Bias and Explainability

Dependable AI Minor-2

Kwanit Gupta, B19EE046[a]

[a]*Indian Institute of Technology, Jodhpur*

March, 2023

Note

With the Options given to either go for Question-2 or Question-3, I decided to go for Question-2 with the CVPR 2021's Submission **"EnD: Entangling and Disentangling deep representations for bias correction"**, authored by Enzo Tartaglione, Carlo Alberto Barbano, and Marco Grangetto (all from University of Turin, Computer Science Dept.) where they emphasized on preserving the classification task as well as mitigating the effects of training data biases, in the model level itself without hindering the originality of classification model.

---

1. Question-1

- There have been several instances of bias reported in the country and globally. Share one instance of bias that you have observed in any of the algorithms. The topic should not have been discussed in class or shared on the web. Show documentary evidence (screenshot or video, etc.) reporting the observation.

- For the above question, I will be discussing the comparatively newer domain, i.e **Generative Diffusion-based Deep Learning**.

1.1. Instance and Possible Description

Following 2 instances somehow went viral in Internet, but researchers still looking for the reasons behind these:-

- Indian Couples:- In the twitter's post (Link) dated 30th December 2022, Generative AI model produced stereotypical images of different wedding couples (maybe the source being some textual descriptions or some reference images). In this controversy, the images of a Gujarati and Bengali Couple received high backlashes from the audience and somehow were trolled altogether.

- Indian Rulers:- In the twitter's post (Link), dated 26th January 2023, Generative AI model this time produced images of different Indian rulers. In the controversy, the facial structures were highly questioned because of the lack of enough descriptions and being historically incorrect too according to each ruler's portrait.

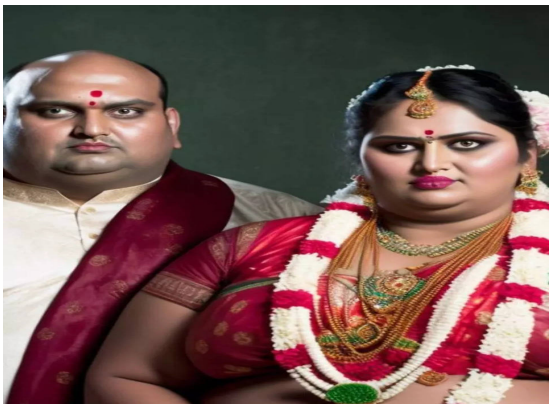

Figure 1: Gujarati Couple's Picture, using AI

Figure 2: King Chandragupta's Image generated, using AI

To get the overall fundamental scenario of Generative-based AI, go to further sub-sections, else here's the discussion on the observation of Generative Diffusion-based Deep Learning Bias.

1.2. Investigation

At a high level, diffusion models rely on a series of transformations of an initial noise variable to generate samples. These transformations are performed through the application of a series of invertible transformations on the noise variable, typically achieved through the use of neural networks. However, because these transformations are learned from data, biases can arise if the data itself is biased or if the model is not able to capture the full distribution of the data.

At the data level, biases can arise if the training data is not representative of the full distribution of the data. For example, if the training data is biased towards a particular set of features or characteristics, the diffusion model may not be able to learn a full representation of the underlying data distribution. This can result in the generation of samples that are biased towards these particular features or characteristics.

At the feature map level, biases can arise if the learned feature representations are not able to capture the full complexity of the underlying data distribution. For example, if the model is only able to capture simple correlations between features, it may not be able to accurately capture more complex correlations, resulting in biased sample generation.

At the model level, biases can arise if the model architecture or training procedure is not appropriate for the task at hand. For example, if the model architecture is too simple, it may not be able to capture the full complexity of the data distribution. On the other hand, if the model architecture is too complex, it may overfit to the training data, resulting in poor generalization performance.

Finally, at the pipeline level, biases can arise if there are biases in the data preprocessing or postprocessing steps. For example, if the data is preprocessed in a way that introduces bias or if the generated samples are postprocessed in a way that amplifies biases, this can result in biased sample generation.

