Explainability, Fairness, and Security for Machine Learning in Sensitive Environments

Tristan Brigham
Yale University: Keidel Grant
New Haven, CT 03301
tristan.brigham@yale.edu

1 Introduction

1.1 Problem Definition

Machine learning models are *not* called black boxes without reason. In most modern frameworks (especially gradient-based methods), it is difficult if not impossible to get a verifiably true understanding of the logic behind a neural network's output. Essentially every explainability method suffers from at least one (if not multiple) of:

1. It is extremely time consuming to implement (e.g. VQA (W. Kim, Son, and I. Kim 2021))
2. It is for simple models (e.g. decision trees, regressions)
3. Inference accuracy suffers (e.g. Cognitive Models (Blazek and Lin 2021))
4. The result is uncertain (e.g. LIME (Ribeiro, Singh, and Guestrin 2016a), SHAP (Lundberg and Lee 2017))

If any of the above is the case for a given model, then it severely constrains where that model can be deployed. However, with ML continually being integrated into more systems with increasingly critical functions, it is becoming increasingly important to find methods that avoid the pitfalls above. Therein lies a key problem in machine learning I seek to address: *is it possible to have explanation methods that avoid every pitfall above?*

There are established best practices for developing models (Clarke 2024, Bachratý 2020, Ray 2024). I seek to provide the first comprehensive method for deploying machine learning in sensitive environments while avoiding the issues above by integrating new explainability and fairness methods into said well-defined best practices.

I define a sensitive environment as one where there exists the potential for unfair bias with an outsized impact. Examples of such environments such as an insurance agency making decisions on whether to insure someone's property, a judicial system carrying out criminal justice, and a hospital making a diagnosis decision for a patient are included to help the reader. In the following sections, I will outline my contributions to any effective deployment of machine learning in sensitive environments. My approach utilizes novel methods to enforce explainability, fairness, and trust in the model. It is called Alaight.

1.2 Acknowledgements

In order to address and learn about the problems with explainable machine learning that I have encountered, I enrolled in Rex Ying's Trustworthy Deep Learning class at Yale in the Spring of 2024. The final project that I completed in the course laid basic groundwork for the following. For the sake of continuity, this work is integrated into the results of the paper.

With that being said, I would like to extend a special thank you to Mr. David Yu of the Andy Keidel Undergraduate Endowment Fund along with the broader Andy Keidel Undergraduate Endowment Fund board and ecosystem for providing me the opportunity to work on this project over the summer. The vast majority of the results found below and development that I experienced over the course of this project is a direct byproduct of the support and flexibility that the Andy Keidel Undergraduate Endowment Fund provides to me and other students.

I am eternally greatful for their support, and Yale's facilitation of such important relationships. For more information on the Andy Keidel Undergraduate Endowment Fund, their message, and to support the organization's cause please check the Yale LinkedIn 2023 page where further resources are posted.

1.3 Background

As non-linear and gradient-based methods in machine learning have become increasingly prevalent, the need for explainability has grown proportionately. However, due to the increasingly complex nature of the models being deployed, explainability and fairness have lagged compared to the accuracy and prowess of the networks.

Without solid, vetted frameworks for understanding how these models think, we fall into the trap of potentially deploying and sinking money into useless systems with no predictive or helpful power. To see why this is the case, though, it's helpful to have a fundamental understanding of how neural networks 'think':

When a neural network processes data (like images, text, etc.), it transforms this data into complex, high-dimensional representations. These transformations and the resulting interactions among features are often complex and not easily interpretable by humans. Therefore, the decision-making process of the network – how it goes from input data to a final decision or prediction – is not readily understandable or explainable to people, making these networks somewhat like "black boxes." Recent novel frameworks have attempted to solve this problem, yet compromise on other metrics which limit their applicability.

One domain that we explore in this paper is criminal justice – a field that is notorious for biased data and over-optimistic machine learning. Several AI algorithms that have been deployed for everything from predictive policing to sentencing guidelines have shown extreme bias on race or proxies for race (e.g. income, address, etc.) and have therefore faced calls to be shut down Heaven 2020.

Moreover, accuracy in models might be misleading. For example, a study by researchers in 2016 discovered that a model's disproportionately successful differentiation between wolves and dogs was not due to animal features, but rather because it identified snow in the wolf images, which was a common background in those specific pictures (Ribeiro, Singh, and Guestrin 2016b).

Even flipping a few pixels' colors from black to white in an enormous image – so few it is imperceptible to humans – can entirely derail sophisticated models. Hence, it's not unreasonable to imagine a situation where powerful neural networks unilaterally are making severely impactful decisions and its facilitators don't understand why. In fact, it's already happening. And, the effects are already potentially deadly.

Epic Systems is a major American software company that specializes in electronic health records (EHR) software for the healthcare industry. It has developed a portfolio of proprietary AI algorithms, including one for predicting sepsis, a leading cause of death in hospitals. However, independent investigations found significant inaccuracies in these predictions. For example, the Epic Sepsis Model (ESM) was found to generate a large number of false alarms, creating a burden of alert fatigue for healthcare professionals.

The Epic Systems sepsis prediction model, widely used in over half of U.S. hospitals, exhibited variable performance in a study analyzing over 800,000 patient encounters across nine hospitals. Research published in JAMA Internal Medicine revealed that the model's accuracy was lower in hospitals with higher sepsis rates, a greater number of patients with multiple health conditions, and a larger oncology patient population. Simply put, the model was substantially less accurate than originally touted.

Here's the problem: after an extensive investigation culminating in a review of internal marketing emails, researchers found that one of the algorithm's input variables during training was whether a doctor had already ordered sepsis medication. See the problem? (Ross 2022, Wong et al. 2021)

A more transparent implementation of such neural networks likely would have caught this training factor's importance in the model early, and made it obvious that a real-world environment such as a hospital where data is not always perfect and cleaned would not be conducive to a model where one of the inputs is essentially the output.

1.4 Alaight

Explainability of networks is only one of four pillars found within the most common framework for trustworthy deep learning (that is robustness, explainability, privacy, and fairness). Yet, it is arguably the most crucial. With robust explainability within a neural network, it can be trivial to ensure that the other 3 pillars are incorporated as well. Therefore, Alaight is fundamentally a method for explainability in a network.

Looking to how Alaight can contribute to the broader goal of trustworthy machine learning most effectively, we find that the framework is most helpful in two domains:

1. **Explainability:** The idea behind the concept-focused training is to push nodes to represent concepts and patterns present in the training data. The goal is to create a machine learning model where one can analyze the activations of concept nodes to understand what patterns the model has found in the data and is making its decisions on. I introduce an approach to more explainable machine learning models. The approach is similar to the CAV framework discussed in class in finding ways to represent data, but has advantages in its ability to find the most important concepts present in data automatically instead of requiring human intervention to define concepts and model integration (Druc et al. 2022).
2. **Fairness:** Using a combination of the broad explainability provided by the concept-focused training and other granular explainability methods such as Layer-wise Relevance Propagation (Binder et al. 2016), LIME (Ribeiro, Singh, and Guestrin 2016a), or Grad-CAM (Selvaraju et al. 2019), the end user can gain insight into the logic behind models' decisions. Using the influence re-weighting approach I propose in this paper, the end user can decrease the influence that any concept or input has on the output. This method can be used to limit unfair or harmful decision factors and thereby increase the fairness of the trained model in substantially less time than full re-training.

2 Related Works

Several existing methods seek to explain machine learning. Some of the most prevalent and related are noted below:

1. OpenAI used GPT-4 to analyze the neuron activations of GPT-2. The goal was to create an automated process for understanding the logic hidden in LLM neurons given how large the models can be. The authors found that GPT-4 could make basic inferences about the neuron activations (Bills et al. 2023).
2. Saliency maps along with their derivative methods have showed promise in explaining what parts of an input a model focuses on and uses to generate its output. I use saliency maps to calculate the individual neuron influence in this project (Simonyan, Vedaldi, and Zisserman 2014).
3. SHAP is one of the leading approaches for model-agnostic explanations of machine learning. It uses Shapley values to estimate the contribution that each feature provides to the model, considering both the feature itself and its combinations with other features (Ribeiro, Singh, and Guestrin 2016a). However, this method runs in an exponential runtime and relies on models providing probabilistic and continuous outputs to score the confidence.
4. LIME creates a locally explainable (typically linear) surrogate model with a weighting function to adjust the influence that points have on the surrogate model to estimate what data points and features inform the output (Ribeiro, Singh, and Guestrin 2016a).

