Unsupervised Latent Debiasing of Time-Series Models

by

Jacob Phillips

B.S, Computer Science & Engineering, Massachusetts Institute of Technology (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

Author .....
Department of Electrical Engineering and Computer Science
December 9, 2021

Certified by .....
Professor Daniela L. Rus
Director, MIT Computer Science and Artificial Intelligence Laboratory
Principal Investigator, MIT Distributed Robotics Laboratory
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Unsupervised Latent Debiasing of Time-Series Models

by

Jacob Phillips

Submitted to the Department of Electrical Engineering and Computer Science
on December 9, 2021, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Traditional training regimens for time-series models have been shown to encode the biases from their training corpora into the models themselves. We aim to train unbiased time-series models using existing biased datasets. However, most debiasing techniques rely on explicit labels that encapsulate the bias, such as pairs of words along some worrying axis of bias such as race or gender for language models. We propose an unsupervised latent debiasing training regimen based on [2] that simultaneously learns the latent distribution of the dataset and a separate language task; datapoints are selected for training batches by sampling weights inverse to their commonality as determined by their placement in the latent space. We adapt [2] to time-series datasets and show algorithmic improvements to bias identification and bias reduction for models trained on toy and real datasets.

Thesis Supervisor: Professor Daniela L. Rus
Title: Director, MIT Computer Science and Artificial Intelligence Laboratory
Principal Investigator, MIT Distributed Robotics Laboratory

Contents

1 Introduction **11**

2 Related Work **15**

2.1 Debiasing Datasets 16

2.2 Debiasing Algorithms 17

3 Methodology **21**

3.1 Latent Debiasing 21

3.2 Our Method 23

4 Experimental Results **27**

4.1 Experiments with Sine-Wave Datasets 27

4.1.1 Sine Wave Dataset 28

4.1.2 Biased Sine Dataset 29

4.1.3 Skewed Sine Dataset 31

4.1.4 Period Biased Sine Dataset 32

4.2 Experiments with Weather Forecasting Datasets 34

4.2.1 One City, Normalized Celsius 34

4.2.2 Mixed Cities Normalized Celsius Datasets 35

4.2.3 Experiments with Reconstruction Loss 37

4.3 Experiments with Language Datasets 39

5 Algorithmic Results **43**

5.1 Latent Dimension Probability Aggregation 43

5.2 Time-Series Dimension Latent Feature Aggregation 45
5.3 Discussion of Algorithmic Results 45

6 Future Work and Conclusion **47**
6.1 Future Work 47
6.2 Conclusion 47

List of Figures

3-1 The algorithmic diagram from [2]. We adapt and improve this algorithm to time-series datasets and other algorithimc improvements. . . 22

3-2 A representation of a given datapoint's latent vector as time-dimension T by latent dimension K. 24

3-3 Consolidation methods of the time-series dimension over sum, mean, and last. Note that sum and mean are scalar multiples with the colors normalized to have the same effect. 25

3-4 A representation of a batch of datapoints's latent vectors, with their time-dimension already consolidated. Presented as batch-dimension B by latent-dimension K. 26

3-5 A representation of the different latent-dimension consolidation methods as given by the max and product operators. 26

4-1 Value Density of a Sine Wave 28

4-2 Scatterplot of sampling probabilities from models trained on a sine wave 29

4-3 Histograms of top and bottom 10% of sequences from models trained on a sine wave. 29

4-4 The wave $y = sin(x + sin(x))$ and the accompanying value distribution. 30

4-5 Scatterplot of resampling values from the wave altered sine wave. . . 30

4-6 Histograms of the top and bottom 10% for the altered sine wave. . . . 31

4-7 Skewed sine wave and value density plot. 31

4-8 Skewed sine wave resampling scatter plot and top and bottom histogram. 32

4-9 Plot of two sine waves with overlapping value distributions 33

4-10 Period biased sine sampling histogram and mean probabilities. 33
4-11 Normalized Celsius readings from one city. X-axis represents average normalizaed sequence value; y-axis represents fraction of the dataset. 34
4-12 Scatterplot of resampling probabilities for one city's celsius readings. . 35
4-13 Top and bottom histograms showing rare and common temperatures. The dataset is roughly Gaussian, so the orange buckets represent identification of the most common datapoints and the blue buckets represents identification of the wings of the dataset. 35
4-14 Value distribution of normalized Celsius temperature readings in Vancouver and LA. 36
4-15 Skewed sine wave resampling scatter plot and top and bottom histogram. 36
4-16 The histogram of reconstruction losses shows a similar histogram to the latent feature resampling weights. Additionally, we see clearly that the model finds better reconstruction loss for the common Vancouver dataset than the minority LA dataset. 38
4-17 Histogram showing the captured top and bottom 10% of losses. The heights are adjusted by the portion of each bin that is represented. . . 39
4-18 The debiased training paradigm reaches lower loss more quickly than the uniform training paradigm due to the unsupervised selection of difficult training examples. 40
4-19 The debiased model achieves lower final loss than the uniform model due to the resampling regime. 41

5-1 A comparison of the latent-dimension probability aggregators, shown as a bar plot of the sampling weights over the first 1000 datapoints. On the left, we show the resampling weights when calculated by the max operator, and on the right we show the resampling weights when calculated by the exp-sum-log trick. 44

List of Tables

4.1 Confusion Matrix for separating Vancouver and LA. 37
4.2 Statistics for separating Vancouver and LA. 37

Chapter 1

Introduction

As machine learning (ML) models are increasingly incorporated into society, we have an obligation to ensure that individuals are treated fairly by models regardless of race, gender, age, wealth, or other axes of bias. Fairness in Artificial Intelligence (AI) has recently come to prime-time news due to researchers who work to ensure fairness as we deploy ML models into the real world. Unfortunately, many ML systems are simply trained and deployed without validation measures to check for bias or unfair treatment of protected characteristics like disability, religion, race, and other classes. There are many examples of organizations exposing ML models for bias, ranging from ProPublica's work with COMPAS [3] exposing racial bias in a tool used by the criminal justice system or [4] exposing how bias is encoded into large-scale language models.

Recent work has examined the bias in language generation and word-embeddings. In [11], researchers have shown bias through coreference resolution and conditional log-likelihood of generated sequences. For example, the phrase "The *doctor* ran because *he* is late" and the phrase "The *doctor* ran because *she* is late" show corefence scores between the italicized words of 5.08 and 1.99, respectively; that is, a trained coreference model finds it far more likely that the words "doctor" and "he" are referring to the same entity than the exact same setup for the words "doctor" and "she". In [6], researchers attempt to expose bias in word embeddings through examining the cosine similarity between specific word embeddings. These researchers find worrying

gender bias encoded in the model as the embeddings for "man" are far more similar to "captain", "skipper" and "architect" whereas the embeddings for "woman" are more similar to "hairdresser", "nanny", and "stylist".

Most bias identification and debiasing approaches are based on identifying worrying axes of bias – such as gender or race – and then determining methods of transforming trained, biased models into unbiased models against the identified axis. However, most of these approaches rely on matched pairs of words that exhibit both sides of some bias axis. Some examples are easy to think of: "male" and "female" identify the gender-bias axis. However, this raises the question: What word pairs identify the race bias axis? What words identify the culture bias axis? What word pairs identify the age bias axis? In order to attempt to debias a model against all kinds of bias inherent in datasets, we need to look towards unsupervised approaches for debiasing.

In contrast to debiasing other applications of machine learning, such as computer vision approaches failing to recognize pictures of dark-skinned or tilted heads at the same rate as light-skinned heads as shown in [2], language models have another dimension to consider: the time dimension. An image presents all of its features at once, whereas a time-series sequence like a sentence may present no bias until the final word. Further, we look to separate value-bias, bias based on the density of the distribution, and sequence-bias, bias based on the relationship between sequential tokens. The time-series dimension allows us to attempt to disentangle these two concepts and raises the question of what makes a sentence truly biased in the absence of labels: weird combinations of words not usually seen together? Grammar mistakes? Rare profane language? We look to explore and develop these definitions as well as algorithmic and experimental approaches for identifying and mitigating bias in time-series datasets.

We present a time-series model training paradigm based on [2], which we apply to time-series datasets such as sine, weather, and langue data, in addition to algorithmic improvements over the original authors implementation. We develop an end-to-end trained model for time-series prediction which utilizes a latent structure similar to a variational auto-encoder (VAE). In the process of learning the primary task of

time-series prediction, the latent space is forced to develop an understanding of the latent structure of the dataset in an unsupervised fashion. By examining the latent features of each datapoint, we can identify and debias against rare and common datapoints. We improve upon the algorithmic approach of [2], including experiments and algorithmic considerations in adapting to time series datasets. Additionally, we present a comparison of using different probability aggregators as well as methods of consolidating the time-series component of the latent dimension. We have shown the identification of bias in time-series datasets, as well as work in disentangling value-bias (dataset bias based on the value distribution of the input time series) as well as sequence-bias (dataset bias based on the relational distribution of the time series).

In our experimental results, we show that the debiased training paradigm can extend to time-series objectives through our new algorithmic methods. Not only does this training paradigm result in higher-accuracy and lower-loss models, we find that analysis of the resampling probabilities can actually act as a high-validity unsupervised binary classifier. Models trained with the debiased training paradigm result in higher accuracy for the rare sequences, which could be important in language, financial markets, or other timer-series objectives.

