Rotation 2: Report

Konstantin Kueffner

February 26, 2021

Abstract The application of common fairness measures, is often limited by the fact that required information is not explicitly provided as input feature, e.g. in face recognition ethnicity is not an explicit feature. To rectify this issues, this text introduces a procedure that given a pre-trained feed forward neural network and a predefined set of concepts, can detect the existence of such concepts in the input. Additionally, it is able to provide insights into the importance of each concept for a single prediction. Concept detection is accomplished using a set of linear classifier and some reasoner requiring domain knowledge. Concept importance, on the other hand, relies on the directional derivatives computed using the vectors extracted from said linear classifier. Having such a method can broaden the applicability of conventional fairness measures, e.g. for monitoring fairness on-line and can lead to the development of new "relevance based" fairness measures.

1 Introduction

In recent years learned classifier have been applied in areas such as crime prediction, credit scoring or recruiting. Thereby, creating situations where the lives of individuals are heavily impacted by the decisions of such models. Hence, it can be particularly damaging, if the applied model has learned to discriminate based on the "wrong" features, e.g. skin colour for the assessment of credit worthiness. Meaning that such models can pick up and amplify implicit biases in the data. This lead to the emergence of a new field of research trying to make decision made by learned systems interpretable and fair. Interpretability, w.r.t. neural networks, is concerned with determining why a network makes a certain decision, whereas fairness revolves around determining whether the predictions made by a network discriminate against certain pre-specified groups (Kusner et al. 2017; Denton et al. 2020). Technically both can be studied independently, however, there may be a benefit to employing interpretability methods in the assessment of fairness (Du et al. 2020). For example, it may be possible to detect a bias against a certain protected group in the networks prediction, while internally group membership is immaterial for the networks decision. In some cases this might even be necessary, e.g. in the area of image recognition where group membership is not made explicit in the data leading to the inability of

applying common fairness measures. To be more precise, if the inputs membership to a protected group, is not explicitly encoded as one of its features, common fairness measures can not be employed, as most of them require the input to be segmented based on those groups. Naturally, this limits their applicability. Moreover, common measures of fairness do not consider whether group membership is actually relevant for the decision or not. Hence, having a reliable method that can determine whether the input is part of a protected group, would broaden the applicability of common fairness measures, to scenarios where labelling is not known, e.g. during on-line monitoring of the network. Moreover, having methods that explain why the network made a certain decision, e.g. whether group membership was important for a particular prediction, allows for the assessment of fairness based on a more nuanced understanding of the networks internal processes.

Section 4 introduces such a method. It requires a trained neural network and datasets containing examples of the concepts one wants to extract. Using this it trains a set of linear classifier to detect concepts from the activations of a network, e.g. group membership in the fairness context. Having completed the off-line calibration phase, the procedure takes some input and uses those classifier to compute the likelihood of the concept being present. In the case of multiple concepts, the procedure requires domain knowledge to construct a consistent description of the input based on the specified concepts, e.g. a set of concepts containing the concept of dark and the concept light skin may be contradictory. Relying on the notion of concept sensitivity developed in (B. Kim et al. 2018) for their TCAV method (see Section 3), those linear classifier are then used to assess the sensitivity of this particular prediction to the detected concepts.

Some rudimentary test were performed, to assess the viability of the procedure. Section 5 describes the experimental set-up and Section 6 presents the corresponding results. While in the aggregate the experiments have shown that the procedure is capable of both detecting the concepts in question, as well as assessing their importance for a prediction, there seem to be some reliability issues. Hence, additional improvements and experiments would be required. For more on this see the discussion in Section 7.

Lastly, this method was designed to be the first step towards on-line monitoring of fairness. The appeal of which is to detect, possible shifts in the input distribution which may lead to unfair predictions.

1.1 Fairness

The notion of fairness in the neural network context can be considered from a prediction outcome and from a prediction quality standpoint. An example for the prior would be a hiring application that consistently selects men over women. Here this bias may arise due to trainings data explicitly discriminating against (possibly) hidden attributes or due to spurious correlations learned by the network from an unbiased dataset. An example of the latter would speech recognition, i.e. a network for speech recognition trained primarily on male

voices has trouble understanding female voices (Du et al. 2020).

To detect unfair treatment of protected groups, several measures have been proposed. Here the distinction between group and individual fairness measures can be made. Group fairness measures are statistical in nature, thus they aggregate inputs based on certain attributes and compare the predictions across groups. Included in this category are measures such as individual demographic parity, equality of opportunity or fairness through unawareness. By contrast, individual fairness measures assert that similar inputs should be treated similarly. That is, for a single input changing the group membership while all else remains equal should not influence the networks prediction, one prominent member of this category is the notion of counterfactual fairness (Kusner et al. 2017; Du et al. 2020).

A particularly prominent issues of group fairness measures is that it can lead to individually unfair decisions, in the name of achieving group fairness. For example, the less qualified person is hired in order to satisfy the quota induced by some group fairness measure. This suggests, that at least on the surface level there may be a conflict between those two approaches. Another problem with group fairness is that it permits fairness gerrymandering, which happens when fairness measures are used on a too coarse scale. For example, the network is fair with respect to skin colour and gender, but is biased against certain skin colour and gender combinations. That is, those measures can fail at the intersection of groups. Individual fairness measures, however, are plagued by the need to define what similarity means, including novel inputs not present in the trainings data (Binns 2020; Du et al. 2020).

1.2 Interpretability through Concepts

The high-dimensional internal state of a network is difficult to interpret by humans, thereby severely limiting the ability to justify why a network produces a certain classification. This lack of interpretability, is in conflict with recent demands on such systems, e.g. see the discussion about the "the right to explanation" (Selbst et al. 2017).

According to (B. Kim et al. 2018) there are two possible avenues one can take to rectify this problem. The first, would be to restrict oneself to interpretable models. The second, presupposes the existence of a pre-trained network and tries to interpret its prediction in a post-hoc manner. A common family of approaches subsumed by the latter, relies on the perturbation of input features and the corresponding network response to construct an interpretation for the prediction. Here it is possible to distinguish between local and global methods. That is, local methods try to provide an interpretation of the networks activity for a single data point, whereas global ones produce interpretations that hold true for most data points of a class (B. Kim et al. 2018). For a detailed survey of various explainable and interpretable artificial intelligence approaches see (Tjoa et al. 2020).

Firmly placed into the group of global, perturbation based methods, TCAV short for "Testing with Concept Activation Vectors" introduced in (B. Kim et

al. 2018), has the capabilities of formulating such interpretation using high-level concepts, rather than merely features of the input. This is accomplished by feeding the network examples of some concept and training a linear classifier on the activations of some layer in the network. This results in a vector separating the concept in question from random input. Finally, by computing the directional derivative w.r.t. the vector, one is able to approximate the sensitivity of a class to the concept in question.

