# The Monte Carlo Transformer:

Alice Martin, Charles Ollion, Florian Strub, Sylvain Le Corff, Olivier Pietquin

# The Monte Carlo Transformer: a stochastic self-attention model for sequence prediction

Alice Martin*[†⊤], Charles Ollion[‡], Florian Strub[⊥], Sylvain Le Corff[†], and Olivier Pietquin[∓]

[†]Samovar, Télécom SudParis, Département CITI, TIPIC, Institut Polytechnique de Paris, France.
[⊤]CMAP, UMR 7641, École Polytechnique, CNRS, Institut Polytechnique de Paris, France.
[‡]Heuritech, Paris, France.
[∓]Google Research, Brain Team.
[⊥]DeepMind.

**Abstract**

This paper introduces the Sequential Monte Carlo Transformer, an original approach that naturally captures the observations distribution in a recurrent architecture. The keys, queries, values and attention vectors of the network are considered as the unobserved stochastic states of its hidden structure. This generative model is such that at each time step the received observation is a random function of these past states in a given attention window. In this general state-space setting, we use Sequential Monte Carlo methods to approximate the posterior distributions of the states given the observations, and then to estimate the gradient of the log-likelihood. We thus propose a generative model providing a predictive distribution, instead of a single-point estimate.

## 1 Introduction

While neural networks excel at predicting a single-point estimate of a given target for complex machine learning problems, an open research question is the design of neural generative models able to output a predictive distribution, that can capture the inherent variability of the observations or the model's level of confidence in its predictions. The main motivation behind uncertainty quantification is the design of AI systems for critical applications that are safe, and are mitigating risks while automatizing decision-making. On one hand, Bayesian statistics offer a mathematically grounded framework to reason about uncertainty; however, such models generally require prohibitive computational costs, which make them not widely used in practice. On the other hand, frequentist methods and metrics have been developed for confidence estimation in neural networks, in particular in the classification setting [Brosse et al., 2020], [Corbière et al., 2019]. Such works address the

*This action benefited from the support of the Chair ≪ New Gen RetAIl ≫ led by l'X – École Polytechnique and the Fondation de l'École Polytechnique, sponsored by CARREFOUR.

issue of neural networks calibration [Guo et al., 2017] and detection of out-of-distribution samples [Lee et al., 2018]. But few works focus on generative models based on recurrent neural networks.

On another note, the Transformer model introduced in [Vaswani et al., 2017] has achieved impressive results on sequential data problems. In the field of Natural Language Processing (NLP), Transformers have indeed repeatedly outperformed recurrent neural networks (RNNs), and are now the go-to network architectures to solve complex tasks such as Machine Translation or Language Modeling. The Transformer model is based on a self-attention mechanism, that computes dot-product attention for every element of a sequence with respect to all others to model their dependency. By computing in parallel multiple heads of self-attention and by stacking layers of such multi-head attention, the model learns long-range dependencies better than previous state-of-the-art models for sequential data.

Since the release of "Attention is all you need" [Vaswani et al., 2017], the Deep Learning community have been eager to seize the nuts and bolts of the model to improve its training algorithm or to adapt its architecture to new use-cases and types of sequential data, see for instance BERT and its variants ([Devlin et al., 2019], [Radford et al., 2018], [Radford et al., 2019], [Raffel et al., 2019]) and [Chen et al., 2018], [Hao et al., 2019]. Yet, few publications actually focus on the dependency structure at the heart of the Transformer architecture which provides a promising approach to model sequential data. We think that designing a recurrent network inspired from the original Transformer with statistical priors in its architecture could provide a powerful statistical model for capturing the distribution of the observations for sequence prediction problems.

To that end, we introduce the Sequential Monte Carlo (SMC) recurrent Transformer which presupposes that the keys, queries, values and self-attention parameters are unobserved latent states evolving randomly through time. The model relies on a dynamical system inspired from the Transformer, capturing the uncertainty by replacing deterministic self-attention sequences by latent trajectories. The combination of self-attention through time with unobserved model noise allows to generate observations with a complex statistical structure. Self-attention vectors being unobserved stochastic variables, the log-likelihood of the observations is intractable and needs to be estimated. In this paper, we propose to use particle filtering and smoothing methods to draw samples from the distribution of hidden states given observations. The proposed algorithm is based on the auxiliary particle filter of [Liu and Chen, 1998, Pitt and Shephard, 1999], the most widely used particle filtering method and a generalization of previous approaches such as the algorithms proposed in [Gordon et al., 1993] and [Kitagawa, 1996]. The theoretical properties of such methods to estimate the unknown distributions of the internal states and observations have been widely studied, usually in the context of hidden Markov models, see for instance [Del Moral, 2004], [Cappé et al., 2005], [Del Moral et al., 2010], [Del Moral et al., 2015], [Douc et al., 2011], [Dubarry and Le Corff, 2013], [Olsson et al., 2017], [Nguyen et al., 2017].

Fitting the Transformer approach to general state space modeling provides a new promising and interpretable statistical framework for sequential data and recurrent neural networks. From a statistical point of view, the SMC Transformer provides an efficient way of writing each observation as a sophisticated mixture of previous data, while the approximated posterior distribution of the unobserved states captures the states dynamics. We evaluate our model on a series of experiments that first assess the model's ability to predict accurately a known distribution of observations on synthetic datasets, and then apply it to uncertainty quantification in a time-series forecasting problem. The results show that the SMC Transformer manages to capture efficiently the known observation models in the synthetic setting. When performing the task of time-series forecasting on a critical dataset (daily deaths from the Covid-19), the SMC Transformer allows to maintain

accurate mean predictions with satisfactory confidence intervals, that take in account the variability of the observations found in each test example.

# 2 Background

## 2.1 The Transformer model

The Transformer model has been originally developed to propose a new sequence transduction model without recurrence or convolution, and as an alternative to recurrent neural networks for sequence modeling, [Vaswani et al., 2017]. This approach relies entirely on the self-attention mechanism ([Lin et al., 2017]) to model global dependencies regardless of their distance in input or output sequences. Let $(X_s)_{s\geqslant 1}$ be a sequence of observations indexed by $\mathbb{N}$. Transformer models are designed to predict an output $X_s$, for a given index $s$, from input data $X_{-s}$. Each input data $X_s$ is associated with a query $q_s$ and a set of key-value $(k_s, v_s)$ computed from linear transformations of the input. From the set of keys and queries, a softmax score function is first computed, which determines how much focus to place on each input in $X_{-s}$ as $X_s$ is processed. Transformer models use a *scaled dot product attention* to compute this score $\pi$: $\pi_s = \text{softmax}(Q_s K_s^T/\sqrt{r})$, where each line of $Q_s$ (resp. $K_s$) contains an input query (resp. keys) and $r$ denotes the dimensionality of keys and queries. Then, this softmax score is used to compute a weighted sum of the values vectors. The final attention is written as: $\text{Attention}(Q_s, K_s, V_s) = \text{softmax}\left(Q_s K_s^T/\sqrt{r}\right) V_s$, where $V_s$ is the matrix whose rows are the values associated with each input data. The Transformer uses *multi-head* self-attention: the data hidden representations are linearly projected in $h$ subspaces where the attention is computed in parallel, leading to $h$ outputs then concatenated back together.

