FlowHMM: Flow-based continuous hidden Markov models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Continuous hidden Markov models (HMMs) assume that observations are generated from a mixture of Gaussian densities, limiting their ability to model more complex distributions. In this work, we address this shortcoming and propose novel continuous HMM models, dubbed FlowHMMs, that enable learning general continuous observation densities without constraining them to follow a Gaussian distribution or their mixtures. To that end, we leverage deep flow-based architectures that model complex, non-Gaussian functions and propose two variants of training FlowHMM model. The first one, based on an expectation-maximization (EM) technique, can be applied directly to continuous multidimensional data, yet its application to larger data sequences remains computationally expensive. Therefore, we also present a second approach to training our FlowHMM that relies on the co-occurrence matrix of discretized observations and considers the joint distribution of pairs of co-observed values, hence rendering the training time independent of the training sequence length. As a result, we obtain a model that can be flexibly adapted to the characteristics and dimensionality of the data. We perform a variety of experiments in which we compare both training strategies with a baseline of Gaussian mixture models. We show, that in terms of quality of the recovered probability distribution, accuracy of prediction of hidden states, and likelihood of unseen data, our approach outperforms the standard Gaussian methods.

1 Introduction

Hidden Markov models (HMMs) are a standard tool in modeling and analysis of time series data. Although structurally simple, they have been successfully applied in a wide variety of applications, ranging from finance [1], speech recognition [2] to computational biology [3] and climate modeling [4].

HMMs are capable of solving complex problems, but their adoption is limited due to several reasons. First, the training process of HMMs typically relies on the Baum-Welch algorithm [5] — a particular case of expectation-maximization (EM) method, which offers a relatively slow convergence to a local maximum. To reduce this burden, several discrete HHMs introduce the so-called co-occurrence matrix that aggregates information about the probability of jointly observed values in the chain and estimate it together with the parameters of the whole model [6, 7, 8, 9, 10]. Although the co-occurrence matrix is computed only once and then used to significantly reduce convergence time during optimization, its application is strictly limited to discrete HMMs and cannot be easily generalized to their continuous variants.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Secondly, continuous HMM models are restricted to follow standard parametrized distributions when modeling the observations. Most of them use either Gaussians or other parametric families of distributions [11, 12], while the others rely on the mixtures of Gaussians or apply semiparametric distribution estimation [13, 14]. As a result, the existing HMMs have limited ability to model complex observations that do not follow the distributions mentioned above. This, in turn, hinders their application in real-life use cases, *e.g.* in human action recognition [15].

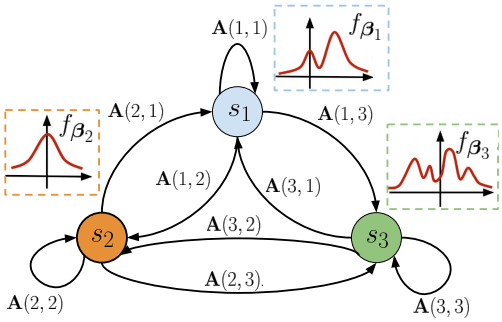

Figure 1: The concept of FlowHMM for $L = 3$ states and transition matrix $\mathbf{A}$. Each emission distribution characterised by density $f_{\boldsymbol{\beta}_l(\cdot)}$ is modeled using a separate flow component. Thanks to this, they can adjust to complex, non-Gaussian distributions.

In this work, we address the shortcomings of the existing HMMs and introduce FlowHMM, a novel continuous hidden Markov model that learns a general continuous distribution of observations by exploiting the properties of flow-based models [16]. The core idea of our approach is to model the distributions of the observations with normalizing flows, instead of Gaussians or their mixtures. Flows map a simple Gaussian prior to more complex distributions using a parametrized neural network. This formulation enables flow-based models to overpass parametric models in their flexibility to handle data samples of unknown distributions. Moreover, flow-based models naturally extend to a multimodal setting, which effectively renders obsolete the tedious process of tuning the number of hidden states.

Practically, training the model with flow-based components requires a gradient-based optimization. To achieve that, we propose two variants of training the FlowHMM model. The first approach is based on an expectation-maximization (EM) technique and can be applied directly to continuous multidimensional data. In the second method, we discretize the continuous values during training and leverage the co-occurrence matrix. As a result, we provide an end-to-end training procedure that jointly optimizes the flow-based component parameters and the co-occurrence matrix using standard gradient-based techniques. Since the model is trained in a discretized form, the optimization process is simple and more efficient than the competing HMM training procedures, while during inference, we can still use a continuous version of our FlowHMM model.

To summarize, the *main contribution of our paper* is a novel continuous HMM method dubbed FlowHMM with two alternative training scenarios, capable of modeling complex multimodal distributions of observations without constraining them to follow Gaussians. Not only does it outperform the competing approaches, but it also increases the efficiency of the required optimization procedure.

2 Background

2.1 Hidden Markov models

Let $\{X_k\}_{k\geq 0}$ be an ergodic, time homogeneous Markov chain over hidden states $\mathcal{S} = \{s_1, \ldots, s_L\}$ with a stationary distribution $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_L)$, and a transition matrix $\mathbf{A}$, *i.e.*, for any k we have $\mathbb{P}(X_{k+1} = s_j | X_k = s_i) = \mathbf{A}(i, j)$. Let $\{Y_k\}_{k\geq 0}$ be a sequence of random variables taking values in $\mathcal{V}$ (the *observation space*), which can be continuous or discrete. Let us fix some time horizon T. Random variables $Y_{0:T} = (Y_0, \ldots, Y_T)$ are independent conditionally on the state sequence $X_{0:T} = (X_0, \ldots, X_T)$, *i.e.*:

$$p(Y_{0:T} = y_{0:T} | X_{0:T} = x_{0:T}) = \prod_{k=0}^{T} p(y_k | x_k), \tag{1}$$

where $p(y|x) \equiv \mathbb{P}(Y = y | X = x)$ represents the so-called *emission probabilities* for discrete observations. We will shortly write $p(y_{0:T})$ for $p(Y_{0:T} = y_{0:T})$ and similarly *e.g.*, $p(y_{0:T} | x_{0:T})$

for $p(Y_{0:T} = y_{0:T}|X_{0:T} = x_{0:T})$ to simplify the notation. Considering the continuous case, $p(y|x)$ represents the conditional emission density function for a given state x. For a Gaussian Mixture HMM (which we simply call Gaussian HMM) we assume that $p(y|x) = \sum_{k=1}^{K} \alpha_{x,k}\mathcal{N}(y; \mu_{x,k}, \sigma^2_{x,k})$, where $\alpha_{x,\cdot}$ is a distribution on $\{1, \ldots, K\}$. The values of parameters $\{\alpha_{x,k}, \mu_{x,k}, \sigma^2_{x,k}\}_{k=1}^{K}$ are determined by the conditioning value x. Note that we assume that each mixture has the same number of components K. For $K = 1$ we have a classic Gaussian HMM.