Let's assume we have a diffusion-based deep learning model, f, that takes an input x and generates an output y. The biases in the model can be expressed as:

$$y = f(x) + b \tag{1}$$

Where b is a bias term that represents the difference between the model's output and the true output. The bias term b can be further decomposed into two components: systematic bias and random bias.

Systematic bias arises from the model architecture and the assumptions made during training. For example, if the model is designed to generate images of people, but the training data is biased towards a particular race, the model may generate biased output that reflects that bias. Systematic bias can be quantified as the expected value of the bias term:

$$E[b] = E[f(x)] - y \tag{2}$$

Random bias, on the other hand, arises from the noise inherent in the data and the modeling process. This type of bias can be quantified as the variance of the bias term:

$$Var[b] = E[(f(x) - E[f(x)] + y - E[y])^2] \tag{3}$$

1.3. Overview and Background Literature

To understand the above domain, its important to get the individual aspects covered first. So following will help to fulfill the purpose:-

1.3.1. Generative Deep Learning

Generative deep learning is a subfield of artificial intelligence (AI) and machine learning that focuses on developing algorithms capable of creating new data instances that resemble the training data. The goal of generative models is to generate new data that is indistinguishable from real data, in terms of quality, distribution, and diversity. Generative models can be used for a wide range of applications, such as image and video synthesis, text and speech generation, music composition, and drug discovery. Following models are in common usage nowadays:-

- Generative Adversarial Networks (GANs):- The GAN architecture consists of two neural networks: a generator network and a discriminator network. The generator network takes random noise as input and produces a sample that is intended to be similar to the training data. The discriminator network takes the generated sample and a real sample from the training data as input and tries to distinguish between them. The two networks are trained in an adversarial process where the generator tries to produce more realistic samples to fool the discriminator, and the discriminator tries to correctly distinguish between real and fake samples. This can be formulated as a minimax game where the generator tries to

minimize the discriminator's ability to distinguish between real and fake samples, and the discriminator tries to maximize its ability to distinguish between the two.

- AutoEncoders:- The basic architecture of an autoencoder consists of an encoder network and a decoder network. The encoder network takes an input x and produces a lower-dimensional representation z, while the decoder network takes z and produces a reconstruction of the original input x'. The encoder and decoder networks are typically neural networks, and the training process involves minimizing a loss function that measures the difference between the input x and the reconstructed output x'. The loss function is often the mean squared error between the two.

- AutoRegressive Models:- Auto-regressive models generate new data points by modeling the conditional probability distribution of a sequence of data points given the previous values. The probability distribution is typically modeled using a neural network with a fixed number of parameters. The probability of generating a new data point is calculated by taking the product of the probabilities of generating all previous data points.

- Flow-based Models:- Flow-based models learn a transformation function that maps a simple distribution, typically a Gaussian distribution, to a more complex target distribution. The transformation function is composed of a sequence of invertible and differentiable transformations, each of which is designed to preserve the Jacobian determinant of the transformation. The Jacobian determinant is a measure of how much the transformation stretches or shrinks the distribution, and preserving it is necessary for the transformation to be reversible and for the target distribution to be tractable.

1.3.2. Diffusion-based Deep Learning

The idea of using the diffusion process for generative modeling dates back to the 1980s, when the Langevin diffusion process was introduced as a method for sampling from high-dimensional distributions. However, it was not until recently that diffusion-based models were introduced as a deep learning technique. In 2019, Ho et al. introduced the first diffusion-based deep learning model, called the Diffractor. This model used a series of diffusion steps to generate samples from a distribution and achieved state-of-the-art results on a number of image synthesis benchmarks. Since then, a number of other diffusion-based models have been introduced, including the Diffusion Autoencoder (DiffusionAE) and the Non-autoregressive Flow-based Model (NAF). These models have shown promising results on a variety of tasks, including image synthesis, video prediction, and speech synthesis.