## 2.2 Sequential Monte Carlo Methods

In real-world machine learning applications, the auxiliary states (queries, keys, values) as well as the observations used to train the neural network are prone to be very noisy. The use of a generative model replacing these deterministic states by random variables evolving according to a dynamical model allows to take into account the uncertainty in the estimation procedure instead of choosing fixed deterministic states. However, considering queries, keys and values as unobserved random variables leads to an intractable likelihood function as the log-likelihood of the observed data $X_{1:T}$, where $X_{u_1:u_2}$ stands for $(X_{u_1}, \ldots, X_{u_2})$ for $u_1 \leqslant u_2$, is obtained by integrating out all latent variables which cannot be done analytically. The exact computation of the likelihood function is therefore not possible in general state spaces. Maximum likelihood estimation cannot be performed directly but a gradient descent algorithm may still be defined using Fisher's identity, see for instance [Cappé et al., 2005]:

$$\nabla_\theta \log p_\theta(X_{1:T}) = \mathbb{E}_\theta[\nabla_\theta \log p_\theta(\zeta_{1:T}, X_{1:T}) | X_{1:T}]\,, \tag{1}$$

where $\theta$ denotes the unknown parameters of the model, $\zeta_{1:T}$ denotes all the unobserved states, $p_\theta$ the joint probability distribution of the observations $X_{1:T}$ and the latent states and $\mathbb{E}_\theta$ the expectation under $p_\theta$. In this paper, we propose to estimate the gradient of the log-likelihood of such general state space model using Sequential Monte Carlo methods, i.e. by a set of random samples associated with non negative importance weights. These particle filters and smoothers approximations combine sequential importance sampling steps to recursively update conditional expectations of the form (1) and importance resampling steps to duplicate or discard particles

according to their importance weights. By (1), $\nabla_\theta \log p_\theta(X_{1:T})$ is then approximated by a weighted sample mean of the form

$$S^M_{\theta,T} = \sum_{m=1}^{M} \omega_n^m \nabla_\theta \log p_\theta(\xi^m_{1:T}, X_{1:T}) \,, \tag{2}$$

where $(\omega_n^m)_{1\leqslant m\leqslant M}$ are nonnegative importance weights such that $\sum_{m=1}^M \omega_T^m = 1$ and where $\xi^m_{1:T}$ are trajectories approximately sampled from the posterior distribution of $\zeta_{1:T}$ given $X_{1:T}$ when the parameter is $\theta$. This approximation of the score function can be plugged into any stochastic gradient algorithm to find local minima of $\theta \mapsto -\log p_\theta(X_{1:T})$.

# 3 The SMC Transformer

## 3.1 Generative model

In the specific case of sequential data, Transformers-based approaches may be introduced to provide a model of the conditional distribution of $X_t$ given $\Delta$ past observations $X_{t-\Delta:t-1}$ where $1 \leqslant \Delta \leqslant t$. In a SMC Transformer with a unique layer, for all $1 \leqslant s \leqslant t$ and all $1 \leqslant h \leqslant n_{\mathsf{heads}}$, define,

$$q^h(s) = W^{h,q} X_s + \Sigma^{1/2}_{h,q} \varepsilon^h_q(s) \,,\; \kappa^h(s) = W^{h,\kappa} X_s + \Sigma^{1/2}_{h,\kappa} \varepsilon^h_\kappa(s) \,,\; v^h(s) = W^{h,v} X_s + \Sigma^{1/2}_{h,v} \varepsilon^h_v(s) \,,$$

where $W^{h,q}$, $W^{h,\kappa}$ and $W^{h,v}$ are unknown $\mathsf{d} \times \mathsf{r}$ matrices, $(\Sigma_{h,q}, \Sigma_{h,\kappa}, \Sigma_{h,v})_{1\leqslant h\leqslant n_{\mathsf{heads}}}$ are unknown semi definite-positive matrices and $(\varepsilon^h_q, \varepsilon^h_\kappa, \varepsilon^h_v)_{1\leqslant h\leqslant n_{\mathsf{heads}}}$ are independent standard Gaussian random vectors in $\mathbb{R}^{\mathsf{r}}$. Then, define the matrix $K^h(t)$ whose columns are the $\kappa^h(t-s)$, $1 \leqslant s \leqslant \Delta$, i.e. the past keys up to time $t-\Delta$. Then, the scores used at time $t$ are

$$\text{score}^h(t) = (q^h(t-1))^\top K^h(t) \quad \text{and} \quad \pi^h(t) = \text{softmax}\left(\text{score}^h(t)/\sqrt{r}\right).$$

Finally, self-attention of the input data is computed, for all $1 \leqslant s \leqslant \Delta$, as,

$$z^h(t) = \sum_{s=1}^{\Delta} \pi^h_s(t) v^h(t-s) + \Sigma^{1/2}_{h,z} \varepsilon^h_z(t) \,, \tag{3}$$

where $\pi^h_s(t)$ denotes the $s$-th component of $\pi^h(t)$ (i.e. the self attention weight of the observation $t-s$), $(\Sigma_{h,z})_{1\leqslant h\leqslant n_{\mathsf{heads}}}$ are unknown semi definite-positive matrices and $(\varepsilon^h_z)_{1\leqslant h\leqslant n_{\mathsf{heads}}}$ are independent standard Gaussian random vectors in $\mathbb{R}^{\mathsf{r}}$. Therefore, $z^h(t)$ is a Gaussian random variable with mean $\mu^h(t) = \sum_{s=1}^{\Delta} \pi^h_s(t) v^h(t-s)$ and covariance matrix $\Sigma_{h,z}$. The *reparametrization trick* is used to write $z^h(t)$ as a deterministic function of $\mu^h(t)$ and $\Sigma_h$ with $\varepsilon^h$ a random variable that does not depend on the parameters. Such trick provides a differentiable transition for the optimization process, see [Kingma and Welling, 2014]. The output $r_t$ is then computed with layer normalization and residual connection steps depending on a parameter $\eta_{state}$.

In a classification setting, the observation model provides a probability vector $G_{\eta_{obs}}(r_t)$ on the finite observation space based on the self-attention vectors. In a regression framework, the observation model is given by

$$X_t = G_{\eta_{obs}}(r_t) + \varepsilon_t \,,$$

where $G_{\eta_{obs}}$ is a FFNN with linear ouput layer and $\varepsilon_t$ is a centered noise, for instance a centered Gaussian random vector with unknown variance $\Sigma_{\text{obs}}$. Let $\theta$ be the vector that contains all the unknown parameters of the model: $\theta = (\eta_{obs}, \eta_{state}, \Sigma_{\text{obs}}, \{\Sigma_h, W^{h,q}, W^{h,\kappa}, W^{h,v}\}_{1\leqslant h\leqslant n_{\mathsf{heads}}})$.

### 3.2 The training algorithm

By Section 3.1, the unobserved state at time $t$ is $\zeta_t = \{z(t), q(t), \kappa(t), v(t)\}$ and the complete-data likelihood may be written

$$p_\theta(X_{1:T}, \zeta_{1:T}) = \prod_{t=1}^{T} p_\theta(\zeta_t|\zeta_{t-\Delta:t-1}, X_{t-\Delta:t-1}) p_\theta(X_t|\zeta_{t-\Delta:t}, X_{t-\Delta:t-1}) \,,$$

where by convention if $t - \Delta \leqslant 1$ then $u_{t-\Delta:s} = u_{1:s}$. The associated probability density functions are

$$p_\theta(X_t|\zeta_{t-\Delta:t}, X_{t-\Delta:t-1}) = G_{\eta_{obs}}\left(r(t)\right)_{X_t} \,,$$

in a classification setting, and

$$p_\theta(X_t|\zeta_{t-\Delta:t}, X_{t-\Delta:t-1}) = \varphi_{G_{\eta_{obs}}(r(t)),\Sigma_{\text{obs}}}(X_t) \,,$$

in a regression setting, where $\varphi_{\mu,\Sigma}$ is the Gaussian probability density function with mean $\mu$ and covariance matrix $\Sigma$. By (1) and (2), the sequential Monte Carlo algorithm approximates $\nabla_\theta \log p_\theta(X_{1:T})$ by a weighted sample mean:

$$S^M_{\theta,T} = \sum_{m=1}^{M} \omega_T^m \sum_{t=1}^{T} \left[\nabla_\theta \log p_\theta(\xi_t^m|\xi_{t-\Delta:t-1}^m, X_{t-\Delta:t-1}) + \nabla_\theta \log p_\theta(X_t|\xi_{t-\Delta:t}^m, X_{t-\Delta:t-1})\right] \,,$$

where the importance weights $(\omega_T^m)_{1\leqslant m\leqslant M}$ and the trajectories $\xi_{1:T}^m$ are sampled according to the particle filter described below. In this paper, we propose to estimate all the parameters of the recurrent architecture based on a gradient descent using $S^M_{\theta,T}$. All parameters related to the noise (the covariance matrices) are estimated using an explicit Expectation Maximization (EM) update [Dempster et al., 1977] each time a batch of observations is processed, see the appendix materials for all details.