HMM model can be parametrized by $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\beta}\}$, where $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1, \ldots \boldsymbol{\beta}_L\}$, and $\boldsymbol{\beta}_l$ stays behind the parameters of $p(y|l)$, for a given state l. For discrete case, $\boldsymbol{\beta}_l$ represents the parameters for categorical distribution, while for a Gaussian HMM we have $\boldsymbol{\beta}_l = \{\alpha_{l,k}\mu_{l,k}, \sigma^2_{l,k}\}_{k=1}^{K}$. The probability of observing the sequence of observations $y_{0:T}$ can be expressed as:

$$p(y_{0:T}; \boldsymbol{\theta}) = \sum_{x_{0:T} \in \mathcal{S}^{T+1}} p(y_{0:T}|x_{0:T}; \boldsymbol{\beta}) p(x_{0:T}; \mathbf{A}, \boldsymbol{\pi}), \tag{2}$$

where $p(y_{0:T}|x_{0:T}; \boldsymbol{\beta})$ is given by Eq. (1), and $p(x_{0:T}; \mathbf{A}, \boldsymbol{\pi})$ can be expressed as:

$$p(x_{0:T}; \mathbf{A}, \boldsymbol{\pi}) = \mathbb{P}(x_0) \prod_{k=1}^{T} \mathbb{P}(x_k|x_{k-1}) = \pi_{x_0} \prod_{k=1}^{T} \mathbf{A}(x_{k-1}, x_k). \tag{3}$$

2.2 Normalizing Flows

Normalizing flows [16] are generative models that can be efficiently trained via direct likelihood estimation thanks to the application of the change-of-variable formula. Practically, they utilize sequence of parametric and invertible transformations: $y = h_n \circ \cdots \circ h_1(z)$. The goal of the transformation is to map z from the known, normal distribution $\mathcal{N}(z; 0, 1)$ to the more complex distribution described by a density function $f(y)$ from the observation domain. The log-probability for y can be expressed as:

$$\log f(y) = \log \mathcal{N}(z; 0, 1) - \sum_{n=1}^{N} \log \left| \det \frac{\partial h_n}{\partial z_{n-1}} \right|. \tag{4}$$

One of the main challenges while designing the normalizing flows is selection of a proper form of transformation functions h_n. The sequence of discrete transformations can be replaced by continuous equivalent by application of Continuous Normalizing Flows (CNFs) [17], where the aim is to solve differential equation of the form $\frac{dz}{dt} = g_{\boldsymbol{\beta}}(z(t), t)$, where $g_{\boldsymbol{\beta}}(z(t), t)$ represents the function of dynamics, described by parameters $\boldsymbol{\beta}$. Our goal is to find a solution of the equation in t_1, $y := z(t_1)$, assuming the given initial state $z := z(t_0)$ with a known prior. The transformation function $h_{\boldsymbol{\beta}}$ and its inverse are defined as defined as:

$$y = h_{\boldsymbol{\beta}}(z) = z + \int_{t_0}^{t_1} g_{\boldsymbol{\beta}}(z(t), t)dt, \quad h_{\boldsymbol{\beta}}^{-1}(y) = y - \int_{t_0}^{t_1} g_{\boldsymbol{\beta}}(z(t), t)dt. \tag{5}$$

The log-probability of y can be computed by (where $h_{\boldsymbol{\beta}}^{-1}(y) = z$):

$$\log f_{\boldsymbol{\beta}}(y) = \log \mathcal{N}(h_{\boldsymbol{\beta}}^{-1}(y; 0, 1)) - \int_{t_0}^{t_1} \left(\frac{dg_{\boldsymbol{\beta}}(z(t), t)}{dz(t)} \right) dt. \tag{6}$$

3 FlowHMM

In this section we introduce FlowHMMs - HMM variants of continuous flow capable to model the observations using complex, non-Gaussian distributions. The idea behind this approach is to model each of conditional densities $p(y|x = s_l) = f_{\boldsymbol{\beta}_l}(y)$, for each of the considered states, $s_l \in S$, using a separate CNF module. In practice, $p(y|x = s_l)$ can be calculated using formula given by Eq. (6) with the parameters $\boldsymbol{\beta}_l$ dedicated for the state s_l. The idea of our approach is illustrated in Fig. 1.

In FlowHMM we assume that $\{X_k\}$ is stationary, *i.e.*, the stationary distribution $\boldsymbol{\pi}$ is also its initial distribution. In such a case, instead of using $\mathbf{A}$, the model can be equivalently represented by the state joint probabilities:

$$\mathbf{S}(i,j) = \mathbb{P}(X_k = s_i, X_{k+1} = s_j) = \mathbb{P}(X_{k+1} = s_j | X_k = s_i)\mathbb{P}(X_k = s_i) = \mathbf{A}(i,j)\pi_i. \tag{7}$$

Note that π_i can be computed as $\sum_j \mathbf{S}(i,j)$, what can be written as $\pi_i = \mathbf{1}_i \mathbf{S} \mathbf{1}^T$, where $\mathbf{1}$ denotes a vectors consisting of ones and $\mathbf{1}_i$ consists of 1 on position i and zeros otherwise. For such a formulation, the FlowHMM model can be parametrized by $\boldsymbol{\theta} = \{\mathbf{S}, \boldsymbol{\beta}_1, \dots \boldsymbol{\beta}_L\}$.