However, TCAV is not the only approach trying to interpret a networks decisions using human understandable concepts. For example, the paper (Ghorbani et al. 2019) builds on TCAV by automatically extracting important concepts from the data rather than requiring them as inputs. Moreover, in (Goyal et al. 2019) the authors leverage causal reasoning as introduced by Halpern and Pearl in (Halpern et al. 2005) and generative adversarial networks to overcome some of the limitations of correlation based methods such as TCAV. Another approach, described in (Carter et al. 2019), is explicitly designed for understanding convolutional neural network. It projects the networks activation space onto a two-dimensional plane, making it explorable by humans and providing insights into how the network represents and relates concepts, e.g. related concepts can be used to induce miss-classifications (Carter et al. 2019).

Lastly, reasoning on a concept level is not limited to mere post-hoc interpretability, as there seems ample literature investigating how to develop networks that allow for compositional and context based reasoning, which is particularly relevant for tasks such as zero- or few-shot learning (Tokmakov et al. 2019; Misra et al. 2017). However, for the purposes of on-line concept detection (T. Kim et al. 2020) and (Koh et al. 2020) are particularly interesting, as both discuss approaches where specific areas of a network are associated with specific concepts. Thereby, providing clear interpretable access points that can be used for monitoring. The prior, relies on a multi-branch neural network architecture where each branch corresponds to a concept. The latter introduces a bottleneck layer into a feed forward network, such that each node of the layer encodes a particular concept. In the case of bottleneck networks, having such an interpretable and modifiable access point, even permits a rudimentary form of counterfactual (and thus to some extend causal) reasoning (T. Kim et al. 2020; Halpern et al. 2005).

2 Preliminaries

This section briefly introduces the terminology used in this report.

2.1 Data, Concepts and other

A dataset is denoted as $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$, with $\mathcal{X}$ being the set of all possible inputs and $\mathcal{Y} := \{y_1, \ldots, y_Y\}$ being the set of all possible labels. Naturally, $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ represent the training and test dataset respectively. Furthermore, for some $y \in \mathcal{Y}$ let $\mathcal{D}^y := \mathcal{D} \cap (\mathcal{X} \times \{y\})$. This text heavily relies on the notion

of concepts, thus for $C \in \mathbb{N}$ let $\mathcal{C} := \{c_1, \dots, c_C\}$ be some set of concepts. Let $\mathcal{D}_c \subset \mathcal{X} \times \{c\}$ be a dataset containing examples of $c \in \mathcal{C}$. Additionally, the described procedures require datasets containing examples of not-concepts, i.e. either random input or concepts not present in $\mathcal{C}$. This set of non-concepts is referred to as $\mathcal{E} := \{e_1, \dots, e_E\}$ for $E \in \mathbb{N}$ with the corresponding datasets being $\mathcal{D}_e$ for some $e \in \mathcal{E}$. Lastly, for some vector $\boldsymbol{x} = (x_1, \dots, x_n)$ and $1 \leqslant i \leqslant n$, let $\pi_i(\boldsymbol{x}) = x_i$ be the projection function.

2.2 Neural Networks and other Classifier

The TCAV and the procedure outlined in Section 4 require two types of classifier.

Firstly, neural networks. This text restrict its attention to end-to-end differentiable, feed forward neural networks in general, see (Goodfellow et al. 2016, p. 163), and convolution neural networks in particular, see (Goodfellow et al. 2016, p. 321). A trained neural network will be understood as a differentiable function $f : \mathbb{R}^n \to \mathbb{R}^m$.

Hence, any input data must be flattened into a vector, and the output $\hat{\boldsymbol{y}} \in \mathbb{R}^m$ is given as one-hot encoded vector, and thus must later be translated into the corresponding $y \in \mathcal{Y}$. This restriction is primarily to employ a single notation for all networks under consideration and can be made without loss of generality (B. Kim et al. 2018). Moreover, the function representing the trained network is the composition of $L \in \mathbb{N}$ separate functions, corresponding to the layers of the network $\mathcal{L} := \{l_1, \dots, l_L\}$, i.e. $f = f^{l_L} \circ \cdots \circ f^{l_1}$ with $f^l : \mathbb{R}^{n_l} \to \mathbb{R}^{m_l}$ for $l \in \mathcal{L}$. Similar to above, inputs are assumed to be flattened, which is relevant in the convolutional network context. Moreover, if f^l is applied to some $\boldsymbol{x} \in \mathbb{R}^{n_l}$, its output is referred to as the activation of the network in layer l. For simplicity, let $f^{\leqslant l}(\boldsymbol{x}) := f^{l_l} \circ \cdots \circ f^{l_1}(\boldsymbol{x})$ and $f^{>l}(\boldsymbol{x}) := f^{l_L} \circ \cdots \circ f^{l_{l+1}}(\boldsymbol{x})$ for the appropriate $\boldsymbol{x}$.

Secondly, support vector machines (SVMs). Given a binary labelled dataset $\mathcal{D}_{train} \subset \mathbb{R}^d \times \{-1, 1\}$ for $d \in \mathbb{N}$, a support vector machine finds a hyperplane segmenting $\mathbb{R}^d$ while complying with the labelling present in $\mathcal{D}_{train}$ and maximising the margin, i.e. distance from the hyperplane to the closest point in $\mathcal{D}_{train}$. This is achieved by solving

$$\min_{\boldsymbol{v} \in \mathbb{R}^d} \frac{1}{2} \|\boldsymbol{v}\|^2 \quad \text{subject to } \forall (\boldsymbol{x}, y) \in \mathcal{D}_{train}\ y \langle \boldsymbol{v}, \boldsymbol{x} \rangle \geqslant 1$$

resulting in the classifier $g(\boldsymbol{x}) = \operatorname{sign} \langle \boldsymbol{v}, \boldsymbol{x} \rangle$. Intuitively, speaking $\boldsymbol{v}$ is a vector orthogonal to the hyperplane representing the decision boundary, directed towards the segment of the dataset labelled with 1 (Mohri et al. 2018, p. 64).

The performance of a classifier h, trained on $\mathcal{D}_{train}$, can be measured by its accuracy. The accuracy is assessed by sapling a test dataset $\mathcal{D}_{test}$ from the same distribution as $\mathcal{D}_{train}$ and computing

$$\operatorname{acc}(h) := \frac{|\{h(\boldsymbol{x}) = y \mid (\boldsymbol{x}, y) \in \mathcal{D}_{test}\}|}{|\mathcal{D}_{test}|}$$

Assuming that the prediction produced by h are sets the accuracy is measured as follows

$$\mathrm{acc_s}(h) := \frac{\sum_{(\boldsymbol{x},y)\in\mathcal{D}_{test}} Eq\big(h(\boldsymbol{x}), y\big)}{|\mathcal{D}_{test}|}$$

with

$$Eq(A,B) := \begin{cases} \frac{|A\cap B|}{B} & if\ |A| \leqslant |B| \\ 0 & otw. \end{cases}$$

2.3 Fairness Measures

Some important fairness measures are briefly introduced here. To that end, let $\hat{y}$ be the prediction of the network f, y the ground truth label, z a variable corresponding to group membership and A and B two groups.

