0.0000**	-0.0005	**0.0081**	**0.0112**	**0.0159**	**0.0015**	0.0020	**0.0076**	**0.0039**	0.0095	**0.0065**	**0.8144**
	(0.0008)	(0.0016)	(0.0021)	(0.0011)	(0.0018)	(0.0029)	(0.0044)	(0.0133)	(0.0257)	(0.0088)	(0.0139)	(0.0159)	(0.1923)
1,000													
CJM	**0.0006**	**0.0000**	**-0.0006**	0.0084	0.0121	0.0155	**0.0012**	**0.0012**	**0.0052**	0.0064	**0.0072**	**0.0070**	0.7526
	(0.0008)	(0.0024)	(0.0023)	(0.0019)	(0.0030)	(0.0033)	(0.0034)	(0.0034)	(0.0139)	(0.0243)	(0.0115)	(0.0145)	(0.1455)
GRSTU	**0.0006**	**-0.0000**	**-0.0006**	**0.0080**	**0.0115**	**0.0162**	0.0013	**0.0012**	0.0071	**0.0033**	0.0102	0.0050	**0.7876**
	(0.0005)	(0.0014)	(0.0019)	(0.0008)	(0.0016)	(0.0024)	(0.0021)	(0.0025)	(0.0220)	(0.0065)	(0.0393)	(0.0089)	(0.1739)
2,000													
CJM	**0.0006**	-0.0001	**-0.0008**	0.0081	0.0124	0.0163	**0.0009**	0.0009	**0.0048**	0.0050	**0.0064**	**0.0060**	0.7800
	(0.0006)	(0.0022)	(0.0021)	(0.0011)	(0.0026)	(0.0025)	(0.0020)	(0.0016)	(0.0088)	(0.0333)	(0.0077)	(0.0181)	(0.1448)
GRSTU	**0.0006**	**-0.0000**	-0.0007	**0.0079**	**0.0116**	**0.0164**	0.0013	**0.001**	0.0065	**0.0029**	0.0082	0.0045	**0.7803**
	(0.0002)	(0.0009)	(0.0014)	(0.0002)	(0.0013)	(0.0018)	(0.0014)	(0.0012)	(0.0364)	(0.0042)	(0.0093)	(0.0072)	(0.1513)

TABLE 3. Parameter estimates based on 1,000 simulated series of different lengths from the three-state HMM. The presented values are the means with their respective standard deviation in parenthesis. Bold entries show the best parameter estimation across models. $^{(*)}$The values for the CJM are from Aydinhan et al. (2023).