We propose two variants of FlowHMM models, which differ mainly in a training process: *EM-based model* $\mathcal{F}^{\mathrm{EM}}$ which is trained with an expectation-maximisation approach directly on continuous data, and *co-occurrence matrix-based model* $\mathcal{F}^{\mathbf{Q}}$, that utilizes co-occurrence matrix and is trained in end-to-end setting using discretized sequences. The first of the proposed methods does not require discretisation step, but is costly and ineffective for larger sequences. $\mathcal{F}^{\mathbf{Q}}$ eliminates that problem, and makes the training time independent to the length of the training sequence (only estimating empirical co-occurrence matrix depends on T, but the time is marginal). In upcoming sections we are going to introduce both models in more details.

3.1 Training FlowHMM $\mathcal{F}^{\mathrm{EM}}$ model

Given a sequence of observations $y_{0:T} = (y_0, \dots, y_T)$, the model is trained by optimizing the log-likelihood $\log p(y_{0:T}; \boldsymbol{\theta})$ (see Eq. (2)), where we aim at finding the optimal values of $\mathbf{S}^*$ (and thus $\mathbf{A}^*$ and $\boldsymbol{\pi}^*$) and parameters of flow models $\boldsymbol{\beta}_l^*$, such that $\boldsymbol{\theta}^* = \{\mathbf{S}^*, \boldsymbol{\beta}_1^*, \dots, \boldsymbol{\beta}_L^*\} = \arg\max_\theta \log p(y_{0:T}; \boldsymbol{\theta})$. In order to satisfy the constraints on $\mathbf{S}$ (i.e., that $\mathbf{1}\mathbf{S}\mathbf{1}^T = 1$ and each entry is non-negative, what we denote by $\mathbf{S} \geq \mathbf{0}$) we parametrize the matrix $\mathbf{S}$ by a real-valued matrix $\tilde{\mathbf{S}}$ also of size $L \times L$, using the following softmax function:

$$\mathbf{S}(s_1, s_2) = \frac{\exp(\tilde{\mathbf{S}}(s_1, s_2))}{\sum_{s_i, s_j} \exp(\tilde{\mathbf{S}}(s_i, s_j))}. \tag{8}$$

Thanks to that representation, we train the $\mathbf{S}$ and $\boldsymbol{\beta}_l$'s iteratively with gradient-based approach, by maximizing the log-likelihood $\log p(y_{0:T}; \boldsymbol{\theta})$.

3.2 Training FlowHMM $\mathcal{F}^{\mathbf{Q}}$ model

Considering $\mathcal{F}^{\mathbf{Q}}$ model we introduce the *co-occurrence matrix* that represents the joint distribution of two consecutive observations: $\mathbf{Q}(y_1, y_2) = p(Y_{k+1} = y_2, Y_k = y_1)$, what can be rewritten as:

$$\mathbf{Q}(y_1, y_2) = \sum_{s_i, s_j \in \mathcal{S}} p(y_1|s_i)\mathbf{S}(i,j)p(y_2|s_j). \tag{9}$$

Assuming the discrete set of observations, $\mathcal{V} = \{v_1, \dots, v_M\}$, the *co-occurrence* matrix $\mathbf{Q}$ can be expressed as $\mathbf{Q} = \mathbf{P}^T\mathbf{S}\mathbf{P}$, where $\mathbf{P}$ collects probabilities of all possible observations at each hidden state in a matrix $\mathbf{P}(s_i, v_j) = p(v_j|s_i)$. In this case there are M^2 possible observation pairs $(y_1, y_2) \in \mathcal{V} \times \mathcal{V}$. Note that matrices $\mathbf{Q}, \mathbf{S}, \mathbf{P}$ are of sizes $M \times M$, $L \times M$ and $L \times L$, respectively. Moreover, we have $\sum_{v_i, y_j \in \mathcal{V}} \mathbf{Q}(v_i, v_j) = \sum_{s_i, s_j} \mathbf{S}(s_i, s_j) = 1$ and $\sum_{v \in \mathcal{V}} \mathbf{P}(s_i, v) = 1$ for all $s_i \in \mathcal{S}$. In a matrix form, these can be written as $\mathbf{1}\mathbf{Q}\mathbf{1}^T = \mathbf{1}\mathbf{S}\mathbf{1}^T = 1$ and $\mathbf{P}\mathbf{1}^T = \mathbf{1}^T$ (note that $\mathbf{1}$ must be of an appropriate size). Given observations $y_{0:T}$ the matrix $\mathbf{Q}$ can be empirically estimated by:

$$\hat{\mathbf{Q}}(v_i, v_j) = \frac{1}{T} \sum_{k=0}^{T-1} \mathbb{I}(y_k = v_i)\mathbb{I}(y_{k+1} = v_j), \tag{10}$$

for all pairs $(v_i, v_j) \in \mathcal{V} \times \mathcal{V}$. The problem of training the HMM is to find such parameters (matrices $\mathbf{S}$ and $\mathbf{P}$) so that the co-occurrence matrix $\mathbf{Q}$ is close (in some sense) to the empirical co-occurrence matrix $\hat{\mathbf{Q}}$. We can formulate the problem as (for some distance dist)

$$\min_{\substack{\mathbf{P} \in \mathbb{R}^{N \times M}, \\ \mathbf{S} \in \mathbb{R}^{L \times L}}} \mathrm{dist}(\hat{\mathbf{Q}}, \mathbf{P}^T\mathbf{S}\mathbf{P}), \qquad \text{subject to} \quad \mathbf{1}\mathbf{S}\mathbf{1}^T = 1, \mathbf{P}\mathbf{1}^T = \mathbf{1}^T, \mathbf{P} \geq \mathbf{0}, \mathbf{S} \geq \mathbf{0}. \tag{11}$$

This problem formulation has a couple of advantages compared to standard likelihood-based optimization. First, the empirical co-occurrence matrix $\hat{\mathbf{Q}}$ is independent of the sequence length T. Second, the given objective can be easily optimized using gradient-based techniques. On the other hand, the constraints for matrices $\mathbf{P}$ and $\mathbf{S}$ should be satisfied. The set $\mathcal{V}$ is discrete, while we aim to design the model for continuous observations. In order to satisfy the constraints for matrix $\mathbf{S}$ we use the representation given by Eq. (8). In order to represent $\mathbf{P}$ matrix we use Flow-based emission probabilities.

