An Interpretable Deep Classifier for Counterfactual Generation

Wei Zhang
Columbia University
New York, NY, USA

Brian Barr
Capital One
New York, NY, USA

John Paisley
Columbia University
New York, NY, USA

ABSTRACT

Counterfactual explanation has been the core of *interpretable machine learning*, which requires a trained model to be able to not only infer but also justify its inference. This problem is crucial in many fields, such as fintech and the healthcare industry, where accurate decisions and their justifications are equally important. Many studies have leveraged the power of *deep generative models* for counterfactual generation. However, most focus on vision data and leave the latent space unsupervised. In this paper, we propose a new and general framework that uses a supervised extension to the Variational Auto-Encoder (VAE) with Normalizing Flow (NF) for simultaneous classification and counterfactual generation. We show experiments on two tabular financial data-sets, Lending Club (LCD) and Give Me Some Credit (GMC), which show that the model can achieve a state-of-art level prediction accuracy while also producing meaningful counterfactual examples to interpret and justify the classifier's decision.

KEYWORDS

supervised autoencoders, intepretable deep learning, explainability, counterfactual methods

ACM Reference Format:
Wei Zhang, Brian Barr, and John Paisley. 2022. An Interpretable Deep Classifier for Counterfactual Generation. In *3rd ACM International Conference on AI in Finance (ICAIF '22), November 2–4, 2022, New York, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3533271.3561722

1 INTRODUCTION

As deep learning has been revolutionizing many fields in the past decade, such as computer vision, robotics, and natural language processing, data scientists outside of machine learning, such as the medical and financial industries, have also looked to leverage deep learning models. However, in such fields where model explainability is required, deep learning's black-box nature still remains a challenge that has limited its adoption.

Explainable machine learning can generally be categorized into two approaches: model explanations and outcome explanations [16]. In this paper, we focus on counterfactual explanations, an instance of the outcome explanation approach. Counterfactual explanations are usually applied in supervised learning settings. While a deep classifier may do well at making correct predictions, counterfactual explainability ultimately aims to answer the question: "what properties of the input data sample were important in its classification," and similarly, "what changes can/should be made to the input to reverse its outcome?" This is where neural networks often come up short. For example, it has been shown by Su et al. [43] that small and meaningless random perturbations to an input are often sufficient to deceive a well-performing classifier.

Given the property of neural networks to learn a complex and highly-nonlinear decision boundary, counterfactual approaches based solely on this decision boundary are often inappropriate for input-specific counterfactual generation. Our goal in this paper is to generate counterfactuals that minimally alter the input to change the decision, yet are still interpretable and personalized. To this end, we propose an algorithm that combines the supervised power of the neural network with the latent space interpretability of the unsupervised autoencoder.

The variational autoencoder (VAE) [24] is a popular and powerful deep generative model that formulates the autoencoder learning problem in the variational setting. It has been used as a framework for anomaly detection [2], recommendation systems [30], temporal tracking [47], image captioning [38] and arithmetic expressions [25]. One limitation of the VAE is that the latent space is only modeled with an isotropic multivariate Gaussian, leading to developments by Rezende and Mohamed [39] and Kingma et al. [22] that embed planar flow (PF) and inverse autoregressive flow (IAF) in the latent space for improved modeling.

Indeed, given its generative power and scalability, the VAE has become one of the natural options for counterfactual generation, leading to several developments of its basic framework [3, 10, 18, 37]. However, these methods all use an unsupervised VAE towards this end. In one related and promising development, Le et al. [28] construct an autoencoder with a linear classifier in its latent space and argue that adding a decoder as an unsupervised auxiliary task will help learn a more distinguishable latent space. Building on this work, we propose a probit regression model to supervise and regularise the latent space of the VAE for *simultaneous classification and counterfactual generation*. We show that the performance of our model on the classification task outperforms linear models – still a popular and standard approach in fintech – and is comparable to a neural network of the same encoder structure, with the added feature of counterfactual explainability.

The paper is organized as follows. In Section 2 we briefly review related works. In Section 3 we present our modeling approach. In Section 4, we discuss a variational inference method with probit/logistic model and the simultaneous approach to classification and counterfactual analysis using the learned model. Section 5 shows experiments on two financial data sets. We conclude in Section 6.

ICAIF '22, November 2–4, 2022, New York, NY, USA

ACM ISBN 978-1-4503-9376-8/22/11…$15.00
https://doi.org/10.1145/3533271.3561722

2 RELATED WORK

Our work is broadly related to the fields of generative models and counterfactual model explanations. More exhaustive literature surveys can by found in Karimi et al. [19] and Verma et al. [45].

2.1 Counterfactual explanation

While counterfactual explanations have a long history in fields such as the social sciences, Wachter et al. [46] first mathematically casts the counterfactual explanation into an optimization problem for machine learning. The objective function aims to solve $\arg\min_{x'} d(x, x')$ subject to $f(x') = y'$ where $f(\cdot)$ is a trained classifier, x is input, and x' and y' are the counterfactual sample and its label, respectively. A potential drawback is that this algorithm tends to find counterfactuals close to the decision boundary, which are likely to be unrealistic and out of the sample density. Many other works construct their own objectives with different distance functions and iteratively search for counterfactuals [6, 8, 11, 13, 15, 32, 44]. While such algorithms have proven powerful for generating counterfactuals, they tend to rely on computationally intensive search algorithms to generate a counterfactual for each input.

Counterfactual samples can also be generated by inverse classification [1], which finds a counterfactual by perturbing the input such that it will be labeled as the counter-class. Lash et al. [26], Laugel et al. [27] generate counterfactual samples with this inverse classification approach. Sharma et al. [42] search for changes in the input space such that the classification result is switched to the counter class. However, the generated sample close to the decision boundary might still be out of the counter-class's data distribution, leading to an unrealistic counterfactual.