Following this introduction, we present related work in language models and debiasing algorithms, followed by our methodology, experimental results, algorithmic analysis, and the conclusion and future work. In this thesis, we present the following contributions:

- the adaptation of [2] to time-series objectives
- experiments on toy, weather, and language datasets
- an analysis of disentangling value-bias and sequence-bias
- an analysis of algorithmic improvements over [2]

Chapter 2

Related Work

In our related work chapter, we will discuss approaches to debiasing datasets and language models. Language models have come a long way since the inception of linguistics as a field; in the mid-1900s, researchers were focused on rule-based systems that tried to lay out massive series of grammar rules and syntax trees. The idea of n-grams, probabilities associated with sequences of words, dates back to [16], published in 1948. From there, natural language processing (NLP) research began to make use of machine learning techniques, giving rise to recurrent neural networks (RNNs) from [14] and [10]. These papers develop the idea of backpropogation-through-time, a gradient-based method for dealing with time-series. The concept of RNNs for time-series tasks was then furthered by the development of the Long Short-Term Memory cell, a recurrent unit that could 'remember' and 'forget' different representations over time, as put forth in [9]. Simultaneously, researchers were developing methods of representing word tokens as feature vectors, which were a much more useful representation for ML models. Bengio et al in [5] put forth the idea of simultaneously learning embedding vectors and a parameterized network for sequence prediction.

The use of word embeddings, especially utilizing pre-trained word embeddings, from GloVe [13] or Word2Vec [12], has enabled many applications to simply build a task on top of pre-trained word embeddings. However, many tasks still struggle to model long-term dependencies due to exploding or vanishing gradients in addition to simply learning connections between words that are far apart in the sequence. This

brings us to modern NLP, which is seemingly dominated by the attention mechanism put forth in [17], which is a building block in the Transformer architecture. While much progress has been made in chasing benchmarks and creating human-like text such as in [7], there has been little progress made in quantifying bias and even less progress in creating unbiased models. This remark from [4] highlights the many ways in which bias is encoded in language models: "Biases can be encoded in ways that form a continuum from subtle patterns like referring to *women doctors* as if *doctor* itself entails *not-woman* or referring to both genders excluding the possibility of non-binary gender identities, through directly contested framings (e.g. undocumented immigrants vs. illegal immigrants or illegals), to language that is widely recognized to be derogatory (e.g. racial slurs) yet still used by some." Clearly, bias in language models is a large scale problem that machine learning researchers should focus on mitigating.

2.1 Debiasing Datasets

Ideally, we could sample sequences from a text corpus in a fashion that selected the perfect mix and curation of sentences that exposed the model to no biases; however, this would be almost impossible due to the cost of identifying how every sentence exhibited bias along every axis, especially when we consider that we would like to generally debias against all existing sources of bias, some of which may be impossible to identify.

One approach to debiasing datasets is to augment datasets using counterfactuals as in [11], then continue with normal training taking advantage of the seemingly unbiased dataset. The approach of Counterfactual Data Augmentation (CDA) is to use a dictionary of paired words along some axis such as gender and augment an existing dataset by replacing some occurences with the opposite word. The authors argue that these "interventions" are effective in reducing bias while maintaining accuracy. However, it may be difficult to adapt these counterfactuals to other tasks such as summarization or inference swapping pronouns could lead to false or confusing state-

ments. Additionally, it may be difficult to imagine dictionaries of word pairs for other bias axes, such as race or age. This approach is inherently limited by the dictionaries that can be produced and thus cannot scale to eliminating all biases.

In other domains, attempts at sampling against specified class imbalances have shown improvements in model fairness; however, this requires labelling of every datapoint and segmentation of datapoints into separate categories, which appears very difficult for language and time-series datasets and tasks. We cannot assume that we will have labels that perfectly encapsulate the input feature bias; as such, we will focus on algorithmic attempts to mitigate bias in language models, as discussed below.

2.2 Debiasing Algorithms

Most effective debiasing algorithms seem to fall into the category of training a model then transforming it into some unbiased state. We examine one such method, based on projecting parts of a learned system into a subspace that is bias-neutral. The authors of [6] look to expose the bias in word embeddings, vector representations of words that are used to transform word tokens into vectors with multidimensional meaning first proposed in [5]. In many applications, word embeddings are learned from data then used in other tasks, meaning that bias from the original train set could be passed into other models and tasks. These researchers find unnecessary alignment between the embedding vectors with specific questionable axes, such as the *she-he* axis, with alignment defined as the cosine similarity between embedding vectors. The researchers found the the occupations "homemaker", "nurse", and "receptionist" were the most aligned with the word "she" while the occupations "maestro", "skipper", and "protege" were most aligned with the "he" vector. However, all of these occupations have gender neutral meaning and should be fairly aligned between the "she" and "he" embeddings.

Our debiasing training paradigm trains unbiased models from biased datasets. Approaches to debiasing are sometimes conflated with interpretability and explainability work. The interpretability of a model is typically defined by how well the users

can understand the inputs and outputs as cause and effect. A model is interpretable if the user knows how to cause changes in the output by changing the input. On the other hand, explainability is typically defined as the user knowing the importance of individual weights and what they represent to the overall model's performance. A model is explainable if the user can understand specific nodes weights and what those weights mean for the input.

One recent paper takes the idea of *interpretability* even further by defining a methodology of "model debugging" [18]: by separating a model into a "deep feature extractor" and a "linear decision layer", the authors propose a structure to interpret ML models by "probing how deep features are combined by the decision layer to make predictions" [18]. The authors use this interpretability tool the analyze the bias found in language and vision tasks. However, while this approach suggest a way to understand the bias found in datasets, it does not take any step to either counter the bias in the model or train a debiased model. A similar paper presents an editable approach where users can directly modify the weights of a model utilizing generative representations adapting the inputs [15]. However, it appears that the user still needs to directly make the changes for all the edits they desire to make; for example, the authors show a typographic attack where they place the text "iPod" into an image; their defense against such attack is to edit the model's prediction to map the text "iPod" to "blank". While this is a successful approach, it would be difficult to make this edit or transformation for every single example or every axis of bias that the users desire.

The same researchers in [6] also present the idea of identifying directions in the embedding space by combining representative word pair differences. The paper focuses on the *gender direction*, which is captured by combining the difference between "she" and "he" as well as "woman" and "man", amongst other pairs. They use this *gender direction* to compute a *gender subspace*, found by using PCA on the representative vector pairs. They then describe a debiasing algorithm which projects all word embeddings onto the *gender subspace*, effectively neutralizing gender bias by ensuring that gender-neutral words are zero in the *gender subspace*. While this approach

showed experimental improvements, it required identification of gender-neutral words, gendered word-pairs, and only minimized one axis of bias: male-female gender bias. It is difficult to scale this approach to other axes of bias as well as to continue to find labelled sources of bias for more vague concepts. While this approach is helpful, we propose a more scalable solution that does not rely on human labelling of the bias.

Chapter 3

Methodology

3.1 Latent Debiasing

In [2], the authors use a VAE to simultaneously learn the latent data strcture in an unsupervised fashion as well as the primary supervised prediction task. The encoder learns to approximate $q_\phi(z|x)$, which is the true distribution of the latent variables in the dataset. The authors describe a VAE with k latent variables and d output variables, creating a structure of an encoder producing $2k + d$ output latent features. The $2k$ outputs are in order to parameterize k Gaussian distributions for the latent space from which z is sampled while the d output features create the prediction $\hat{y}$. The decoder structure reconstructs the input by approximating $p_\theta(x|z)$, where z is sampled as $z = \mu(x) + \Sigma^{\frac{1}{2}}(x) * \epsilon$, where $\epsilon \sim N(0, I)$. The output of the decoder is the reconstruction of the input, forcing supervision over the latent variables but without handcrafted labels. The authors adapt a facial recognition task and a sum of loss functions: the Kullback-Liebler divergence loss over the latent variables; a p-norm reconstruction loss between the input and reconstructed output; and a crossentropy binary classification loss. This structure informs our methodology, as presented later. The algorithmic diagram for [2] is presented below in 3-1.

With this VAE structure, the authors of [2] describe a debiasing algorithm that utilizes histograms over the latent space to discover rare datapoints. Generally, the latent distributions are used to determine which points are rare and should be sampled

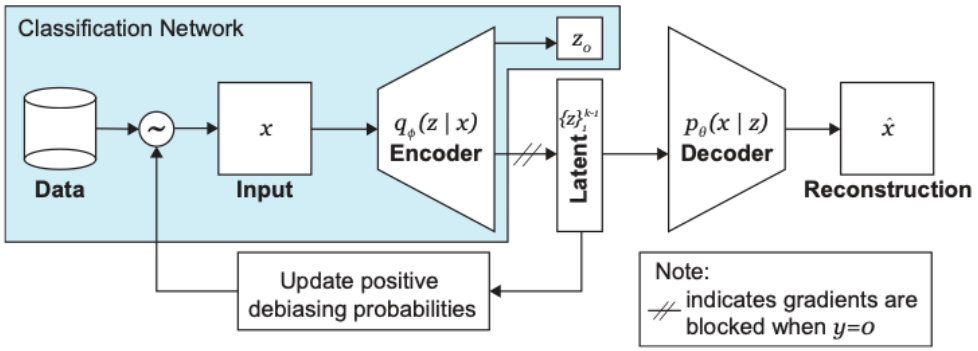

Figure 3-1: The algorithmic diagram from [2]. We adapt and improve this algorithm to time-series datasets and other algorithimc improvements.

with a higher probability and which points are common and should be sampled at a lower probability for training, resulting in an effectively unbiased dataset from the model's point of view. The algorithm describes encoding the entire dataset, resulting in an estimation of the latent distribution: $Q(z|X)$. In an effort to determine the likelihood of each point, the algorithm creates bucketed histograms of each dimension in the latent distribution as an approximation of the true joint distribution, calculated as:

$$\hat{Q}(z|X) \propto \prod_i \hat{Q}_i(z_i|X) \tag{3.1}$$

Using these frequency distributions, we can estimate the sampling weight of each datapoint, represented as:

$$W(z(x)|X) \propto \prod \frac{1}{\hat{Q}_i(z_i(x)|X) + \alpha} \tag{3.2}$$

where α is a tunable debiasing parameter that forces a smoothing of the sampling weights. When the debiasing parameter α goes to 0, we see uniform sampling inverse to the latent distribution; that is, pure debiasing sampling. As α increases, we see smoothing of the sampling weights leading to uniform sampling over the train dataset; that is, normal uniform sampling over the dataset. The authors show that this approach allows debiasing against characteristics present in the facial recogni-

tion dataset without labels or supervision, showing increases in performance against biased categories like race and head tilt. This approach effectively eliminates bias in datasets by the relative sampling of varying biased and unbiased points, resulting in an unbiased model despite training from a biased dataset.