- *Fairness Through Unawareness*, simply requires that no protected attributes are used explicitly in the decision making process (Kusner et al. 2017)
- *Demographic Parity*, requires that each group should have a similar probability of a certain classification, i.e. $\frac{p(\hat{y}|z=A)}{p(\hat{y}|z=B)} \geqslant \epsilon$ where $\epsilon \in [0,1]$ is some threshold permitting regulating similarity.
- *Equality of Opportunity*, acknowledges that different groups exhibit different distributions of y, essentially comparing the false- or the true-positive rates between groups, i.e. $p(\hat{y}=1|z=A, y=0) - p(\hat{y}=1|z=B, y=0)$ and $p(\hat{y}=1|z=A, y=1) - p(\hat{y}=1|z=B, y=1)$.
- *Individual Fairness (general)*, is fair if similar individuals have similar predictions, i.e. for some distance metric d_{Fair} and two data points $\boldsymbol{x}$ and $\boldsymbol{x'}$ and some $\epsilon \geqslant 0$, if $d_{Fair}(\boldsymbol{x}, \boldsymbol{x'}) < \epsilon$ then $f(\boldsymbol{x}) = f(\boldsymbol{x'})$.

For more on those metrics see (Kusner et al. 2017) and (Du et al. 2020). Moreover, the introduction of counterfactual fairness would be out of the scope of this text, thus the reader may be referred to the prior reference.

3 TCAV

The TCAV method is designed to globally gauge how sensitive the class of a neural network is to a human defined concept. It requires as input a neural network f, which was previously trained on a dataset $\mathcal{D}_{train}$ sampled from some distribution $p(x,y)$. Using this network, a single TCAV-score is computed with respect to a specific concept $c \in \mathcal{C}$, a target class $y \in \mathcal{Y}$ and a layer $l \in \mathcal{L}$. Furthermore, the procedure requires the user to provide examples for the concept, i.e. $\mathcal{D}_c$, as well as examples for the target class, i.e. $\mathcal{D}^y$. Moreover, it is necessary to provide some additional non-concept dataset $\mathcal{D}_e$, e.g. sampled from $p(x,y)$.

A concept activation vector (CAV) of a layer l for the concept c is computed by training a SVM to distinguish between inputs from $\mathcal{D}_c$ and $\mathcal{D}_e$ based on the activations of layer l. To be more precise, a CAV $\boldsymbol{v}^l_{c,e}$ is computed as follows. First, let $\mathcal{D}_{c,e} := \{(f^{\leqslant l}(\boldsymbol{x}), 1) \mid (\boldsymbol{x}, c) \in \mathcal{D}_c\} \cup \{(f^{\leqslant l}(\boldsymbol{x}), -1) \mid (\boldsymbol{x}, y') \in \mathcal{D}_e\}$, and let it be split into a training and a test dataset. The resulting SVM $g^l_{c,e}$, trained on the training part of $\mathcal{D}_{c,e}$, has the form $g^l_{c,e}(\boldsymbol{x}) = \text{sign}\left(\langle \boldsymbol{v}^l_{c,e}, \boldsymbol{x}\rangle\right)$. Hence, the computed CAV $\boldsymbol{v}^l_{c,e}$ is a vector pointing into the direction of the activations corresponding to the concept c. Lastly, the accuracy of the classifier is assessed using the test part of this dataset.

CAVs are integral for the computation of a TCAV-score $\text{tcav}^{l,y}_{c,e}$, because they provide the direction for the directional derivative used in the assessment of the networks sensitivity to the concept for a particular prediction. That is, rather than gauging the sensitivity to a single feature, as common in applications such as saliency maps, it allows one to approximate the sensitivity to a specific concepts. Let $S^{l,y}_{c,e}$ refer to the sensitivity of the prediction y to perturbations of the activations of l in the direction of $\boldsymbol{v}^l_{c,e}$, and it is defined as

$$S^{l,y}_{c,e}(\boldsymbol{x}) := \lim_{\epsilon \to 0} \frac{\pi_y\big(f^{>l}(f^{\leqslant l}(\boldsymbol{x}) + \epsilon)\big) - \pi_y\big(f(\boldsymbol{x})\big)}{\epsilon} = \nabla \pi_y\big(f(\boldsymbol{x})\big) \cdot \boldsymbol{v}^l_{c,e}$$

This sensitivity measure only reflects the sensitivity of a single prediction to the concept in question. Yet, with TCAV being a global measure, input independence must be achieved. This is accomplished by aggregating the directional derivatives for all $\boldsymbol{x} \in \mathcal{D}^y$. To be precise, $\text{tcav}^{l,y}_{c,e}$ is defined as

$$\text{tcav}^{l,y}_{c,e} := \frac{|\{(\boldsymbol{x}, y) \in \mathcal{D}^y \mid S^{l,y}_{c,e}(\boldsymbol{x}) > 0\}|}{|\mathcal{D}^y|}$$

One particular disadvantage of computing a single TCAV-score is that its viability heavily depends on the "quality" of the learned CAV, i.e. it assumes that $\boldsymbol{v}^l_{c,e}$ provides a meaningful segmentation of the activation space. To make the TCAV procedure more robust, a guard against potentially spurious CAVs must be established. This is accomplished by relying on a two-sided t-test. To be precise, rather than providing a single negative dataset, the user has to prepare several of such datasets, i.e. one $\mathcal{D}_e$ for each $e \in \mathcal{E} := \{e_1, \ldots, e_E\}$. Then, one computes for each e the corresponding TCAV-score $\text{tcav}^{l,y}_{c,e}$. If the two-sided t-test rejects the null hypothesis $\mathcal{H}_0 : \text{tcav}^{l,y}_{c,e} = 0.5$, the concept is considered to be significant for the target class y. The returned TCAV-score amounts in this scenario to the average over all computed scores. Furthermore, to reduce false positives even further, a Bonferroni correction is employed, i.e. $p < \frac{\alpha}{m}$ with $m = 2$ (B. Kim et al. 2018).

4 On-line Concept Detection

The long term goal of this work is to monitor fairness. This, however, requires access to specific information about the input, e.g. membership of the input to

a protected group. Unfortunately, such information is not always made explicit, e.g. there is no feature in the input data that references group membership. This lack of information is particularly apparent in image data, as the only features present are pixel values, thus requiring the need for a method that can extract the necessary information from a pre-trained network. This section, will use ideas from TCAV to develop such a method. It consists of two phases, an on- and an off-line phase, with the latter being used to calibrate the procedure for its on-line deployment.

During the *off-line* stage the procedure requires datasets containing examples of each concept that should be considered by the procedure, i.e. the user has to provide $\mathcal{D}_c$ for each $c \in \mathcal{C}$. Similar to TCAV, the procedure requires non-concept examples as well, i.e. $\mathcal{D}_e$ for each $e \in \mathcal{E}$. Using this it computes for each layer $l \in \mathcal{L}$, each concept c and each negative concept e the CAV $\boldsymbol{v}^l_{c,e}$. Those CAVs or more precisely their corresponding linear classifier $g^l_{c,e}$ and their accuracies are stored for the on-line stage of this procedure.