2.2 Generative models

Our work also draws on the generative modeling framework. In general, such models fall into three branches: Generative Adversarial Networks (GAN) [12], Variational Autoencoders (VAE) [24], and Normalizing Flows (NF) [39].

Several works take advantage the autoencoding framework to generate counterfactual examples [3, 7, 10, 18, 31, 37, 44]. In this paper, we will compare with the following: Pawelczyk et al. [37] trains a Conditional VAE model called CCHVAE and uses a different trained classifier to classify the output from the decoder. A nearest neighbor search algorithm is then implemented in the learned latent space. Joshi et al. [18] proposes REVISE, which implements a VAE model with a gradient step in the latent space. Both tend to generate counterfactuals that are close to the decision boundary and out of the target distribution's density. Downs et al. [10] designed CRUDS, which builds a conditional subspace with VAE and only changes relevant features for generating counterfactuals. Antorán et al. [3] propose CLUE, which takes the uncertainty of the classifier's prediction into account. CRUDS and CLUE achieve a better distribution matching but often produce invariant counterfactuals. All approaches leave the latent space of the autoencoder unsupervised and rely on input-specific search algorithms.

GANs have also received attention recently: Samangouei et al. [40] use a GAN designed for image data. In addition to the typical GAN loss, they add a classification loss, a prior loss and a reconstruction loss. Nemirovsky et al. [34] extend this with a residual connection in GAN architecture which can take an input and directly generate its counterfactual image. Normalized flows have also been proposed for density estimation [9, 21, 23, 35] to improve learning; our approach also leverages the modeling power of NF in the context of variational inference. Recently, Chang et al. [4], Cohen et al. [5], Sauer and Geiger [41] aim to explain a deep classifier using the counterfactual approach. Despite these useful methods, they primarily focus on vision tasks and the models are less appropriate for tabular data such as is used in finance, which is our main interest.

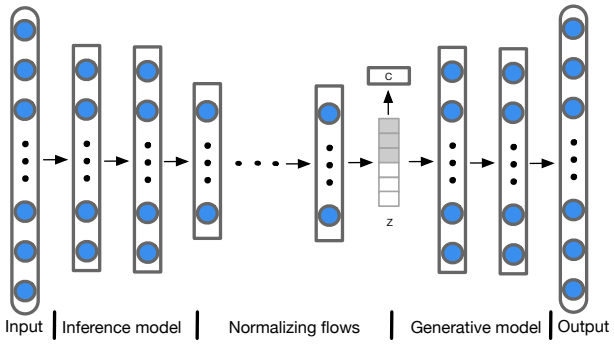

Figure 1: An illustration of our pipeline. Left part: An encoder maps the input to the latent "z-space." Middle (gray) part: A linear classifier makes decisions, in our case for credit lending, based on data in this mapped space. Right part: A decoder reconstructs the input based on its location in z-space. All parts are learned simultaneously by optimizing a single objective function. Using the linear classifier in the latent space to move a z-mapped point into the counter-class region according to the shortest distance (a closed form calculation defined by the linear classifier) the decoder can generate its counterfactual based on its new location. We also observe that the combined encoder/linear classifier are identical in structure to a deep neural network, resulting in improved classification performance over a linear classifier in the original data space.

3 THE MODEL

We build upon the VAE model with normalizing flows using probit regression for classification and for better interpretability of its decision through counterfactual analysis. In the next two sections, we present a technical overview of our approach. A graphical layout and high-level explanation is given in Figure 1.

3.1 Basic setup: VAE and NF

We are given a set of n labeled training samples $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \in \mathbb{R}^D$ is the feature vector for the i-th data point and $y_i \in \{-1, 1\}$ is its class label, which is a binary set in this scenario. We also denote $z_i^k \in \mathbb{R}^d$ as the latent vector from k-th coupling block for the i-th data point. Usually $D \gg d$. A VAE learns an encoder parameterized by θ such that $f_e^\theta(\cdot) : \mathbb{R}^D \to \mathbb{R}^d$, which maps a data feature vector x_i to a latent vector in $\mathbb{R}^d$. It also learns a decoder parameterized by ϕ such that $f_d^\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^D$, which

Algorithm 1 Supervised VAE with probit model

Input: $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, η
Set t=0
Initialize $\theta_t, \phi_t, \Sigma'_t, \mu'_t, \mu'_{0,t}$ and $\mu_{\lambda_i t}, \forall i$ randomly
while not converged **do**
  Sample $S_t \sim \mathcal{D}$
  Update μ_{λ_i} and $\mu'_t, \Sigma'_t, \mu'_{0,t}$
  Update $\theta_t \leftarrow \theta_{t-1} + \eta \nabla_\theta \mathcal{L}(\theta, \phi),\ \phi_t \leftarrow \phi_{t-1} + \eta \nabla_\phi \mathcal{L}(\theta, \phi)$
end while
return $\theta_t, \phi_t, \mu'_t, \Sigma'_t, \mu'_{0,t}$

maps a latent vector back to a reconstructed sample in feature space. The goal is for $f_d^\phi(f_e^\theta(x)) \approx x$. Both $f_e^\theta(\cdot)$ and $f_d^\phi(\cdot)$ can be parameterized as neural networks with nonlinear activation functions in hidden layers. As proposed by [24], we select the Gaussian distribution for both the prior distribution $p(z_i^K)$ and the likelihood distribution $p(x_i|z_i^K)$ where z_i^K is the output from K-th normalizing flow. With normalizing flows, we can transform the posterior isotropic Gaussian $q(z_i^0|x_i)$ through a flow of bijective functions to obtain a more written latent representations. The hierarchical structure of our VAE model can be introduced as