1.4. How these even work?

- Generative Adversarial Networks (GANs):-
  - The generator network takes as input a random noise vector z of dimensionality d, and outputs a generated sample x of dimensionality D. The generator network can be represented as a differentiable function G(z;θ_g), where θ_g are the parameters of the generator network. The generator tries to map the input noise vector to a realistic-looking output.
  - The discriminator network takes as input a sample x of dimensionality D and outputs a scalar value between 0 and 1 that represents the probability that x is a real sample. The discriminator network can be represented as a differentiable function D(x;θ_d), where θ_d are the parameters of the discriminator network. The discriminator tries to distinguish between real and generated samples.
  - The objective function of GANs is to minimize the following expression:

$$L_{GD} = E[\log\big(D(x)\big)]+E[\log\big(1 - D(G(z))\big)] \tag{4}$$

    The discriminator network is trained on both the real and generated samples by minimizing the following binary cross-entropy loss:

$$L_D = -[log(D(x)) + log(1 - D(G(z)))] \tag{5}$$

    The generator network is trained by minimizing the following binary cross-entropy loss:

$$L_G = -log(D(G(z))) \tag{6}$$

- AutoEncoders:-
  - It consists of two main components, i.e an encoder and a decoder. The encoder maps the input data to a lower-dimensional space, while the decoder maps the lower-dimensional representation back to the original data space.

- Encoder: $h = f(x)$. The encoder function takes the input data x and maps it to a lower-dimensional representation h.
- Decoder: $\hat{x} = g(h)$. The decoder function takes the lower-dimensional representation h and maps it back to the original data space $\hat{x}$. This function is denoted by $g(h)$.
- Loss function: $L(x, \hat{x})$ The loss function is used to measure the difference between the input data x and the reconstructed data $\hat{x}$, represented by the following:-

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x_i})^2 \tag{7}$$

The goal of training an autoencoder is to minimize the loss function by adjusting the learnable parameters of the encoder and decoder.

- AutoRegressive Models:-
  - Let's assume that we have a sequence of observations represented by the variable $x = x_1, x_2, ..., x_T$. An autoregressive model predicts the probability distribution of each observation x_t given the previous observations $x_{t-1}, x_{t-2}, ..., x_1$. This can be written as follows:-

$$p(x_t|x_{t-1}, x_{t-2}, \ldots, x_1) \tag{8}$$

  - One common type of autoregressive model is the autoregressive moving average (ARMA) model. The ARMA model is defined by the following equation:

$$x_t = c + \sum_{i=1}^{p}\phi_i x_{t-i} + \sum_{j=1}^{q}\theta_j \epsilon_{t-j} + \epsilon_t \tag{9}$$

where c is a constant, ϕ_i are the autoregressive coefficients, θ_j are the moving average coefficients, ϵ_t is the error term at time t, and p and q are the orders of the autoregressive and moving average components, respectively.

  - Another common type of autoregressive model is the auto-regressive integrated moving average (ARIMA) model, which extends the ARMA model by adding a differencing step to make the time series stationary. The ARIMA model is defined by the following equation:

$$(1 - \sum_{i=1}^{p}\phi_i B^i)(1-B)^d x_t = c + (1 + \sum_{j=1}^{q}\theta_j B^j)\epsilon_t \tag{10}$$

where B is the backshift operator, d is the degree of differencing, and all other terms are the same as in the ARMA model.

  - In general, autoregressive models are trained by maximizing the log-likelihood of the observed data, given the previous observations. The log-likelihood is given by:

$$\log p(x_1, \ldots, x_T) = \sum_{t=1}^{T}\log p(x_t|x_{t-1}, \ldots, x_1) \tag{11}$$

- Flow-based Models:-
  - The basic idea behind flow-based models is to transform a simple probability distribution such as a Gaussian distribution into a more complex one using a series of invertible transformations. If we start with a random variable $z \sim \mathcal{N}(0, I)$, we can define a transformation function f that maps this random variable to a new random variable $x = f(z)$. The inverse transformation f^{-1} maps x back to z.
  - The probability density function of x can be expressed in terms of the probability density function of z using the change of variables formula:

$$p(x) = p(z)\left|\left|\frac{\partial z}{\partial f}\right|\right|^{-1} \tag{12}$$

The determinant of the Jacobian matrix $\frac{\partial f}{\partial z}$ is easy to compute for invertible transformations, so we can compute the probability density of x given the probability density of z. Multiple invertible transformations can be composed to form a flow-based deep learning model.