During the *on-line stage*, the processing of a single input $\boldsymbol{x} \in \mathcal{X}$ can be split into three components. First, the procedure has to assess what concepts are present in the input, which is accomplished by aggregating the predictions of each $g^l_{c,e}$ such that each concept c can be assigned a likelihood score. Second, the resulting list has to be reduced into a consistent set of concepts $\mathcal{K} := \{k_1, \ldots, k_K\} \subseteq \mathcal{C}$, representing the verdict of what concepts are present in the input. This step permits the user to provide a specification characterising, what a consistent set of concepts is and how it should be computed. Third, the procedure calculates the directional derivatives $S^l_{k,e}(\boldsymbol{x})$ for each $k \in \mathcal{K}$, which are subsequently combined to determine the importance of each concept for the networks prediction. The following makes this intuition more precise.

The first step computes for each concept the likelihood of being present in input $\boldsymbol{x} \in \mathcal{X}$. This is accomplished by averaging the classifiers decisions weighted by their normalised accuracies, i.e.

$$q^l_c(\boldsymbol{x}) = \sum_{e \in \mathcal{E}} \left(\max\left(0, g^l_{c,e}(f^{\leqslant l}(\boldsymbol{x}))\right) \cdot \frac{\operatorname{acc}(g^l_{c,e})}{\sum_{e' \in \mathcal{E}} \operatorname{acc}(g^l_{c,e'})} \right)$$

which is then averaged across all layers, i.e.

$$q_c(\boldsymbol{x}) = \frac{1}{L} \sum_{l \in \mathcal{L}} q^l_c$$

Using $q_c(\boldsymbol{x})$ the vector $q_{\mathcal{C}}(\boldsymbol{x}) := \big((c_1, q_{c_1}(\boldsymbol{x})), \ldots, (c_C, q_{c_C}(\boldsymbol{x}))\big)$ is constructed and passed to the subsequent step.

The second step, allows the user to pass a function $\lambda : \times_{i=1}^{C} \big(\{c_i\} \times [0,1]\big) \to \wp(\mathcal{C})$ encoding knowledge about the environment in which the neural network operates. The purpose of this function is to select the set of concepts that is most likely describes the input $\boldsymbol{x}$. This requires for example that any set of concepts, which contains contradictory entries is removed, i.e. $\boldsymbol{x}$ can not be member of two disjointed groups, e.g. $\boldsymbol{x}$ can not be rich and poor. Section 5 will present

several of those functions for the dataset used in the experiments. However, if no such function is provided, implying that any subset of $\mathcal{C}$ is permitted, the procedure uses the the minima of the Gaussian kernel density estimate or short KDE (Bishop 2006, p. 123) of $q_{\mathcal{C}}(\boldsymbol{x})$ to determine $\mathcal{K} := \lambda(q_{\mathcal{C}}(\boldsymbol{x}))$. That is, it sorts the $q_{\mathcal{C}}(\boldsymbol{x})$ based on the likelihoods, computes the Gaussian KDE, finds the minima of the resulting function (over the domain $[0, 1]$), and returns those concepts with a likelihood greater than the value of last minima. Let this function be called λ_0.

The third step determines which concepts found in $\mathcal{K}$ were important for the prediction y of the neural network f given input $\boldsymbol{x}$, i.e. $y = f(\boldsymbol{x})$. To accomplish this the weighted average of the directional derivatives for each concept, with respect to the corresponding classifiers accuracy is computed, i.e.

$$S_k^{l,y}(\boldsymbol{x}) = \sum_{e \in \mathcal{E}} \left(S_{k,e}^{l,y}(\boldsymbol{x}) \cdot \frac{\operatorname{acc}(g_{k,e}^l)}{\sum_{e \in \mathcal{E}} \operatorname{acc}(g_{k,e'}^l)} \right)$$

Using those aggregated derivatives, construct $S_{\mathcal{K}}^{l,y}(\boldsymbol{x}) := \big(S_{k_1}^{l,y}(\boldsymbol{x}), \ldots, S_{k_K}^{l,y}(\boldsymbol{x})\big)$, which is subsequently normalised using the distance between the maximal and a minimal entries, i.e. $d\big(S_{\mathcal{K}}^{l,y}(\boldsymbol{x})\big) := |\max\big(S_{\mathcal{K}}^{l,y}(\boldsymbol{x})\big) - \min\big(S_{\mathcal{K}}^{l,y}(\boldsymbol{x})\big)|$ and averaged across all layers

$$S_k^y(\boldsymbol{x}) = \frac{1}{L} \sum_{i=1}^{L} \left(\frac{S_{\mathcal{K}}^{l_i,y}(\boldsymbol{x})}{d\big(S_{\mathcal{K}}^{l_i,y}(\boldsymbol{x})\big)} \right)$$

The resulting vector is an estimate of how important the observed concepts are for this particular prediction. Information, that can subsequently be passed to some reasoner constructed to detect violations of some specified notion of fairness.

5 Experiments

The experimental set-up used to test this procedure is rather insufficient. That is, the experiments revolve around some very simple computer generated datasets, containing small images depicting only basic shapes and colours, and a relatively small neural network, originally designed for hand written digit detection. The benefits of this set-up are, the possibility for quick testing on small machines with limited RAM and complete control over certain biases in the network. While suitable to for this early developmental stage, this prohibits any conclusive remarks on the performance of the procedure sketched in Section 4.

5.1 Classifier

The neural network used in all experiments is an adaptation of the convolutional neural network called "LeNet-5" developed in (LeCun et al. 1998). and was originally designed for classifying the hand written image digits found in the MNIST-dataset (LeCun et al. 1998). Once trained the network in question will

be a function $f_{LN} : \mathbb{R}^{32\times32\times3} \to \mathbb{R}^3$ that takes 32 by 32 pixel images with 3 colour channels and maps them to three output nodes. It consists of 8 layers, $f^1_{LN} : \mathbb{R}^{32\times32\times3} \to \mathbb{R}^{30\times30\times6}$ is a two-dimensional convolutional layer, $f^2_{LN} : \mathbb{R}^{30\times30\times6} \to \mathbb{R}^{15\times15\times6}$ is an average pooling layer, $f^3_{LN} : \mathbb{R}^{15\times15\times6} \to \mathbb{R}^{13\times13\times16}$ is a two-dimensional convolutional layer, $f^4_{LN} : \mathbb{R}^{13\times13\times16} \to \mathbb{R}^{6\times6\times16}$ is an average pooling layer, $f^5_{LN} : \mathbb{R}^{6\times6\times16} \to \mathbb{R}^{576}$ transitions input to the densly connected part of the network, the remaining layers are all densely connected, i.e. $f^6_{LN} : \mathbb{R}^{576} \to \mathbb{R}^{120}$, $f^7_{LN} : \mathbb{R}^{120} \to \mathbb{R}^{84}$ and $f^8_{LN} : \mathbb{R}^{84} \to \mathbb{R}^3$. Moreover, the activation function used throughout the network is the ReLU-function (rectified linear unit) and the kernel size of the all convolutional layers is 3 by 3.[1] The linear SVMs required to separate concept activations from random activations use a regularisation of 0.01

5.2 Datasets

All the datasets are generated by drawing shapes of various colours with random size, position and proportions on a black canvas with 32 by 32 pixel. The shapes in question are circles, triangles and rectangles, whereas the colours are red, green and blue. The set of concepts is thus defined as $\mathcal{C} := \mathcal{C}_{colour} \cup \mathcal{C}_{shape} \cup \mathcal{C}_{comp}$ with $\mathcal{C}_{colour} := \{cr, cg, cb\}$, $\mathcal{C}_{shape} := \{sc, st, sr\}$ and $\mathcal{C}_{comp} := \{(c_c, c_s) \mid c_c \in \mathcal{C}_{colour}\ c_s \in \mathcal{C}_{shape}\}$. The corresponding datasets are constructed as follows. For each $c = (c_c, c_s) \in \mathcal{C}_{comp}$, the dataset $\mathcal{D}_c$ contains only the shape c_s of the colour c_c, whereas for $c \in \mathcal{C}_{colour}$ the dataset $\mathcal{D}_c$ contains all shapes of a single colour and the dataset $c \in \mathcal{C}_{shape}$ contains a single shape coloured with every colour in $\mathcal{C}_{colour}$. In the latter two concept datasets, the shapes and colours are represented in equal proportions.