Flow-based emission probabilities. With each hidden state l we associate the flow model described by density function $f_{\boldsymbol{\beta}_l}$, that can be calculated from Eq. (6), where $\boldsymbol{\beta}_l$ is a set of trainable parameters of the flow. We construct $\mathbf{P}$ based on these models (in such a case $\mathbf{P} \equiv \mathbf{P}_{\boldsymbol{\beta}}$). The i-th row of the matrix is a density $f_{\boldsymbol{\beta}_i}(\cdot)$ evaluated at $v_1, \ldots, v_M$ and normalized as follows:

$$\mathbf{P}_{\boldsymbol{\beta}}(s_i, v_j) = \frac{f_{\boldsymbol{\beta}_i}(v_j)}{\sum_{k=1}^{M} f_{\boldsymbol{\beta}_i}(v_k)}. \tag{12}$$

Similarly, the optimization problem (11) becomes:

$$\min_{\boldsymbol{\beta}, \mathbf{S} \in \mathbb{R}^{L \times L}} \operatorname{dist}(\hat{\mathbf{Q}}, \mathbf{P}_{\boldsymbol{\beta}}^T \mathbf{S} \mathbf{P}_{\boldsymbol{\beta}}). \tag{13}$$

As a distance function, we propose to use *Kullback–Leibler divergence*, i.e.

$$\min_{\boldsymbol{\beta}, \mathbf{S} \in \mathbb{R}^{L \times L}} \mathcal{L}, \quad \mathcal{L} = \sum_{i,j} \mathbf{Q}_{\boldsymbol{\beta}}(i,j) \log \frac{\mathbf{Q}_{\boldsymbol{\beta}}(i,j)}{\hat{\mathbf{Q}}(i,j)}, \tag{14}$$

where $\mathbf{Q}_{\boldsymbol{\beta}} = \mathbf{P}_{\boldsymbol{\beta}}^T \mathbf{S} \mathbf{P}_{\boldsymbol{\beta}}$. We postulate to apply divergence instead to L^2 distance used e.g. in [7]., because it is more natural measure for comparing two distributions. We observed during empirical evaluation, that using this divergence gives better stability and convergence of the training process.

The final training procedure of FlowHMM $\mathcal{F}^{\mathbf{Q}}$ model is as follows. Let $y_{0:T}^{\text{train}}$ be the training set and let $y_{0:T'}^{\text{test}}$ be the test set. For the fixed M we construct the grid $\Gamma = (\gamma_1, \ldots, \gamma_M)$ using one of the approaches described in A.1. Next, we create the discretized training data $y_{0:T}^{\text{train},\Gamma}$, where $y_i^{\text{train},\Gamma} = \arg\min_{\gamma \in \Gamma} ||\gamma - y_i^{\text{train}}||^2$. The values of the grid represent further the set $\mathcal{V}$, i.e., $\mathcal{V} = \Gamma$. Next, we calculate the empirical co-occurrence matrix $\hat{\mathbf{Q}}$:

$$\hat{\mathbf{Q}}(v_i, v_j) = \frac{1}{T} \sum_{k=0}^{T-1} \mathbb{I}(y_k^{\text{train},\Gamma} = v_i) \mathbb{I}(y_{k+1}^{\text{train},\Gamma} = v_j), \tag{15}$$

and matrices $\mathbf{S}$ and $\mathbf{P}_{\boldsymbol{\beta}}$ (using Eqs. (8), (12)) that are further used to calculate the loss given by Eq. (14). Practically, we add Gaussian perturbations to the grid values, while calculating the matrix $\mathbf{P}$:

$$\mathbf{P}_{\boldsymbol{\beta}}(s_i, v_j) = \frac{f_{\boldsymbol{\beta}_i}(v_j + \epsilon)}{\sum_{k=1}^{M} f_{\boldsymbol{\beta}_i}(v_k + \epsilon)}, \tag{16}$$

where ϵ is a random sample from $\mathcal{N}(0, \sigma_{noise}^2)$, and σ_{noise}^2 is a hyperparameter of the method. These perturbations prevent from overfitting of the flow-based components caused by observing the same grid while training. Next, all of the parameters of FlowHMM are updated with gradient based approach. The procedure is repeated until convergence. The procedures of training is presented in Algorithm 1.

3.3 Inference with FlowHMM

As we postulated before, our model is designed for continuous problems, due ti the fact, that both of the training techniques return estimated $\mathbf{S}$, and the parameters, that represent emission distributions for each of the states. Thus, during the inference stage, we simply calculate $\mathbf{A}$ and $\boldsymbol{\pi}$ from $\mathbf{S}$ and apply forward procedure on the test data $y_{0:T'}^{\text{test}}$ to calculate $p(y_{0:T'}^{\text{test}}; \boldsymbol{\theta})$. We can also determine the hidden state values using *Viterbi* procedure. Concluding, FlowHMM can be used in the same applications as standard continuous HMM models, but with no restrictions to emission distributions.

3.4 FlowHMM for multidimensional observations

Our approach can be easily extended to multidimensional observations, i.e., to the case where $y_{0:T}^{\text{train}}$ are from $\mathbb{R}^d$. For such a case, the $\mathcal{F}^{\text{EM}}$ model is straightforward, let us focus on a description of $\mathcal{F}^{\mathbf{Q}}$. On one hand the extension is straightforward: we construct some grid $\Gamma = (\gamma_1, \ldots, \gamma_M)$ of d-dimensional points, we create discretized training set $y_{0:T}^{\text{train},\Gamma}$ and the empirical co-occurrence matrix $\hat{\mathbf{Q}}$ from (15), afterwards we compute $\mathbf{P}_{\boldsymbol{\beta}}$ from Eq. (16) and the gradient $\nabla_{\boldsymbol{\beta}_l}\mathcal{L}$ of $\mathcal{L}$ given in (14). In other words we proceed with an Algorithm 1. Note that CNFs are well-suited for multivariate observations. On the other hand, there is a challenging aspect of training $\mathcal{F}^{\mathbf{Q}}$ model for multidimensional data: an effective discretization technique. We elaborate on that in A.1 proposing a new grid search method.