$$z_i^0 \sim \mathcal{N}(0, I), \quad x_i|z_i^K \sim \mathcal{N}(f_d^\phi(z_i^K), \beta I),$$
$$z_i^0|x_i \sim \mathcal{N}\big(m(f_e^\theta(x_i)), \mathrm{diag}(v(f_e^\theta(x_i)))\big) \tag{1}$$

where $z_i^K = f(z_i^0)$, $f = f_1 \circ f_2 \cdots \circ f_K$, $\mathcal{N}$ denotes the Gaussian distribution, β is a hyper-parameter that controls the variance of the Gaussian likelihood, $m(\cdot)$ and $v(\cdot)$ are linear transformations and each f_k defines a bijective transformation. We also implement the reparameterization trick to allow gradients to back-propagate smoothly [24].

3.2 Probit regression model

In keeping with the Bayesian approach, we select the probit regression model as a binary linear classifier in the latent space of the VAE. This includes a weight vector $w \in \mathbb{R}^d$, a bias term $w_0 \in \mathbb{R}$ and a hidden variable $\lambda_i \in \mathbb{R}$ for the i-th data point. The class label y_i is modeled as $y_i = \mathrm{sign}(\lambda_i)$ where λ_i depends on w and w_0 as described below. We also select Gaussian prior distribution for w, w_0. The hierarchical structure of the probit regression model can be written as:

$$w \sim \mathcal{N}(0, I), \quad w_0 \sim \mathcal{N}(0, \sigma_0^2), \quad \lambda_i \sim \mathcal{N}\big(\langle z_i^K, w\rangle + w_0, 1\big) \tag{2}$$

As mentioned, the label of the i-th data point is the sign of λ_i.

4 VARIATIONAL INFERENCE AND COUNTERFACTUAL GENERATION

We next give a high-level overview of the variational inference algorithm for learning the proposed supervised VAE model and discuss how it can be used for counterfactual generation.

4.1 Autoencoder: VAE model with NF

As derived by Rezende and Mohamed [39], the objective function for the VAE can be written

$$\mathcal{L}_{\mathrm{VAE}} = \sum_{i=1}^{N} -\mathbb{E}_{z_i^k}[\log(q(z_i^0))] + \mathbb{E}_{z_i^k}[\log(p(x_i|z_i^k))]$$
$$+\mathbb{E}_{z_i^k}[\log(p(z_i^k))] + \sum_{k=1}^{K} \mathbb{E}_{z_i^k}[\log(q(z_i^k))] \tag{3}$$

where the expectation is with respect to the chosen q distribution and K is the number of flows. (We use the now-standard settings for these, including amortization of q.) We use stochastic gradient ascent to iteratively learn the parameters of $q(z^k|x)$ and $p(x|z^k)$. By changing the value of β in Equation 1 we can control the variance of squared errors resulting from the decoder network.

4.2 Classifier: Probit regression model

To approximate the full posterior distribution of w, w_0 and each λ_i, we apply the mean-field assumption,

$$q(\lambda, w, z, w_0) \equiv q(w)q(w_0)\prod_{i=1}^{N} q(\lambda_i)q(z_i).$$

The objective for this part of the model can be written as

$$\mathcal{L}_{\mathrm{probit}} = \mathbb{E}_q\left[\ln \frac{p(y_i, w, w_0, \lambda, z|x)}{q(w)q(w_0)\prod_{i=1}^{N} q(\lambda_i)q(z_i|x_i)}\right] \tag{4}$$

where the joint likelihood factorizes as

$$p(y, w, \lambda, z, w_0|x) = p(w)p(w_0)\textstyle\prod_i p(y_i|\lambda_i)p(\lambda_i|w, w_0, z_i)p(z_i|x_i). \tag{5}$$

Combining the terms from the two subsections above gives the overall objective

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{VAE} + \mathcal{L}_{\mathrm{probit}}.$$

We use coordinate ascent to iteratively update the following variational parameters in closed-form. All expectations are with respect to the q distribution. Algorithm 1 gives the learning outline:

$q(\lambda_i)$: A truncated normal with parameter $\mu_{\lambda_i} = \vec{\mu_i}^T \mathrm{E}[w]$.
$q(w)$: Gaussian, $\mu' = \Sigma'(\sum_i \mathrm{E}[\lambda_i]\vec{\mu_i}),\ \Sigma' = (I + \sum_i \mathrm{E}[z_i z_i^T])^{-1}$.
$q(w_0)$: Gaussian, $\sigma_0'^2 = \frac{1}{N+\sigma_0^{-2}}$ and $\mu'_0 = \sigma_0'^2 \sum_i (\mathbb{E}[\lambda_i] - \mathbb{E}[z_i]^T \mathbb{E}[w])$.
$q(z)$: This update is more complicated, but can directly use the VAE-NF updates built into the Pytorch or Tenserflow libraries.