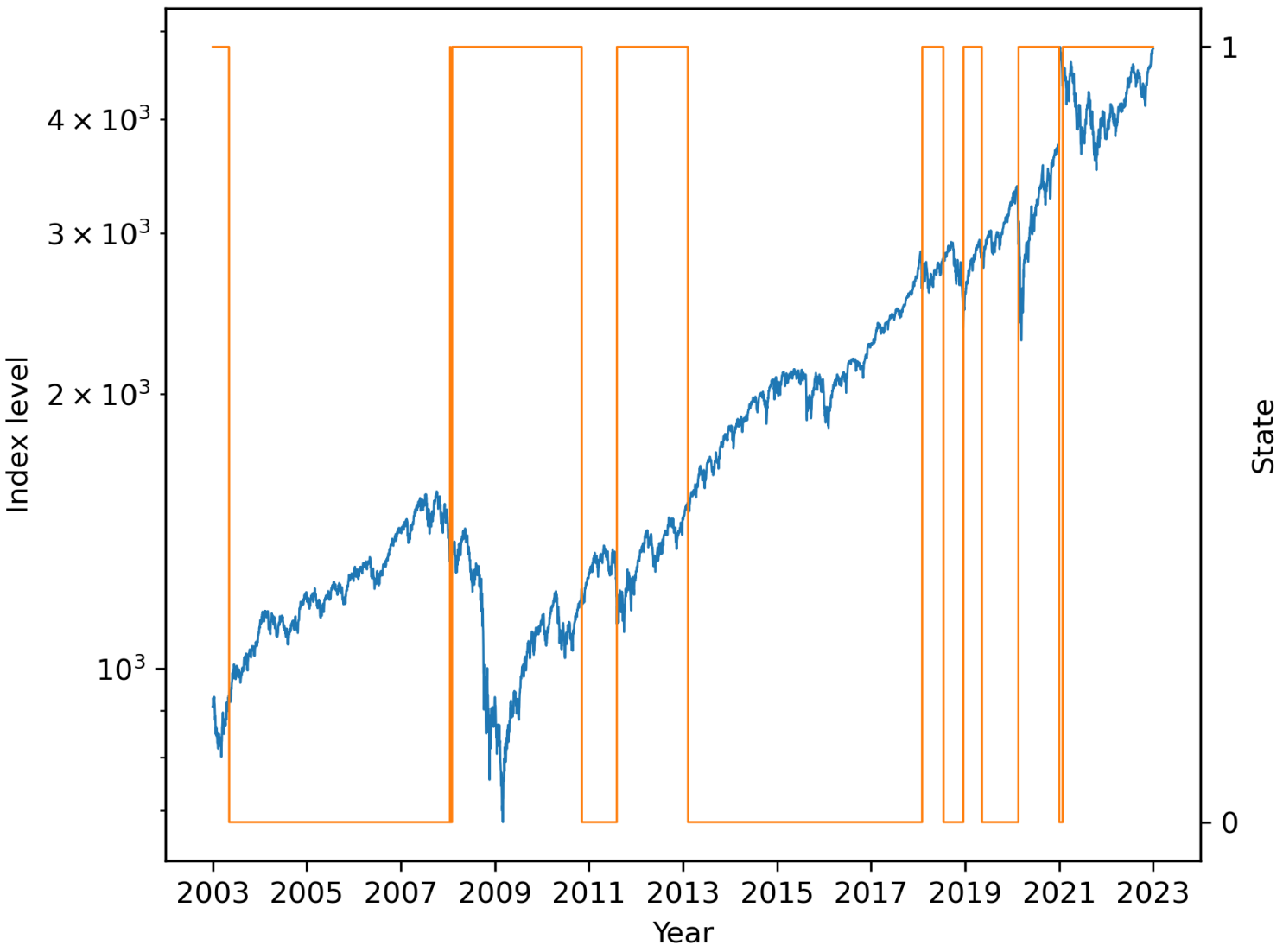

FIGURE 2. Regimes (in orange) of the S&P500 Index (in blue) from 2003 through 2023 as estimated by the GRSTU. State 0 and 1 correspond to bull and bear regimes, respectively.