5. Researchers have created neural networks where the sub-components are decision networks to decide whether a concept is present or not in the input. The outputs of that network are agglomerated and passed to another network that generates the final network output (Blazek and Lin 2021). Using the sub-decision networks guarantees that the model makes decisions based off of the concept model outputs which can improve interpretability with extreme added overhead.
6. Concept activation vectors are a key part of the deep learning explainability regime. The process to utilize concept activation vectors for machine learning models is effectively to investigate a latent activation space deep in a model to analyze what neurons fire and outputs are generated when a given concept is present in the input space, and then utilize these vectors for things like understanding the model's understanding of concepts in the input space (B. Kim et al. 2018).

Less exploration of intrinsically explainable machine learning methods has been done, and minimal research has focused on the topic since 2019 (Card, Zhang, and Smith 2019, Alvarez-Melis and Jaakkola 2018).

3 Proposed Approaches

Using my approach as an addition to best practices, machine learning models can begin to be deployed in sensitive environments with verifiable logic and results. My approach involves 3 aspects, the first two of which are novel:

3.1 Concept-Focused Training

Using a custom-defined loss function, I manipulate the gradients during training such that the parameters force the activations of neurons to represent patterns or ideas that the model will decide on. The inspiration for this process comes from Druc et al. 2022. In my method, instead of training surrogate models to differentiate concepts, the concepts are simply integrated into the node activations. The idea is that for any parameter configuration that is not at the absolute local minimum for data (which most models are not), there exists a manifold of model parameters that produce the same loss value. I aim to train the model such that it finds the optimal parameter configuration on that manifold for explainability.

Before training the model, I run PCA on the input data to get embeddings with minimized noise for the data points. By doing so, we can exploit the fact that PCA finds a low-dimensional representation while maintaining the most important features and variance for distinguishing classes (Kwak and Choi 2003). The embeddings can be found in Figures **??**, **??**, and **??**. An example number of data points with the minimum distance to each centroid is found in Figure 12b.

After running PCA, I run a clustering algorithm to generate C centroids for the data where C is a user-defined hyperparameter. The best results come when C is roughly $\frac{\sqrt{n}}{5}$ for the dimensionality of the data n. These centroids will essentially represent distinct concepts or ideas present in the data. A point's distance to these centroids judges how likely the point is to include the centroids' concepts.

For each of the training data points we compute the Mahalanobis Distance between the training point and each of the C centroids found above in the PCA representations of the data (Mclachlan 1999). Using the Mahalanobis distance accounts for any skew or covariance remaining within sub-clusters of the data.

Then, train the model using the original training data (not the PCA-decomposed version) where the same loss as normal is used. However, instead of applying the normal gradient to each parameter at each update step, transform the gradient g_i into g_i' for a single parameter i in the model as follows before clipping the gradient to mitigate infinite values:

$$g_i' = \epsilon \cdot \frac{s}{C} \sum_{j=1}^{C} \Bigg(((1 - \sigma(g_i, w_i)) \cdot (1 - m_{j,\text{norm}}) + \sigma(g_i, w_i) \cdot m_{j,\text{norm}}) \cdot \sigma(\|\Delta_o\|) \\ + (\sigma(g_i, w_i) \cdot (1 - m_{j,\text{norm}}) + (1 - \sigma(g_i, w_i)) \cdot m_{j,\text{norm}}) \cdot (1 - \sigma(\|\Delta_o\|)) \Bigg) + g_i$$

where m_{max} is the maximum Mahalanobis distance in the PCA space between any two points across all of the input data. $m_{j,\text{norm}} = \frac{m_j}{m_{\text{max}}}$ is the normalized mahalanobis distance from the original training input data to the j^{th} centroid computed above since m_j is the raw Mahalanobis distance.

$\sigma(g_j, w_j) = \sigma(\text{sign}(g_j) \cdot \text{sign}(w_j))$ represents a 1 if the sign of each of the values g_j and w_j is the same and 0 otherwise. g_j represents the j^{th} entry in the gradient and w_j represents the j^{th} weight in the weight matrix. It is used to update the gradient in the manner such that the outputs at the target layer are similar for inputs with a small Mahalanobis distance, and are disparate for different inputs.

o_j is the output of the layer that g' is updating when the j^{th} centroid was used as input, and o_c is the output of the layer given the current training point as input. Hence, $\Delta_o = o_j - o_c$. ϵ is a hyperparameter with a small value. s is a parameter that controls the strength of the gradient with respect to how deep we want the concept nodes to be $s = e^{-\left(\frac{l_c}{l_t} - p_e\right)^2}$ where l_c represents the index of the layer that the gradient is being applied to, l_t is the total number of layers there are, p_e is a value between 0 and 1 that controls how deep the concept nodes should be (0 means the inputs will represent the concepts, or the features are the concepts themselves, and 1 means the model outputs will represent the learned concepts). It is maximized and equal to 1 when $\frac{l_c}{l_t}$ is closest to p_e, and a value of p_e such that the second-to-last or third-to-last layers contain the concept nodes is best as the direct impacts of certain concepts being present or not in the input data can be understood best there.

In order to further understand exactly the impetus behind such decisions, a simplified version of the equation is:

$$g_i' = \epsilon \cdot \frac{s}{C} \sum_{j=1}^{C} \Bigg(((1 - \sigma_{gi}) \cdot (1 - m_{j,\text{norm}}) + \sigma_{gi} \cdot m_{j,\text{norm}}) \cdot \sigma_{\Delta o} + (\sigma_{gi} \cdot (1 - m_{j,\text{norm}}) + (1 - \sigma_{gi}) \cdot m_{j,\text{norm}} \cdot (1 - \sigma_{\Delta o}) \Bigg) + g_i$$

where

$$\sigma_{gi} = \sigma(g_i, w_i)$$
$$\sigma_{\Delta o} = \sigma(\|\Delta_o\|)$$

We combine the terms inside the summation:

$$g_i' = \epsilon \cdot \frac{s}{C} \sum_{j=1}^{C} \Bigg(((1 - \sigma_{gi}) \cdot (1 - m_{j,\text{norm}}) + \sigma_{gi} \cdot m_{j,\text{norm}}) \cdot \sigma_{\Delta o} + (\sigma_{gi} \cdot (1 - m_{j,\text{norm}}) + (1 - \sigma_{gi}) \cdot m_{j,\text{norm}}) \cdot (1 - \sigma_{\Delta o}) \Bigg) + g_i$$

We denote the two inner terms as A and B:

$$A = ((1 - \sigma_{gi}) \cdot (1 - m_{j,\text{norm}}) + \sigma_{gi} \cdot m_{j,\text{norm}})$$
$$B = (\sigma_{gi} \cdot (1 - m_{j,\text{norm}}) + (1 - \sigma_{gi}) \cdot m_{j,\text{norm}})$$

Thus, the equation simplifies to:

$$g_i' = \epsilon \cdot \frac{s}{C} \sum_{j=1}^{C} (A \cdot \sigma_{\Delta o} + B \cdot (1 - \sigma_{\Delta o})) + g_i$$

The idea behind this is to make the output of a given layer be close to a weighted average of the outputs where o_j is upweighted inversely with the mahalanobis distance from the current input to

centroid j in the PCA space, and to make the outputs as different as possible when the distance between a centroid and the point is large.

We also limit the number of gradients that we perturb so that we only change the gradients that have the strongest negative or positive activations (typically the top 30% has shown good results).

Finally, to push the neurons to represent concepts, I apply weight-decay to the intermediate layers. Given the difference constraint imposed above with the weight-decay, the neurons start to represent a low-dimensional form of the required concepts to generate the correct outputs.

The output is a deep layer which activates different neurons for the presence of different ideas in the input data and contains enough information to generate correct outputs. In essence, these are the model-defined most important concepts in the data.

3.2 Influence Re-Weighting

Once a user finds what the model believes is the most important set of concepts in the input data through concept-focused training, the user can limit the influence of features deemed harmful or unfair through influence re-weighting without re-training the model from scratch. Influence of concepts are measured using layer-wise relevance propagation and saliency maps. We explore novel techniques such as stochastic fine-tuning, inversion fine-tuning, and amplitude influence-reduction as candidates for removing feature influence.

Stochastic fine-tuning trains the machine learning model using the original training data but replacing the harmful features with stochastic variables. The model should learn that these stochastic features should be ignored, and their influence is minimized. I have attempted to generalize this to concepts defined by combinations of input features without luck, so it only works for input features currently.

