Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (**TCAV**)

Paper Präsentation

Babar Ayan, Artiom Blinovas, Manyue Zhang

Einleitung: XAI

Das Verständnis, wie sich (komplexe) ML Modelle verhalten, wie z. B. neuronale Netze, und wieso sie bestimmte Entscheidungen treffen, bleibt eine große Herausforderung.

Es geht nicht nur darum, genaue Vorhersagen zu machen (Kluger Hans). Interpretierbarkeit liefert uns bessere Möglichkeiten bei ML Modellen:

- Entwurf
- Entwicklung
- Fehlerbehebung
- Fehlersuche

Abbildung: Kluger Hans. Quelle: [1].

Insbesondere um sicherzustellen, dass ML-Modelle unsere Werte widerspiegeln.

Saliency Maps

Saliency Maps verwenden die Ableitung zur Messung der Sensitivität der Ausgangsklasse k auf Veränderungen in der Größe des Pixels (a, b):

$$\frac{\partial h_k(x)}{\partial x_{a,b}}$$

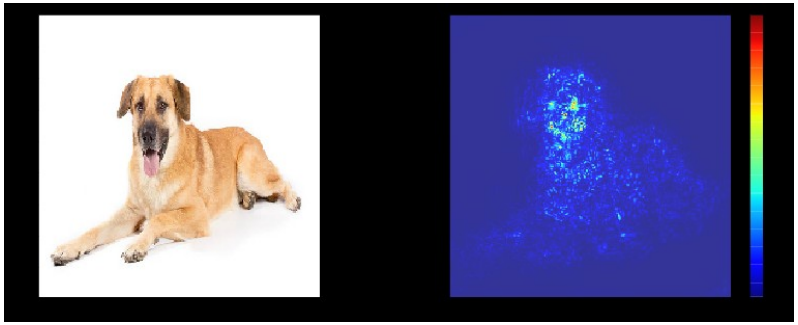

Abbildung: Saliency Map Hund. Quelle: [2].

TCAV

1. Erstellen eines Datensatzes für benutzerdefiniertes Konzept
2. Concept Activation Vectors ermitteln
3. Richtungsableitung berechnen

$$S_{C,k,l}(\boldsymbol{x}) = \lim_{\epsilon \to 0} \frac{h_{l,k}(f_l(\boldsymbol{x}) + \epsilon \boldsymbol{v}_C^l) - h_{l,k}(f_l(\boldsymbol{x}))}{\epsilon} = \nabla h_{l,k}(f_l(\boldsymbol{x})) \cdot \boldsymbol{v}_C^l, \quad (1)$$

4. TCAV Score berechnen

$$\text{TCAV}_{Q_{C,k,l}} = \frac{|\{\boldsymbol{x} \in X_k : S_{C,k,l}(\boldsymbol{x}) > 0\}|}{|X_k|} \quad (2)$$

5. Hypothesentest

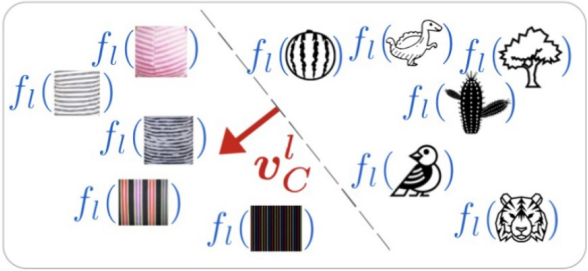

Abbildung: CAV. Quelle: [3].

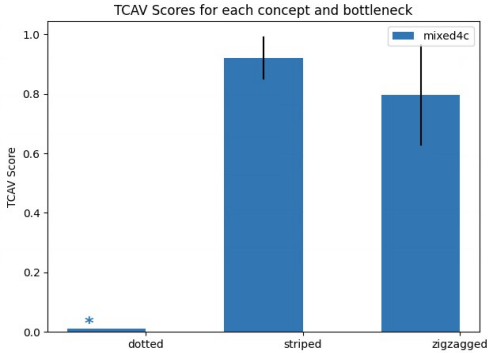

Abbildung: TCAV Score Quelle: Eigene Darstellung.