3.2 Our Method

We look to adapt and improve the framework presented in [2] to time-series datasets and language tasks due to its unsupervised approach to eliminating bias. By adding the time dimension, as opposed to just an image, we attempt to disentangle and debias against two kinds of bias: value-bias and sequence-bias. In time series datasets, the value bias is identifying sequences with values that are unusual; sequence-bias is identifying sequences with unusual relationships. In language datasets, value bias may be sequences with weird or rare inputs while sequence-bias may describe rare relationships between words or meanings of sentences. These notions are difficult to separate but may be vital to truly debiasing language models. The additional time-series dimension adds an additional axis of complexity but also potentially a new avenue for the interpretability of language models with respect to bias.

We adapt [2] to time-series by examining three methods of consolidating the time-series dimension of the latent distributions: sum, mean, and the last index. These approaches reduce the time-dimension of the latent space in order to apply the debiasing approach shown above. In addition, we show an algorithmic improvement over [2] by showing the difference in the probability aggregators during the sampling weight scheme. In the formula, the frequency distribution probabilities are aggregated using the product operator; however, due to underflow errors, the authors used a max operator in their implementation. We show a comparison in using the max operator and avoiding underflow errors by taking the exponent of the sum of the log of the probabilities; that is, for the datapoint p^i with latent dimension of size K, the difference between aggregating probabilities as:

$$max(p_0^i, p_1^i, ...p_K^i) \tag{3.3}$$

and

$$\exp\left(\sum_{k=0}^{K} log(p_k^i)\right) \tag{3.4}$$

Intuitively, these two aggregation methods align with different ideas: the max operator would focus just on which value was the largest, leading to one latent dimension dominating the sampling weight for each datapoint respectively, whereas the exp-sum-log trick provides a true product over all of the latent dimensions.

We can visualize the differences between the algorithmic approaches as follows. First, if we consider one datapoint's latent vector, which is presented as shape [*sequence dimension T, latent dimension K*], we can examine the different methods of the sequence-dimension consolidation to fit the debiasing training paradigm. First, we show an example datapoint's latent vector, shown as time-dimension by latent-dimension latent vector in 3-2.

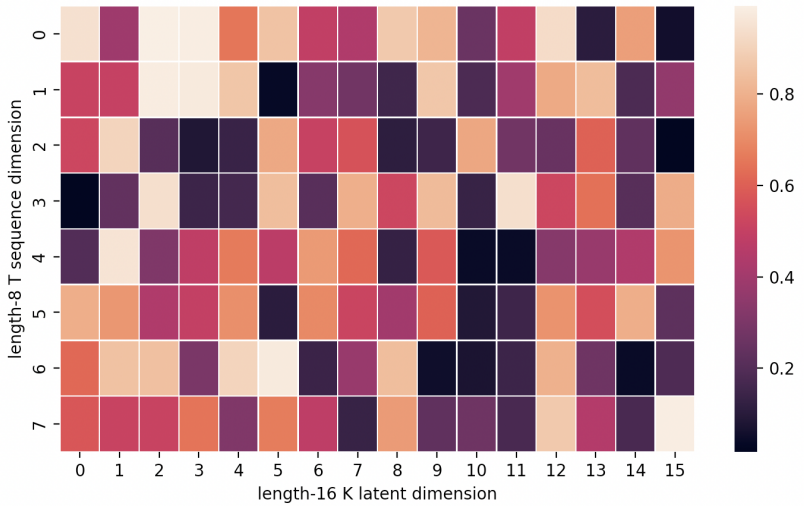

Figure 3-2: A representation of a given datapoint's latent vector as time-dimension T by latent dimension K.

With this latent vector, we can show the three approaches for time-series consolidation: sum, mean, and last. Sum is a sum over the time-steps; mean is a mean

over the time-steps; and last is an index into the last time-step. The impacts of these approaches are shown in 3-3

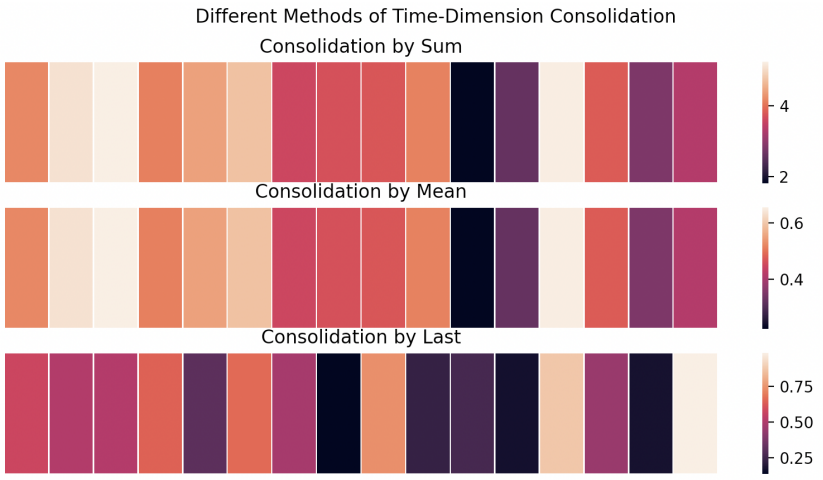

Figure 3-3: Consolidation methods of the time-series dimension over sum, mean, and last. Note that sum and mean are scalar multiples with the colors normalized to have the same effect.

Following the time-series consolidation, we show the impacts of different latent-dimension consolidation methods, max and product. In [2], the authors formulate the resampling algorithm as a product; however, in their implementation, they use the max operator to overcome underflow errors in the product approach. We add the product approach by utilizing the exp-sum-log trick, where products of small numbers can be found by the exponentiation of the sum of the logs of the small numbers. This time, in 3-4, we show a batch of latent-dimension vectors which have already had their time-series dimension consolidated by the last method. 3-4 shows the batch-dimension by latent-dimension heatmap of 32 latent vectors.

Given a batch of latent vectors, we can examine the different latent-dimension consolidation methods, max and product. The consolidations of these latent-dimension vectors combine into the weights used to predict resampling probabilities. In 3-5, we show the difference between the max and product approaches.

We explore the differences between these approaches later in this thesis. It is useful to see that the product operator produces a more helpful histogram of outputs, as we expect to see many similar values and a few different values, whereas the max

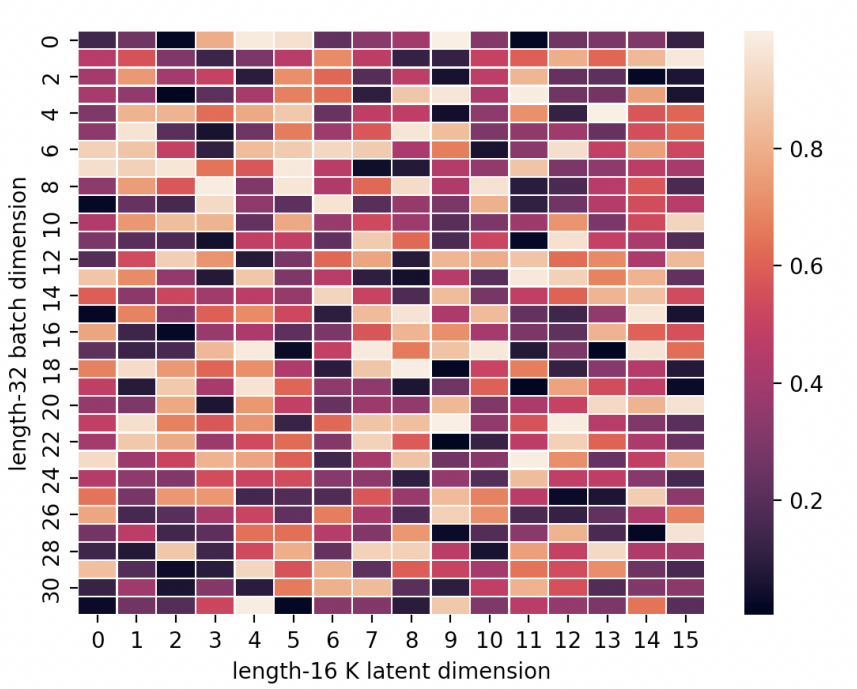

Figure 3-4: A representation of a batch of datapoints's latent vectors, with their time-dimension already consolidated. Presented as batch-dimension B by latent-dimension K.

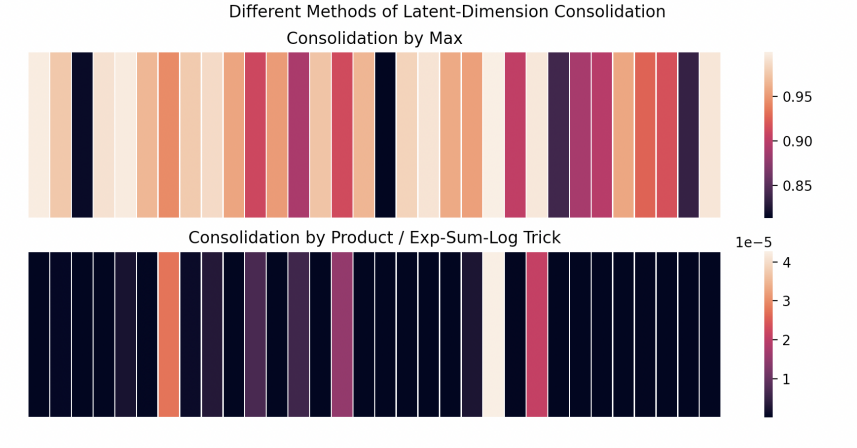

Figure 3-5: A representation of the different latent-dimension consolidation methods as given by the max and product operators.

operator produces a more uniform distribution.