During the experiments two neural networks were trained, while both have the structure outline above, they differ in the trainings dataset. That is, the function f_{LN_c} was obtained by training f_{LN} to map shapes to output nodes irrespective of the colour of the shape. That is, the dataset $\mathcal{D}_{cont}$ was constructed using a function τ_{cont} mapping triangles to 0, squares to 1 and circles to 2, while at the same time ensuring each colour and shape combination occurs in equal proportions. By contrast, the dataset $\mathcal{D}_{bias}$ used for learning f_{LN_b}, has a disproportionally high green population. That is, 80% of the images present in $\mathcal{D}_{bias}$ contain green shapes, while the remaining amount to 10% of the population each. Similar to $\mathcal{D}_{cont}$ the shapes within each colour-population are equally distributed. Moreover, the labelling is again computed by a deterministic function τ_{bias}. However, it shifts red shapes by -1 and blue shapes by $+1$. For details see Table 1. Lastly, the viability of the outlined procedure is tested by constructing a dataset $\mathcal{D}_{online}$ containing all shape and colour combinations in equal proportions. Moreover, all images are labelled with the corresponding colour-shape pair.

[1] See (Goodfellow et al. 2016, p. 330) for more on pooling, (Goodfellow et al. 2016, p. 330) for more on the ReLU-function and (Goodfellow et al. 2016, p. 321) for more on convolutional neural networks in general.

τ_{cont}	Red	Green	Blue
Triangle	0	0	0
Rectangle	1	1	1
Circle	2	2	2

τ_{bias}	Red	Green	Blue
Triangle	0	0	1
Rectangle	0	1	2
Circle	1	2	2

Table 1: Function used to label the datasets $\mathcal{D}_{cont}$ and $\mathcal{D}_{bias}$

5.3 Concept Aggregation

The outline procedure requires some form of function to determine what permissible sets of concepts are. Considering that during the simulated on-line stage only images similar to those in $\mathcal{D}_{online}$ can be observed, the following conditions for a viable concept set can be formulated. First, there is exactly one colour, exactly one shape and one composite concept present in the image. Second, the colour and the shape of the composite concept must be consistent with the colour and shape in the set. Any set violating those conditions is dismissed. To encode the relationships between the concepts, each of the subsequent functions has access to a labelled directed acyclic graph embedding of the concept and their relations. To be precise $\mathcal{G} := (V, E)$ with $V := \mathcal{C}$ and $E := \{(c, c_c) \mid c \in \mathcal{C}_{comp}\ c_c \in \mathcal{C}_{colour}\} \cup \{(c, c_s) \mid c \in \mathcal{C}_{comp}\ c_s \in \mathcal{C}_{shape}\}$. For some $v \in V$ let $\deg^+_{\mathcal{G}}(v)$ and $\deg^-_{\mathcal{G}}(v)$ be the out and in-degree respectively, in analogue let $N^+_{\mathcal{G}}(v)$ be the set of successors and $N^-_{\mathcal{G}}(v)$ be the set of predecessors.

Considering the input vector $\boldsymbol{w}$ fed into all functions listed below is of the form $\big((c_1, q_{c_1}(\boldsymbol{x})), \ldots, (c_C, q_{c_C}(\boldsymbol{x}))\big)$, let $\mu_{\boldsymbol{w}} : \mathcal{C} \to [0, 1]$ be the mapping from concepts to the concept likelihood, as specified by the input vector $\boldsymbol{w}$. Moreover, let filter be the function checking whether a set of concepts has the correct size.

$$\text{filter}(A) := \begin{cases} A & \text{if } |A| = 3 \\ \{\} & otw. \end{cases}$$

Moreover, let Γ be the function that completes the set of concept with respect to the graph $\mathcal{G}$, i.e.

$$\Gamma(A) := \begin{cases} A & \text{if } A = \Gamma(A) \\ \Gamma\left(A \cup \bigcup_{c \in A} N^+_{\mathcal{G}}(c) \cup \{c \mid c \in A \wedge N^-_{\mathcal{G}}(c) \subseteq A\}\right) & \text{otw.} \end{cases}$$

The function λ_1, simply selects $c_c \in \mathcal{C}_{colour}$ and $c_s \in \mathcal{C}_{shape}$ such that the total likelihood is maximised. If there is a tie the function returns the empty set. That is, let

$$\lambda_1(\boldsymbol{w}) := \text{filter}\left(\Gamma\left(\operatorname*{argmax}_{c \in \mathcal{C}_{colour}}\left(\mu_{\boldsymbol{w}}(c)\right) \cup \operatorname*{argmax}_{c \in \mathcal{C}_{shape}}\left(\mu_{\boldsymbol{w}}(c)\right)\right)\right)$$

The function λ_2, simply selects the maximal $c \in \mathcal{C}_{comp}$. If there is a tie the function returns the empty set. That is, let

$$\lambda_2(\boldsymbol{w}) := \text{filter}\left(\Gamma\left(\operatorname*{argmax}_{c \in \mathcal{C}_{comp}}\left(\mu_{\boldsymbol{w}}(c)\right)\right)\right)$$

The function λ_3 combines the likelihoods of concepts with those of their sub-concepts using a weighted average. Subsequently it selects the most likely concept without any super-concepts. To that end let

$$\mu'_{\boldsymbol{w}}(c) := \begin{cases} \mu_{\boldsymbol{w}}(c) & \text{if } \deg^+_{\mathcal{G}}(c) = 0 \\ \frac{1}{2}\mu_{\boldsymbol{w}}(c) + \frac{1}{2 \cdot \deg^+_{\mathcal{G}}(c)} \sum_{c' \in N^+_{\mathcal{G}}} \mu'_{\boldsymbol{w}}(c') \} & \text{otw.} \end{cases}$$

and thus

$$\lambda_3(\boldsymbol{w}) := \text{filter}\left(\Gamma\left(\underset{c \in \mathcal{C}_{comp}}{\text{argmax}}\left(\mu'_{\boldsymbol{w}}(c)\right)\right)\right)$$

5.4 Implementation Details

The complete procedure and the corresponding experiments were implemented in python and can be found on the internal GitLab page of IST-Austria.[2].