  - Define the first transformation to map from a simple distribution to a more complex distribution:

$$\begin{aligned} x_0 &= z \\ x_k &= f_k(x_{k-1}; \theta_k) \end{aligned} \tag{13}$$

  - The probability density function of x can be expressed in terms of the proba-

bility density function of z:

$$\log p(x) = \log p(z) - \sum_{k=1}^{K} \log \left| \left| \frac{\partial x_{k-1}}{\partial f_k} \right| \right| \tag{14}$$

– To sample from the model, we first sample from the simple distribution $z \sim \mathcal{N}(0, I)$, and then pass it through the series of invertible transformations:

$$x_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(z) \tag{15}$$

- Diffusion Models:-
  – The core idea behind diffusion-based deep models is to model the dynamics of a stochastic process that transforms the initial noise vector to the data space. This process can be modeled by a sequence of simple diffusion steps, where each step introduces a small amount of noise into the current state of the process. Formally, at each step, the diffusion process can be expressed as follows:

$$x_{t+1} = x_t + \sqrt{\beta_t} \cdot \epsilon_t \tag{16}$$

where x_t is the state of the diffusion process at time step t, ϵ_t is a sample from a standard Gaussian distribution, and β_t is a non-negative scalar that controls the amount of noise introduced at each step. The sequence of β_t values is typically chosen such that $\beta_1 > \beta_2 > \cdots > \beta_T$, where T is the total number of diffusion steps.

– The diffusion process can be seen as a Markov chain, where the transition probability density function at each step is given by:

$$p(x_{t+1} \mid x_t) = \mathcal{N}(x_{t+1} \mid x_t, \beta_t I) \tag{17}$$

where $\mathcal{N}$ denotes the Gaussian probability density function and I is the identity matrix.

– Given a set of training samples ${x_i}_{i=1}^{N}$, the goal of training a diffusion-based deep model is to learn the sequence of transformations that maps the initial noise vector to the data distribution. This can be achieved by minimizing the negative log-likelihood of the data under the diffusion process, which can be expressed as:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log p(x_t \mid x_{t-1}, \beta_t, \theta) \tag{18}$$

where θ denotes the parameters of the diffusion-based deep model, which can be learned by stochastic gradient descent.

– One of the main advantages of diffusion-based deep models is that they allow for efficient generation of high-quality samples from the learned distribution. This can be achieved by reversing the sequence of diffusion steps, starting from a sample from the data distribution and gradually removing the noise until the original noise vector is obtained. Specifically, given a sample x_T from the data distribution, a corresponding noise vector z can be obtained as follows:

$$z = \frac{x_T - \sqrt{\beta_T} \cdot \epsilon_T}{\sqrt{1 - \beta_T}} \tag{19}$$

Then, the sequence of transformations can be applied in reverse order to obtain the initial noise vector z_0, which can be used to generate new samples from the learned distribution.

1.5. Limitations

- Generative Adversarial Networks (GANs):- They experience mode collapse when the generator produces a limited number of samples that resemble each other, resulting in a lack of diversity in the generated data and the discriminator becoming too strong to even stop the training process of the generator. The lack of explicit representation of the underlying data distribution also intensifies the cons, making it difficult to condition the generator on specific attributes.

- AutoEncoders:- They can suffer from overfitting, where the model becomes too specialized to the training data and fails to generalize well to new data. They can also struggle to effectively encode complex information, where there is a lot of variation and noise, due to the encoding layer's insufficient capacity to represent all of the necessary information. They also struggle to capture the global structure of complex data, such as the overall layout of an image or the semantic relationships between words in a sentence. This is because the model focuses on local details rather than the larger context.

- AutoRegressive Models:- They have a limited view of the data because they can only consider the information from previous time steps, making it difficult for them to capture

long-term dependencies in the data. Other cons are the exponential growth of parameters with respect to higher dimensional data, crystallized independence assumptions, brittle flexibility due to heavier reliance on patterns, and the inability to handle variable-sized inputs.