**Particle filtering/smoothing algorithm.** For all $t \geqslant 1$, once the observation $X_t$ is available, the weighted particle sample $\{(\omega_t^m, \xi_{1:t}^m)\}_{m=1}^N$ is transformed into a new weighted particle sample. This update step is carried through in two steps, *selection* and *mutation*, using the auxiliary sampler introduced in [Pitt and Shephard, 1999]. New indices and particles $\{(I_{t+1}^m, \xi_{t+1}^m)\}_{m=1}^N$ are simulated independently as follows:

1. Sample $I_{t+1}^m$ in $\{1, \ldots, N\}$ with probabilities proportional to $\{\omega_t^j\}_{1\leqslant j\leqslant N}$.
2. Sample $\xi_{t+1}^m$ using the model introduced in Section 3.1 with the resampled trajectories.

For any $m \in \{1, \ldots, N\}$, the ancestral line $\xi_{1:t+1}^\ell$ is updated as follows $\xi_{1:t+1}^m = (\xi_{1:t}^{I_{t+1}^m}, \xi_{t+1}^m)$ and is associated with the importance weight defined by

$$\omega_{t+1}^m \propto p_\theta(X_{t+1}|\xi_{t+1-\Delta:t+1}^m, X_{t+1-\Delta:t}) \,.$$

Therefore, in a classification setting, $\omega_{t+1}^m \propto [G_{\eta_{obs}}(r_{t+1}^m)]_{X_{t+1}}$, and in a regression setting, $\omega_{t+1}^m \propto \exp\{-\|X_{t+1} - G_{\eta_{obs}}(r_{t+1}^m)\|^2_{\Sigma_{\text{obs}}}/2\}$, where for any vector $u$, $\|u\|^2_{\Sigma_{\text{obs}}} = u^T \Sigma_{\text{obs}}^{-1} u$. This smoother introduced in [Kitagawa, 1996] (see also [Del Moral, 2004] for a discussion) approximates the joint

smoothing distributions of the latent states given the observations using the genealogy of the particles produced by the auxiliary particle filter. The genealogical trajectories are defined recursively and updated at each time step with the particles and indices $(\xi_{k+1}^m, I_{k+1}^m)$. As a result, at each time step, the algorithm selects an ancestral trajectory by choosing its last state at time $k$ which is extended using the newly sampled particle $\xi_{k+1}^m$. As explained for instance in [Kitagawa, 1996, Kitagawa and Sato, 2001], [Fearnhead et al., 2010] and [Poyiadjis et al., 2011], this algorithm suffers from the path degeneracy issue. At each time $t \geqslant 1$, the first step to build a new trajectory is to select an ancestral trajectory chosen among $M$ existing trajectories, as the number of resampling steps increases, the number of ancestral trajectories which are likely to be discarded increases. There are many solutions to improve the approximation $S_{\theta,T}^M$; in this paper, we propose to use the fixed-lag smoother of [Olsson et al., 2008], which means that for each $1 \leqslant t \leqslant n$, the trajectories $\xi_{t-\Delta:t-1}^m$ involved in $S_{\theta,T}^M$ are only resampled up to a few time steps after $t$.

**Inference and predictive distribution.** Based on the generative model proposed in this paper, usual objectives are the *state estimation problem*, which aims at recovering the latent attention parameter $z_t$ at time $t$ given the observations $X_{1:t}$ and the *inference problem* which aims at approximating the distribution of $X_t$ given $X_{1:t-1}$. After a training phase which produces an estimate $\widehat{\theta}$ of $\theta$, the state estimation problem is usually solved by approximating the posterior mean of $z_t$ given the observations $X_{1:t}$ when the model is driven by the parameter $\widehat{\theta}$: $\widehat{z}_t = \sum_{m=1}^M \omega_t^m \xi_t^m$. To solve the inference problem, note that

$$p_{\widehat{\theta}}(X_t|X_{1:t-1}) = \int p_{\widehat{\theta}}(X_t, z_{1:t}|X_{1:t-1})\mathrm{d}z_{1:t}\,,$$

which may be approximated using the weighted samples at time $t-1$ by

$$\widehat{p}_{\widehat{\theta}}^M(X_t|X_{1:t-1}) = \sum_{m=1}^M \omega_{t-1}^m \int p_{\widehat{\theta}}(X_t|z_t) p_{\widehat{\theta}}(z_t|\xi_{1:t-1}^m, X_{1:t-1})\mathrm{d}z_t\,.$$

This distribution may be computed using the states dynamics which implies that $p_{\widehat{\theta}}(z_t|\xi_{1:t-1}^m, X_{1:t-1})$ is a Gaussian probability density function. A Monte Carlo approximation of the predictive probability $\widehat{p}_{\widehat{\theta}}^M(X_t|X_{1:t-1})$ may be obtained straightforwardly by sampling from $p_{\widehat{\theta}}(z_t|\xi_{1:t-1}^m, X_{1:t-1})$. This Monte Carlo estimate can be extended straightforwardly to predictions at future time steps.

# 4 Experiments

## 4.1 Results on synthetic datasets

Consider first an experimental setting where the observations model is known to assess the ability of our model to capture the distributions of the observations. To that end, we designed two synthetic auto-regressive time-series with a sequence length of 24 observations. For **model I**, one data sample $X = (X_0, X_1, ..., X_{24})$ is drawn as follows:

$$X_0 \sim \mathcal{N}(0,\,1)\,,\; X_{t+1} = \alpha X_t + \sigma\varepsilon_{t+1}\,,$$

where $(\varepsilon_t)_{1\leqslant t\leqslant 24}$ are i.i.d standard Gaussian variables independent of $X_0$. For **model II**, the law of a new observation given the past is multimodal and drawn as follows:

$$X_0 \sim \mathcal{N}(0,\,1)\,,\; X_{t+1} = \alpha U_{t+1} X_t + \beta(1-U_{t+1})X_t + \sigma\varepsilon_{t+1}\,,$$

Table 1: Train and validation losses ($\times 10^{-2}$) for synthetic data.