Moreover, the implementation uses TensorFlow[3] 2.x to construct and train the neural networks described below (Martin Abadi et al. 2015). However, the original implementation of TCAV, found on GitHub[4], was at the time of writing implemented using TensorFlow 1.x only. Therefore, any results discussed in Section 6, were obtained using a modified version of TCAV[5] which is the result of migrating TCAV to TensorFlow 2.x.

Additional, the datasets were drawn using OpenCV[6] (Bradski 2000) and the linear classifier used in TCAV rely on Scikit-learn[7] (Pedregosa et al. 2011) for their implementation.

5.5 Experimental Set-up

The testing and the evaluation of the procedure was done as follows. First, the procedure is calibrated during its off-line phase using the network and the datasets mentioned above. To do so, the procedure samples 500 images from each class and non-class dataset to train the linear classifier.

To assess the performance of the procedure with respect to concept detection. The procedure is fed with 100 batches from $\mathcal{D}_{online}$ each containing 50 entries. The per batch accuracy is assessed using acc_s. Subsequently, the standard deviation of the batch accuracy is calculated.

To assess the performance of the procedure with respect to concept importance. The procedure is fed with 50 batches from $\mathcal{D}_{online}$ each containing 50 entries. The best performing λ function, i.e. λ_3, is used. First, the results are cleaned by setting all results smaller than 0 to 0, because for a concept to be important the directional derivatives have to be positive. Second, the distributions of the results for shape and colour are contrasted. Third, a table summarising

[2] https://git.ist.ac.at/kkueffne/fcm
[3] https://www.tensorflow.org/
[4] https://github.com/tensorflow/tcav/tree/master/tcav
[5] https://git.ist.ac.at/kkueffne/fcm
[6] https://opencv.org/
[7] https://scikit-learn.org/stable/

which concept dominated in how many decision is constructed. Lastly, to provide additional insight into the results, the confusion matrices for the concept detection in this scenario is provided as well.

6 Results

By executing the outlined test procedure, one can observe that for the network trained on $\mathcal{D}_{bias}$ the concept detection accuracy is $40.9(\pm 3.9)\%$ for λ_0; $63.68(\pm 4.59)\%$ for λ_1; $59.27(\pm 6.07)\%$ for λ_2; $72.31(\pm 4.58)\%$ for λ_3. Similarly for the network trained on $\mathcal{D}_{cont}$ one can observe an accuracy of $34.2(\pm 2.5)\%$ for λ_0; $78.5(\pm 4.97)\%$ for λ_1; $85.02(\pm 4.38)\%$ for λ_2; $86.0(\pm 4.16)\%$ for λ_3. Figure 1 provides an overview of the accuracy rates for the experiments.

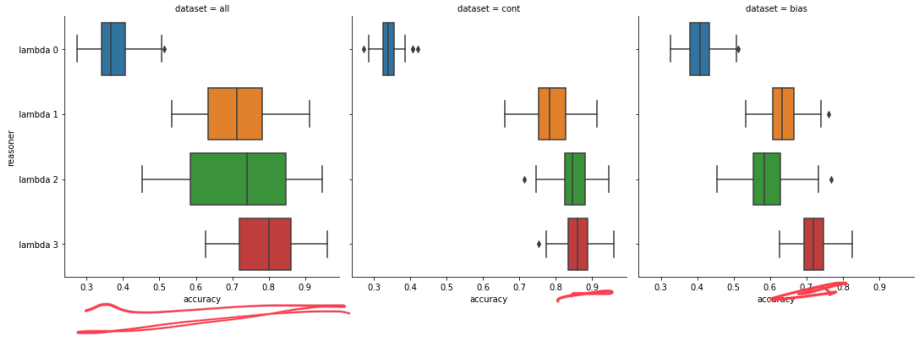

Figure 1: Box plot comparing the accuracy rates over all experiments and over the experiments using f_{LN_b} and f_{LN_c} respectively.

As for the experiments concerning concept importance, Figure 2 summarises the values obtained by the procedure of Section 4. One can observe that, in the control scenario, i.e. where colour concepts are not relevant for the labelling function (see Table 1), within each of the three colour groups, the median for the shape concepts is greater than the median for colour concepts. Particularly, of note is the unexpected results for the colour blue in the control setting, considering that colour was not part of the labelling function. By contrast, in the biased scenario, i.e. where the labels are shifted based on colour, one can observe that the colour concepts are considered to be more important, based on the median within each group. Moreover, the distance between the shape and colour median is more pronounced within the groups of red and the group of blue shapes. Additionally, Figure 3 depicts the KDE of the distribution aggregated in Figure 2. Lastly, in Figure 4 one can observe that a significant amount of miss-classification occurred within the group of blue and red shapes.

A less cohesive picture arises when aggregating the data based on the number of predictions where the colour concepts was more important and contrast them with the number of predictions where the shape concepts were more important. Table 2 summarises this particular view on the data based on group membership.

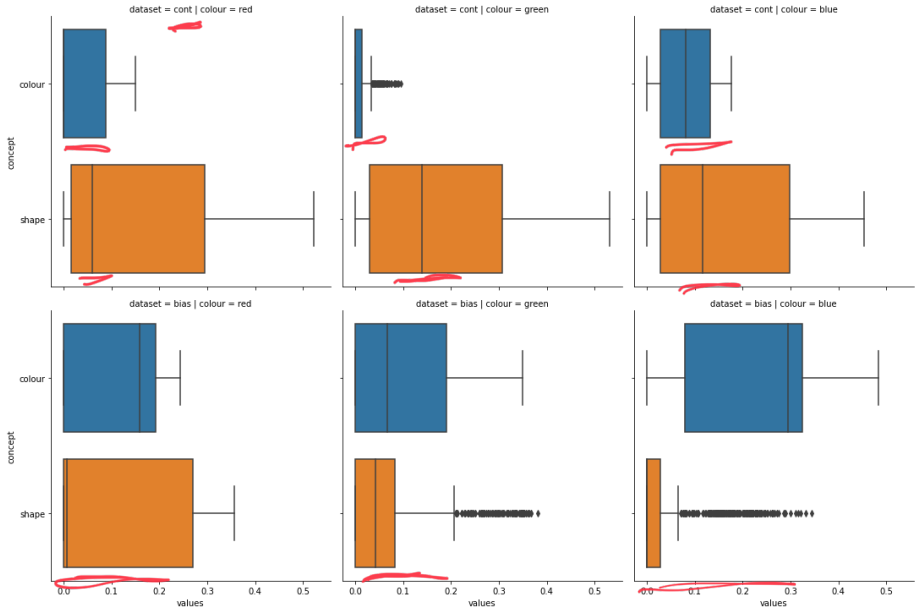

Figure 2: Box plot comparing the median of the values computed by the procedure described in Section 4. The rows represent the experiments, i.e. based f_{LN_c} or f_{LN_b}. The columns represent predicted group membership, i.e. whether the input shape was deemed to have colour red, green or blue. The hue aggregates importance values computed by the procedure for either the shape or the colour concept.

in % network	colour	shape	colour	undef.	shape	shape	colour	undef.
f_{LN_c}	blue	53.21	46.15	0.64	circle	67.10	30.19	2.71
	green	91.12	5.52	3.36	rectangle	85.69	13.41	0.89
	red	85.79	8.12	6.09	triangle	86.85	5.75	7.40
f_{LN_b}	blue	23.97	75.85	0.18	circle	14.70	72.70	12.60
	green	51.45	44.33	4.22	rectangle	55.38	41.70	2.91
	red	57.81	23.65	18.53	triangle	52.17	43.50	4.33

Table 2: The data is grouped based on either the colour concepts or the shape concepts. Within this grouping this table depicts which percentage of the predictions deems either shape or colour more important. "undef." implies that the respective values are identical.