Chapter 4

Experimental Results

Our work has shown positive signals towards identifying bias in time-series datasets. Below, we show results from both toy sine-wave datasets, weather prediction datasets, and language tasks. These small-scale and relatively simple datasets inform our debiasing approach to time-series datasets while larger-scale experimentation with language models verify theoretical hypotheses. While the toy datasets do not display the same level of complexity as language, they are useful experiments to begin determining which metrics, charts, and graphs are helpful at capturing and displaying model performance and bias. In addition, these experiments help to determine methods of separating value-bias and sequence-bias in simpler environments, which enable us to apply the same approaches to language models. These experiments are conducted with an LSTM network learning a sequence-prediction task, effectively enabling us to learn the latent distribution of the dataset without any labels – only those features provided by the data.

4.1 Experiments with Sine-Wave Datasets

We began experimenting with toy sine-wave datasets in order to experimentally validate both the code and the effectiveness of the debiasing algorithm against time-series datasets. We show the value distribution of multiple sine and sine-like waves, accompanied by the resampling probabilities associated with those sequences. In order to

debias against value-bias, we expect to see the opposite of the value distribution reflected in the resampling probabilities. Additionally, we show the top 10% and bottom 10% predicted by the models, which should reflect the most rare and most common points.

4.1.1 Sine Wave Dataset

We demonstrate that the sine wave dataset exhibits high value-bias following $y = sin(x)$. That is, when the time series is broken into short sequences as inputs for the model, we can determine that the mean value of those sequences are not uniform and thus invite bias into the model's predictions. The wave and value density distribution are shown in Figure 4-1.

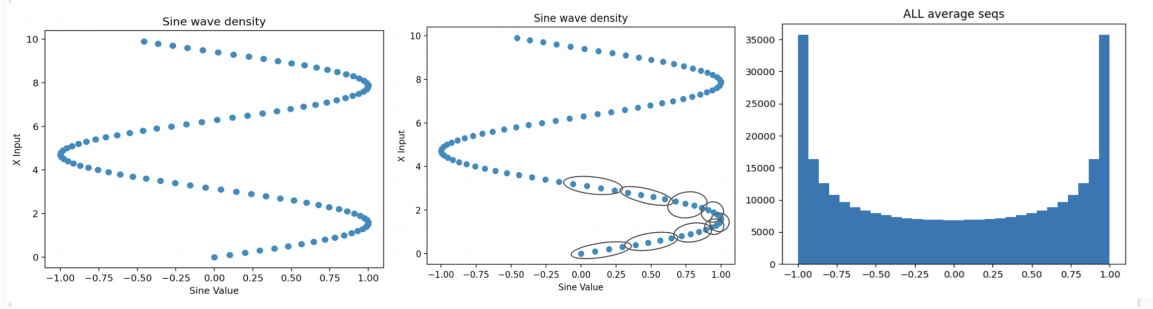

Figure 4-1: Value Density of a Sine Wave

In order to debias against this distribution, we need to sample inverse to the value density distribution. That is, given the parabola of the value density, we want to find a negative parabola describing the opposite distribution. After training the models, we can sample the latent space to find the sampling probability of scattered points on the x-axis. In Figure 4-2, we see that the sampling distributions peak around 0 and are lowest on the outside of the range, exactly inverse to the value distribution which is lowest at 0 and peaks on the outside edges of the range.

In Figure 4-3 we show histograms showing the top and bottom 10% of sampling weights, which confirms that the models find the center points to be the most rare and the edge points to be the most common.

These results from the toy sine wave dataset shows that we can identify value-bias

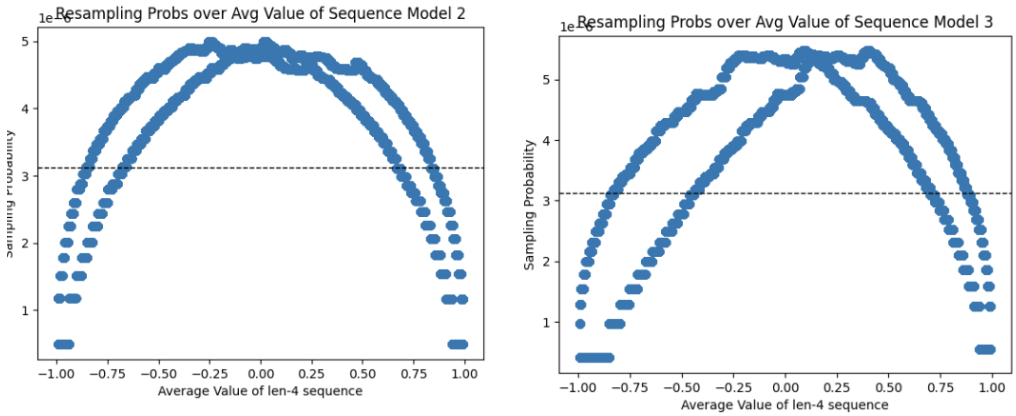

Figure 4-2: Scatterplot of sampling probabilities from models trained on a sine wave

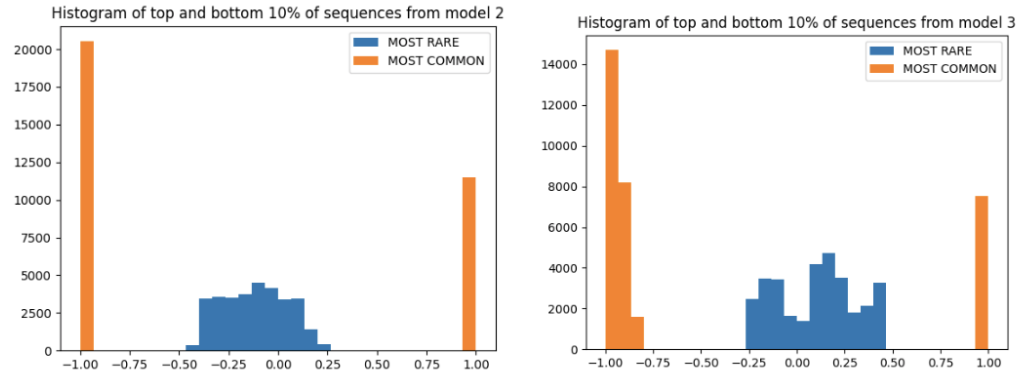

Figure 4-3: Histograms of top and bottom 10% of sequences from models trained on a sine wave.

in time-series forecasting without any supervision or labels. Next, we will show that the algorithm performs well on altered sine wave datasets in order to explore the entaglement of value-bias and sequence-bias.

4.1.2 Biased Sine Dataset

We show the same experiments on a slightly different sine wave dataset: $y = sin(x + sin(x))$. While the range of values is the same, there is a very different value distribution, as you can see in Figure 4-4. To correctly debias against this value distribution, we would expect to see a bimodal distribution peak at -0.5 and +0.5, somewhat like an "M" shape.

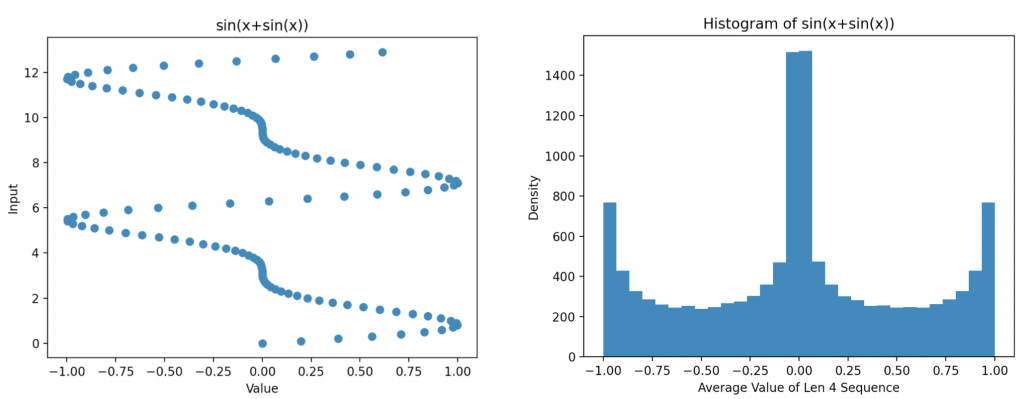

Figure 4-4: The wave $y = sin(x + sin(x))$ and the accompanying value distribution.

If we examine some of the trained models in Figure 4-5, we can see positive identification of the two modes of rare values near -0.5 and 0.5 as well as identification of the primary value mode at 0.0. This is roughly the inverse of the data distribution.

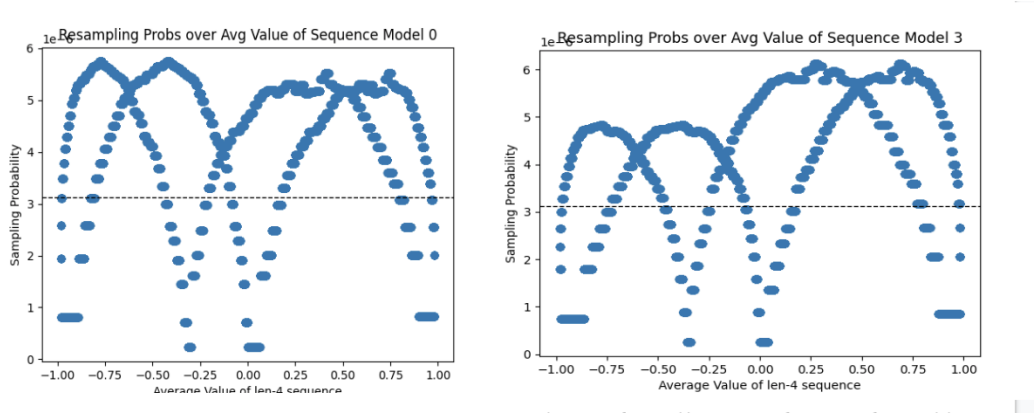

Figure 4-5: Scatterplot of resampling values from the wave altered sine wave.