4.3 Counterfactual generation

As the probit regression model supervises the latent space of the VAE during training, we observe that the data distribution of each class in the latent space will be nearly symmetric around the learned decision boundary. This allows us to project a latent vector directly to or beyond the decision boundary into the counter-class density and decode a counterfactual. To this end, we observe that any vector z_i^K can be projected onto the decision boundary of a linear classifier by calculating $z_i^K + \alpha_i * w$ where α_i satisfies the condition

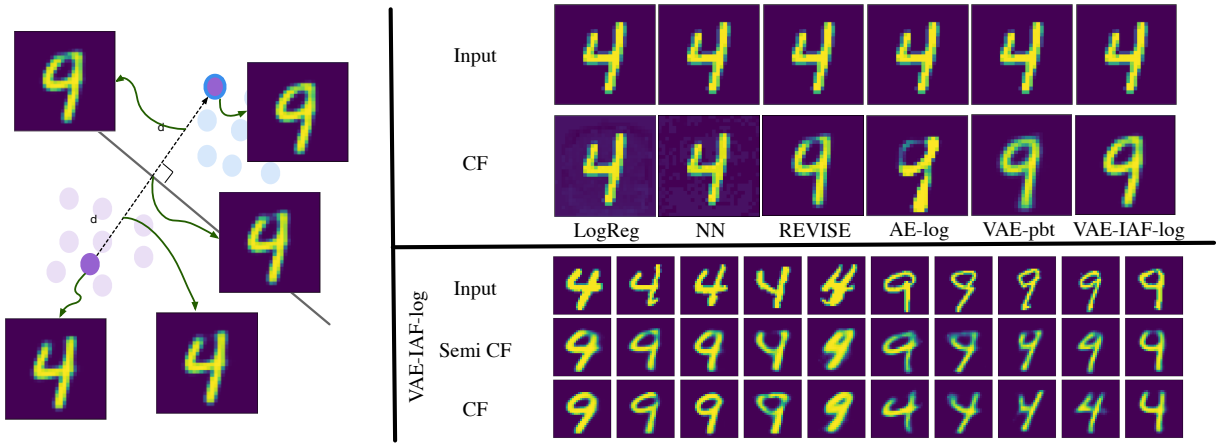

Figure 2: A visual example for illustrative purposes. Left: A rendering of the VAE's latent space and the corresponding decoded images at various locations. Using the decision boundary of linear classifier in this space, we can generate a sequence of counterfactuals that look increasingly like the counter-class. Top right: Counterfactual examples for the same query input for multiple models (see Table 1). Bottom right: Examples of different factual inputs (row 1), the corresponding counterfactual on the latent decision boundary (row 2) and the CF moving into the counter-class latent space (row 3).

$(z_i^K + \alpha_i w)^T w + w_0 = 0$. Solving gives $\alpha_i = -((z_i^K)^T w + w_0)/w^T w$. A counterfactual latent vector $(z_i^K)^c$ can be found by setting

$$(z_i^K)^c = z_i^K + \text{step} \times \alpha_i w, \tag{6}$$

where step ≥ 1, which determines how far into the counter-class region to move. A counterfactual sample can be generated by passing $(z_i^K)^c$ through the decoder network.

5 EXPERIMENTS

While we emphasize that image data is not the focus of this paper, we first illustrate our idea on the MNIST handwritten digit data set [29] to provide additional intuition. We then focus on two large scale financial tabular data sets for credit lending. For all experiments, we use PyTorch to implement the algorithm.[1]

5.1 Illustrative example: MNIST data set

MNIST is a public benchmark data set of handwritten digits. The data set contains 60,000 training samples and 10,000 in testing samples, both distributed across the ten digits. In our scenario, we select only 4 and 9, which gives 10,761 training samples for binary classification.

In all autoencoder models we consider, the network architecture of the encoder consists of 2 fully connected layers, starting from 400 nodes in the first layer and 20 nodes in the second layer. The dimension of the latent space is 20. The architecture of the decoder is the reverse of the encoder. We use the ReLU [33] as the nonlinear function, except for the output layer where we have no activation function. For this data, we set the variance parameter $\beta = 0.5$ as defined in Equation 1. See Table 1 for a description of models.

Counterfactual generation. We illustrate our idea by presenting a qualitative evaluation for counterfactual generation. Figure 2 shows the visualization of counterfactual samples and comparison with other models. In the left plot, we show the progressive transformation of a 4 to its counter class. We first sample 4 latent space vectors by setting the step size in Equation 6 to 0.5, 1, 1.5 and 2. We then pass them through the decoder network. We observe that the visualizations present a meaningful and arguably minimal change to the input when mapping to the counter-class. In the upper right plot, we compare the quality of a counterfactual sample (setting "step" to 2) for several models. The plot shows that adding a small amount of perturbation is enough to deceive logistic regression. The NN model also simply adds spurious noise to the sample space. The REVISE model seems to find a sample around the decision boundary, leading to a semi-counterfactual appearance. The AE-log shows a counterfactual sample but is unable to make it solid, indicating the possibly lack of regularization in latent space. We see that the VAE-pbt and VAE-IAF-log achieve the best counterfactual performance. In the bottom right, we show input samples, semi-counterfactual and counterfactual samples from 4s and 9s using our VAE-IAF-log model. These samples again demonstrate that our model makes progressively more meaningful modifications to the input to generate a datum-specific counterfactual.

5.2 Methods and evaluation

We evaluate our method in comparison with several other counterfactual approaches summarized in Table 1. We briefly describe these below.

As a baseline, we compare with logistic regression for both classification and counterfactual generation. Logistic regression directly learns a weight vector for the input space. Thus, counterfactual samples can be generated by directly projecting an input using the learned weight vector, which is the same for all inputs. We also evaluate classification performance on a neural network that is identical to the encoder architecture of our model. The counterfactual sample in this setting can be generated by iteratively optimizing the input space such that the predicted label is turned to the counter-class,

[1]For our Python implementation, see https://github.com/Wei2624/SVAE

Table 1: Methods evaluated in our experiments. Cited methods are learned using the CARLA library [36].