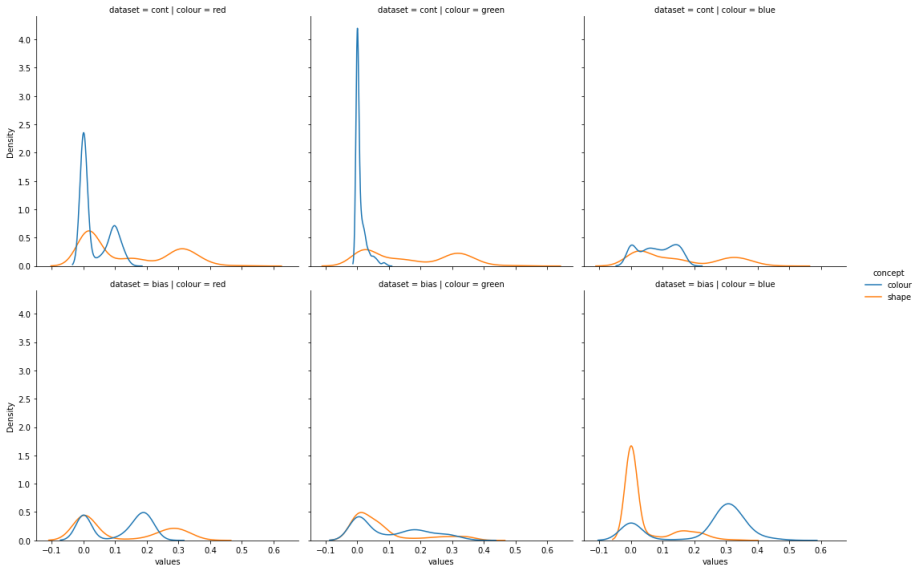

Figure 3: Plot comparing the KDE of the values computed by the procedure described in Section 4. Rows, columns and hue are to be interpreted as in Figure 2

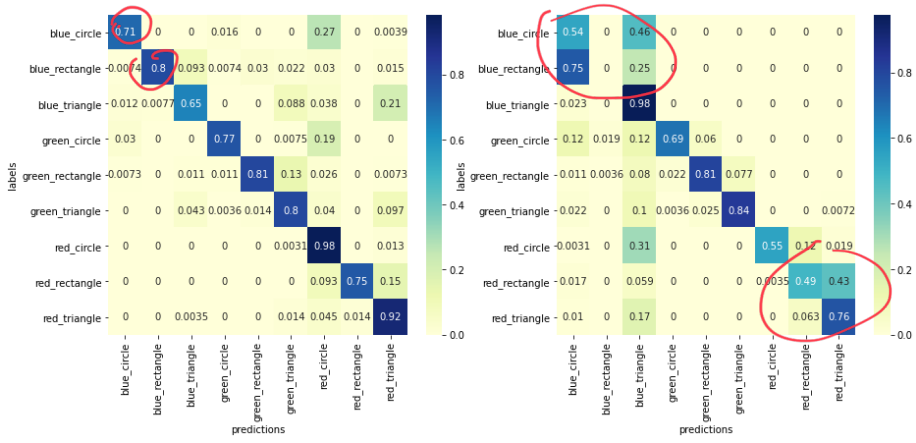

Figure 4: Confusion matrix for λ_3 computed using f_{LN_c} (left) and f_{LN_b} (right) in the concept sensitivity experiments

7 Discussion

The approach takes ideas from TCAV and translates them into an on-line setting. The challenge hereby, is that it is not known whether a certain input contains a particular concept. Simply computing the directional derivatives with respect to all CAVs is insufficient, as this can lead to non-existing concepts being considered as important. Given that the CAVs have to be computed to gauge importance, it was the economical choice to re-purpose those classifier to resolve this issue. Moreover, as shown in the experiments doing so while providing the algorithm with domain knowledge can lead to drastic improvements. An additional benefit over TCAV is that its output is well suited for the direct employment of common fairness measures (see Section 2).

The contributions of this approach in the context of fairness are twofold. Firstly, the procedure translates what is seen by the network into human understandable pre-defined concepts. This is crucial, as it permits the use of the fairness measures introduced in Section 2. Hence, it is possible to estimate fairness based on attributes that are not explicitly encoded in the features of the data. For example, consider the setting described in Section 5. The inputs in this case are merely images. Hence, the group membership (i.e. the colour) of each input is not known, thus it is not possible to use a measure such as demographic parity to assess the fairness of the networks prediction in an on-line setting. However, by having the ability to extract colour from the networks activation, one can now collect the necessary information to compute $\frac{p(\hat{y}|z=red)}{p(\hat{y}|z=green)}$. Moreover, by using the classifier on the networks activations, rather than the data itself, one may obtain additional assurance that the network actually observes colours. This claim, however, must be tested. Secondly, and more significantly, it opens up the possibility for more nuanced measures of fairness, that is rather than merely observing how the predictions change between inputs differing in certain attributes, one can make the measures dependent on whether the attribute was actually relevant for the prediction. For example, one could modify demographic parity, such that only those predictions are considered, where the importance of the protected group exceeds some threshold. That is, if one observes during on-line monitoring of the network $f_{\mathcal{LN}_c}$ that $p(\hat{y} = 2|z = red) > p(\hat{y} = 2|z = green)$. This would imply that under demographic parity unfair behaviour was detected. However, considering that almost none of the decisions made by the network deem colour to be important, such a judgement may have been premature.

7.1 Results

Above all, given the results additional tests and improvements are necessary.

The results for concept detection seem promising for the control dataset. However, on the biased dataset, a noticeable dip in performance can be detected. This may be explained by the fact that blue and red shapes are underrepresented in the dataset, a conjecture which is supported by the confusion matrix depicted in Figure 4. Moreover, the fluctuations in performance of λ_2,

observed between the biased and controlled case, could be explained by the lack of tie-breaking present in the function.

The expected outcome for the concept sensitivity experiments would have been that in the control setting almost all decisions were made based on the shape concepts. Moreover, in the biased setting a mixture of both would have been expected. However, it within this mixture one would expect to observe that colour concepts are more relevant for the red and blue shapes. While in the aggregate those expectations were satisfied. There are some irregularities that require additional attention.