Algorithm 1 Training using $\mathcal{F}^{\mathbf{Q}}$ technique

Require: $\hat{\mathbf{Q}}$: — empirical co-occurrence matrix from Eq. (10).
Parameters: $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_L\}$ — flow parameters, $\tilde{\mathbf{S}}$ parameters representing un-normalised co-occurrence matrix.
Hyperparameters: L - number of hidden states, α- step size, noise variance σ^2_{noise}.

```
function TRAIN(Q̂, L, α)
    Initialize β, and S̃.
    while not convergent do
        Calculate S from Eq. (8). and P_β from Eq. (16).
        Calculate loss function L from Eq. (14) using Q̂.
        S̃ ← S̃ − α∇_S̃ L
        for each l ∈ {1, ..., L} do
            β_l ← β_l − α∇_{β_l} L
        end for
    end while
    return β_1, ..., β_L, S̃
end function
```

4 Related work

One of the most popular way to train HMM models is the Baum-Welch algorithm (a name for EM applied to HMMs). However, one of (several) drawbacks of using Baum-Welch algorithm for training is that it is prohibitively slow for long sequences. The complexity of the forward-backward algorithm (which must be run for each epoch) is $\mathcal{O}(N^2T)$, where N is the number of hidden states and T is the length of the observation sequence. In order to eliminate this issue, the authors [6] propose to use pairwise co-occurrence probabilities (also higher order statistics are also discussed there) and *prove* that it is possible to recover the structure of an HMM based only on co-occurrences. An interesting extension (both, of pairs and triplets) was considered in [18]. Authors used there non-consecutive tuples, which outperforms consecutive tuples in some cases.

Our approach is, in a sense, close to the technique used in [7], with however significant differences: authors use alternating least square methods for optimization (in our case, all the matrices are real-valued and, softmax is applied whenever needed) and for the continuous case, they assume that the observation densities are a mixture of the predefined number of kernels.The "softmax trick" is also used in [9] (their model learns so-called *dense* representations of hidden states and observation probabilities – it is designed only for discrete HMMs).

As already mentioned, typically continuous HMMs assume that observations were sampled from a Gaussian distribution or a mixture of such distributions. For a non-Gaussian distribution, one usually assumes some parametric family of distributions. In [11] authors propose a *decoupling* method to learning the parametric HMMs which are stationary. Instead of estimating the parameters of hidden states and observations jointly, they learn the parameters of observation densities using some parametric mixture learner, and then hidden states by solving some convex quadratic programming. In [13] a method is proposed for a case where at least one state-dependent distribution is modeled with some nonparametric technique (*e.g.*, maximum likelihood estimation under shape constraints), some broader review of methods is presented in [19]. In [14] authors use a Gaussian copula to model the dependence structure. A semiparametric data transformation is also proposed to ensure one may indeed use such copulas (the final observation distribution is a finite mixture of the copula models). In

[20] authors present an efficient learning for parametric continuous HMMs sampled at finite irregular time instants (they incorporate some ideas from the theory of continuous-time Markov chains).

In [21], the authors consider the extension of a Kalman filter (which you can think of as a version of a HMM with continuous both, observations and hidden states), where the *pseudo-observations* (as authors call it) are transformed through a normalizing flow producing the actual observation.

The most similar model to our approach was introduced in [22], where the authors employ normalizing flows in observation densities and apply it to classification problems. However, they utilize the flow with coupling layers that cannot be directly applied to one-dimensional data. We use CNFs that can be easily used for any type of data, and are characterized by better quality for low-dimensional examples compared to coupling-based flows. Moreover, their approach relies on EM training, which makes it impractical for longer sequences of data. We propose an alternative form of training the model and show advantages empirically compared to EM, for one-dimensional examples. Finally, the authors of [22] apply their model only on sequence classification as *black box*. At the same time, our paper investigates the proprieties of specific examples in order to discover the capabilities of distribution adjustment and discover the hidden states of data unobserved during training.

5 Experimental results

Evaluation. Here we consider our FlowHMM models $\mathcal{F}^{\mathrm{EM}}$ and $\mathcal{F}^{\mathbf{Q}}$ with training procedures described in Sections 3.1 and 3.2 respectively.

We compare our results with classic HMM models utilizing Gaussian mixtures observation densities with $K = 1, 10, 20$ components; we denote them by $\mathcal{G}^{(K)}$ and for simplicity we write $\mathcal{G} \equiv \mathcal{G}^{(1)}$. We used `hmmlearn` library[1] for the computations of all $\mathcal{G}^{(K)}$ models and provide the source code for FlowHMM models[2]. We evaluate the quality of the proposed methods computing *Normalized log-likelihood* (`normLL`) of unseen observations, *Total variation distance* (d_{tv}) between learned and true emission probabilities, and accuracy of predicted hidden states (`accuracy`). We define all the metrics and motivate their choice in A.2, the experimental settings are described in details in A.3.

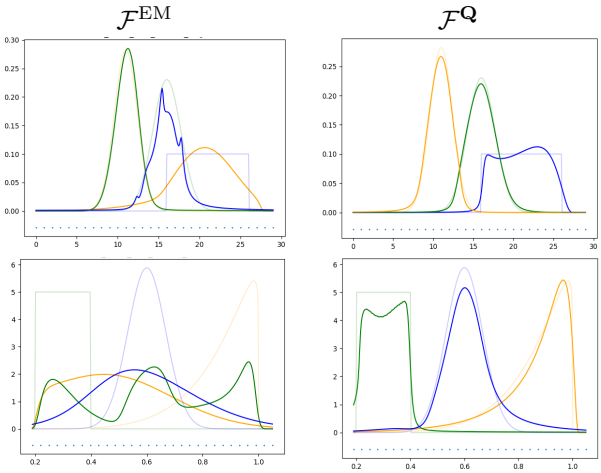

Figure 2: Distributions learned by $\mathcal{F}^{\mathrm{EM}}$ and $\mathcal{F}^{\mathbf{Q}}$ for $T = 10^4$ observations for Example 1 (top row) and Example 2a (bottom row). Original distributions shaded.