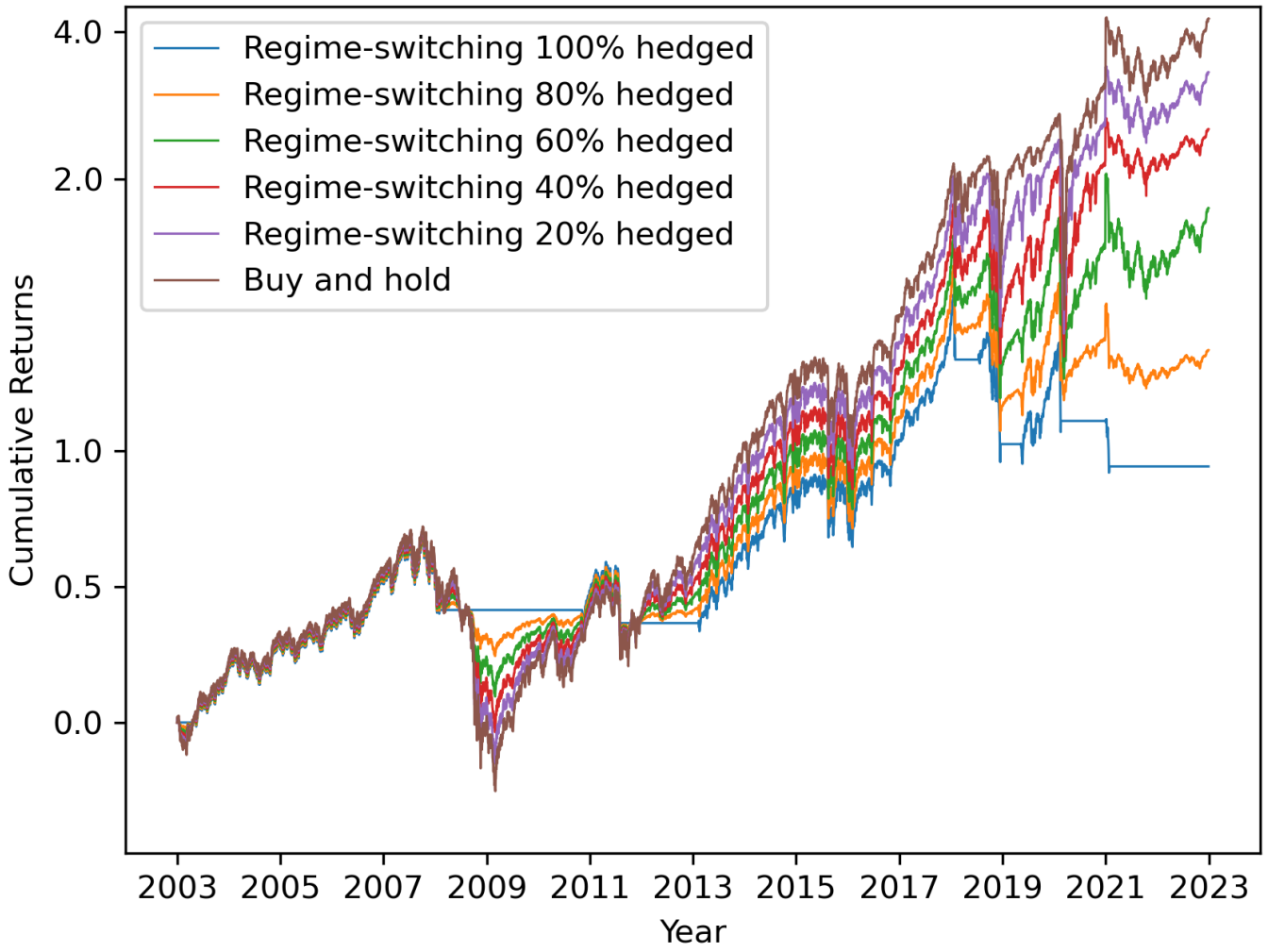

FIGURE 3. Cumulative log returns of the strategy with different levels of hedging compared to the buy and hold baseline of the S&P500 index from 2003 through 2023.

% hedged	CVaR	MDD	Log Return	Volatility	Sharpe Ratio
100%	**-1.66%**	**-25.4%**	3.32%	**9.8%**	0.340
80%	-1.68%	-26.2%	4.32%	10.3%	0.421
60%	-1.81%	-35.3%	5.31%	11.6%	0.458
40%	-2.06%	-43.4%	6.31%	13.5%	**0.466**
20%	-2.39%	-50.4%	7.31%	15.9%	0.461
Baseline	-2.78%	-56.6%	**8.32%**	18.4%	0.451

FIGURE 4. Summary statistics for the S&P500 index strategies. The most favorable results for each statistic are highlighted in bold. The first column indicates the cash exposure of the strategies during the bear regime, where 100% represents complete absence from the market. CVaR is the daily conditional value at risk at the 95% level using log returns. MDD is the maximum drawdown, calculated as the worst peak-to-trough decline expressed as a percentage of the peak. Returns, volatilities and Sharpe ratios are all annualized. The baseline is a buy and hold strategy of the index.