Finally, we can show that the top and bottom 10% of values are identified by the model in Figure 4-6. While the bottom 10% are correctly identified as in the center, the models seem to struggle to capture both modes of rare sequences.

Our last sine-wave value-bias based experiment will be to examine an asymmetrical value distribution..

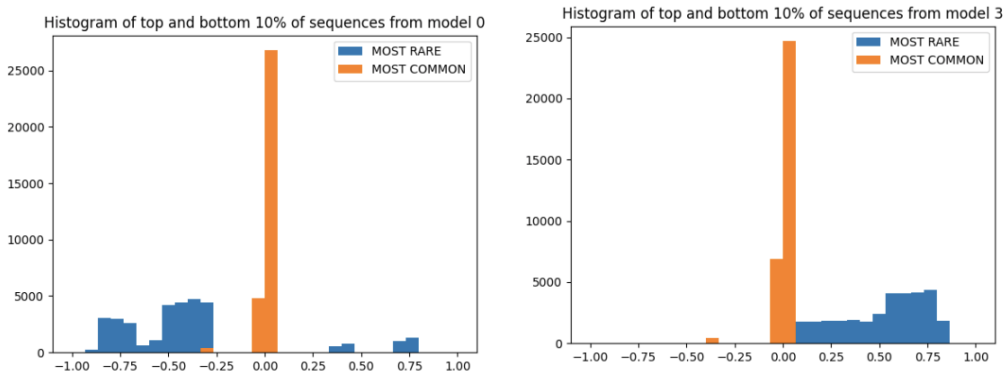

Figure 4-6: Histograms of the top and bottom 10% for the altered sine wave.

4.1.3 Skewed Sine Dataset

The skewed sine dataset will repeat three times on the negative side of the x-axis for every one time on the positive side of the y-axis, resulting in a value density heavily weighted towards negative values, as shown in Figure 4-7.

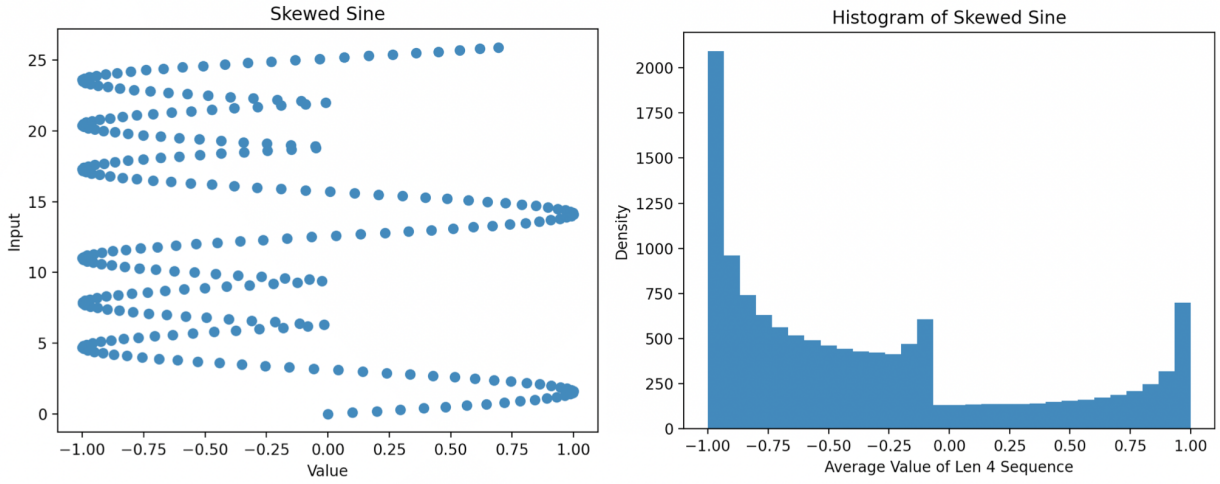

Figure 4-7: Skewed sine wave and value density plot.

This asymmetric value density presents a new distribution for the latent variables of our model to learn. It is more noisy, but the model still appears to learn the latent distribution and find the correct resampling probabilities, as seen in Figure 4-8.

These experiments show that we can understand the latent features of the dataset and debias accordingly towards the value distribution. In the next section, we show

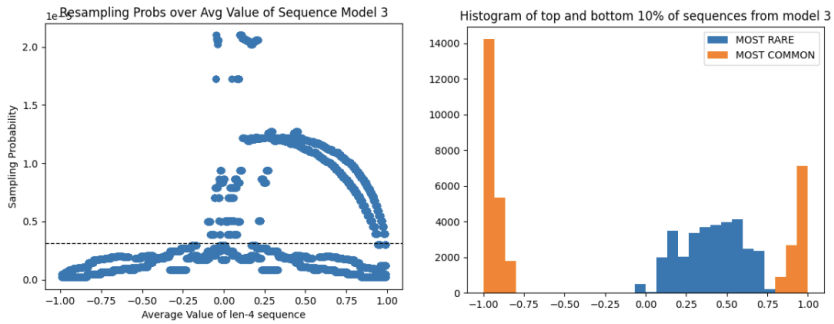

Figure 4-8: Skewed sine wave resampling scatter plot and top and bottom histogram.

one way to judge sequence-bias by utilizing two functions with entirely overlapping value-densities.

4.1.4 Period Biased Sine Dataset

In order to see sequence-bias by explicitly eliminating value-bias, we need to use two functions with perfectly overlapping value densities. In order to accomplish this, we construct a dataset of 90% samples of the wave $y = sin(\frac{\pi * x}{16})$ and of 10% samples of the wave $y = sin(\frac{\pi * x}{8})$ and make the sequence length equal to 32; that is, one full period for 90% of the data and two full periods for 10% of the data. This ensures that the value distributions are the same, as you can can see in Figure 4-9.

As these functions have overlapping value densities, any differences picked up by the model must be sequence-level bias. As you can see in Figure 4-10, the model still finds differences in the two portions of the datasets. In Figure 4-10, you can see the difference in sampling probability, with all of the normal data beneath the true uniform sampling probability, and all of the modulated data above the uniform sampling probability. Likewise, the mean value of sampling any of the normal data is well below the dashed line representing the uniform sampling probability while the mean value of sampling the modulated data is well above the dashed line.

The graphs in Figure 4-10 clearly show that the model is able to distinguish between the two types of data despite there being no difference in value-bias, meaning that the model is able to pick up and identify sequence-bias. Now that we have shown

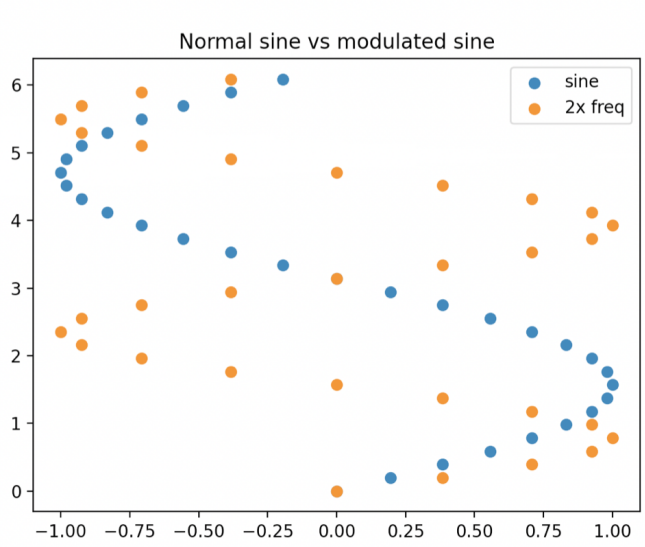

Figure 4-9: Plot of two sine waves with overlapping value distributions

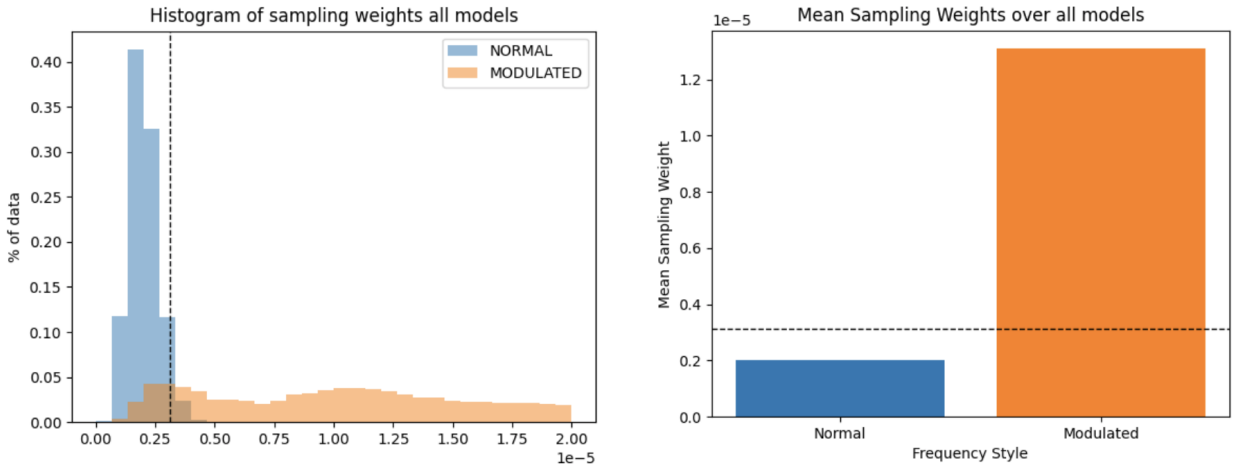

Figure 4-10: Period biased sine sampling histogram and mean probabilities.

results on toy datasets, we move to real-world time-series datasets.

4.2 Experiments with Weather Forecasting Datasets

4.2.1 One City, Normalized Celsius

We run a series of similar experiments on the weather forecasting datasets. First, we show that models can learn the latent value distribution of a normalized Celsius dataset. Figure 4-11 shows the value histogram, which invites a positive parabola for the resampling probability.