Notation	Description
LogReg	Binary logistic regression model
NN	Only encoder of VAE as classifier
Wachter	Proposed by Wachter et al. [46]
CCHVAE	Proposed by Pawelczyk et al. [37]
REVISE	Proposed by Joshi et al. [18]
CRUDS	Proposed by Downs et al. [10]
CLUE	Proposed by Antorán et al. [3]
AE-log	Autoencoder with logistic regression
VAE-pbt	VAE with probit regression
VAE-IAF-log	VAE with IAF flow and log. reg.

namely $x' = x + \partial f(x)/\partial x$ where f is the deep classifier. For the option of normalizing flows, we use inverse autoregressive flow proposed by Kingma et al. [22]. While there are many advanced flows in recent years, our idea can be easily adapted to those cases. We also benchmark with the popular counterfactual model [46] and models that use VAEs for generating counterfactuals [3, 10, 18, 37]. In all cases, we first train a deep neural network classifier that has the identical encoder structure of our model. We use the CARLA library [36] to implement these benchmarking methods. We use the default parameters of all benchmarking methods.

To evaluate quantitatively, we first compare the classification performance on the original data sets across different models. Then, we evaluate the plausibility of the generated counterfactuals using the Maximum Mean Discrepancy (MMD) measure [14]. Here, we evaluate the MMD distance between the generated counterfactuals of class 1 (from class 0) and the true samples of class 1. A lower MMD indicates a better matching of two distributions. In this case, a lower MMD indicates that a counterfactual looks like data from the counter-class.

However, counterfactuals that look like the counter-class are not by themselves sufficient. (For example, every counterfactual could map to the same point in the counter-class.) We therefore introduce a secondary logistic regression detector that aims to distinguish true counter-class data from counterfactuals. The closer to 0.5 accuracy (random guessing) the harder time the detector has telling counterfactuals from real data. Finally, because not all methods generate counterfactuals that change labels when reclassified (which is not desireable if being used to advise how to change an outcome), we also evaluate the frequency of failures. Finally, we show computation time requirements for finding a counterfactual.

For all experiments, the training batch size for each update is 128 random samples with nearly proportional number of samples from each class. The initial learning rate is 10^{-4} and decreases by 10 times after the loss fails to significantly decrease for 10 consecutive epochs. The training stops after the loss fails to significantly decrease for 20 consecutive epochs. We set the step size $\rho_t = 1/(t+1)^{0.9}$ for the stochastic variational inference [17], $\sigma_0^2 = 10^{-4}$ in Equation 2. We use 10 IAF normalizing flows in flow setting. We use Adam [20] for stochastic gradient method. We run our experiments 5 times for each setting and show error bars in the results.

5.3 Lending Club Data (LCD)

LCD consists of loan application profiles with the label of non-default loan or default loan. We select 10,000 samples for training and 1,000 for testing. The number of positive and negative sample in both the training and testing set is equally distributed. The data set is publicly available at Kaggle.com. The feature dimension for this data set is 7, which contains 5 scalar features and 1 binary feature mapped to one-hot vectors. We first standardize the real-valued features. In all autoencoder models, the network architecture is similar to the MNIST example, except that the first layer consist of 200 nodes and the latent dimension is 4. For this data, we set the variance parameter $\beta = 0.005$ in Equation 1.

Quantitative results. We evaluate the classification performance of good versus bad loan for LCD. We also consider a wider neural network (VAE-IAF-wide) which has twice the number of nodes at each layer, and a deeper neural network (VAE-IAF-deep) which has two more layers with 20 nodes for each layer. We present our results in Table 2. The results show that our model can serve as a state-of-art classifier for supervised learning applications in finance. We observe that VAE-IAF-log does not score identical to a NN that has an identical architecture as the encoder of VAE-IAF-log, which is because we regularize the latent space. However, we can add more layers and nodes to the encoder to improve classification performance. We also evaluate our model with different latent dimensionality, showing those results in Table 3.

Counterfactual analysis. We generate counterfactual samples by again projecting a latent vector to its counter-class's distribution based on Equation 6 and then reconstruct using the decoder. We compare different models qualitatively and show our results in Figure 3. We then apply MMD to evaluate the plausibility of our method, i.e. the similarity of the distributions between the samples' counterfactual their original inputs. We also introduce a secondary detector, trying to classify between factuals and counterfactuals. The frequency of failing to find a counterfactual is also measured. All results are shown in Table 4.

Despite the zero failure rates, LogReg, NN, Wachter and REVISE score a higher MMD, implying less distribution matching between the target class and the generated counterfactuals. NN, Wachter and REVISE also generate counterfactual samples that can be easily detected (high DC score). This is clearly observed in Figure 3. For example, REVISE proposes to increase loan amount and increase interest rate, which are usually considered to have a negative impact on loan applications. Wachter, NN and CLUE even downplay the impact of FICO score and cannot drive it as high as the factual targets. On the other hand, CRUDS achieves an even better MMD score. However, CRUDS tends to generate invariant counterfactual samples. Indeed, as shown in Figure 3, regardless of the noticeable range of input queries, CRUDS produces invariant counterfactuals. Though leading to a better MMD, the invariant proposals make counterfactuals less informative at the observation level. On the contrary, as our models make tailored modifications on a factual query, our models preserve the variations as shown in Figure 3. The plots also demonstrate that our counterfactual features' distribution look almost the same as the target factuals. LogReg, often thought

Table 2: Classification results on LCD and GMC for different models and settings. All methods perform roughly comparable, with slight improvement using deep learning. Thus, the proposed method is a state of the art classifier. However, Table 4 shows that it also has advantages as a counterfactual generator that are *not* available to other methods.