1D Synthetic sequences. We consider HMMs with $L = 3$ hidden states and experiment with transition matrices $\mathbf{A}_1$ (used in [7, Eq. (11)]) or $\mathbf{A}_2$, where

$$\mathbf{A}_1 = \begin{pmatrix} 0.0 & 0.9 & 0.1 \\ 0.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & 0.0 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 0.4 & 0.2 & 0.4 \\ 0.25 & 0.5 & 0.25 \\ 0.4 & 0.2 & 0.4 \end{pmatrix}. \tag{17}$$

We consider 2 examples. **Example 1** - transition matrix $\mathbf{A}_1$ and emission probabilities: two Gaussian and one Uniform. **Example 2a** - transition matrix $\mathbf{A}_2$, emission probabilities: one Gaussian, one Uniform, and Beta distribution. In addition, we consider **Example 2b**, where we aim at adjusting the models assuming only two hidden states. The examples are described in details in Appendix A.4.

For each example we sample train and test observations of the same length $T \in \{10^3, 10^4, 10^5\}$. We train $\mathcal{F}^{\mathrm{EM}}$ and $\mathcal{F}^{\mathbf{Q}}$ models using 500 (Example 1) and 1000 (Examples 2a and 2b) epochs. For $\mathcal{F}^{\mathbf{Q}}$

[1] `https://github.com/hmmlearn`
[2] `https://github.com/anonimized_link`

	T	Example 1					Example 2a					Example 2b				
		$\mathcal{G}$	$\mathcal{G}^{(10)}$	$\mathcal{G}^{(20)}$	$\mathcal{F}^{\mathrm{EM}}$	$\mathcal{F}^{\mathbf{Q}}$	$\mathcal{G}$	$\mathcal{G}^{(10)}$	$\mathcal{G}^{(20)}$	$\mathcal{F}^{\mathrm{EM}}$	$\mathcal{F}^{\mathbf{Q}}$	$\mathcal{G}$	$\mathcal{G}^{(10)}$	$\mathcal{G}^{(20)}$	$\mathcal{F}^{\mathrm{EM}}$	$\mathcal{F}^{\mathbf{Q}}$
normLL	10^3	-2.21	**-2.21**	-2.21	-2.46	-3.06	0.17	0.17	0.17	**0.33**	0.11	0.09	0.17	0.17	**0.32**	0.09
	10^4	-2.19	-2.19	-2.19	-2.28	**-2.18**	0.18	0.17	0.17	0.33	**0.34**	0.09	0.18	0.18	**0.32**	0.30
	10^5	-2.19	-2.19	-2.19	-2.28	**-2.16**	0.18	0.17	0.17	**0.34**	0.33	0.09	0.18	0.18	**0.32**	0.18
d_{tv}	10^3	**0.09**	0.09	0.09	0.13	0.39	**0.28**	0.31	0.31	0.38	0.43	–	–	–	–	
	10^4	0.07	0.07	0.07	0.11	**0.05**	0.29	0.32	0.32	0.30	**0.15**	–	–	–	–	–
	10^5	0.07	0.07	0.07	0.11	**0.04**	0.29	0.32	0.32	0.28	**0.15**	–	–	–	–	–

Table 1: normLL and d_{tv} metrics for synthetics Examples 1, 2a-b. Best results **bolded**.

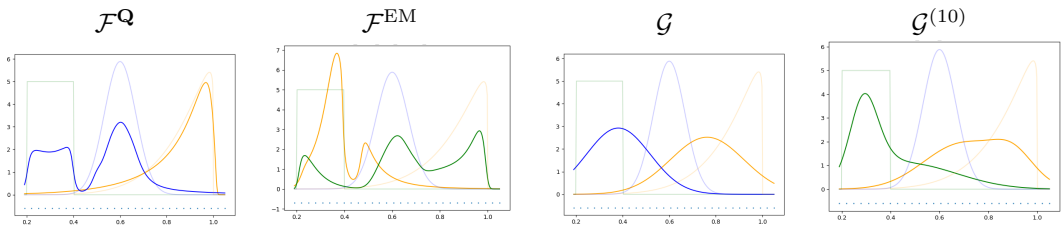

Figure 3: Results for Example 2b (three original distributions shaded) for $T = 10^4$ observations.

we used a grid of size $M = 30$. For each case (*i.e.*, fixed example and T) we perform 10 simulations and report means of normLLs and total variation distances in Table 1. For observations of length $T \geq 10^4$ flow models clearly outperform Gaussian ones in all the examples. Note that in these examples we had a grid of size $M = 30$, thus we estimated the matrix $\mathbf{Q}$ with 900 entries (number of observation pairs). It is intuitively clear that $T = 10^3$ observations is not enough then for $\mathcal{F}^{\mathbf{Q}}$. In this case $\mathcal{F}^{\mathrm{EM}}$ outperforms $\mathcal{F}^{\mathbf{Q}}$ in terms of normLLs; models are comparable for larger T. For $T \geq 10^4$ the $\mathcal{F}^{\mathbf{Q}}$ model "recovers" original distributions much more accurately, the small values of d_{tv} from Table 1 are confirmed in Fig. 2, where distributions learned by our flow models trained with $T = 10^4$ observations are depicted (from single simulations). In A.4 in Fig. 5, the trained distributions for Examples 1, 2a-b trained with $T = 10^3$ observations are depicted.

We encourage the reader to confront Fig. 2 for $\mathcal{F}^{\mathbf{Q}}$ and $T = 10^4$ for Example 1 with [7, Fig. 4]. In Fig. 3 we depict the distributions learned models with only $L = 2$ hidden states for $T = 10^4$ observations (recall observations were sampled from a model with 3 different distributions). Since flow models are capable of fitting multimodal distributions, they outperform Gaussian ones. Moreover, the co-occurrence matrix-based model $\mathcal{F}^{\mathbf{Q}}$ outperforms the EM-based model $\mathcal{F}^{\mathrm{EM}}$.