| | *Train loss* | *Val loss* | *Train loss* | *Val loss* |
|---|---|---|---|---|
| | Model I | Model I | Model II | Model II |
| LSTM | | | | |
| $d = 64$, drop $= 0.2$ | 50.02 (0.13) | 50.39 (0.70) | 32.21 (0.13) | 32.37 (0.67) |
| $d = 64$, drop $= 0.3$ | 50.37 (0.11) | 50.81 (0.55) | 32.40 (0.14) | 32.46 (0.68) |
| $d = 64$, drop $= 0.5$ | 51.33 (0.41) | 51.59 (1.29) | 33.70 (0.24) | 33.99 (0.79) |
| SMC-Transf. | | | | |
| $d = 16$, $M = 10$ | 49.92 (0.27) | 50.20 (0.66) | 32,23 (0.21) | 32.18 (0.86) |
| $d = 16$, $M = 30$ | 49.98 (0.11) | 50.23 (0.54) | 32.82 (0.18) | 32.83 (0.63) |

where $(\varepsilon_t)_{1\leqslant t\leqslant 24}$ are i.i.d standard Gaussian variables independent of $X_0$ and $(U_t)_{1\leqslant t\leqslant 24}$ are i.i.d Bernoulli random variables variables with parameter $p$ independent of $X_0$ and of $(\varepsilon_t)_{1\leqslant t\leqslant 24}$. In the first experiment, the dataset is sampled with $\alpha = 0.8$ and $\sigma^2 = 0.5$. In the second experiment, the dataset is sampled with $\alpha = 0.9$, $\beta = 0.6\alpha$, $p = 0.7$ and $\sigma^2 = 0.3$. More details on the hyper-parameters used for such experiments are provided in the appendix. Here, all covariance matrices of the SMC Transformer are assumed to be scalar. Table 1 displays the loss values and their associated standard deviations at the end of training over a 5-fold cross-validation. The displayed losses are the mean squared errors between the predictions (resp. weighted mean of the output particles) and the true observations for the LSTM (resp. the SMC Transformer).

The main interest of our generative model and our estimation procedure is displayed in Figure 1, Table 2 and Figure 2. For each test example, at each time step $t$, 1000 samples are drawn from the SMC estimate of the law of $X_{t+1}$ given $X_{0:t}$. A *MC-Dropout* approach as described in [Gal and Ghahramani, 2015, Section 4] is also used to estimate the uncertainty of $X_{t+1}$ from 1000 stochastic forward passes through the LSTM network with dropout. In this table, dropout is added after the output of the LSTM layer. Figure 1 illustrates that the samples from the SMC estimate match the true distribution at each time step while the dropout samples highly underestimate its variability. These samples are compared in Figure 1 to the true 95% confidence interval at each time step which is available in this synthetic setting. Our generative model captures the probability distribution of the observations while the LSTM models fail to estimate the known variance of the observations. Based on these 1000 samples, Table 2 provides the empirical estimate of the mean squared error of the predictive distribution of $X_{t+1}$ given the past for all time steps $t$. For **Model I**, it is given by $\mathbb{E}[(X_{t+1} - \alpha X_t)^2 | X_t]$ (and the true value is $\sigma^2 = 0.5$) and by $\mathbb{E}[(X_{t+1} - \alpha U_{t+1} X_t - \beta(1 - U_{t+1})X_t)^2 | X_t] = p\mathbb{E}[(X_{t+1} - \alpha X_t)^2 | X_t] + (1-p)\mathbb{E}[(X_{t+1} - \beta X_t)^2 | X_t]$ for **Model II**. The means and standard deviations over the all time steps and all test examples illustrate how the SMC samples capture the true variability of the law of a new observation given the past. In this table, LSTM with dropout also in the LSTM layer are displayed. Dropout networks highly underestimate the variability which yields a predictive MSE too small for both the monomodal distribution (**Model I**) and the multimodal distribution (**Model II**). This is confirmed by Figure 2 which shows histograms over the 100 test examples and over all time steps of the frequency of the 1000 samples which fall in the true confidence interval. Over all test samples and time steps, 78.4% (resp. 93.6%) of the SMC with $M = 30$ (resp. LSTM with $d = 64$ and dropout $= 0.5$) samples fall into the 80% confidence interval. Additional results are available in the appendix.

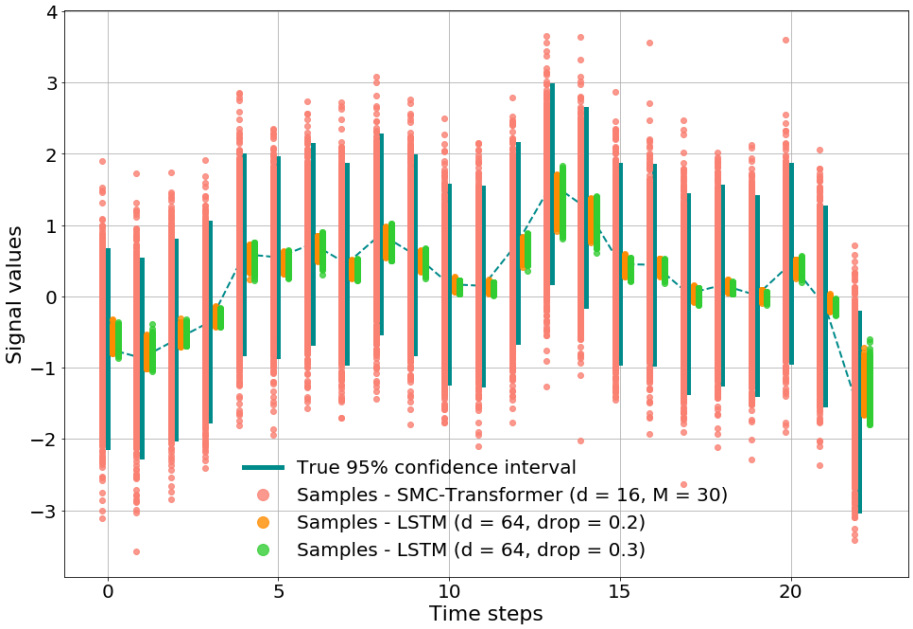


Figure 1: Samples distribution from each approach on a test example (Model I).

Table 2: Mean squared error of the predictive distribution over 1000 samples on 100 test examples. The symbol ‖ separates between dropout only after the output of the LSTM layer (left) and full dropout (right). $\mathbb{E}^t_\alpha = \mathbb{E}[(X_{t+1} - \alpha X_t)^2 | X_t]$ and $\mathbb{E}^t_\beta = \mathbb{E}[(X_{t+1} - \beta X_t)^2 | X_t]$.

| | Model I $\mathbb{E}^t_\alpha$ | Model II $p\mathbb{E}^t_\alpha + (1-p)\mathbb{E}^t_\beta$ |
|---|---|---|
| True model | 0.499 (0.032) | 0.348 (0.070) |
| LSTM | | |
| $d = 64$, drop $= 0.2$ | 0.023 (0.037) ‖ 0.077 (0.115) | 0.027 (0.038) |
| $d = 64$, drop $= 0.3$ | 0.029 (0.043) ‖ 0.116 (0.170) | 0.029 (0.042) |
| $d = 64$, drop $= 0.5$ | 0.032 (0.047) ‖ 0.223 (0.338) | 0.032 (0.048) |
| SMC-Transf. | | |
| $d = 16$, $M = 10$ | 0.543 (0.071) | 0.481 (0.189) |
| $d = 16$, $M = 30$ | 0.515 (0.050) | 0.353 (0.043) |

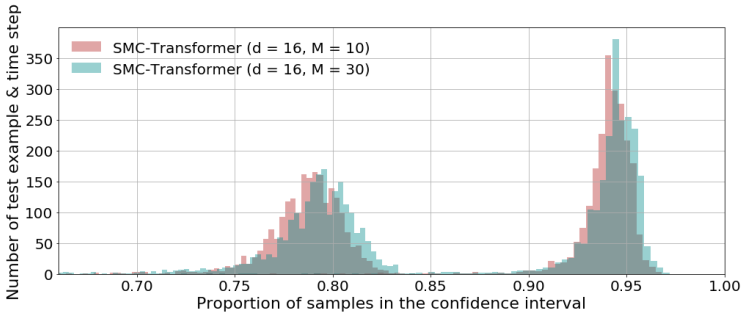


Figure 2: Frequency of the 1000 samples in the true 95% (right) and 80% (left) confidence intervals (Model I).