Inversion fine-tuning is the same process but instead of replacing the harmful features with noise, the features are inverted about their dataset means in the input data. I have attempted to generalize this to concepts defined by combinations of input features without luck.

$$x[i] = \text{mean}(x[:, i]) - 2 \cdot (x[i] - \text{mean}(x[:, i]))$$

In amplitude influence-reduction, we simply try to reconstruct the concept that we are attempting to remove using the other activations in the preceding layer, and adjust the other features' weights to offset this loss. I explored several algorithms to construct the optimal weighting, but the best outcome comes from running a regression using the other features in the training data as input values and adjusting the weights in the subsequent layer to offset the change in the harmful feature's weights. The weights that are multiplied with the harmful feature are set to zero, effectively turning off this feature. The error increases with this new regime, so some fine-tuning is required at the cost of re-introducing the concept slightly. Essentially, to remove the influence of node i' in a layer, we adjust the weights such that:

$$w'_j = \begin{cases} 0 & \text{if } j = i' \\ w_j \cdot \left(1 + \frac{\|w_{i'}\|}{\sum_{k=1}^{n} w_k}\right) & \text{otherwise} \end{cases}$$

If we are trying to remove a concept, we must limit our fine-tuning or else the concept will simply be re-learned in another neuron or set of neurons. Else, with concepts as input features one can fine-tune and enforce that the harmful feature's weights are zero without issue.

3.3 Weight Freezing

Once a model is certified to possess certain properties by some trusted body, the deployed model must keep those properties. The stakeholders actually deploying the model likely do not want to re-run the tests locally to limit computation costs, comparing the weights of the local model against a remote version can be expensive given communication overheads, and it is risky to send weights over communication networks incase malicious actors intercept the weights and potentially run attacks to infer data or model attributes.

Stakeholders can run a deterministic hashing algorithm over the important attributes (e.g. weights, activations, layer sizes, etc) of a model and compare this against the trusted body's hash of the

model to see if the model still possesses the attributes that the trusted body said it does in a single communication step without communicating the weights or sensitive data. Given that hashing is not a novel concept, I refer the user to Martín-Fernández and Caballero-Gil 2022 for more information and use Geeks for Geeks 2023 as a resource for my implementation.

3.4 GNN's for Concept Definition

One of the key problems with the approaches above is that the concept definition can be difficult. A trivial way to define the concepts is to run it through an algorithm such as PCA which is able to find the concepts and ideas which theoretically have the largest influence on the output. However, in order to take full advantage of the power behind explainability in inherently interpretable machine learning models, the concepts must have high fidelity.

One method that has been lightly explored to increase the fidelity of machine learning models is the incorporation of graph neural networks into the data analysis. Meng et al. 2022 propose a large language model in which the text corpus is dynamically integrated to produce more accurate outputs. One can think of it as a more complex version of PCA for its ability to better find nonlinear and hidden relationships between input concepts.

Because other portions of this project took much longer to implement during this summer, I leave graph neural network integration into the process to future exploration.

3.5 Advanced Concept Activation Vectors

Concept Activation Vectors are a very well-known method for finding the influence of concepts on the output. In essence, one can train a linear classifier on a deep latent activation space from one of the later layers in a network and use this to find which concepts are most salient in an input.

Given a neural network f and a set of inputs X, let $h_l(X)$ represent the activations at layer l. For a concept C, let X_C be the set of inputs that represent the concept and X_R be the set of random counterexamples.

The activations for these sets are:

$$H_C = h_l(X_C)$$
$$H_R = h_l(X_R)$$

A linear classifier is trained to distinguish H_C from H_R, solving the following optimization problem:

$$\min_{\mathbf{w},b} \sum_i \mathcal{L}(\mathbf{w} \cdot h_l(X_i) + b, y_i)$$

where $\mathcal{L}$ is the loss function, $\mathbf{w}$ is the weight vector (representing the CAV), and y_i are the labels indicating whether the activation comes from H_C or H_R.

The Concept Activation Vector is:

$$\mathbf{v}_C = \mathbf{w}$$

To test the influence of a concept on the network's decision, we use the directional derivative of the prediction function f with respect to the CAV:

$$S_{C,k} = \frac{\partial f_k}{\partial h_l} \cdot \mathbf{v}_C$$

where $S_{C,k}$ indicates the sensitivity of the class k prediction to the concept C B. Kim et al. 2018.

I take this concept a step further in my explainability strategy. Given that the linear classifier is able to decipher when certain attributes or concepts exist in the input, it should also be helpful in finding ways to perturb the latent space activation such that the impact of this concept is mitigated.

That is, given the previously defined activation of layer l $h_l(X)$, we can pair this information with the input to the model on a given inference to create a highly informative vector that can be used to perturb the latent activation space such that the output is closer to what I would expect.

We consider two methods to limit the effect of specific concepts on the output of a neural network: a linear classifier and a Multi-Layer Perceptron (MLP).

Linear Classifier Approach

Let $\mathbf{z}$ be the activation vector in the latent space. We want to remove the effect of a concept C from $\mathbf{z}$. Suppose $\mathbf{v}_C$ is the Concept Activation Vector (CAV) corresponding to concept C.

The modified activation vector $\mathbf{z}'$ can be obtained by subtracting a scaled version of $\mathbf{v}_C$:

$$\mathbf{z}' = \mathbf{z} - \alpha \mathbf{v}_C$$

Here, α is a scalar representing the prevalence of the concept C in the input space. This scalar can be determined as:

$$\alpha = \beta \cdot \sigma(\mathbf{z} \cdot \mathbf{v}_C)$$

where β is a hyperparameter that controls the strength of the adjustment, and σ is the sigmoid function.

Multi-Layer Perceptron Approach

For the MLP approach, we train a neural network P that takes the activation vector $\mathbf{z}$ and produces a perturbation $\Delta\mathbf{z}$ such that the final output remains the same for two input feature sets differing only by concept C.

Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two input feature sets where only the concept C differs. Their corresponding activation vectors are $\mathbf{z}_1$ and $\mathbf{z}_2$.

The perturbation $\Delta\mathbf{z}_1$ for $\mathbf{z}_1$ is given by:

$$\Delta\mathbf{z}_1 = P(\mathbf{z}_1)$$

Similarly, the perturbation $\Delta\mathbf{z}_2$ for $\mathbf{z}_2$ is:

$$\Delta\mathbf{z}_2 = P(\mathbf{z}_2)$$

The modified activation vectors are:

$$\mathbf{z}_1' = \mathbf{z}_1 + \Delta\mathbf{z}_1$$
$$\mathbf{z}_2' = \mathbf{z}_2 + \Delta\mathbf{z}_2$$

We train the MLP to minimize the difference in the final outputs of the neural network for $\mathbf{z}_1'$ and $\mathbf{z}_2'$:

$$\mathcal{L} = \|f(\mathbf{z}_1') - f(\mathbf{z}_2')\|_2^2$$

where f is the function representing the rest of the neural network after the latent space. By training P to minimize $\mathcal{L}$, we ensure that the final output is insensitive to the presence of concept C.

Using the concept activation vectors to limit the influence of concepts in the decision making process of the machine learning model will be referred to as concept-dampening for the rest of this study.

4 Process Section

In order to provide more granular insight into the process that I followed this summer, I walk through the steps that I took in order to achieve my results below.

1. In the beginning of the summer, my primary focus was on getting as broad of an understanding of the field of trustworthy deep learning as possible. Besides reviewing all of the class material and papers that I could find from Professor Rex Ying's Trustworthy Deep Learning course, I also read 100+ papers to solidify my understanding. Many of the concepts that I gleaned from this paper were instrumental in informing the final products of the paper.

2. As a next step after completing my literature review, I searched for spaces in industry that were currently lacking explainable machine learning and were in the most dire need of it. This review of industry standards and practices led me to believe that criminal justice and healthcare were the two fields that were most in need of trustworthy deep learning. These were fortunately the same two fields that I had targeted with my final project in Trustworthy Deep Learning as well.
3. In my third step this summer, I worked to implement a more salient version of the PCA process which is described above using several different methods. The first and most obvious version of dimensionality reduction (and concept isolation) that I tried was using autoencoders to compress the information from the input into a latent space.
4. Because of the autoencoder's ability to capture non-linear features in the data when the models compress the input information into a latent space, this was able to create salient embeddings of the input information.