- Flow-based Models:- They are limited in modeling complex probability distributions and high-dimensional data due to the invertibility requirement of the transformation function. They require the computation of the Jacobian determinant of the transformation function, which is computationally expensive. Poor initialization can lead to non-invertible transformations, which makes it impossible to train the model. Moreover, overfitting can occur if the model is too complex or if the data has too little variation. They are also continuous models and are therefore unsuitable for modeling discrete data directly.

- Diffusion Models:- They can be difficult to interpret, and it can be challenging to understand how the model generates its outputs. This can make it difficult to diagnose issues with the model or understand how to improve its performance. They are not suitable for all types of data, and they may not be the best choice for certain tasks. For instance, they may not be the best choice for tasks that require generating sharp, high-resolution images, as they can produce blurry outputs. Their training can be computationally expensive since they perform multiple iterations of the diffusion process to generate high-quality samples.

2. Question-2

- Review a paper of your choice published in top conference or journal, on bias or explainability. Submit the review of the paper. All of you should review a different paper, include the paper you selected in the spreadsheet Link – first come, first serve for paper selection.

- Present an extension to the paper you reviewed and explain what is limitation your solution will address and how.

- The modification to the algorithm should be clearly explained.

2.1. Overview

Nowadays, Artificial Neural Networks (ANNs) and their family of variants are capable of performing any specialized task with the least chance of mishappenings. But the aspect of universal usage and generalizability (even being in the same domain, but the distribution of data being differently sourced) is questioned by the working of the models and their inability to mitigate it due to data's intrinsic biases being unknown and non-evident altogether.
The authors proposed EnD, a regularization strategy that can be practiced with any loss function and any deep classification network without heavily relying on another module or training process, or otherwise a pipeline component. Its purpose is to introduce an "Information Bottleneck" at any feature layer inside the model to disentangle the intrinsic set of biases without compromising on the critical classification characteristics of data, mapped to target classes.

2.2. Mathematical Outlook

EnD acts like any regularization term that assists to minimize the loss without overfitting the model, as follows:-

$$J = L + R = L + \alpha R_{||} + \beta R_{\perp} \tag{20}$$

Here, $R_{||}$ represents the EnD term responsible to bind the classification-related characteristics of data intact to the task, and $R_{\perp}$ represents the EnD term responsible to disentangle and extract out the intrinsic biases of data samples to be decorrelated. To understand it more clearly, the following are the intuitions:-

- Authors highly took inspiration from Correlation Matrix and prepared the special case of the Gramian Matrix, using the following:-

$$\begin{aligned} G &= Y^T Y \\ g_{i,j} &= y_i^T y_j \end{aligned} \tag{21}$$

  where Y represents the norm-2 normalized matrix of samples' extracted features. Since each value will lie between -1 and +1, it will hereby showcase how strongly each sample relates to the other (i-th and j-th samples in the equation).

- They also had the following nomenclatures while formulating their propositions:-
  - G - Gramian Matrix
  - M - Batch Size
  - T - Cardinality of Target Class

- B - Cardinality of Bias Class
- N_τ - Output Feature Size of Sample τ
- y_i - i-th sample in mini-batch
- $M^{t,\sim}, M^{\sim,b}, M^{t,b}$ - Cardinality of Samples with target class and no-bias class, no-target class and bias class, & target-bias classes together, respectively.
- $y^{t,\sim}, y^{\sim,b}, y^{t,b}$ - Sample of the target class and no-bias class, no-target class and bias class, & target-bias classes together, respectively.