References

Ang, A. and G. Bekaert (Apr. 2002). "Regime Switches in Interest Rates". In: *Journal of Business & Economic Statistics* 20.2, pp. 163–182.

Aydinhan, A. O., P. N. Kolm, J. M. Mulvey, and Y. Shu (2023). "Continuous Statistical Jump Models for Identifying Financial Regimes". In: *Available at SSRN*.

Baum, L. E. and J. A. Eagon (1967). "An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology". In: *Bulletin of the American Mathematical Society* 73.3, pp. 360–363.

Baum, L. E., T. Petrie, G. Soules, and N. Weiss (1970). "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains". In: *The Annals of Mathematical Statistics* 41.1, pp. 164–171.

Bemporad, A., V. Breschi, D. Piga, and S. P. Boyd (Oct. 2018). "Fitting Jump Models". In: *Automatica* 96, pp. 11–21.

Bengio, Y., N. Léonard, and A. Courville (Aug. 2013). *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*.

Bottou, L. (2010). "Large-Scale Machine Learning with Stochastic Gradient Descent". In: *Proceedings of COMPSTAT'2010*. Ed. by Y. Lechevallier and G. Saporta. Heidelberg: Physica-Verlag HD, pp. 177–186.

Bulla, J. (2011). "Hidden Markov Models with t Components. Increased Persistence and Other Aspects". In: *Quantitative Finance* 11.3, pp. 459–475.

Chung, J., C. Gulcehre, K. Cho, and Y. Bengio (2014). "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *arXiv preprint arXiv:1412.3555*.

Dasgupta, S. (Oct. 1999). "Learning Mixtures of Gaussians". In: *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*. ISSN: 0272-5428, pp. 634–644.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1, pp. 1–22.

Dueker, M. J. (Jan. 1997). "Markov Switching in GARCH Processes and Mean-Reverting Stock-Market Volatility". In: *Journal of Business & Economic Statistics* 15.1, pp. 26–34.

Elman, J. L. (Apr. 1990). "Finding Structure in Time". In: *Cognitive Science* 14.2, pp. 179–211.

Fine, S., Y. Singer, and N. Tishby (July 1998). "The Hierarchical Hidden Markov Model: Analysis and Applications". In: *Machine Learning* 32.1, pp. 41–62.

Fisher, R. (1921). "On the 'Probable Error' of a Coefficient of Correlation Deduced from a Small Sample." In: *Fisher014OT*.

Francq, C. and J. M. Zakoian (June 2001). "Stationarity of Multivariate Markov–Wwitching ARMA Models". In: *Journal of Econometrics* 102.2, pp. 339–364.

Gales, M. and S. Young (2007). "The Application of Hidden Markov Models in Speech Recognition". In: *Foundations and Trends in Signal Processing* 1.3, pp. 195–304.

Goodfellow, I., Y. Bengio, and A. Courville (Nov. 2016). *Deep Learning*. MIT Press.

Gray, S. F. (Sept. 1996). "Modeling the Conditional Distribution of Interest Rates as a Regime-Switching Process". In: *Journal of Financial Economics* 42.1, pp. 27–62.

Hamilton, J. D. (1989). "A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle". In: *Econometrica* 57.2, pp. 357–384.

Hamilton, J. D. and B. Raj (June 2013). *Advances in Markov-Switching Models: Applications in Business Cycle Research and Finance*. Springer Science & Business Media.

Hamilton, J. D. and R. Susmel (1994). "Autoregressive Conditional Heteroskedasticity and Changes in Regime". In: *Journal of Econometrics* 64.1-2, pp. 307–333.

Hardy, M. R. (Apr. 2001). "A Regime-Switching Model of Long-Term Stock Returns". In: *North American Actuarial Journal* 5.2, pp. 41–53.

Hochreiter, S. and J. Schmidhuber (Dec. 1997). "Long Short-term Memory". In: *Neural Computation* 9, pp. 1735–1780.