Table 3: Train and validation losses for covid data.

| | *Train loss* | *Val loss* |
|---|---|---|
| LSTM - $d = 64$, drop $= 0.2 \parallel 0.5$ | $0.023 \parallel 0.039$ | $0.025 \parallel 0.044$ |
| SMC-Transf. - $d = 8$, $M = 10 \parallel 30$ | $0.021 \parallel 0.020$ | $0.024 \parallel 0.023$ |

## 4.2 Covid-19 data

The performance of the stochastic Transformer is analyzed using Covid-19 data[1] which gathers daily deaths from the Covid-19 disease in 3261 US cities. Cities with less than 100 deaths over the time period considered were discarded from the dataset: all algorithms are thus trained on 886 cities decomposed into 80% for training and 20% for test and validation. Table 3 displays the training results for different LSTM with Dropout and SMC Transformer architectures. Figure 3 displays the 20 days ahead predictions, where the true observation is not available after day 40 and replaced by a sample from the estimated model, for three cities in the test dataset. Here, the variability of the observations is different for every sample: for our model, we first estimated a global variance of the observations at training time, and then fine-tuned the estimated noise per test sample at inference, using 30 iterations of an EM algorithm on the first 40 days. As the time horizon increases, the variability of our predictions increases, thus maintaining a satisfactory combination of mean prediction and error bars: the latter are crucial to model both the uncertainty in the observations measurements (e.g potential errors in data collection), and the natural randomness found in such problem, where daily death rates depend on complex dynamics and a multitude of external factors. On the other hand, the underestimation of the output variability given by the MC-Dropout samples for a rate of 0.2 is highlighted as it leads in some test examples to drifted predictions (see Maryland example on the graph). When increasing the dropout rate up to 0.5, the increase in variability comes at the cost of performance's degradation (see Fairfax example and the training table): this suggests that the confidence interval given by MC-Dropout is ill-calibrated, and that there is no easy tuning of the dropout rate that could give satisfactory error bars. Additional graph, results and full details about the hyper-parameters used for training are available in the appendix.

[1] https://github.com/CSSEGISandData/COVID-19

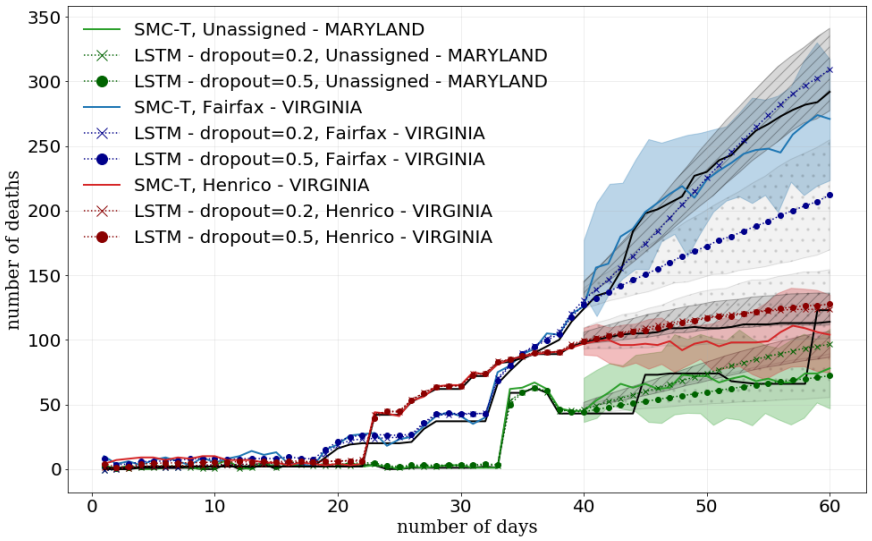


Figure 3: 20 days ahead predictions given by a SMC Transformer ($d = 8$, $M = 10$), and 2 MC-Dropout LSTM ($d = 64$, dropout rate of 0.2 and 0.5). The black lines correspond to the ground truth. The colored (resp. grey stripped and grey dotted) areas correspond to the variability of the SMC Transformer (resp. 2 MC-Dropout LSTM with dropout of 0.2 and 0.5) predictions, i.e an interval of mean $\pm$ std over the 1000 samples at each time step.

# 5 Related work

The SMC Transformer is part of an emerging line of research which focuses on reconciling traditional model-based approaches and model-free ones that arose from the development of Deep Learning. One such approach similar to the SMC Transformer is the Particle Filter Recurrent Neural Network (PF-RNN) from [Ma et al., 2019]. Our work differs in several ways. First, the importance weights of the PF-RNN do not depend directly on the output of the RNN model, i.e. on an observation model estimated thanks to the SMC approach, but on a exterior learned function. The SMC Transformer uses Fisher's Identity to estimate the gradient of the likelihood of the observations, while the PF-RNN directly optimizes the classic cross-entropy loss and adds an evidence lower bound term in the objective function. The prediction algorithm of the PF-RNN only leverages the SMC algorithm to improve performance at training time and outputs a single-point estimate, thus failing to capture the observations distribution, which is the focus of our paper.

On this topic, our work is related to Bayesian Deep Learning: two popular Bayesian methods to quantify uncertainty are *MC-Dropout* from [Gal and Ghahramani, 2015] and *Bayes by Backprop* from [Blundell et al., 2015] and its extension to RNNs [Fortunato et al., 2017]. By introducing randomness in the network's parameters or in the training algorithm, they both train an ensemble of neural networks giving a predictive distribution which tends to be overconfident (as illustrated in Section 4); while our method, based on stochastic states, learns directly an observation model almost able to capture perfectly the true variability of the observations.

# 6 Conclusion

In this paper, we proposed the SMC Transformer, a novel recurrent network naturally capturing the observations distribution. The model maintains a distribution of self-attention parameters as latent states, estimated by a set of particles. It then outputs a distribution of predictions instead of a single-point estimate, and our inference method gives a flexible framework to quantify the observations variability.

The number of particles of the embedded algorithm is a hyper-parameter that can be tuned depending on the complexity of the task and dataset considered, and by taking in account the trade-off between training time and precision. To our knowledge, this is the first method dedicated to estimate uncertainty in the newly-developed Transformer model, and one of the few focusing on uncertainty quantification in the context of sequence prediction. This SMC Transformer layer could be used as a "plug-and-play" layer for uncertainty quantification in a deeper neural network representing sequential data: the data could be first encoded in a multi-layer neural network before being processed sequentially by the SMC Transformer layer. We specifically chose to focus on the Transformer model by having in mind future applications in the NLP field. We are indeed particularly interested in the diversity in language generation that could arise from the particle filter algorithm at inference. For such task, the distributions of words created by the SMC Transformer could naturally mimic the diversity and richness found in natural language.

# A SMC Transformer and the training algorithm

The unobserved state at time $t$ is $\zeta_t = \{z(t), q(t), \kappa(t), v(t)\}$ and the complete-data likelihood may be written

$$p_\theta(X_{1:T}, \zeta_{1:T}) = \prod_{t=1}^{T} p_\theta(\zeta_t | \zeta_{t-\Delta:t-1}, X_{t-\Delta:t-1}) p_\theta(X_t | \zeta_{t-\Delta:t}, X_{t-\Delta:t-1}) \,.$$

In the regression framework of this paper, the associated probability density functions is

$$p_\theta(X_t | \zeta_{t-\Delta:t}, X_{t-\Delta:t-1}) = \varphi_{G_{\eta_{obs}}(r(t)), \Sigma_{\text{obs}}}(X_t) \,,$$

where $\varphi_{\mu,\Sigma}$ is the Gaussian probability density function with mean $\mu$ and covariance matrix $\Sigma$. A graphical representation of our model which describes the dependency between states and observations is proposed in Figure A.