Model	LCD			GMC		
	Accuracy(%)	Precision(%)	Recall(%)	Accuracy(%)	Precision(%)	Recall(%)
LogReg	88.5±0.1	87.4±0.1	90.1±0.1	75.0±0.1	69.7±0.1	88.6±0.1
NN	89.1±0.1	87.7±0.1	91.3±0.1	77.6±0.1	74.5±0.1	85.2±0.1
AE-log	88.9±0.3	87.2±0.3	91.2±0.3	77.5±0.3	73.7±0.3	85.2±0.3
VAE-pbt	88.5±0.6	87.1±0.4	90.4±0.5	75.4±0.6	70.0±0.5	89.0±0.6
VAE-IAF-log	88.7±0.4	87.3±0.4	90.6±0.3	75.8±0.6	70.3±0.7	89.4±0.6
VAE-IAF-wide	88.9±0.4	87.5±0.5	90.8±0.5	75.8±0.7	70.2±0.6	89.6±0.5
VAE-IAF-deep	89.0±0.5	87.5±0.4	91.2±0.4	76.5±0.7	70.9±0.6	90.0±0.6

Table 3: Classification results of VAE-IAF-d model on LCD and GMC with different latent dimensionalities d.

LCD	Accuracy(%)	Precision(%)	Recall(%)	MMD+	MMD-	DC+	DC-
VAE-IAF-2	88.1±0.4	86.7±0.5	90.0±0.5	0.21±0.03	0.22±0.02	0.53±0.03	0.55±0.02
VAE-IAF-3	88.2±0.4	86.9±0.5	90.0±0.5	0.20±0.03	0.25±0.02	0.53±0.03	0.53±0.02
GMC	Accuracy	Precision	Recall	MMD+	MMD-	DC+	DC-
VAE-IAF-3	74±0.4	68.8±0.5	88.0±0.6	0.43±0.06	033±0.05	0.68±0.05	0.59±0.04
VAE-IAF-4	74.4±0.5	68.8±0.3	88.0±0.5	0.45±0.06	0.36±0.05	0.65±0.05	0.56±0.04
VAE-IAF-5	75.2±0.5	69.7±0.4	89.0±0.4	0.40±0.03	0.29±0.04	0.63±0.05	0.56±0.04

Table 4: MMD distance and detector classification (DC) between one class's cluster and the projected cluster on LCD and GMC in input space. MMD+/- is the distance between true positive/negative cluster and projected negative/positive cluster using maximum mean discrepancy. DC+/- is the classification accuracy for detecting those respective counterfactuals from the true counter-class data. Failure rate (FR) is the percentage of the samples for which the counterfactual does not change label.

Model	LCD					GMC				
	MMD+	MMD-	DC+	DC-	FR(%)	MMD+	MMD-	DC+	DC-	FR(%)
LogReg	0.60±0.01	0.57±0.02	0.58±0.01	0.60±0.01	0	0.68±0.01	0.44±0.01	0.65±0.02	0.83±0.01	0
NN	0.89±0.01	0.93±0.01	0.80±0.02	0.82±0.01	0	0.90±0.01	0.48±0.01	0.97±0.01	0.99±0.01	0
Wachter	0.98±0.01	0.87±0.02	0.87±0.01	0.87±0.01	0	0.49±0.02	0.65±0.03	0.77±0.03	0.75±0.02	0
CCHVAE	0.94±0.02	0.25±0.01	0.98±0.01	0.86±0.01	7	0.20±0.02	0.58±0.03	0.99±0.01	0.99±0.01	5
REVISE	0.44±0.03	0.25±0.02	0.95±0.03	0.88±0.03	0	0.22±0.04	0.29±0.03	0.99±0.01	0.99±0.01	0
CRUDS	0.12±0.02	0.13±0.02	0.82±0.01	0.81±0.01	13	0.10±0.01	0.11±0.01	0.81±0.02	0.86±0.01	15
CLUES	0.19±0.01	0.21±0.01	0.90±0.02	0.93±0.02	11	0.19±0.02	0.19±0.02	0.97±0.01	1.00±0.00	9
AE-log	0.63±0.04	0.50±0.04	0.65±0.03	0.63±0.02	1	0.65±0.05	0.56±0.05	0.73±0.04	0.75±0.05	1
VAE-pbt	0.54±0.06	0.51±0.05	0.57±0.01	0.59±0.05	2	0.63±0.05	0.56±0.05	0.72±0.02	0.71±0.01	1
VAE-IAF-log	0.18±0.03	0.17±0.02	0.51±0.03	0.52±0.02	2	0.38±0.04	0.27±0.03	0.63±0.03	0.56±0.02	2

Table 5: Running time (seconds/sample) across different models for generating counterfactuals.

	LogReg	NN	Wachter	CCHVAE	REVISE	CRUDS	CLUE	AE-log	VAE-pbt	VAE-IAF-log
LCD	<0.0001	1.9	1.9	1.0	4.7	28.5	2.2	<0.001	<0.001	<0.001
GMC	<0.0001	2.0	0.4	0.6	4.1	14.1	2.7	<0.001	<0.001	<0.001

of as the most explainable model in the financial field, fails to lower the interest rate enough.

Our models run significantly faster than all the benchmark methods to generate counterfactuals. This is shown in Table 5. This is intuitive in that all the benchmark methods rely on a search algorithm to find each sample's counterfactual, whereas ours only depend on a closed-form linear projection and no iterations.