Real datasets. **Example 3.** S&P 500 and Dow Jones are popular measures of market performance. We retrieved data [23] for a period of 9/1977-8/2017 that consist of 2082 points. We trained models on $T = 1000$ observations (and computed normLLs for the remaining data).

Example 4. We used mean air pressure from [24] dataset collected between 02/15/18 and 02/28/19. There were 54 531 observations, the models were trained on the first $T = 40\,000$ ones and tested them on the remaining observations.

L	$\mathcal{G}$	$\mathcal{G}^{(10)}$	$\mathcal{G}^{(20)}$	$\mathcal{F}^{\mathrm{EM}}$	$\mathcal{F}^{\mathbf{Q}}$
	S&P 500				
2	-10.476	-10.157	-10.686	-6.850	**-5.479**
3	-10.800	-10.194	-10.806	-6.837	**-4.966**
4	-12.462	-9.7641	-11.130	-6.908	**-5.043**
	Dow Jones				
2	-12.317	-14.154	-13.138	-8.730	**-7.120**
3	-12.645	-13.747	-15.007	-8.897	**-7.426**
4	-13.956	-13.313	-14.669	-8.902	**-7.514**
	Air pressure				
2	-0.704	0.169	0.598	-0.984	**0.823**
3	-0.743	0.646	0.603	0.540	**0.861**
4	-0.758	0.710	0.692	0.661	**0.783**

Table 2: normLL for Examples 3 and 4.

In both, Example 3 and Example 4, we applied a standard difference transform (to remove a trend), *i.e.*, we considered time series $y'_k = y_k - y_{k-1}$. We trained the models with $L \in \{2, 3, 4\}$ hidden states, models $\mathcal{F}^{\mathrm{EM}}$ and $\mathcal{F}^{\mathbf{Q}}$ were trained for 500 epochs, for the latter we used a uniform grid of size $M = 30$. The values of normLL are reported in Table 2. As can be seen, in all cases $\mathcal{F}^{\mathbf{Q}}$ yields the best results and for Example 3 model $\mathcal{F}^{\mathrm{EM}}$ is second best. Note also that although a number of observations for Example 3 is not large (we used only $T = 1000$ observations, while for $m = 30$ we have 900 observation pairs) $\mathcal{F}^{\mathbf{Q}}$ outperforms $\mathcal{F}^{\mathrm{EM}}$. Fig. 6 in A.4 shows that the learned observation densities for $L \in \{2, 3, 4\}$ are non-Gaussian.

	T	Example 5a					Example 5b				
		$\mathcal{G}$	$\mathcal{G}^{(10)}$	$\mathcal{G}^{(20)}$	$\mathcal{F}^{\mathrm{EM}}$	$\mathcal{F}^{\mathbf{Q}}$	$\mathcal{G}$	$\mathcal{G}^{(10)}$	$\mathcal{G}^{(20)}$	$\mathcal{F}^{\mathrm{EM}}$	$\mathcal{F}^{\mathbf{Q}}$
normLL	10^3	-1.6572	-0.6521	-0.6637	**-0.6003**	-2.7090	-1.6560	-1.4459	-1.4486	**-1.3309**	-2.6789
	10^4	-1.6083	-0.6176	-0.6081	**-0.4822**	-0.6711	-1.6574	-1.4392	-1.4356	**-1.2286**	-1.4185
	10^5	-1.6117	-0.6001	-0.5988	**-0.4898**	-3.4500	-1.6510	-1.4160	-1.4178	**-1.2538**	-3.0717
accuracy	10^3	0.6010	**1.0000**	1.0000	1.0000	0.5600	0.5450	0.6560	0.6480	**0.7390**	0.5360
	10^4	0.6325	0.9995	**0.9998**	0.9998	0.9956	0.5680	0.6677	0.6723	0.7088	**0.7414**
	10^5	0.6274	**1.0000**	1.0000	0.9998	0.6097	0.5726	0.6775	0.6727	**0.7448**	0.5615
		Example 6a					Example 6b				
normLL	10^3	0.8652	0.9443	0.9472	**1.0531**	-2.1823	0.2270	0.3128	0.2915	**0.3636**	-2.1891
	10^4	0.8755	1.0009	1.0014	**1.0807**	0.9189	0.2378	0.3161	0.3173	**0.3551**	0.2120
	10^5	0.8843	0.9978	0.9992	**1.0688**	-1.8349	0.2186	0.3076	0.3103	**0.3496**	-0.3917
accuracy	10^3	**0.9960**	0.9960	0.9960	0.9940	0.6700	0.8270	0.6580	0.6590	**0.8550**	0.6490
	10^4	0.9971	0.9987	0.9987	**0.9989**	0.9934	0.8173	0.6533	0.6558	0.5419	**0.8428**
	10^5	0.9964	**0.9994**	0.9994	0.9987	0.9219	0.8077	0.6503	0.6443	**0.8796**	0.7376

Table 3: `normLL` and `accuracy` for Examples 5 and 6. Best results **bolded**.

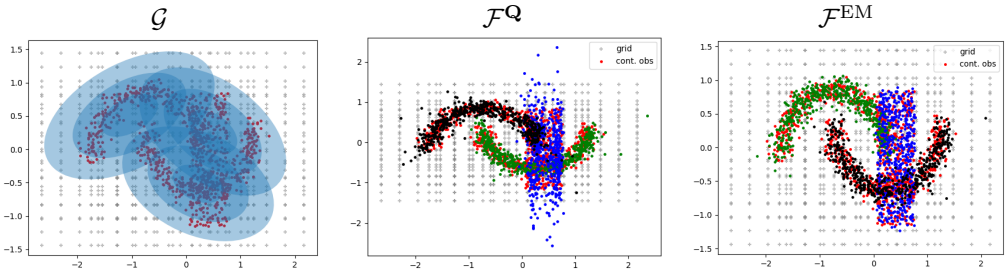

Figure 4: Trained distributions for Example 5a for $T = 10^4$: original observations denoted as red dots •, grid Γ presented as gray crosses $+$; samples from trained distributions $f_{\boldsymbol{\beta}_l}, l = 1, 2, 3$ depicted as black •, green • and blue • dots (colors correspond to state l).