Ilhan, F., O. Karaahmetoglu, I. Balaban, and S. S. Kozat (Feb. 2023). "Markovian RNN: An Adaptive Time Series Prediction Network with HMM-Based Switching for Nonstationary Environments". In: *IEEE Transactions on Neural Networks and Learning Systems* 34.2, pp. 715–728.

Karayiannis, N. (June 1994). "MECA: Maximum Entropy Clustering Algorithm". In: *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*. Vol. 1, pp. 630–635.

Kingma, D. and J. Ba (Dec. 2014). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*.

Krogh, A., M. Brown, I. S. Mian, K. Sjölander, and D. Haussler (Feb. 1994). "Hidden Markov Models in Computational Biology: Applications to Protein Modeling". In: *Journal of Molecular Biology* 235.5, pp. 1501–1531.

Mamon, R. S. and R. J. Elliott, eds. (2014). *Hidden Markov Models in Finance: Further Developments and Applications, Volume II*. Vol. 209. International Series in Operations Research & Management Science. Boston, MA: Springer US.

Nadkarni, P. M., L. Ohno-Machado, and W. W. Chapman (2011). "Natural Language Processing: An Introduction". In: *Journal of the American Medical Informatics Association* 18.5, pp. 544–551.

Nystrup, P., P. N. Kolm, and E. Lindström (2020a). "Greedy Online Classification of Persistent Market States Using Realized Intraday Volatility Features". In: *Journal of Financial Data Science* 2.3, pp. 25–39.

– (2021). "Feature Selection in Jump Models". In: *Expert Systems with Applications* 184, p. 115558.

Nystrup, P., E. Lindström, and H. Madsen (July 2020b). "Learning Hidden Markov Models with Persistent States by Penalizing Jumps". In: *Expert Systems with Applications* 150, p. 113307.

Rydén, T. (Dec. 2008). "EM versus Markov Chain Monte Carlo for Estimation of Hidden Markov Models: A Computational Perspective". In: *Bayesian Analysis* 3.4, pp. 659–688.

Rydén, T., T. Teräsvirta, and S. Åsbrink (1998a). "Stylized Facts of Daily Return Series and the Hidden Markov Model". In: *Journal of Applied Econometrics* 13.3, pp. 217–244.

– (1998b). "Stylized Facts of Daily Return Series and the Hidden Markov Model". In: *Journal of Applied Econometrics* 13.3, pp. 217–244.

Schaller, H. and S. V. Norden (Apr. 1997). "Regime Switching in Stock Market Returns". In: *Applied Financial Economics* 7.2, pp. 177–191.

Shumway, R. H. and D. S. Stoffer (2017). "Characteristics of Time Series". In: *Time Series Analysis and Its Applications: With R Examples*. Ed. by R. H. Shumway and D. S. Stoffer. Springer Texts in Statistics. Cham: Springer International Publishing, pp. 1–44.

Stock, J. H. and M. W. Watson (1996). "Evidence on Structural Instability in Macroeconomic Time Series Relations". In: *Journal of Business & Economic Statistics* 14.1, pp. 11–30.

Turin, W. (2004). "Continuous State HMM". In: *Performance Analysis and Modeling of Digital Transmission Systems*. Ed. by W. Turin. Information Technology: Transmission, Processing and Storage. Boston, MA: Springer US, pp. 295–340.

Wang, X., E. Lebarbier, J. Aubert, and S. Robin (June 2017). "Variational Inference for Coupled Hidden Markov Models Applied to the Joint Detection of Copy Number Variations". In: *The International Journal of Biostatistics* 15.

Wright, S. J. (June 2015). "Coordinate Descent Algorithms". In: *Mathematical Programming* 151.1, pp. 3–34.

Zheng, K., Y. Li, and W. Xu (Aug. 2021). "Regime Switching Model Estimation: Spectral Clustering Hidden Markov Model". In: *Annals of Operations Research* 303.1, pp. 297–319.