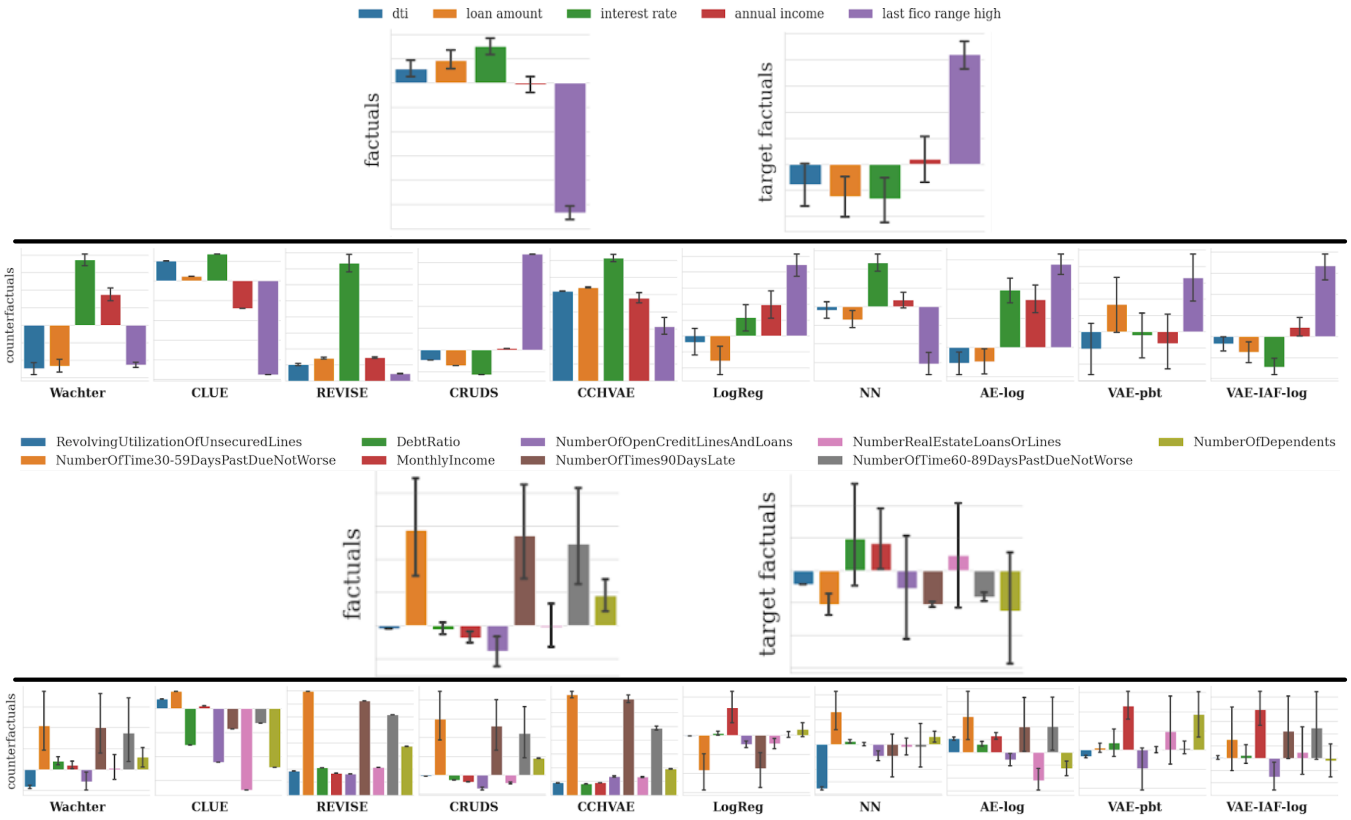

Figure 3: Comparisons of feature distributions on LCD (top half of figure) and GMC (bottom half of figure) data sets, showing the mean and variance for each dimension. The features have been standardized. The first row (above line) of each plot shows the rejected factual data (class 0) and accepted factual data (class 1). The second row (below line) of each plot shows features distributions of counterfactuals generated by different models. A closer appearance to class 1 from row 1 indicates that the distribution of the counterfactuals looks like the distribution of the true data in the counter-class. The labels of each dimension are shown at top of each plot.

5.4 GMC data set

GMC consists of credit application profiles with label of delinquency or no delinquency. We select 15,000 samples for training and 1,000 for testing. The number of positive and negative samples in both sets are equal. The data set is publicly available at Kaggle.com. The feature dimension for this data set is 9.

Quantitative results. We again evaluate the classification performance of having delinquency or no delinquency from the selected GMC. We consider a wider neural network (VAE-IAF-wide) and a deeper neural network (VAE-IAF-deep). We present our results in Table 2. The results show that our models perform consistently. AE-log, which has the least regularization in latent space achieves similar performance to the NN. We also evaluate our model with different latent dimensions. Results are shown in Table 3.

Counterfactual analysis. We generate counterfactual samples by again projecting a latent vector to its counter-class's distribution based on Equation 6 and then reconstruct with the decoder. We show qualitative results in Figure 3. We again apply MMD to evaluate the plausibility of all the methods, and then introduce the secondary detector to classify between factuals and counterfactauls and measure the running time per sample. The results are shown in Table 4 and Table 5, respectively.

We again observe that CLUE and CRUDS generate almost invariant counterfactual samples that can be easily classified through the secondary detector. This is reflected in a low MMD but a high DC in Table 4. In addition, only our proposed models, LogReg, and REVISE suggest significantly increasing monthly income in order for an application to be accepted. However, we again see that REVISE propose much less diversified counterfactuals, indicating that the counterfactuals are not tailored to the input. In this scenario, we may still be able to explain the black-box model. However, tailored recommendations are impossible, which is one of the advantages of using counterfactaul explanation. We also measure the running time on GMC and show the results in Table 5. Not surprisingly, our models again run much faster than all the benchmark methods.

6 CONCLUSION

In this paper, we propose a supervised VAE model using normalizing flows and a probit classifier on the latent space of the VAE.

Using two financial tabular data sets, we show that this approach can achieve excellent classification performance, which was expected since the encoder/probit model is equivalent to a neural network. However, we also demonstrated that the learned linear decision boundary in the latent space of the VAE can help direct the counterfactual generation in a more meaningful way using a simple closed-form equation that requires no additional optimization. This allows for a simple unified method for explainable deep learning for problems such as credit lending.