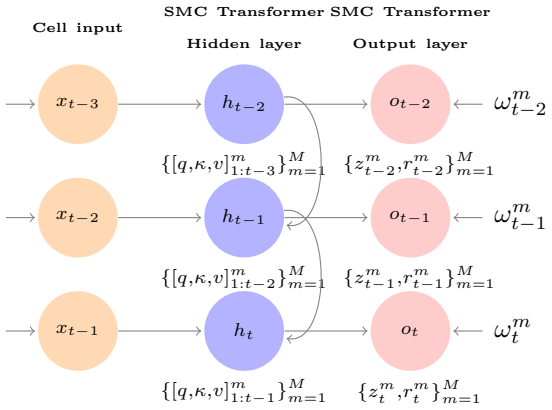


Figure 4: Graphical representation of the SMC transformer for sequential data.

**Particle filtering/smoothing algorithm.** For all $t \geqslant 1$, once the observation $X_t$ is available, the weighted particle sample $\{(\omega_t^m, \xi_{1:t}^m)\}_{m=1}^N$ is transformed into a new weighted particle sample. This update step is carried through in two steps, *selection* and *mutation*, using the auxiliary sampler introduced in [Pitt and Shephard, 1999]. New indices and particles $\{(I_{t+1}^m, \xi_{t+1}^m)\}_{m=1}^N$ are simulated independently as follows:

1. Sample $I_{t+1}^m$ in $\{1, \ldots, N\}$ with probabilities proportional to $\{\omega_t^j\}_{1 \leqslant j \leqslant N}$.
2. Sample $\xi_{t+1}^m$ using the model with the resampled trajectories.

In the regression framework of the paper, for any $m \in \{1, \ldots, N\}$, the ancestral line $\xi_{1:t+1}^\ell$ is updated as follows $\xi_{1:t+1}^m = (\xi_{1:t}^{I_{t+1}^m}, \xi_{t+1}^m)$ and is associated with the importance weight defined by

$$\omega_{t+1}^m \propto p_\theta(X_{t+1} | \xi_{t+1-\Delta:t+1}^m, X_{t+1-\Delta:t}) = \exp\{-\|X_{t+1} - G_{\eta_{obs}}(r_{t+1}^m)\|^2_{\Sigma_{\text{obs}}}/2\} \,.$$

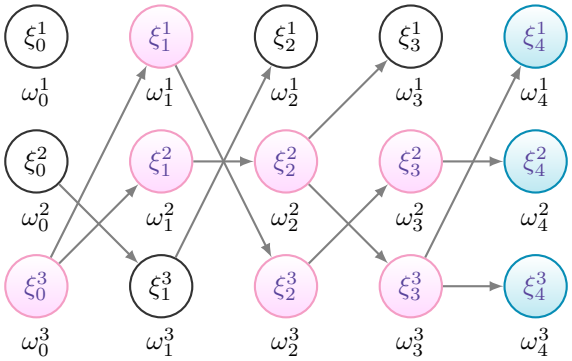

Figure 5: Particle filter: $N = 3$, $n = 4$.

The algorithm is illustrated in Figure 5: particles at the last time step are in blue and pink particles are the ones which appear in the genealogy of at least one blue particle. White particle have not been selected to give birth to a path up to the last time. In Figure 5, the $N = 3$ genealogical trajectories are $\xi^1_{0:4} = (\xi^3_0, \xi^2_1, \xi^2_2, \xi^3_3, \xi^1_4)$, $\xi^2_{0:4} = (\xi^3_0, \xi^1_1, \xi^3_2, \xi^2_3, \xi^2_4)$, $\xi^3_{0:4} = (\xi^3_0, \xi^2_1, \xi^2_2, \xi^3_3, \xi^3_4)$.

**The training algorithm** The sequential Monte Carlo algorithm approximates $\nabla_\theta \log p_\theta(X_{1:T})$ by a weighted sample mean:

$$S^M_{\theta,T} = \sum_{m=1}^{M} \omega^m_T \sum_{t=1}^{T} \left[\nabla_\theta \log p_\theta(\xi^m_t | \xi^m_{t-\Delta:t-1}, X_{t-\Delta:t-1}) + \nabla_\theta \log p_\theta(X_t | \xi^m_{t-\Delta:t}, X_{t-\Delta:t-1})\right] ,$$

where the importance weights $(\omega^m_T)_{1\leqslant m\leqslant M}$ and the trajectories $\xi^m_{1:T}$ are sampled according to the particle filter described below. Thanks to Fisher's identity, this approximation only requires to compute the gradient of the state model loglikelihood $\theta \mapsto \log p_\theta(\xi^m_t | \xi^m_{t-\Delta:t-1}, X_{t-\Delta:t-1})$ and the gradient of the observation model loglikelihood $\theta \mapsto \log p_\theta(X_t | \xi^m_{t-\Delta:t}, X_{t-\Delta:t-1})$. There is no need to compute the gradient of the weights $\omega^m_T$ which depend on the parameter $\theta$. Using TensorFlow function *tf.stop_gradient* on these weights, this allows to train the model with the following loss function:

$$\theta \mapsto -\sum_{m=1}^{M} \omega^m_T \sum_{t=1}^{T} \left[\log p_\theta(\xi^m_t | \xi^m_{t-\Delta:t-1}, X_{t-\Delta:t-1}) + \log p_\theta(X_t | \xi^m_{t-\Delta:t}, X_{t-\Delta:t-1})\right] .$$

In this paper, we propose to estimate all the parameters of the recurrent architecture based on a gradient descent using $S^M_{\theta,T}$. All parameters related to the noise (the covariance matrices) are estimated using an explicit Expectation Maximization (EM) update [Dempster et al., 1977] each time a batch of observations is processed. For each sequence of observations, the EM update relies on the approximation of the intermediate quantity

$$\begin{aligned}&\mathbb{E}[\log p_\theta(X_{1:T}, \zeta_{1:T}) | X_{1:T}] \\ &\qquad = \mathbb{E}[\sum_{t=1}^{T} \log p_\theta(\zeta_t | \zeta_{t-\Delta:t-1}, X_{t-\Delta:t-1}) + \log p_\theta(X_t | \zeta_{t-\Delta:t}, X_{t-\Delta:t-1}) | X_{1:T}]\end{aligned}$$

by the following particle-based estimator:

$$Q^M_{\theta,T} = \sum_{m=1}^{M} \omega_T^m \sum_{t=1}^{T} \left[\log p_\theta(\xi_t^m | \xi^m_{t-\Delta:t-1}, X_{t-\Delta:t-1}) + \log p_\theta(X_t | \xi^m_{t-\Delta:t}, X_{t-\Delta:t-1})\right] .$$

Then, $Q^M_{\theta,T}$ may be maximized with respect to all covariances to obtain the new estimates. This is a straightforward update which yields for instance for $\Sigma_{\text{obs}}$ for the $p$-th update:

$$\Sigma^p_{\text{obs}} = \frac{1}{T} \sum_{m=1}^{M} \omega_T^m \sum_{t=1}^{T} (X_t - G_{\eta_{obs}}(r_t^m))^\top (X_t - G_{\eta_{obs}}(r_t^m)) ,$$

where $r_t^m$ are the resampled particles at time $t$. The estimate $\widehat{\Sigma}_{\text{obs}}$ of $\Sigma_{\text{obs}}$ is then updated as follows:

$$\widehat{\Sigma}_{\text{obs}} = (1 - \eta_p)\widehat{\Sigma}_{\text{obs}} + \eta_p \Sigma^p_{\text{obs}}$$

where $\eta_p$ is a learning rate chosen by the user ($\eta_p = p^{-0.6}$ in the experiments).