   Mathematically, an autoencoder consists of an encoder function f_θ and a decoder function g_ϕ. The encoder f_θ maps the input $\mathbf{x}$ to a latent representation $\mathbf{z}$:

   $$\mathbf{z} = f_\theta(\mathbf{x})$$

   The decoder g_ϕ maps the latent representation $\mathbf{z}$ back to the reconstructed input $\hat{\mathbf{x}}$:

   $$\hat{\mathbf{x}} = g_\phi(\mathbf{z})$$

   The model is trained to minimize the reconstruction error, typically measured by the mean squared error (MSE):

   $$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

   By minimizing this loss function, the autoencoder learns to capture the essential features of the input data in the latent space $\mathbf{z}$. However, the latent space embeddings alone do not necessarily correspond to interpretable concepts.

   To extract meaningful concepts from the latent space, we introduce a regularization term that encourages the latent representations to align with predefined concepts. Let $\mathbf{c}$ be a vector representing a concept, and $\mathbf{z}_i$ be the latent representation of the i-th input. We can add a regularization term to the loss function that penalizes deviations from the concept vector:

   $$\mathcal{L}_{reg} = \sum_i \|\mathbf{z}_i - \mathbf{c}\|^2$$

   The total loss function thus becomes:

   $$\mathcal{L}_{total} = \mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) + \lambda \mathcal{L}_{reg}$$

   where λ is a regularization parameter that controls the tradeoff between reconstruction accuracy and concept alignment.

   However, the only way to get concepts defined by the model out of the latent space was to add intense regularization, which resulted in a tradeoff between accuracy and concept prevalence. When λ is too high, the model might overly constrain the latent representations to match the concept vectors, leading to a loss of reconstruction accuracy:

   $$\text{Accuracy} \propto \frac{1}{\lambda}$$

   Conversely, if λ is too low, the latent representations might not align well with the predefined concepts, reducing the prevalence of these concepts in the latent space:

   $$\text{Concept Prevalence} \propto \lambda$$

   Therefore, finding an optimal balance of λ is crucial to ensure both high reconstruction accuracy and meaningful concept embeddings.

5. In order to try to find a method that had better concept-defining abilities without compromising on accuracy, I tried t-Distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP). UMAP is a faster method than t-SNE with better techniques for preserving both local and global features and was implemented after t-SNE, but despite the non-linearity of both methods I was not able to define concepts with high fidelity in the output.

   t-SNE operates by minimizing the Kullback-Leibler divergence between the joint probabilities of the high-dimensional data and the low-dimensional embedding:

$$\text{KL}(P \parallel Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

   where p_{ij} and q_{ij} represent the pairwise similarities in the high-dimensional space and low-dimensional space, respectively. Despite this, t-SNE often struggles to preserve both local and global structure.

   UMAP, on the other hand, constructs a weighted k-nearest neighbor graph in the high-dimensional space and optimizes a cross-entropy between fuzzy topological representations of the data:

$$\mathcal{L}_{UMAP} = \sum_{(i,j) \in \text{edges}} w_{ij} \log \frac{1}{d_{ij} + \epsilon} + (1 - w_{ij}) \log \frac{d_{ij} + \epsilon}{1 + d_{ij} + \epsilon}$$

   where w_{ij} are weights representing local connectivity probabilities, and d_{ij} are distances in the low-dimensional space. Although UMAP can better preserve both local and global structures compared to t-SNE, it did not achieve high fidelity in concept definition for my data.

6. Independent Component Analysis (ICA) was the next method attempted. ICA has the principal goal of decomposing given feature sets into additive sets. ICA assumes that the observed data $\mathbf{X}$ are linear mixtures of some unknown latent variables (sources) $\mathbf{S}$:

$$\mathbf{X} = \mathbf{AS}$$

   where $\mathbf{A}$ is an unknown mixing matrix, and $\mathbf{S}$ contains the source signals which are assumed to be statistically independent and non-Gaussian.

   To find the independent components, ICA estimates both the mixing matrix $\mathbf{A}$ and the sources $\mathbf{S}$ given only the observed data $\mathbf{X}$. This can be achieved by finding an unmixing matrix $\mathbf{W}$ such that:

$$\mathbf{S} = \mathbf{WX}$$

   Several algorithms can be used to perform ICA, including the FastICA algorithm. The key idea behind these algorithms is to maximize the statistical independence of the estimated components $\mathbf{S}$. One common approach is to maximize the non-Gaussianity of the components, which can be quantified using measures such as kurtosis or negentropy.

   One way to maximize non-Gaussianity is to maximize the kurtosis, defined for a random variable y as:

$$\text{Kurt}(y) = \mathbb{E}[y^4] - 3(\mathbb{E}[y^2])^2$$

   Another approach is to maximize negentropy, which is based on the entropy $H(y)$ of a random variable y. The negentropy $J(y)$ is defined as:

$$J(y) = H(y_{\text{Gaussian}}) - H(y)$$

   where y_{Gaussian} is a Gaussian random variable with the same covariance matrix as y. Negentropy is always non-negative and is zero if and only if y is Gaussian.

   The FastICA algorithm is an efficient and popular method for performing ICA. It involves the following steps:

1. **Centering**: Subtract the mean of $\mathbf{X}$ to make it zero-mean. 2. **Whitening**: Transform $\mathbf{X}$ to make it uncorrelated and unit variance. 3. **Iterative Optimization**: Use a fixed-point iteration scheme to maximize the non-Gaussianity of the estimated components. The fixed-point iteration for one component w involves:

$$w^{(k+1)} = \mathbb{E}[\mathbf{X}g(\mathbf{X}^T w^{(k)})] - \mathbb{E}[g'(\mathbf{X}^T w^{(k)})]w^{(k)}$$

where g is a non-quadratic function (e.g., $g(u) = \tanh(u)$) and g' is its derivative. The vector w is then normalized to unit length.

This method was used to try to find the individual factors (or signals) that result in given outputs from the neural network. By applying ICA to the latent space representations of the network, I sought to find the constituent features and concepts that made up an input to the neural network that I could limit. While there was some success with this, I found that it was not as good as PCA in defining features that could be integrated into the explainability model.

7. Continuing with the explainability portion of the project, I was fortunate to have a framework in place from the Trustworthy Deep Learning class that could help me think about different ways to create models that inherently explain themselves.
8. My first step was to continue the improvement of the base framework that I had in place. Although it generally worked, there were several performance improvements that I made to ensure that the training process was as close to vanilla training as possible. This ensures that integration of my method into production environments is cost-efficient and we can draw on continued improvements in the torch framework due to the deep integration of its components into my code. I decreased the runtime of the training process by roughly 80% which put it within a 10% deviation from the runtime of vanilla, unchanged models.
9. The next step was to attempt to limit the influence of select concepts on the output. The driving theory was that if I can find activations representing salient concepts in deeper layers of the neural network, then I could alter these activations on the fly or the weights in the neural network matrix multiplication scheme to limit the influence of the weights. The method that I had originally implemented suffered from out-sized performance losses with the new concept-dampening scheme and therefore had to be replaced. I tried several loss functions in attempts to keep the distributions of deep layers in the neural network from shifting or changing with the removal of certain concepts from nodes, but I suffered from tradeoffs once again: either the concept was removed or the accuracy of the network was low compared to a baseline without the concept being in the training data at all. Simple fine-tuning on the data seemed to fix this problem, but resulted in catastropic forgetting or the reintroduction of the target concepts in further distributed node activations. Both of these problems were not overcome in the course of this project and are left to further exploration.
10. Finally, I sought to integrate both the MLP and regression variants of the Concept Activation Vector-driven into the framework. The process to incorporate the frameworks were rather arduous since I had to recreate much of the processing that I had for the baseline explainability model for the new fairness-enforcing variant. However, once both of the types were implemented, they thankfully worked very well. With minimal hyperparameter tuning, it was possible to take a vanilla model and turn it into a fully explainable model according to pre-defined concepts through both the PCA space and normal node inputs. I call this process concept-dampening.

5 Experiment Section

5.1 Datasets

The first dataset I use is a life insurance decisions dataset from Prudential. The dataset was released in 2016 as part of a Kaggle competition. It provides anonymized insurance decisions with numerical representations of applicant attributes like income, height, and weight (Montoya et al. 2015).

There are 59,381 entries across 128 columns with a mean height and weight across the dataset of 0.707 and 0.293 respectively. Medical keywords are sparse (as evidenced by the low mean values) and the vast majority of the data is normalized to a distribution between 0 and 1.