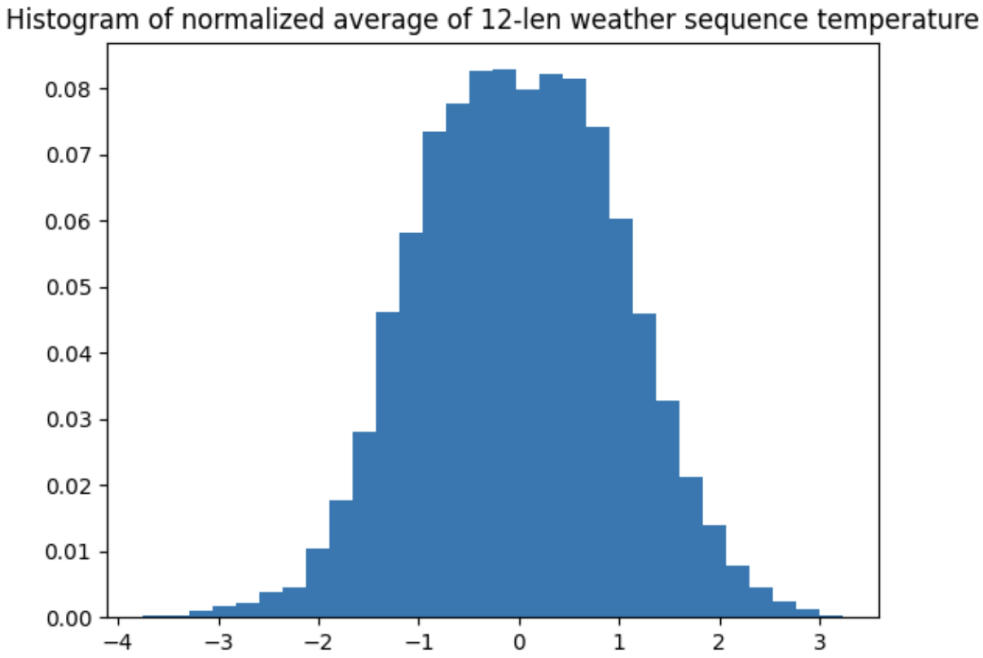

Figure 4-11: Normalized Celsius readings from one city. X-axis represents average normalizaed sequence value; y-axis represents fraction of the dataset.

Our trained models find the latent distribution and identify the fact that the common data is in the middle while there is more rare data on the outsides, as seen in Figure 4-12. However, we see some denigration of the model's ability to classify the truly rare outside edges.

Similarly, in the top and bottom histograms in Figure 4-13, we see identification of the center and the top and bottom quartiles, but the true edges of the distribution are not seen are particularly rare. While we do not know why that is the case, we

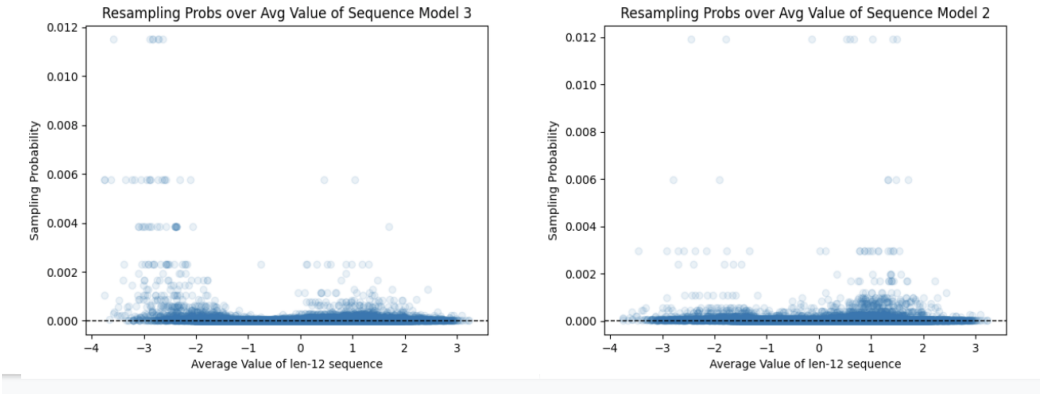

Figure 4-12: Scatterplot of resampling probabilities for one city's celsius readings.

suppose that the data may be too rare or too unlikely to fit well into our smooth latent space. Data outside the input domain of the VAE has little guarantee to be projected into reasonable latent space.

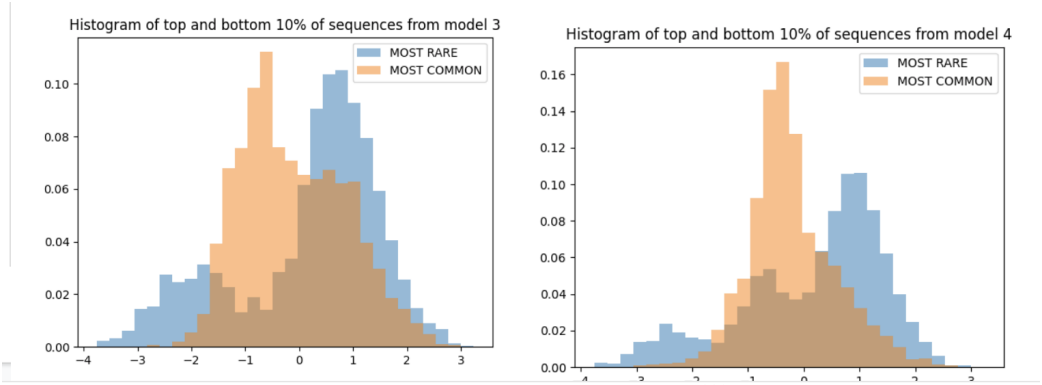

Figure 4-13: Top and bottom histograms showing rare and common temperatures. The dataset is roughly Gaussian, so the orange buckets represent identification of the most common datapoints and the blue buckets represents identification of the wings of the dataset.

4.2.2 Mixed Cities Normalized Celsius Datasets

Having seen that our algorithm can determine classification based on value-density, we look to see if the trained model's latent space can separate data taken from different cities. For example, a histogram of LA's and Vancouver's normalized Celsius sequences are shown in Figure 4-14, both as an even mixture (left) and as a mixture of 90% Vancouver and 10% LA (right).

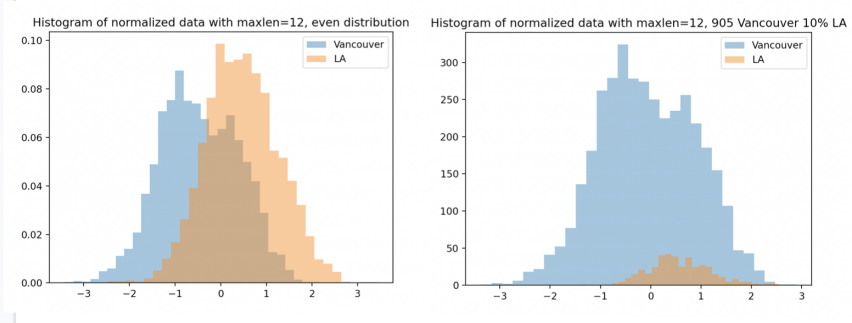

Figure 4-14: Value distribution of normalized Celsius temperature readings in Vancouver and LA.

Since the value distributions of the heavily skewed 90/10 datasets are entirely overlapping, we can say that the differences found by the models may be from sequence-bias instead of value-bias. After training on this mixed dataset, we see plausible identification of rare datapoints and common datapoints that follow the sequence-density separation between Vancouver and LA in Figure 4-15.

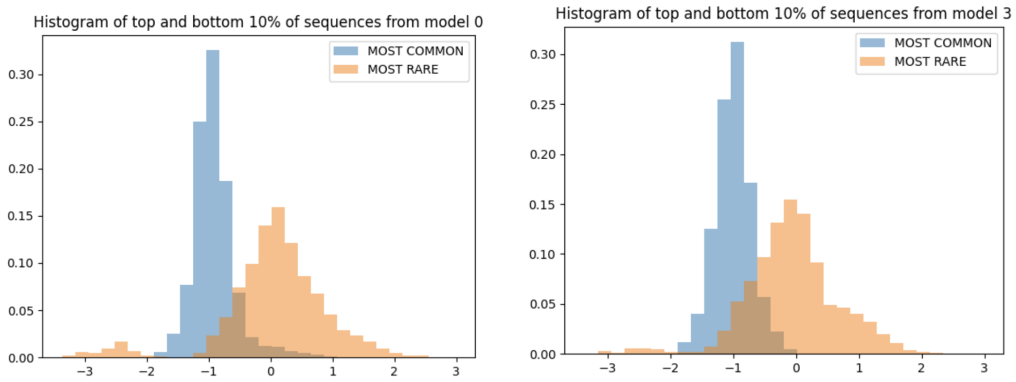

Figure 4-15: Skewed sine wave resampling scatter plot and top and bottom histogram.

In Figure 4-15, we see that the models can effectively act as classifiers using their unsupervised latent space as a separator. If we mix two cities weather data, such as Vancouver and LA as demonstrated above in Figure 4-14, with a mixing ratio of 90% Vancouver and 10% LA, and then examine the model's selection of top 10% most rare and bottom 90% most common as a binary classifier, we can score the effectiveness of the model at truly understanding and separating the two datasets. Given a test dataset of 1000 points (meaning 900 Vancouver, 100 LA), we display the resulting

accuracies as a confusion matrix in Table 4.1, along with accompanying metrics in Table 4.2.

	Predicted Rare	Predicted Common
Actually Rare	71 (True Positive)	29 (False Negative)
Actually Common	29 (False Positive)	871 (True Negative)

Table 4.1: Confusion Matrix for separating Vancouver and LA.

	Worst Possible	**This Model**	Best Possible
Sensitivity	0%	**71%**	100%
Specificity	88.8%	**96.7%**	100%
Precision	0%	**71%**	100%
Accuracy	80%	**94.2%**	100%

Table 4.2: Statistics for separating Vancouver and LA.