**Inference and predictive distribution used in the experiments.** To solve the inference problem, note that

$$p_{\widehat{\theta}}(X_t | X_{1:t-1}) = \int p_{\widehat{\theta}}(X_t, z_{1:t} | X_{1:t-1}) \mathrm{d}z_{1:t} ,$$

which may be approximated using the weighted samples at time $t-1$ by

$$\widehat{p}^M_{\widehat{\theta}}(X_t | X_{1:t-1}) = \sum_{m=1}^{M} \omega_{t-1}^m \int p_{\widehat{\theta}}(X_t | z_t) p_{\widehat{\theta}}(z_t | \xi^m_{1:t-1}, X_{1:t-1}) \mathrm{d}z_t .$$

A Monte Carlo approximation of the integral involved in the predictive probability $\widehat{p}^M_{\widehat{\theta}}(X_t | X_{1:t-1})$ may be obtained straightforwardly. In the experiments, $\widehat{p}^M_{\widehat{\theta}}$ was approximated as follows. For all $1 \leqslant k \leqslant N$ (with $N = 1000$ in our experiments), sample independently $I_k$ in $\{1, \dots, M\}$ with weights proportional to $(\omega^m_{t-1})_{1 \leqslant m \leqslant M}$ and then $z_t^k$ from $p_{\widehat{\theta}}(z_t | \xi^{I_k}_{1:t-1}, X_{1:t-1})$, i.e. from our transformer model with the selected past trajectory of hidden states $\xi^{I_k}_{1:t-1}$. Finally $r_t^k$ can be computed from $z_t^k$ and $\widehat{p}^M_{\widehat{\theta}}$ is approximated by $N^{-1} \sum_{k=1}^{N} \varphi_{G_{\eta_{obs}}(r_t^k), \widehat{\Sigma}_{\text{obs}}}(X_t)$ i.e. by a mixture of Gaussian distributions with means given by $(r_t^k)_{1 \leqslant k \leqslant N}$ and variance $\widehat{\Sigma}_{\text{obs}}$.

# B Experiments

## B.1 Additional results on synthetic datasets

The synthetic datasets were generated with 1000 samples: we used 800 of them for training and 100 of them for test and validation. For training the models, we used a batch size of 32, a number of epochs equal to 50, and the ADAM algorithm with a learning rate of 0.001 and the original custom schedule found in [Vaswani et al., 2017].

Table 4 displays the loss values and their associated standard deviations at the end of training over a 5-fold cross-validation. The displayed losses are the mean squared errors between the predictions (resp. weighted mean of the output particles) and the true observations for the LSTM (resp.

Table 4: Train and validation losses ($\times 10^{-2}$) for synthetic data.

| | *Train loss* | *Val loss* | *Train loss* | *Val loss* |
|---|---|---|---|---|
| | Model I | Model I | Model II | Model II |
| LSTM | | | | |
| $d = 32$, drop $= 0.1$ | 49.91 (0.10) | 50.24 (0.58) | 32.21 (0.09) | 32.28 (0.69) |
| $d = 32$, drop $= 0.2$ | 50.43 (0.04) | 50.79 (0.73) | 32.49 (0.10) | 32.50 (0.65) |
| $d = 32$, drop $= 0.3$ | 50.93 (0.26) | 51.13 (0.54) | 32.87 (0.07) | 32.88 (0.65) |
| $d = 32$, drop $= 0.4$ | 51.89 (0.29) | 51.98 (1.17) | 34.11 (0.15) | 34.02 (0.87) |
| $d = 32$, drop $= 0.5$ | 52.78 (0.36) | 53.26 (1.10) | 34.75 (0.18) | 34.81 (0.90) |
| $d = 64$, drop $= 0.1$ | 49.76 (0.10) | 50.14 (0.66) | 32.06 (0.10) | 32.17 (0.58) |
| $d = 64$, drop $= 0.2$ | 50.02 (0.13) | 50.39 (0.70) | 32.21 (0.13) | 32.37 (0.67) |
| $d = 64$, drop $= 0.3$ | 50.37 (0.11) | 50.81 (0.55) | 32.40 (0.14) | 32.46 (0.68) |
| $d = 64$, drop $= 0.4$ | 50.77 (0.36) | 51.07 (1.15) | 33.39 (0.23) | 33.34 (0.90) |
| $d = 64$, drop $= 0.5$ | 51.33 (0.41) | 51.59 (1.29) | 33.70 (0.24) | 33.99 (0.79) |
| SMC-Transf. | | | | |
| $d = 8$, $M = 10$ | 49.96 (0.18) | 50.34 (0.33) | 32,26 (0.14) | 32,23 (0.76) |
| $d = 16$, $M = 10$ | 49.92 (0.27) | 50.20 (0.66) | 32,23 (0.21) | 32.18 (0.86) |
| $d = 16$, $M = 30$ | 49.98 (0.11) | 50.23 (0.54) | 32.82 (0.18) | 32.83 (0.63) |

the SMC Transformer). Additional dropout rates and a Transformer with $d = 8$ were considered to complete the experiments given in the main part of the paper.

These neural networks were also considered in Table 5.

### B.2 Additional results on the Covid-19 data

The hyper-parameters used for training the SMC Transformer and the LSTM with dropout were the following: the models were trained during 50 epochs with a batch size of 32, with a learning rate of 0.001 for the LSTM and the original learning rate with custom schedule from [Vaswani et al., 2017]. The LSTM with dropout include a dropout layer inside the LSTM layer, and before the output layer.

Figure 6 displays another example of 20 days ahead predictions, where the true observation is not available after day 40 and replaced by a sample from the estimated model, for two cities in the test dataset. The figure shows that the confidence intervals given by the SMC Transformer capture almost entirely the true sequence of predictions for these two cities with high variability in daily death rates, which is not the case for the two LSTM with dropout.

## References

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 37.

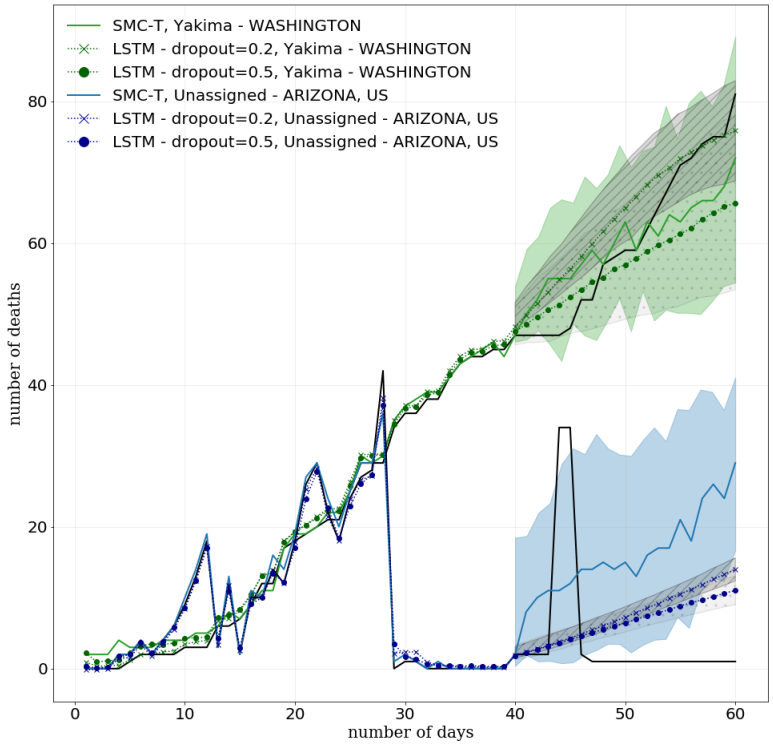


Figure 6: 20 days ahead predictions of covid-19 daily deaths for 2 cities. The black lines represent the true predictions, and the colored areas represent the confidence interval given by a SMC Transformer with 10 particles, while the grey stripped (resp. dotted) areas represent the confidence intervals given by a LSTM with a dropout rate equal to 0.2 (resp. to 0.5).