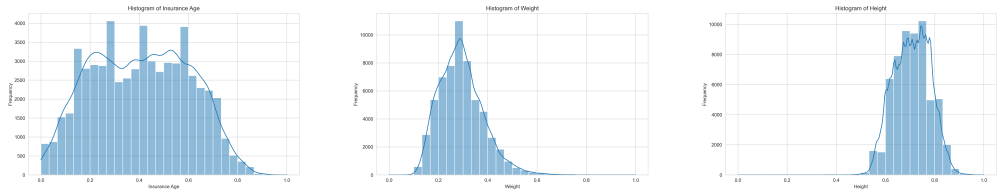


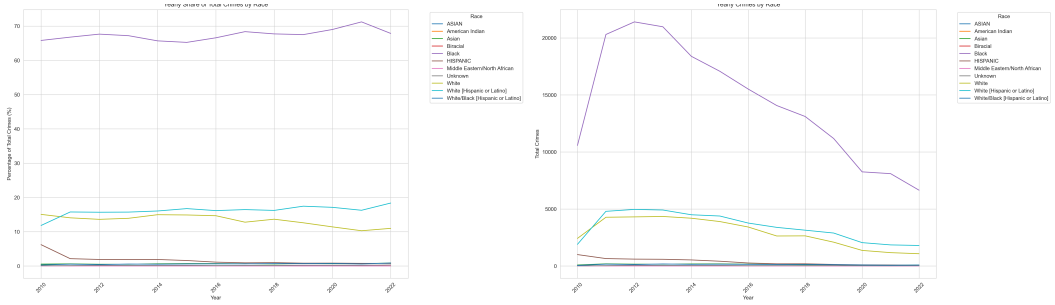
Figure 1: Histograms of Age, Weight, and Height

Because the dataset is anonymized, we cannot be certain what the response values included in the dataset mean. But, an educated guess given the relative prevalence of different values might be Declined, Postponed, Rated, Standard, Preferred, Elite, Smoker, and Accepted for values 0-7.

The Cook County Criminal Justice Dataset comes from the Chicago Open Data City Initiative (City of Chicago 2024, Cook County, Illinois Data Catalog 2024). This dataset will serve as a validation dataset given its larger size and easier interpretability due to lesser privacy constraints.

Analyzing the data, I find that young men are dramatically over-represented in the data compared to their share of the population. Additionally, as confirmed by numerous studies, people of minority backgrounds are systemically more likely to experience the criminal justice system in Chicago (M. K. Chen et al. 2022, Minor 2023, American Civil Liberties Union of Illinois 2015).

Given this data, it becomes apparent that it is crucial to find a way in order to limit the prevalence of race in MLP's as a decision factor.



(a) Share of Reported and Processed Crimes by Race (b) Yearly Reported and Processed Crimes by Race

Figure 2: Racial distribution of Reported Crime (does not normalize for racial makeup of communities)

5.2 Benchmarking

Training a basic neural network without concept-focused training yields medial results with uninterruptible intermediate layers. As a test, I run LIME on the base neural network which has a mediocre result on the life insurance dataset, and a poor result on the criminal justice dataset. The computed relative importance values are extreme and incorrect. LIME seems to worsen with the complexity of the dataset.

Given concept-focused training is a method to create intrinsically explainable machine learning, I fit an XGBoost model to the data as a baseline. XGBoost is a tree-based model that has been shown to be one of the most accurate explainable models (T. Chen and Guestrin 2016). We look to the validation accuracy and Cohen's Kappa values of the models (McHugh 2012).

The concept-focused neural network uses the gradient boosting equation explained above to generate concept-activated neurons in the second-to-last layer of the model. The Single Layer Concept-focused Neural Network only applies the gradient boosting to the second-to-last layer (none of the layers before it). All networks have a structure the same as the structure in Figure 12a.

I find that the introduction of concept-focused training does not degrade the neural network performance when the gradient boosting is scaled up from the start of the model to the target layer.

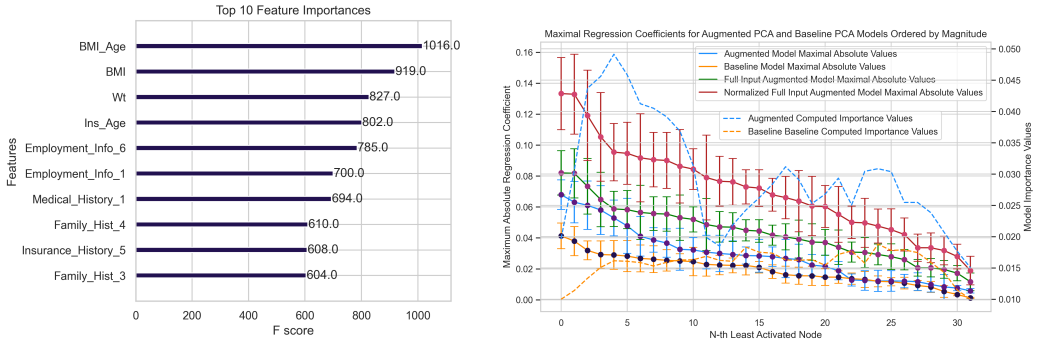
Table 1: Average Model Performance

Model	Cohen's Kappa	Validation Accuracy (%)
Vanilla XGBoost	0.6510	28.45
Unchanged Neural Network	0.4326	44.14
Concept-focused Neural Network	0.4127	43.91
Single Layer Concept-focused Neural Network	-0.01647	10.37

However, when the gradient boosting is only applied to a single layer the model collapses. This likely indicates that the concepts must be built up over the course of the neural network layers – they cannot be constructed immediately from an arbitrary layer's output.

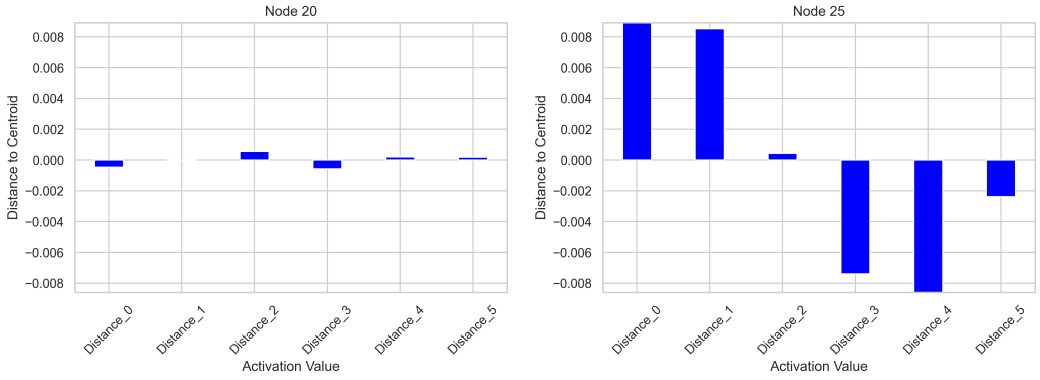
The higher Cohen's Kappa value but lower accuracy in XGBoost comes from the tree model being better at not over-predicting a single class.

Now, looking to the relative explainability of the models, I find that the concept-focused neural network has better explainability than the XGBoost model.



(a) XGBoost's Self-Reported Top Features (b) Concept Activation of Second-to-Last Layer Neurons

Figure 3: Explainability of Investigated Models



(a) Concept neuron with weak concept utilization (b) Concept neuron with strong concept utilization

Figure 4: Concept Activation of Neurons

The XGBoost model is only able to provide high-accuracy feature importance scores for the input features. Although tree-based models can give somewhat inaccurate concept-based importances, the definitions of the concepts are rather arbitrary and it is difficult to define concepts that the model is actually using with any degree of confidence.

On the other hand, in the concept-focused model, I use established methods such as SHAP and GradCAM to get average individual feature importance in the model in only slightly more time than

XGBoost's explanation framework. The model yields concept-relevance explanations instantly given the concept-focused training. On an instance level, we can compute the importance of any neuron in the model with existing methods, and cross-check what concepts the important neurons represent. In Figure 3b, we plot the augmented model trained on the PCA's maximal coefficients (which has the largest spike on the left), the augmented model trained on the full input with PCA distance (the second highest on the left), the un-augmented model trained on PCA, and the un-augmented model trained on the full input. The augmented model has several neurons which represent combinations of subsets of the model-defined concepts with high fidelity (the left side of the graph), while the rest of the neurons possess residual information for generating accurate outputs.

We run a regression on each of the neurons to find out which are most correlated with weighted combinations of concepts. A subset of the neurons in augmented models use the PCA-defined concepts (nodes represented on the left of the graph). These represent the concepts that the model find most informative. The latter neuron regressions revert to normal values with relatively high output influence which signifies important residual information.