It should be noted that due to the class imbalances and the artificial classification of the top 10% and bottom 90%, the worst possible metrics for some categories are not that low, as seen in the 'Worst Possible' column of Table 4.2. Due to the vastly overlapping value densities between LA and Vancouver, we can assume that the predictive capacity of the model is due to differences in sequence-level bias.

4.2.3 Experiments with Reconstruction Loss

In addition to utilizing the latent features of the dataset, we also examine replacing the bias estimation with the reconstruction loss. This is powerful as models would need to be adapted to produce a latent vector whereas we can create our append reconstruction models to any given model by cutting off the final layers. Reconstruction loss as a predictor for bias follows a similar formulation as the bias estimation constructed off the latent features. We use the same histogram approach but input the reconstruction loss, essentially acting as though the reconstruction loss is a one-dimensional latent vector. Using the same skewed dataset, with 90% Vancouver and 10% LA weather data, we show that reconstruction loss is also a helpful predictor

and separator of the datasets, although less meaningful than the latent feature bias estimation. We show results in 4-16.

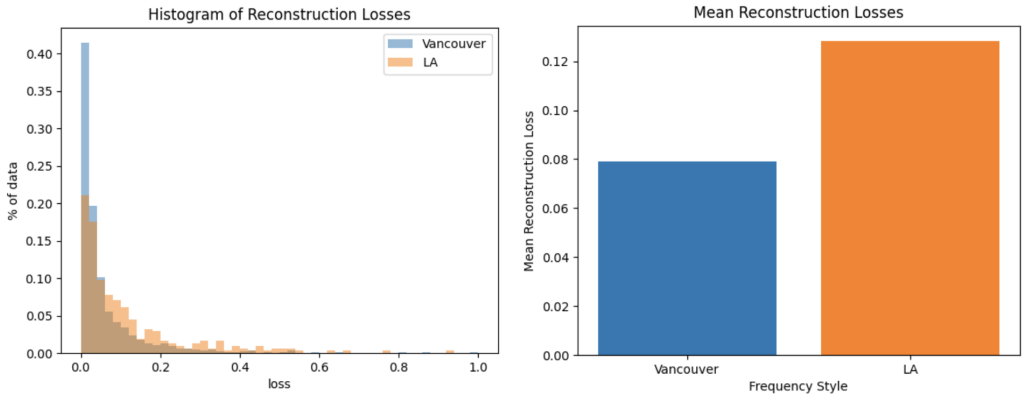

Figure 4-16: The histogram of reconstruction losses shows a similar histogram to the latent feature resampling weights. Additionally, we see clearly that the model finds better reconstruction loss for the common Vancouver dataset than the minority LA dataset.

In 4-16, we show that the histogram of reconstruction losses is very similar to the histogram of resampling weights from the latent feature bias resampling weights. This suggests we can achieve similar objectives utilizing the reconstruction loss as the bias estimation latent vector. Likewise, the reconstruction loss average for Vancouver is far lower than the reconstruction loss average for LA, again suggesting we can utilize the reconstruction loss as a helpful metric for bias estimation.

Additionally, we show that the value-bias and sequence-bias can both be explained by a plot. While the histogram of sequences by loss has been shown in prior portions of this paper, we this time adjust the heights of each bar to show amount of each bin captured. This corrects for the fact that there is little data on the wings of the dataset; if there is only one data point in a given bin and it is captured in the top or bottom 10% histogram, that should show a height of 1 and not a small fraction of the entire dataset. In 4-17, we show that the reconstruction losses capture significant portions of the center of the dataset as the lowest reconstruction losses and also significant portions of the wings of the dataset as the highest reconstruction losses.

In 4-17, we show that reconstruction losses capture the density-bias without regard for the sequential-bias, as there is no separation between the two portions of the

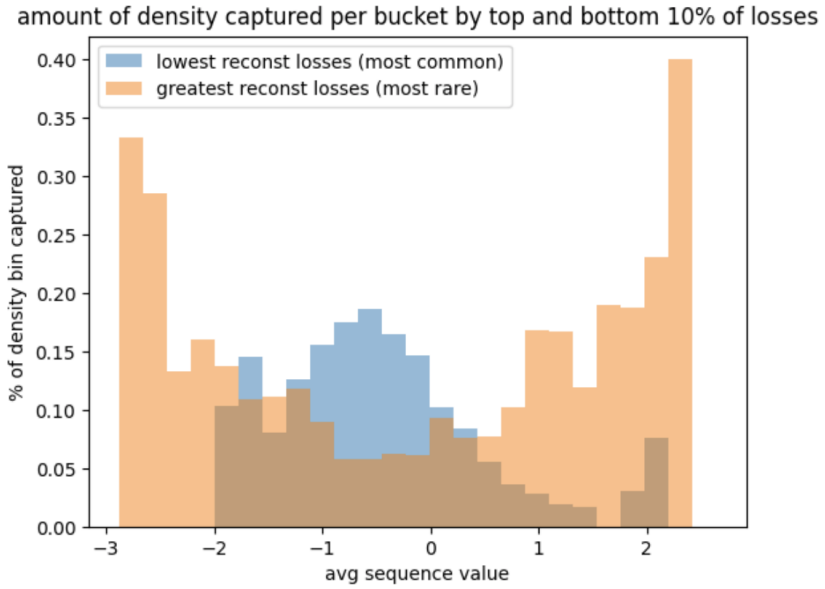

Figure 4-17: Histogram showing the captured top and bottom 10% of losses. The heights are adjusted by the portion of each bin that is represented.

datasets.

4.3 Experiments with Language Datasets

When shifting experiments from toy and 1-dimensional datasets, it becomes very difficult to capture concepts like value- and sequence-density. However, we can still show metrics based on accuracy and loss to compare biased and debiased models. We conduct self-supervised training on next-word prediction on the Corpus of Contemporary American English (COCA) Dataset [8].

We show increased train and test accuracy from using the debiased paradigm, both in terms of final accuracy of the converged model and in terms of the efficiency in terms of number of epochs that accuracy levels are reached. This occurs because the debiased training paradigm effectively implements unsupervised hard-negative mining, an approach close to active learning. In our approach, easy common examples produce latent features close to the Gaussian normals. If there are many easy common examples, their histogram bucket will have many examples included, thus

resulting in low resampling weights, effectively eliminating the easy examples from the training data in terms of expected sampling probability. Likewise, the difficult training examples product latent features far from the Gaussian normals, resulting in sparse histogram buckets and high resampling weights. This forces the training paradigm to predominantly sample difficult examples, which increases the training efficiency of the model.

We evaluate these experiments by training a model using a normal uniform training paradigms and a model using our debiased training paradigm on the same dataset for the same number of epochs. In 4-18, we show the difference between a uniform training paradigm and a debiased training paradigm to compare the speed-up in the decrease of train loss between the two approaches.

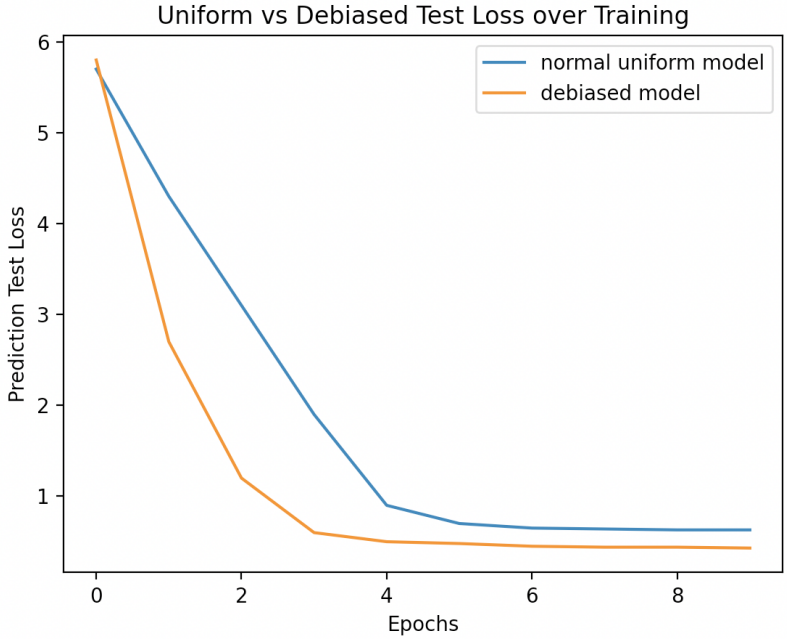

Figure 4-18: The debiased training paradigm reaches lower loss more quickly than the uniform training paradigm due to the unsupervised selection of difficult training examples.

We can also show the experimental different in final test loss between the two paradigms. Through upsampling of the difficult (or biased) datapoints, the model's increased repeated exposure to underrepresented datapoints results in higher test accuracy for the debiased model. In 4-19, we show that the debiased training paradigm

reaches significantly lower test loss than the uniform model on the COCA dataset.

We also examine the most-biased samples for the given dataset. However, it is difficult to attach semantic meaning to these samples, as they appear to be mostly indecipherable errors in the datasets or errors in the data collecting and tokenization process.