- Talking about the disentangling term $R_\perp$, its main focus is to de-correlate the intrinsic biases as much as possible with all patterns into the same bias class b. An ideal situation is hereby to make the following for bias class b:-

$$G^{\sim,b} = (y^{\sim,b})^T . y^{\sim,b} = I \tag{22}$$

implying that the $R_\perp$ should be prepared as follows:-

$$R_\perp = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{(M^{\sim,b})^2} \sum_{i,j} |g_{i,j}^{\sim,b}|^2 \tag{23}$$

- Talking about the entangling term $R_{||}$, its main focus is to force correlate the classification characteristics between data of different bias groups but of the same target class t. Hereby, the following would be the case:-

$$G^{t,\sim} = (y^{t,\sim})^T . y^{t,\sim} \tag{24}$$

implying that the $R_{||}$ should be prepared as follows:-

$$R_{||} = 1 - \frac{1}{T} \sum_{t=1}^{T} \frac{1}{(M^{t,\sim})^2} \sum_{i,j} g_{i,j}^{t,\sim} \tag{25}$$

But this will introduce the bias because of target class t, which will bring the problem back to square one, regardless of any bias class.
So to correct this, they obeyed the following changes:-

$$T(y_i) = T(y_j)$$
$$B(y_i)! = B(y_j)$$
$$R_{||} = 1 - \frac{1}{M} \sum_{i=1}^{M} \frac{\sum^j \delta(B(y_i), B(y_j)) . g_{i,j}^{T(y_i),\sim}}{\sum_{b!=B(y_i)} M^{T(y_i),b}} \tag{26}$$

where the target class should remain the same, but the bias classes differ. Also, $\delta(B(y_i), B(y_j))$ signifies the bias samples relation by giving 1 for different bias classes and 0 for vice versa.

2.3. Results

Authors experimented on 4 different benchmarks with 3 different models (Paper-Link) and produced results there accordingly:-

- Biased Coloured MNIST (with the suggested model by Bahng)
- CelebA and IMDB Face (Pretrained ResNet18 on ImageNet Dataset)
- CORDA (Pretrained DenseNet121 on CXR Dataset)

2.4. Possible Extensions

Following can be the directions, where extension can go, alongside their possible reasoning:-

- Inclusion of More Raw Intrinsic Biases:- Using the same formulation, but extending it towards the middle of the feature extraction module in CNNs (apart from loss component from the last layer and EnD in between Feature and Classifier Modules), as following:-

$$J = L + + R_{EnD} + l_{Features} \tag{27}$$

Depending on the choice of the user, it can be calculated via any similarity metrics (Jaccard Similarity, Cosine Similarity or ArcFace-based Loss) or indeed a modified version of EnD for 2D matrices.
The reason is to introduce more spatial biases apart from the domain-based intrinsic biases from logits and linear feature maps.

- Usage of other matrices norms:- In the proposed implementation, authors made use of Norm-2 normalization on sample-wise features. But following norms can be useful as well with their own set of advantages:-
  - 0-norm: It measures the number of non-zero elements in a vector. Using the 0-norm for normalization can be beneficial in situations where we want to encourage sparsity in the matrix. This can help reduce overfitting and improve interpretability, as it makes the matrix easier to understand.
  - 1-norm: It measures the sum of the absolute values of the elements in a vector. Using the 1-norm for normalization can be beneficial in situations where we want to encourage sparsity and promote interpretability, similar to the 0-norm. However, it is less strict than the 0-norm, as it allows for a small number of

non-zero elements. It can also help reduce the impact of outliers and improve the robustness of the model, similar to the infinite-norm.

  - p-norm: The 2-norm tends to shrink all the entries in a matrix uniformly, which can result in loss of information. On the other hand, using a p-norm with p¿2 can better preserve the original structure of the matrix.
  - infinite-norm: It measures the maximum absolute value of the elements in a vector. Using the infinite-norm for normalization can be beneficial in situations where we want to focus on the most significant elements in the matrix. This can help reduce the impact of outliers and improve the robustness of the model.

- Investigation on disentangling the influence of intrinsic biases over target class distribution and vice versa:- As of right now, intrinsic biases are seen at the model level on feature maps. But it's still a curiosity to see whether there are existing patterns and overlapping influences of bias classes (either at data, feature, or model level) on the behavior of logits and target classes. It can be vice versa too since imbalanced classes-based datasets themselves construct the biases, which earlier on, weren't even existent.