2D synthetic sequences. We consider HMM models with $L = 3$ states. We consider 2 examples: **Example 5** with emission probabilities: two "`Moons`" and one Uniform and **Example 6** with emission probabilities: one bivariate Gaussian, one Uniform and one related to geometric Brownian motion. We used non-uniform grids of size $M = 900$. Details provided in A.5. In both examples we consider the transition matrix $\mathbf{A}_1$ (variant **(a)**) and $\mathbf{A}_2$ (variant **(b)**), defined in (17).

In Table 3 values of `normLL` and `accuracy` are reported, whereas in Fig. 4 the trained distributions for Example 5a for $T = 10^4$ are depicted. Similar plots for Example 6a are presented in A.5 in Fig. 7. One can observe that in all the considered cases and variants, model $\mathcal{F}^{\mathrm{EM}}$ gives the best `normLL` values. When it comes to the other metric, i.e. the `accuracy`, all the classic Gaussian mixture models, as well as $\mathcal{F}^{\mathrm{EM}}$, yield almost the same accuracy if transition matrix $\mathbf{A}_1$ was used (variant **(a)**). However, for the transition matrix $\mathbf{A}_2$ (variant **(b)**), the models $\mathcal{F}^{\mathrm{EM}}$ and $\mathcal{F}^{\mathbf{Q}}$ outperform the classic models with significant margin (note that $\mathbf{A}_2$ yields a "more complex" model compared to $\mathbf{A}_2$).

6 Conclusions and limitations

In this work, we proposed a continuous hidden Markov model that leverages deep flow-based network architectures to model complex, non-Gaussian distributions. Although our approach outperforms several baselines and competing approaches, it relies on the co-occurrence matrix that can be only computed with the assumption that the Markov chain on a set of hidden states is stationary - a limitation not present in the EM-based approaches. Addressing this constraint, *e.g.* by re-computing temporally stationary co-occurrence matrix during training, is part of our future work. So is in-depth validating our model on multivariate time series or distributions, where our flexible approach can potentially offer more benefits over the existing works. While we do not identify any straightforward negative societal implications of our work, we acknowledge that its application to large data corpora can lead to significant energy consumption, despite our main contributions being focused on increased efficiency.

References

[1] Rogemar S Mamon and Robert James Elliott. *Hidden Markov models in finance*, volume 4. Springer, 2007.

[2] Mark Gales and Steve Young. *The application of hidden Markov models in speech recognition*. Now Publishers Inc, 2008.

[3] Mathukumalli Vidyasagar. *Hidden Markov Processes: Theory and Applications to Biology*. Princeton University Press, 2014.

[4] Pierre Ailliot, Craig Thompson, and Peter Thomson. Space–time modelling of precipitation by using a hidden markov model and censored gaussian distributions. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 58(3):405–426, 2009.

[5] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

[6] Kejun Huang, Xiao Fu, and Nicholas Sidiropoulos. Learning hidden markov models from pairwise co-occurrences with application to topic modeling. In *International Conference on Machine Learning*, pages 2068–2077. PMLR, 2018.

[7] Balaji Lakshminarayanan and Raviv Raich. Non-negative matrix factorization for parameter estimation in hidden markov models. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*, pages 89–94, 2010.

[8] Daniel Hsu, Sham M Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012.

[9] Joachim Sicking, Maximilian Pintz, Maram Akila, and Tim Wirtz. DenseHMM: Learning hidden markov models by learning dense representations. *CoRR*, abs/2012.09783, 2020.

[10] Robert Mattila, Cristian R Rojas, Vikram Krishnamurthy, and Bo Wahlberg. Identification of hidden markov models using spectral learning with likelihood maximization. In *2017 IEEE 56th annual conference on decision and control (CDC)*, pages 5859–5864. IEEE, 2017.

[11] Aryeh Kontorovich, Boaz Nadler, and Roi Weiss. On learning parametric-output hmms. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–702–III–710. JMLR.org, 2013.

[12] Oscar Darwin and Stefan Kiefer. Equivalence of hidden markov models with continuous observations. *arXiv preprint arXiv:2009.12978*, 2020.

[13] Jörn Dannemann. Semiparametric hidden markov models. *Journal of computational and graphical statistics*, 21(3):677–692, 2012.

[14] Hongyang Yu. A novel semiparametric hidden markov model for process failure mode identification. *IEEE Transactions on Automation Science and Engineering*, 15(2):506–518, 2018.

[15] Lu Xia, Chia-Chih Chen, and J. K. Aggarwal. View invariant human action recognition using histograms of 3d joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–27, 2012.

[16] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*. PMLR, 2015.

[17] Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv*, 2018.

[18] Robert Mattila, Cristian Rojas, Eric Moulines, Vikram Krishnamurthy, and Bo Wahlberg. Fast and consistent learning of hidden Markov models by incorporating non-consecutive correlations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6785–6796. PMLR, 13–18 Jul 2020.

[19] Jörn Dannemann, Hajo Holzmann, and Anna Leister. Semiparametric hidden markov models: identifiability and estimation. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):418–425, 2014.

[20] Yu-Ying Liu, Shuang Li, Fuxin Li, Le Song, and James M Rehg. Efficient learning of continuous-time hidden markov models for disease progression. *Advances in neural information processing systems*, 28:3599, 2015.

[21] Emmanuel de Bézenac, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski. Normalizing kalman filters for multivariate time series analysis. In *NeurIPS*, 2020.

[22] Anubhab Ghosh, Antoine Honoré, Dong Liu, Gustav Eje Henter, and Saikat Chatterjee. Robust classification using hidden markov models and mixtures of normalizing flows. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2020.

[23] `data.world`. DowJones and S&P500 datasets, accessed 2022. `https://data.world/chasewillden/stock-market-from-a-high-level/workspace/`.

[24] `AmeriGEOSS`. Wind-measurementds in Papua New Guinea, 2019. data retrieved from `https://data.amerigeoss.org/dataset/857a9652-1a8f-4098-9f51-433a81583387/resource/a8db6cec-8faf-4c8e-aee2-4fb45d1e6f14`.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please see Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see Appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] The references [23] and [24]
   (b) Did you mention the license of the assets? [No] The used datasets are common and publicly available.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]