Because neurons' computed influence has strongly positive covariance with the concept alignment, we can infer that the final layer of the model is using the concepts to compute the final model output. The total effect of the concept on the output is the magnitude of the regression coefficient times the node importance. Additional explanation information is plotted in the supplementary portion of this report at Figures 7a, 7b, 8a, and 8b.

Figure 4a shows a neuron in the deeper layers that does not necessarily use the model-defined concepts to make its decision. It is also a neuron with very weak influence on the output after training because of the weight-decay. The existence of these neurons shows that the model is selective in the concepts it defines. It leaves as many neurons with low influence and concept influence as possible. On the other hand, Figure 4b shows a neuron that uses a concept which is a combination of concepts 0 and 1, and the opposite of 3 and 4. To give an example, if the model decides concept 0 is a combination of features to represent violence exposure during childhood, concept 1 is the crime's violence level, concept 3 represents the convicted's employment status, and concept 4 is the racial bias of the arresting county's police force, this neuron could indicate a concept like the individual's likelihood to recidivate. Analyzing the activations of the strong concept neurons for any specific inference explains whether the concept is present and utilized in the inference, generating instance-level concept explanation.

We test how well each of the methods do in removing concept influence compared to baseline retraining. The average runtime of each method is found in Table 4. Refer to Table 5 for the full study.

Table 2: Ablation Study of Removing Different Principal Components' Influence Impact on Val Loss

Reduction Method	PC 0 (%)	PC 2 (%)	PC 5 (%)	PC 7 (%)	PC 9 (%)
Amplitude Reduction	1.90	1.86	1.86	1.86	1.88
Stochastic Fine Tuning	1.23	1.23	1.25	1.41	1.29
Inversion Fine Tuning	0.05	0.01	0.09	0.19	0.05
Simple Retraining	0.99	0.98	0.96	1.25	1.16

Table 3: Average Influence Reduction per Function from Removing Principal Components

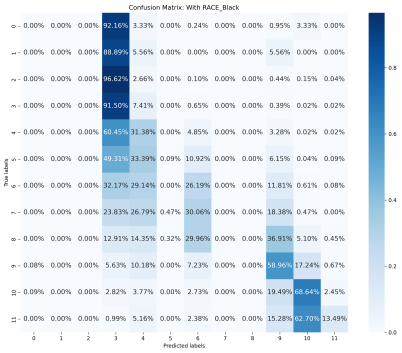
Reduction Method	PC 0 (%)	PC 2 (%)	PC 5 (%)	PC 7 (%)	PC 9 (%)
Amplitude Reduction	-100.00	-100.00	-100.00	-100.00	-100.00
Stochastic Fine Tuning	-65.93	-68.57	-73.66	-66.78	-65.39
Inversion Fine Tuning	-22.73	-23.48	-24.75	-22.88	-21.85
Simple Retraining	-100.00	-100.00	-100.00	-100.00	-100.00

As seen by comparing Table 2 and Table 3, each of the PCA components have relatively equal influence on the output of the model. This seems counterintuitive at first since earlier PC's should represent larger shares of the data information. It should get easier to mitigate the loss increase due to the decreasing variance. Removing any single concept increases the validation loss. Additionally, the

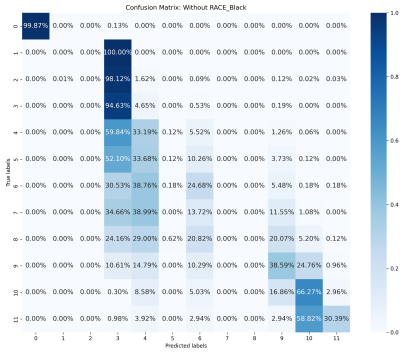
model doesn't seem to be affected until the concept is at least 50% removed when validation loss shoots up. The best method is the amplitude influence-reduction method with similar runtime to stochastic fine-tuning but the performance guarantees of re-training the model.

Looking now to the concept-dampening model which uses concept activation vectors to limit the effect of harmful concepts on decision processes, we find that the introduction of explainability of the output predictions and paired with the developed adjustment frameworks for fairness into the machine learning model drastically increases the fairness of predictions. As an example, we can look to court-prescribed Black people and their treatment by machine learning models. In the following study, I use the data from the Cook County Courts system to try to predict the severity of punishments handed down to people who go through the courts system accused of crimes. The higher values correspond with more severe punishments. For example, 0 corresponds with small fines while 5 corresponds to medium-length prison terms and 11 corresponds with life terms.

It is important to note that as a proof of concept, a simple MLP with Relu activations in middle layers and a softmax activation layer at the end was used. Hence, the base performance of the model is not great. With that being said, given the novelty of my approach one can assume that more advanced models fall victim to the same bias issues as found in my simple MLP implementation.



(a) Confusion matrix of simple model incarceration outcomes for Black people



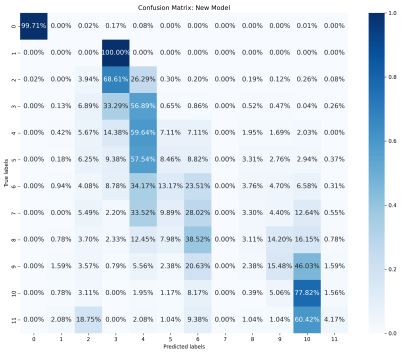
(b) Confusion matrix of simple model incarceration outcomes for non-Black people

Figure 5: Confusion Matrixes for Model Performance

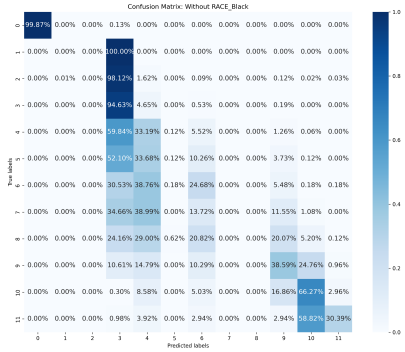
As one can see in 5a and 5b, the base network is highly biased against Black people. Chicago has dealt with racist policing in the past, so it is not surprising that Black people are likely to get substantially harsher penalties compared to non-Black people in the criminal justice system. In fact, the model almost never predicts that Black people get the lightest punishment while there rest of the population has very high accuracy for most categories. While Black people get stiff penalties for low level offenses, other people get more accurate predictions for valid sentences. This means that being Black substantially increases the likelihood that the model predicts a stiffer sentence for you.

However, when we implement the CAV approach to enforced fairness, we find that this effect is almost entirely mitigated. While it takes tuning to get the coefficients correct, one can trivially use distribution difference methods to tune the parameters to make the distributions of predicted classes as similar as possible to enforce fairness. Figures 6a and 6b show that the introduction of the latent activation delta can force the distribution for targeted populations to represent the population distribution substantially better.

And, because of the versatility of this approach, one can control for essentially any concept that is in the inputs. For instance, in this situation, one could simultaneously control for the arresting police department, the judge presiding over the court, the income of the accused, and the race of the accused amongst other variables to enforce fairness in pursuit of a truly just criminal justice system.



(a) Concept-Dampening-Model Confusion matrix of simple model incarceration outcomes for Black people



(b) Confusion matrix of simple model incarceration outcomes for non-Black people

Figure 6: Confusion Matrixes for Concept-Dampening-Model Performance

6 Conclusion

6.1 Key Takeaways

I sought to introduce new frameworks for observable and fair neural networks without compromising on model effectiveness to open the black box of neural networks. The concept-focused training approach detailed in this project allows for stakeholders to understand the concepts that the model believes are most important to inference while influence re-weighting operations remove such harmful concepts' impact on the output of the model to enforce fairness. Finally, when models are trained and deployed, stakeholders can be certain that their models are being provably fair through using concept activation vectors to minimize the prevalence of harmful concepts in the decision process.

These approaches do not compromise the model's accuracy or inference speed and should be integrated to ensure models in sensitive environments are fair and explainable. Without such broad methods, stakeholders take on too much risk and cannot claim to be deciding fairly. Using this paper's methods, one can understand individual features' and model-defined concepts' instance-level contributions and remove said influence – a game changer for sensitive ML.

6.2 Limitations of Method

Despite the forward progress this study has made into explainability, it suffers from some drawbacks which warrant further exploration. First: sometimes the explainability method just doesn't work. I estimate that I am able to guide the network towards defining and utilizing the PCA concepts roughly 70% of the time. With further tuning and optimization of the gradient boosting, this success probability can likely be increased. But, the potential for failure increases training time and costs.