REFERENCES

[1] Charu C Aggarwal, Chen Chen, and Jiawei Han. 2010. The inverse classification problem. *Journal of Computer Science and Technology* 25, 3 (2010), 458–468.

[2] Jinwon An and Sungzoon Cho. 2015. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE* 2, 1 (2015), 1–18.

[3] Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. 2020. Getting a clue: A method for explaining uncertainty estimates. *arXiv preprint arXiv:2006.06848* (2020).

[4] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. 2018. Explaining image classifiers by counterfactual generation. *arXiv preprint arXiv:1807.08024* (2018).

[5] Joseph Paul Cohen, Rupert Brooks, Sovann En, Evan Zucker, Anuj Pareek, Matthew P Lungren, and Akshay Chaudhari. 2021. Gifsplanation via Latent Shift: A Simple Autoencoder Approach to Counterfactual Generation for Chest X-rays. *arXiv preprint arXiv:2102.09475* (2021).

[6] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. 2020. Multi-Objective Counterfactual Explanations. *arXiv preprint arXiv:2004.11165* (2020).

[7] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. 2018. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in neural information processing systems*. 592–603.

[8] Amit Dhurandhar, Tejaswini Pedapati, Avinash Balakrishnan, Pin-Yu Chen, Karthikeyan Shanmugam, and Ruchir Puri. 2019. Model agnostic contrastive explanations for structured data. *arXiv preprint arXiv:1906.00117* (2019).

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016).

[10] Michael Downs, Jonathan L Chu, Yaniv Yacoby, Finale Doshi-Velez, and Weiwei Pan. 2020. Cruds: Counterfactual recourse using disentangled subspaces. *ICML WHI* 2020 (2020), 1–23.

[11] Rubén R Fernández, Isaac Martín de Diego, Víctor Aceña, Alberto Fernández-Isabel, and Javier M Moguerza. 2020. Random forest explainability using counterfactual sets. *Information Fusion* 63 (2020), 196–207.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014), 2672–2680.

[13] Rory Mc Grath, Luca Costabello, Chan Le Van, Paul Sweeney, Farbod Kamiab, Zhao Shen, and Freddy Lecue. 2018. Interpretable credit application predictions with counterfactual explanations. *arXiv preprint arXiv:1811.05245* (2018).

[14] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research* 13, 1 (2012), 723–773.

[15] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. 2018. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820* (2018).

[16] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)* 51, 5 (2018), 1–42.

[17] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *Journal of Machine Learning Research* 14, 5 (2013).

[18] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. 2019. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615* (2019).

[19] Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. 2020. Algorithmic Recourse: from Counterfactual Explanations to Interventions. *arXiv preprint arXiv:2002.06278* (2020).

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

[21] Diederik P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039* (2018).

[22] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems* 29 (2016), 4743–4751.

[23] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934* (2016).

[24] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[25] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar variational autoencoder. In *International Conference on Machine Learning*. PMLR, 1945–1954.

[26] Michael T Lash, Qihang Lin, Nick Street, Jennifer G Robinson, and Jeffrey Ohlmann. 2017. Generalized inverse classification. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 162–170.

[27] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2018. Comparison-based inverse classification for interpretability in machine learning. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, 100–111.

[28] Lei Le, Andrew Patterson, and Martha White. 2018. Supervised autoencoders: Improving generalization performance with unsupervised regularizers. *Advances in Neural Information Processing Systems* 31 (2018), 107–117.

[29] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010). http://yann.lecun.com/exdb/mnist/

[30] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 305–314.

[31] Divyat Mahajan, Chenhao Tan, and Amit Sharma. 2019. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277* (2019).

[32] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 607–617.

[33] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Icml*.

[34] Daniel Nemirovsky, Nicolas Thiebaut, Ye Xu, and Abhishek Gupta. 2020. CounteRGAN: Generating Realistic Counterfactuals with Residual Generative Adversarial Nets. *arXiv preprint arXiv:2009.05199* (2020).

[35] George Papamakarios, Theo Pavlakou, and Iain Murray. 2017. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057* (2017).

[36] Martin Pawelczyk, Sascha Bielawski, Johannes van den Heuvel, Tobias Richter, and Gjergji Kasneci. 2021. Carla: a python library to benchmark algorithmic recourse and counterfactual explanation algorithms. *arXiv preprint arXiv:2108.00783* (2021).

[37] Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. 2020. Learning Model-Agnostic Counterfactual Explanations for Tabular Data. In *Proceedings of The Web Conference 2020*. 3126–3132.

[38] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. 2016. Variational autoencoder for deep learning of images, labels and captions. *arXiv preprint arXiv:1609.08976* (2016).

[39] Danilo Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *International Conference on Machine Learning*. PMLR, 1530–1538.

[40] Pouya Samangouei, Ardavan Saeedi, Liam Nakagawa, and Nathan Silberman. 2018. Explaingan: Model explanation via decision boundary crossing transformations. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 666–681.

[41] Axel Sauer and Andreas Geiger. 2021. Counterfactual generative networks. *arXiv preprint arXiv:2101.06046* (2021).

[42] Shubham Sharma, Jette Henderson, and Joydeep Ghosh. 2019. Certifai: Counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. *arXiv preprint arXiv:1905.07857* (2019).

[43] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (2019), 828–841.

[44] Arnaud Van Looveren and Janis Klaise. 2019. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584* (2019).

[45] Sahil Verma, John Dickerson, and Keegan Hines. 2020. Counterfactual Explanations for Machine Learning: A Review. *arXiv preprint arXiv:2010.10596* (2020).

[46] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* 31 (2017), 841.

[47] Aonan Zhang and John Paisley. 2018. Deep Bayesian nonparametric tracking. In *International Conference on Machine Learning*. PMLR, 5833–5841.