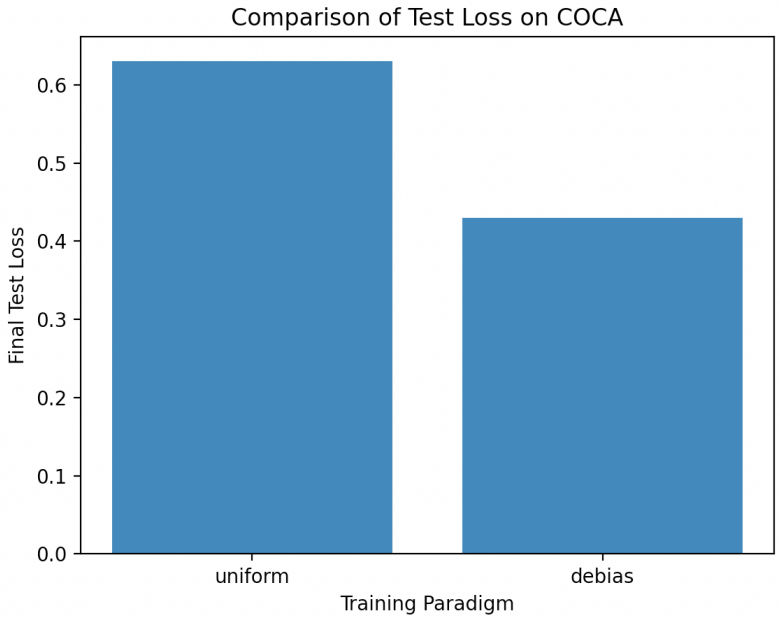

Figure 4-19: The debiased model achieves lower final loss than the uniform model due to the resampling regime.

Chapter 5

Algorithmic Results

We also consider multiple algorithmic changes to the algorithms produced by [2]. These algorithmic improvements fall into two domains: the latent-dimension probability aggregation style and the time-series dimension latent feature aggregation. The latent features are presented as batches tensors of shape [*batch dimension, sequence length, latent dimension*]. Both approaches deal with condensing one datapoint's latent vector into its given resampling weight. The latent-dimension probability aggregation deals with combining all of the k latent dimension probabilities (the last index of the shape) whereas the time-series dimension latent feature aggregation deals with combining the successive timesteps of the latent vectors (the second to last index of the shape).

5.1 Latent Dimension Probability Aggregation

In the original paper by [2], the authors formulate the latent-dimension probability aggregation as a product over the probability in each dimension for a given datapoint. That is, for a given datapoint p with latent vector of length K, the weight is calculated as $w = \prod_{k=0}^{K} p^k$. However, due to the underflow errors that result from multiplying many small probabilities, the authors used the max operator, redefining the equation as $w = max(p^0, p^1, ...p^K)$. However, this technique may introduce unintended bias into the sampling formulation as each datapoint's weight is calculated off of just the

largest probability instead of taking into account each aspect of the latent vector. We experiment adjusting between the two techniques, with the product technique made possible by the exp-sum-log trick: finding the product of small numbers by taking the exponent of the sum of the log of the small numbers. We formulate the exp-sum-log trick as $w = \exp\left(\sum_{k=0}^{K} log(p^k)\right)$.

We find that the max operator may introduce a normalizing effect over the latent-dimension probability aggregation phase. We examine the first 1000 latent vectors of a dataset and their returned weights when sampled by the max approach and the exp-sum-log trick and observe that the max operator tends to force the sampling weights towards the uniform sampling weight while the exp-sum-log trick produces much more diverse weights. This is demonstrated in 5-1.

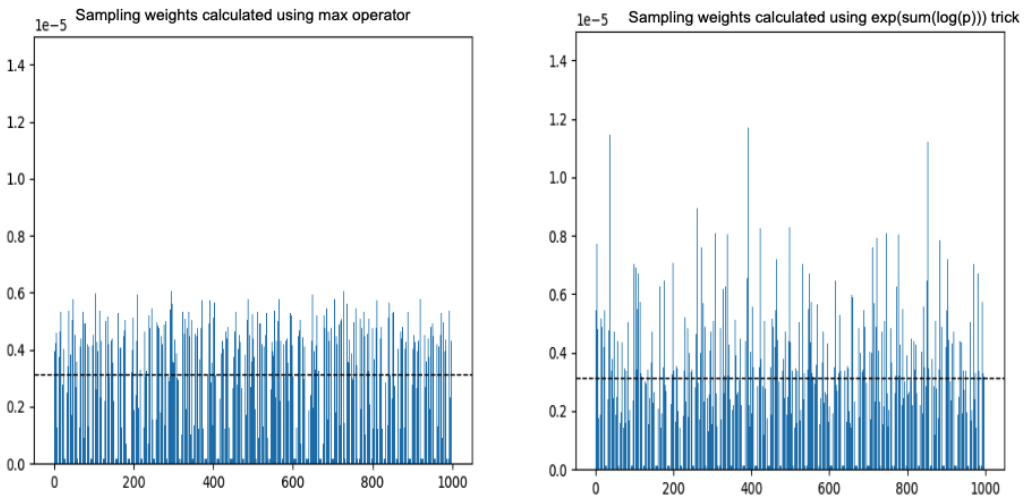

Figure 5-1: A comparison of the latent-dimension probability aggregators, shown as a bar plot of the sampling weights over the first 1000 datapoints. On the left, we show the resampling weights when calculated by the max operator, and on the right we show the resampling weights when calculated by the exp-sum-log trick.

5.2 Time-Series Dimension Latent Feature Aggregation

In order to utilize the histogram based debiasing paradigm, we need to compress the time dimension from the latent vector shapes. As such, we propose and experiment over three methods for consolidating the *sequence length* dimension: mean, sum, and last. The *mean* approach would take over the vectors; the *sum* approach would compress the time dimension by summing the vectors; and the *last* approach just utilizes the last latent vector which should hold all the latent features for the entire input. We find little difference in using each of the approaches and use last for all experiments. This is interesting yet sensible, as mean and sum are effectively scalar multiples of one another, so differences between the two approaches should be minimal unless accounting for magnitude. Likewise, the last timestep of the latent dimension should encode all of the features found in the input and thus hold all the information that would be found by taking the sum or mean of the timesteps.

5.3 Discussion of Algorithmic Results

We show a difference between the max approach and the product approach, but find little difference in the experimental methods for the time-dimension aggregation. However, there may be interesting future work in this space to account for differences between each vector at each timestep. We suggest an attention-based mechanism over the entire sequence which may provide insights into sequence-bias, but leave it as future work. We believe that comparing latent vectors on a timestep-by-timestep basis may produce insights into sequence-bias because this would capture when the n^{th} timestep is different or unexpected from the first $n-1$ inputs. This may have large impacts on time-series tasks such as financial markets, where the identification of a change or bias must occur immediately and not after a set sequence has been found. Likewise, in language tasks, immediately identifying the bias on input could prevent downstream impacts on the rest of the task.

In addition to studying the impact of different methods on the time-dimension and latent-dimension consolidation and aggregation techniques, we also survey the impact of the α smoothing parameter, which adjusts the distribution of the resampling weights. From [2], we know $\alpha = 0$ implements pure debiased training, whereas $\alpha = Z+$ or any positive number implements closer and closer to typical uniform training as the α term dominates the resampling weights. We find that this parameter acts as a tuning mechanism to modify the amount of accuracy gained in the debiased process, ranging from uniform to full debiased training. However, we observe that once this hyperparameter is not 0, it becomes necessary to tune α to the required task and leave analysis of this to future work.

Chapter 6

Future Work and Conclusion

6.1 Future Work

For future work, we propose experiments over other time-series tasks where exhibited feature bias is not easily labelled or calculated. One example may be financial markets, where the prevailing upward trend of historical data may produce biased models that are poor performers in neutral or bear markets. Additionally, we hope to expand this work to other types of data and tasks, such as tabular data or speech-to-text tasks, where quantifying bias and data density is difficult.

Specifically, we would like to see experiments over other methods of bias estimation, such as uncertainty or other Bayesian approaches as in [1]. Additionally, learning explainable and interpretable approaches to tuning the α hyperparameter to adjust the accuracy distribution of the trained model could be an interesting avenue of inquiry.

6.2 Conclusion

To summarize our completed work: we have verified that the latent debiasing approach brought forth by [2] can be adapted to time-series data, especially in the case of disentangling value-bias and sequence-bias, which is instrumental in our approach to training unbiased time-series models. Additionally, we have shown the impact of

the debiasing approach for time-series tasks over toy datasets, weather prediction datasets, and language datasets. Further, we have shown algorithmic improvements over [2] which enable our approach to unsupervised latent debiasing. Finally, we investigate the entangled issues of value-bias and sequence-bias in an attempt to understand the impact of biased data and biased models in time-series data.

We hope to see more investigation into disentangling value- and sequence-bias. While the two concepts are very related, further research into isolating each is necessary for truly debiasing and understanding the bias encoded in time-series models via their training datasets. Additionally, we hope that the field of debiasing machine learning algorithms will garner more attention and authors as these systems are increasingly deployed in the real world. While some of the experiments and datasets used in this thesis may seem removed from the lives of individual people, it is easy to see how biased datasets and biased models will continually and increasingly impact the lives of everyone. We hope to see a paradigm shift where both the datasets and the models are inspected for bias at multiple stages of the machine learning lifecycle, starting in data collection and all the way through to deployment.

Bibliography

[1] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. *CoRR*, abs/1910.02600, 2019.

[2] Alexander Amini, Ava P. Soleimany, Wilko Schwarting, Sangeeta N. Bhatia, and Daniela Rus. Uncovering and mitigating algorithmic bias through learned latent structure. *AAAI/ACM Conference on AI, Ethics, and Society*, 2019.

[3] J. Angwin, J. Larson, and L. Kirchner. Machine bias. there's software used across the country to predict future criminals. and it's biased against blacks. *ProPublica*, 2016.

[4] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021.

[5] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 2003.

[6] Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Neural Information Processsing Systems*, 2016.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Neural Information Processing Systems*, 2020.

[8] Mark Davies. Corpus of Contemporary American English (COCA), 2015.

[9] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[10] Michael Jordan. Serial order: a parallel distributed processing approach. *Neural-Network Models of Cognition*, 1986.

[11] Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. Gender bias in neural natural language processing. *Logic Language, and Security. Lecture Notes in Computer Science, vol 12300*, 2019.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Neural Information Processing Systems*, 2013.

[13] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. *Conference on Empirical Methods in Natural Language Processing*, 2014.

[14] David Rumelhart, Geoffrey Hiton, and Ronald Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, 1985.

[15] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules, 2021.

[16] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 1948.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Neural Information Processing Systems*, 2017.

[18] Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. *CoRR*, abs/2105.04857, 2021.