Additionally, I found that the optimal clustering scheme is data-dependent. K-Means seems to have the most reliably good results, but is often not the best.

Finally, the method is time-consuming. Even though I was able to dramatically decrease the computational time required through both the ingrained explainability and fairness methods, any gradient manipulation will increase the training time for a machine learning model – especially at scale. The method is a linearly-scaling method with the complexity of the model and number of input features, but on complex systems even a 10% increase in training time can lead to millions of dollars in costs.

Cited References

[1] David Alvarez-Melis and Tommi S. Jaakkola. "Towards Robust Interpretability with Self-Explaining Neural Networks". In: *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS)*. Montréal, Canada, 2018.

[2] American Civil Liberties Union of Illinois. *Stop and Frisk in Chicago*. Mar. 2015. URL: https://www.aclu-il.org/en/publications/stop-and-frisk-chicago.

[3] Viktor Bachratý. "Machine Learning With Highly Sensitive Data". In: *Jumio Engineering & Data Science* (Aug. 2020). Accessed: 2024-05-03. URL: https://medium.com/jumio-engineering-data-science/machine-learning-with-highly-sensitive-data-%3Carticle-id%3E.

[4] Steven Bills et al. *Language models can explain neurons in language models*. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html. 2023.

[5] Alexander Binder et al. *Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers*. 2016. arXiv: 1604.00825 [cs.CV].

[6] Paul J. Blazek and Milo M. Lin. "Explainable neural networks that simulate reasoning". In: *Nature Computational Science* 1 (2021), pp. 607–618. DOI: 10.1038/s43588-021-00132-w.

[7] Dallas Card, Michael Zhang, and Noah A. Smith. "Deep Weighted Averaging Classifiers". In: *FAT* '19: Conference on Fairness, Accountability, and Transparency*. ACM. Atlanta, GA, USA, 2019, p. 13. DOI: 10.1145/3287560.3287595.

[8] M. Keith Chen et al. *Smartphone Data Reveal Neighborhood-Level Racial Disparities in Police Presence*. 2022. arXiv: 2109.12491 [econ.GN].

[9] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. ACM, Aug. 2016. DOI: 10.1145/2939672.2939785. URL: http://dx.doi.org/10.1145/2939672.2939785.

[10] City of Chicago. *City of Chicago Data Portal*. 2024. URL: https://data.cityofchicago.org/.

[11] Harrison Clarke. *Mastering MLOps: Best Practices for Secure Machine Learning Systems*. Accessed: 2024-05-03. 2024. URL: https://www.harrisonclarke.com/blog/mastering-mlops-best-practices-for-secure-machine-learning-systems.

[12] Cook County, Illinois Data Catalog. *Sentencing*. 2024. URL: https://datacatalog.cookcountyil.gov/Courts/Sentencing/tg8v-tm6u/data.

[13] Stefan Druc et al. *Concept Activation Vectors for Generating User-Defined 3D Shapes*. 2022. arXiv: 2205.02102 [cs.CV].

[14] Geeks for Geeks. *How to encrypt and decrypt strings in Python*. https://www.geeksforgeeks.org/how-to-encrypt-and-decrypt-strings-in-python/. Accessed: 2024-05-03. 2023.

[15] Will Douglas Heaven. "Predictive policing algorithms are racist. They need to be dismantled." In: *MIT Technology Review* (July 2020). Accessed: date-of-access. URL: https://www.technologyreview.com/2020/07/17/1005396/predictive-policing-algorithms-ai-racism/.

[16] Been Kim et al. *Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)*. 2018. arXiv: 1711.11279 [stat.ML]. URL: https://arxiv.org/abs/1711.11279.

[17] Wonjae Kim, Bokyung Son, and Ildoo Kim. *ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision*. 2021. arXiv: 2102.03334 [stat.ML].

[18] Nojun Kwak and Chong-Ho Choi. "Feature Extraction Based on ICA for Binary Classification Problems". In: *IEEE Trans. Knowl. Data Eng.* 15 (2003), pp. 1374–1388. DOI: 10.1109/TKDE.2003.1245279.

[19] Scott Lundberg and Su-In Lee. *A Unified Approach to Interpreting Model Predictions*. 2017. arXiv: 1705.07874 [cs.AI].

[20] F Martín-Fernández and P Caballero-Gil. *Analysis of the new standard hash function*. 2022. arXiv: 2209.11857 [cs.CR].

[21] Mary L McHugh. “Interrater reliability: the kappa statistic”. In: *Biochem Med (Zagreb)* 22.3 (2012), pp. 276–282.

[22] G. Mclachlan. “Mahalanobis Distance”. In: *Resonance* 4 (June 1999), pp. 20–26. DOI: 10.1007/BF02834632.

[23] Yuxian Meng et al. “GNN-LM: Language Modeling based on Global Contexts via GNN”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=BS49l-B5Bql.

[24] Jasmine Minor. *Traffic study finds CPD 6 times more likely to stop Black drivers*. ABC7 Chicago. Nov. 2023. URL: https://abc7chicago.com/chicago-police-traffic-stops-free2move-study-racial-profiling/14054056/.

[25] Anna Montoya et al. *Prudential Life Insurance Assessment*. https://www.kaggle.com/competitions/prudential-life-insurance-assessment. Kaggle competition. 2015.

[26] Josh Ray. *Generative AI Security: 11 Best Practices*. https://www.cloudticity.com/blog/generative-ai-security-11-best-practices. Accessed: 2024-05-03. Mar. 2024.

[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. arXiv: 1602.04938 [cs.LG].

[28] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. arXiv: 1602.04938 [cs.LG]. URL: https://arxiv.org/abs/1602.04938.

[29] Casey Ross. “Epic’s Overhaul of a Flawed Algorithm Shows Why AI Oversight is a Life-or-Death Issue”. In: *STAT* (2022). October 24, 2022. URL: https://www.statnews.com/2022/10/24/epics-overhaul-of-a-flawed-algorithm-shows-why-ai-oversight-is-a-life-or-death-issue/.

[30] Ramprasaath R. Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7.

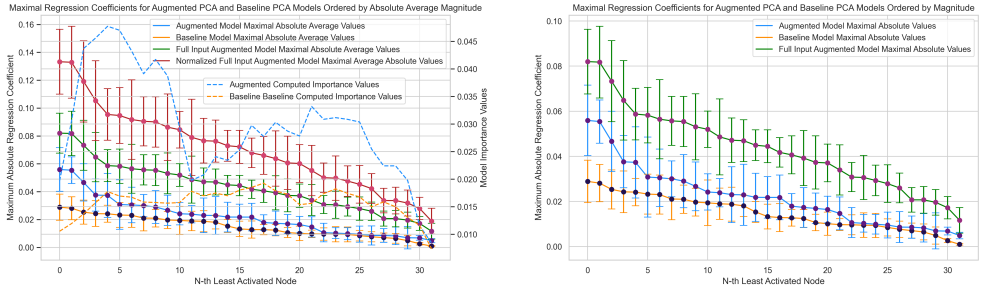
[31] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2014. arXiv: 1312.6034 [cs.CV].

[32] Andrew Wong et al. “External Validation of a Widely Implemented Proprietary Sepsis Prediction Model in Hospitalized Patients”. In: *JAMA Internal Medicine* 181.8 (Aug. 2021), pp. 1065–1070. ISSN: 2168-6106. DOI: 10.1001/jamainternmed.2021.2626. eprint: https://jamanetwork.com/journals/jamainternalmedicine/articlepdf/2781307/jamainternal_wong_2021_oi_210027_1627674961.11707.pdf. URL: https://doi.org/10.1001/jamainternmed.2021.2626.

[33] Andy Keidel Undergraduate Endowment Fund at Yale LinkedIn. *Andy Keidel Undergraduate Endowment Fund at Yale LinkedIn*. Accessed: 2023-08-04. 2023. URL: https://www.linkedin.com/company/andy-keidel-undergraduate-endowment-fund-at-yale/.

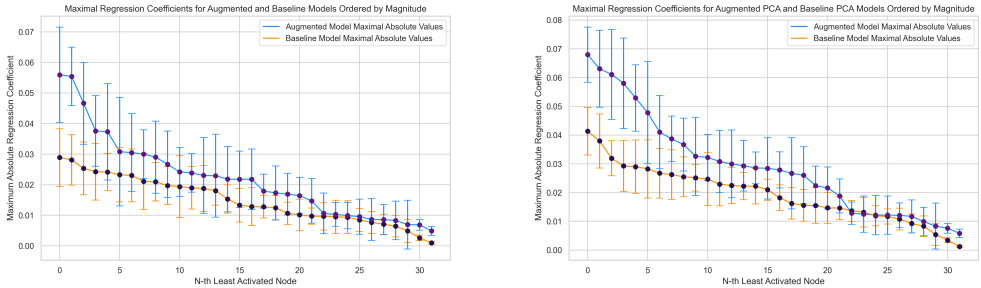
7 Supplementary Material

Below, the reader can find additional material that is helpful to the understanding of the project. Figures are included with brief explanations of their relevance to the project.