In particular the observed asymmetry between the red and blue shapes is troublesome. That is, considering that the labelling function (see Table 1) treats them symmetrically, one would expect that they would exhibit similar behaviours. However, in the experiments the group of blue shapes perform as expected on the biased dataset, while the group of red shapes performs as expected on the control dataset, yet both fail in the respective other case. This, however, referrers to the "individual" perspective summarised in Table 2. By contrast, Figure 2 provides a more aggregated picture, in which the procedure performs as expected. Indicating that there might be large fluctuations between individual predictions, an assertion which seems to be supported by the more or less bi-modal distributions depicted in Figure 3.

Overall, more extensive testing and further improvements are required to hopefully construct a suitably reliable method. Some of which are discussed in the subsequent subsection.

7.2 Improvements and Critique

Several areas for improvement can be identified. Firstly, the average may hide important information. Particularly, worrisome is the average used for gauging the sensitivity of a prediction to a particular concept. The normalisation step that occurs within this average, is particularly susceptible to outliers. Hence, it may be sensible to either use the average distance between concepts, which is computationally quite expensive, or simply ignore possible long tails when calculating the distance. Alternatively, this average could be replaced entirely by a voting mechanism based on the relative positions of the concepts in each layer. This, however, would hide the actual distance between the concepts. Moreover, such a modification would still aggregate indiscriminately over all layers, which is not only inefficient, but could also limit the capabilities of the procedure. That is, the paper (B. Kim et al. 2018) mentions that concepts are not uniformly detected throughout the network, a claim supported by (Carter et al. 2019). Therefore, rather than aggregating over all layers, it may be sensible to focus on a single relevant layer (or a relevant subset of layers) per concept. The assessment of relevant layers, could rely on the information provided by TCAV. Hence, resulting in a procedure with a slightly more involved off-line phase and a more efficient on-line phase. Additionally, implementing a t-test similar to the one employed in the TCAV-procedure, could also improve accuracy and would aid in the tuning of the procedure.

An important addition to the procedure would be to replace λ with a specific reasoner operating over some specification language. Two possible formalisms seem to be natural candidates for this task. The first, being some kind of description logic (Krötzsch et al. 2013) and the second would be some answer set programming extension (Eiter et al. 2009). The prior, would provide the advantage of being already employed to model concepts and their relationships. Moreover, this would allow the user to access a large collection of predefined concept knowledge bases. The latter, comes to mind, because it its semantic revolves around the computation of maximal consistent sets. Furthermore, considering, the rather rudimentary nature of the common fairness measures and the fact that some of them fail in the intersection of attributes (Binns 2020), it may be desirable to develop a specification language that can not only model the relationships between concepts, but also can be used to express complex fairness criteria.

Lastly, this procedure has been constructed by re-purposing components of TCAV to allow for the on-line concept detection and the assessment of concept importance. However, an entirely different approach, building on the work in (Henzinger et al. 2019) could be envisioned. That is, one can replace the linear classifier with a box-abstraction and use those to detect which concepts are observed by the network. Moreover, a box naturally provides several decision boundaries, the vectors orthogonal to those boundaries can be used to compute the directional derivative "towards" the concept encased by the box.

8 Conclusion

The introduced procedure allows for rudimentary on-line concept detection, as well as the assessment of concept importance for a particular prediction. This broadens the applicability of conventional fairness measures, e.g. making them applicable in the context of on-line monitoring of fairness, and allows for the creation of new fairness measures based on concept importance. Experiments have shown that concept detection, in combination with additional reasoning using domain knowledge, seems to be quite reliable. Although, given the limited tests and the perceived unreliability, the concept importance part of the procedure requires additional testing. However, in the aggregate the results seem promising as well. Lastly, given the outlined potential shortcomings, further improvements seem likely.

References

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Bradski, G. (2000). "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools*.

Halpern, Joseph Y and Judea Pearl (2005). "Causes and explanations: A structural-model approach. Part I: Causes". In: *The British journal for the philosophy of science* 56.4, pp. 843–887.

Bishop, Christopher M (2006). *Pattern recognition and machine learning*. springer.

Eiter, Thomas, Giovambattista Ianni, and Thomas Krennwallner (2009). "Answer set programming: A primer". In: *Reasoning Web International Summer School*. Springer, pp. 40–110.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Krötzsch, Markus, Frantisek Simancik, and Ian Horrocks (2013). "Description logics". In: *IEEE Intelligent Systems* 29.1, pp. 12–19.

Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: `http://tensorflow.org/`.

Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio (2016). *Deep learning*. Vol. 1. 2. MIT press Cambridge.

Kusner, Matt J, Joshua Loftus, Chris Russell, and Ricardo Silva (2017). "Counterfactual fairness". In: *Advances in neural information processing systems*, pp. 4066–4076.

Misra, Ishan, Abhinav Gupta, and Martial Hebert (2017). "From red wine to red tomato: Composition with context". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1792–1801.

Selbst, Andrew D and Julia Powles (Dec. 2017). "Meaningful information and the right to explanation". In: *International Data Privacy Law* 7.4, pp. 233–242. ISSN: 2044-3994. DOI: `10.1093/idpl/ipx022`. eprint: `https://academic.oup.com/idpl/article-pdf/7/4/233/22923065/ipx022.pdf`. URL: `https://doi.org/10.1093/idpl/ipx022`.

Kim, Been, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. (2018). "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)". In: *International conference on machine learning*. PMLR, pp. 2668–2677.

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2018). *Foundations of machine learning*. MIT press.

Carter, Shan, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah (2019). "Activation Atlas". In: *Distill*. https://distill.pub/2019/activation-atlas. DOI: 10.23915/distill.00015.

Ghorbani, Amirata, James Wexler, James Y Zou, and Been Kim (2019). "Towards automatic concept-based explanations". In: *Advances in Neural Information Processing Systems*, pp. 9277–9286.

Goyal, Yash, Amir Feder, Uri Shalit, and Been Kim (2019). "Explaining classifiers with causal concept effect (cace)". In: *arXiv preprint arXiv:1907.07165*.

Henzinger, Thomas A, Anna Lukina, and Christian Schilling (2019). "Outside the box: Abstraction-based monitoring of neural networks". In: *arXiv preprint arXiv:1911.09032*.

Tokmakov, Pavel, Yu-Xiong Wang, and Martial Hebert (2019). "Learning compositional representations for few-shot recognition". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6372–6381.

Binns, Reuben (2020). "On the apparent conflict between individual and group fairness". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 514–524.

Denton, Emily, Ben Hutchinson, Margaret Mitchell, Timnit Gebru, and Andrew Zaldivar (2020). *Image Counterfactual Sensitivity Analysis for Detecting Unintended Bias*. arXiv: 1906.06439 [cs.CV].

Du, Mengnan, Fan Yang, Na Zou, and Xia Hu (2020). "Fairness in deep learning: A computational perspective". In: *IEEE Intelligent Systems*.

Kim, Taesup, Sungwoong Kim, and Yoshua Bengio (2020). "Visual Concept Reasoning Networks". In: *arXiv preprint arXiv:2008.11783*.

Koh, Pang Wei, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang (2020). "Concept bottleneck models". In: *International Conference on Machine Learning*. PMLR, pp. 5338–5348.

Tjoa, Erico and Cuntai Guan (2020). "A survey on explainable artificial intelligence (xai): Toward medical xai". In: *IEEE Transactions on Neural Networks and Learning Systems*.