Table 5: Mean squared error of the predictive distribution over 1000 samples on 100 test examples. The symbol ‖ separates between dropout only after the output of the LSTM layer (left) and full dropout (right). $\mathbb{E}^t_\alpha = \mathbb{E}[(X_{t+1} - \alpha X_t)^2 | X_t]$ and $\mathbb{E}^t_\beta = \mathbb{E}[(X_{t+1} - \beta X_t)^2 | X_t]$.

| | Model I $\mathbb{E}^t_\alpha$ | Model II $p\mathbb{E}^t_\alpha + (1-p)\mathbb{E}^t_\beta$ |
|---|---|---|
| True model | 0.499 (0.032) | 0.348 (0.070) |
| LSTM | | |
| $d = 32$, drop $= 0.1$ | 0.024 (0.042) | 0.027 (0.038) |
| $d = 32$, drop $= 0.2$ | 0.028 (0.051) | 0.030 (0.042) |
| $d = 32$, drop $= 0.3$ | 0.040 (0.064) | 0.033 (0.047) |
| $d = 32$, drop $= 0.4$ | 0.039 (0.061) | 0.036 (0.054) |
| $d = 32$, drop $= 0.5$ | 0.051 (0.077) | 0.041 (0.061) |
| $d = 64$, drop $= 0.1$ | 0.019 (0.030) ‖ 0.038 (0.059) | 0.026 (0.036) |
| $d = 64$, drop $= 0.2$ | 0.023 (0.037) ‖ 0.077 (0.115) | 0.027 (0.038) |
| $d = 64$, drop $= 0.3$ | 0.029 (0.043) ‖ 0.116 (0.170) | 0.029 (0.042) |
| $d = 64$, drop $= 0.4$ | 0.027 (0.041) ‖ 0.155 (0.241) | 0.029 (0.044) |
| $d = 64$, drop $= 0.5$ | 0.032 (0.047) ‖ 0.223 (0.338) | 0.032 (0.048) |
| SMC-Transf. | | |
| $d = 16$, $M = 10$ | 0.543 (0.071) | 0.481 (0.189) |
| $d = 16$, $M = 30$ | 0.515 (0.050) | 0.353 (0.043) |

[Brosse et al., 2020] Brosse, N., Riquelme, C., Martin, A., Gelly, S., and Moulines, É. (2020). On last-layer algorithms for classification: Decoupling representation from uncertainty estimation. *arXiv preprint arXiv:2001.08049*.

[Cappé et al., 2005] Cappé, O., Moulines, E., and Rydén, T. (2005). *Inference in Hidden Markov Models*. Springer.

[Chen et al., 2018] Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G., Jones, L., Parmar, N., Schuster, M., Chen, Z., et al. (2018). The best of both worlds: Combining recent advances in neural machine translation. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 1:76–86.

[Corbière et al., 2019] Corbière, C., Thome, N., Bar-Hen, A., Cord, M., and Pérez, P. (2019). Addressing failure prediction by learning model confidence. In *Advances in Neural Information Processing Systems*, pages 2898–2909.

[Del Moral, 2004] Del Moral, P. (2004). *Feynman-Kac Formulae. Genealogical and Interacting Particle Systems with Applications*. Springer.

[Del Moral et al., 2010] Del Moral, P., Doucet, A., and Singh, S. S. (2010). A backward particle interpretation of feynman-kac formulae. *ESAIM: Mathematical Modelling and Numerical Analysis*, 44(5):947–975.

[Del Moral et al., 2015] Del Moral, P., Doucet, A., and Singh, S. S. (2015). Uniform stability of a particle approximation of the optimal filter derivative. *SIAM Journal on Control and Optimization*, 53(3):1278–1304.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–38 (with discussion).

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186.

[Douc et al., 2011] Douc, R., Garivier, A., Moulines, E., Olsson, J., et al. (2011). Sequential monte carlo smoothing for general state space hidden markov models. *The Annals of Applied Probability*, 21(6):2109–2145.

[Dubarry and Le Corff, 2013] Dubarry, C. and Le Corff, S. (2013). Non-asymptotic deviation inequalities for smoothed additive functionals in nonlinear state-space models. *Bernoulli*, 19(5B):2222–2249.

[Fearnhead et al., 2010] Fearnhead, P., Wyncoll, D., and Tawn, J. (2010). A sequential smoothing algorithm with linear computational cost. *Biometrika*, 97(2):447–464.

[Fortunato et al., 2017] Fortunato, M., Blundell, C., and Vinyals, O. (2017). Bayesian recurrent neural networks. *CoRR*, abs/1704.02798.

[Gal and Ghahramani, 2015] Gal, Y. and Ghahramani, Z. (2015). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 48.

[Gordon et al., 1993] Gordon, N., Salmond, D., and Smith, A. (1993). Novel approach to nonlinear/non-Gaussian bayesian state estimation. *IEE Proc. F, Radar Signal Process*, 140:107–113.

[Guo et al., 2017] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org.

[Hao et al., 2019] Hao, J., Wang, X., Yang, B., Wang, L., Zhang, J., and Tu, Z. (2019). Modeling recurrence for transformer. *Proceedings of the 2019 Conference of the North*.

[Kingma and Welling, 2014] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

[Kitagawa, 1996] Kitagawa, G. (1996). Monte-Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 1:1–25.

[Kitagawa and Sato, 2001] Kitagawa, G. and Sato, S. (2001). Monte carlo smoothing and self-organizing state-space model. In Doucet, A., De Freitas, N., and Gordon, N., editors, *Sequential Monte Carlo methods in Practice*. Springer.

[Lee et al., 2018] Lee, K., Lee, H., Lee, K., and Shin, J. (2018). Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[Lin et al., 2017] Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130.

[Liu and Chen, 1998] Liu, J. and Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93:1032–1044.

[Ma et al., 2019] Ma, X., Karkus, P., Hsu, D., and Lee, W. S. (2019). Particle filter recurrent neural networks. *CoRR*, abs/1905.12885.

[Nguyen et al., 2017] Nguyen, T., Le Corff, S., and Moulines, É. (2017). On the two-filter approximations of marginal smoothing distributions in general state-space models. *Advances in Applied Probability*, 50(1):154–177.

[Olsson et al., 2008] Olsson, J., Cappe, O., Douc, R., and Moulines, E. (2008). Sequential monte carlo smoothing with application to parameter estimation in nonlinear state space models. *Bernoulli*, 14(1):155–179.

[Olsson et al., 2017] Olsson, J., Westerborn, J., et al. (2017). Efficient particle-based online smoothing in general hidden markov models: the paris algorithm. *Bernoulli*, 23(3):1951–1996.

[Pitt and Shephard, 1999] Pitt, M. K. and Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599.

[Poyiadjis et al., 2011] Poyiadjis, G., Doucet, A., and Singh, S. (2011). Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98:65–80.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

[Raffel et al., 2019] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.