(a) Average Concept Importances with Normalized (b) Concept Importances without Normalized

Figure 7: Concept Importance Graphs



(a) Augmented vs. Baseline with no PCA concepts (b) Augmented vs. Baseline with PCA concepts

Figure 8: Concept Importance Graphs

Method	**Average Runtime (seconds)**
Amplitude Reduction	3.5191
Stochastic Fine Tuning	3.1223
Inversion Fine Tuning	3.1770
Simple Retraining	16.9567

Table 4: Average runtime across all removed PCA components for each method

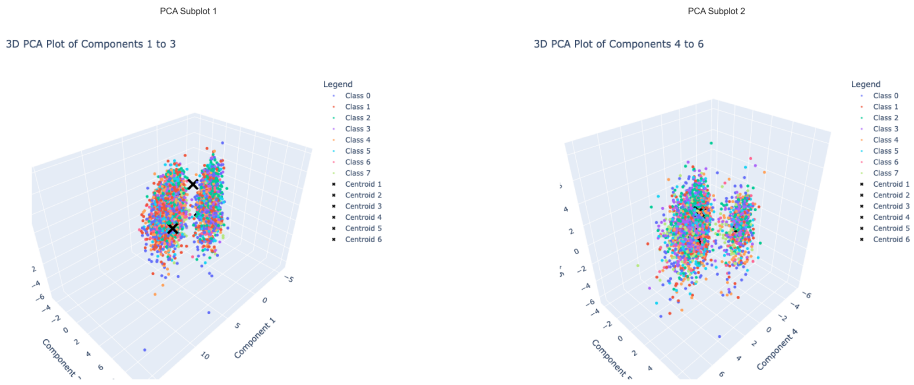


Figure 9: First and Second Sets of Principal Components with Centroids

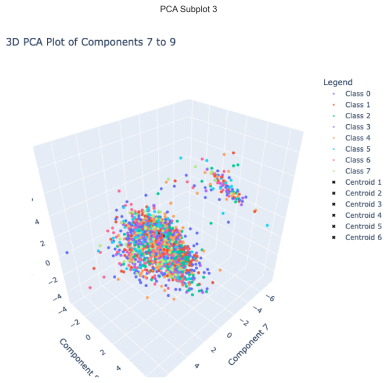
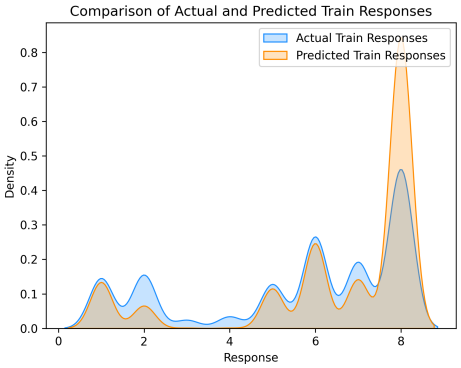
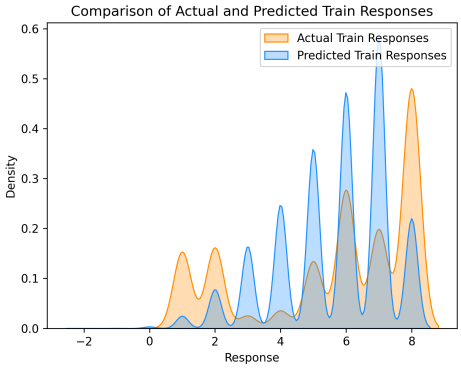


Figure 10: Third set of Principal Components with Centroids

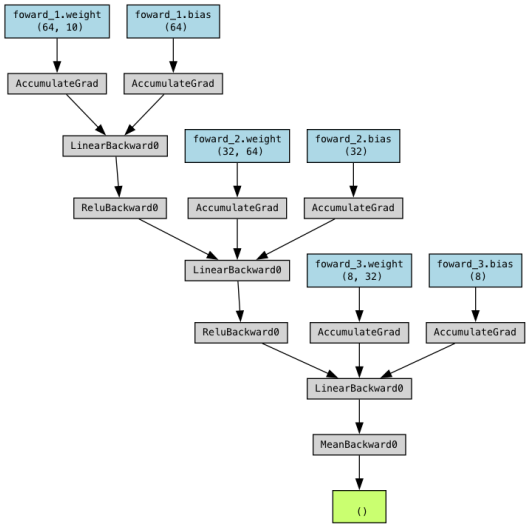


(a) Base Neural Network Train Accuracy

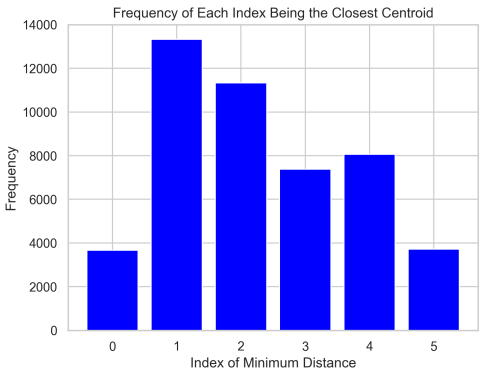


(b) XGBoost Train Accuracy

Figure 11: Baseline Model Accuracy



(a) Neural Network Structure



(b) Minimum Distance from Centroids Histogram

Table 5: Results for Removing Feature Influence by Different Methods

	Method	Validation Loss	Loss % Change	Importance Reduction %
Feature 0				
	amplitude_reduction	1.5313	1.90	-100.00
	stochastic_fine_tuning	1.5212	1.23	-65.93
	inversion_fine_tuning	1.5035	0.05	-22.73
	simple_retraining	1.5177	0.99	-100.00
Feature 1				
	amplitude_reduction	1.5306	1.85	-100.00
	stochastic_fine_tuning	1.5208	1.20	-67.40
	inversion_fine_tuning	1.5049	0.14	-21.84
	simple_retraining	1.5220	1.28	-100.00
Feature 2				
	amplitude_reduction	1.5307	1.86	-100.00
	stochastic_fine_tuning	1.5213	1.23	-68.57
	inversion_fine_tuning	1.5043	0.10	-23.48
	simple_retraining	1.5175	0.98	-100.00
Feature 3				
	amplitude_reduction	1.5298	1.80	-100.00
	stochastic_fine_tuning	1.5221	1.28	-68.39
	inversion_fine_tuning	1.5036	0.05	-25.63
	simple_retraining	1.5246	1.45	-100.00
Feature 4				
	amplitude_reduction	1.5325	1.98	-100.00
	stochastic_fine_tuning	1.5223	1.30	-68.70
	inversion_fine_tuning	1.5042	0.09	-23.88
	simple_retraining	1.5250	1.48	-100.00
Feature 5				
	amplitude_reduction	1.5308	1.86	-100.00
	stochastic_fine_tuning	1.5216	1.25	-73.66
	inversion_fine_tuning	1.5042	0.09	-24.75
	simple_retraining	1.5172	0.96	-100.00
Feature 6				
	amplitude_reduction	1.5308	1.87	-100.00
	stochastic_fine_tuning	1.5223	1.30	-65.87
	inversion_fine_tuning	1.5037	0.06	-23.23
	simple_retraining	1.5199	1.14	-100.00
Feature 7				
	amplitude_reduction	1.5307	1.86	-100.00
	stochastic_fine_tuning	1.5239	1.41	-66.78
	inversion_fine_tuning	1.5057	0.19	-22.88
	simple_retraining	1.5216	1.25	-100.00
Feature 8				
	amplitude_reduction	1.5306	1.85	-100.00
	stochastic_fine_tuning	1.5223	1.30	-66.80
	inversion_fine_tuning	1.5041	0.09	-18.94
	simple_retraining	1.5209	1.21	-100.00
Feature 9				
	amplitude_reduction	1.5311	1.88	-100.00
	stochastic_fine_tuning	1.5222	1.29	-65.39
	inversion_fine_tuning	1.5036	0.05	-21.85
	simple_retraining	1.5202	1.16	-100.00