Die Ziele von CAV

- Accessibility
- Customization
- Plug-in readiness
- Global quantification

Validierung der gelernten CAVs

- Bilder nach ihrer Relevanz für Konzepte zu sortieren
- Überprüfung, ob ein CAV mit dem Zielkonzept übereinstimmt
- Cosinus-Ähnlichkeit zwischen einer Reihe von Bildern berechnen

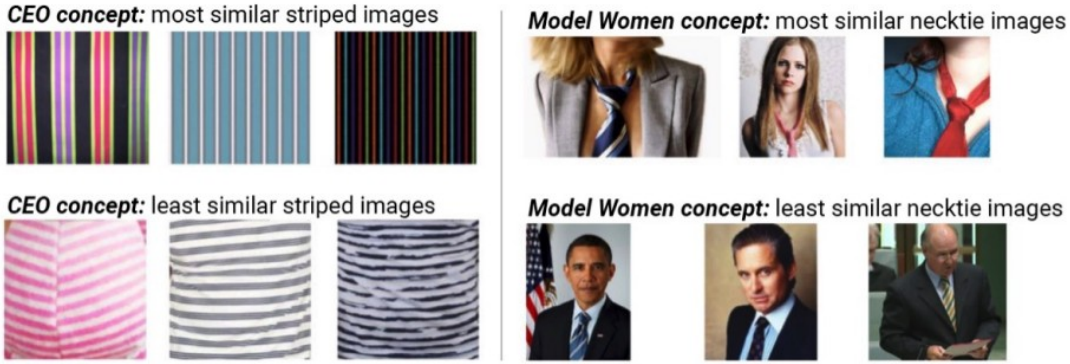

Abbildung: Concepts and similar images Quelle: [3].

Erkenntnisse aus Datensätzen Gewinnen mit TCAVs

Zwei weitverbreitete Modelle: GoogleNet und Inception V3

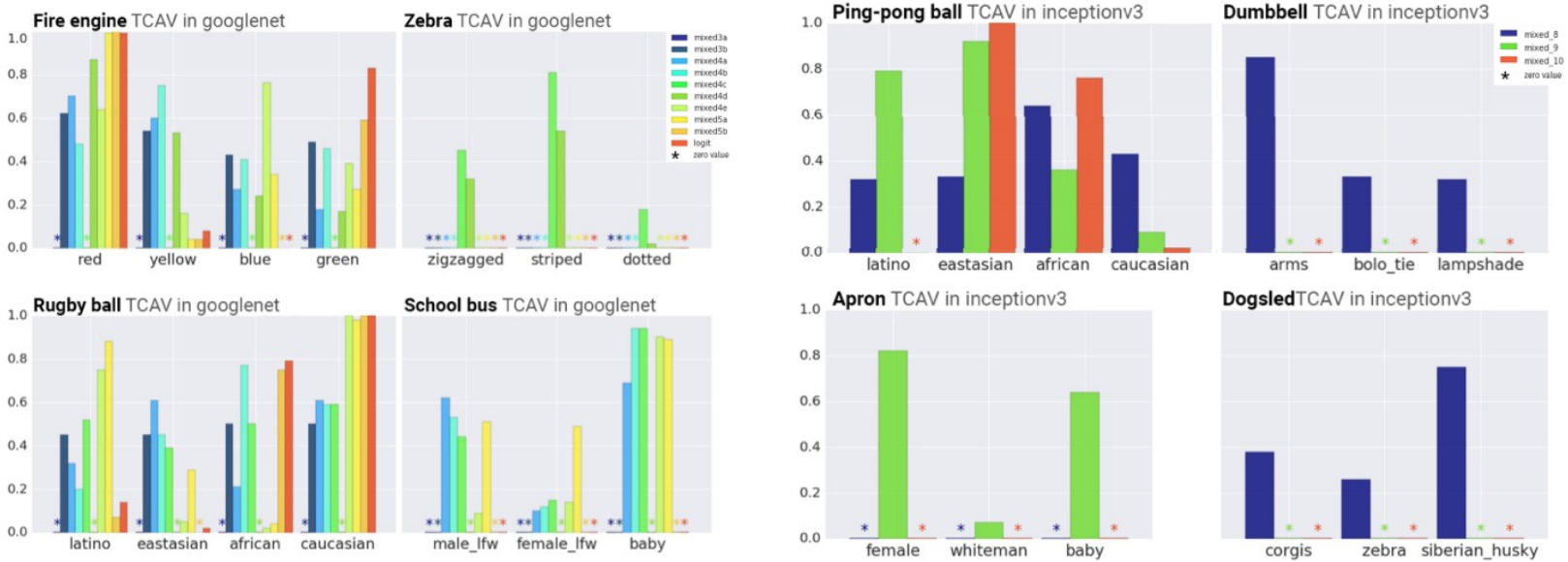

Abbildung: TCAV Score, angewandt auf Datensätze. Quelle: [3].

TCAV, Wo Konzepte erlernt werden

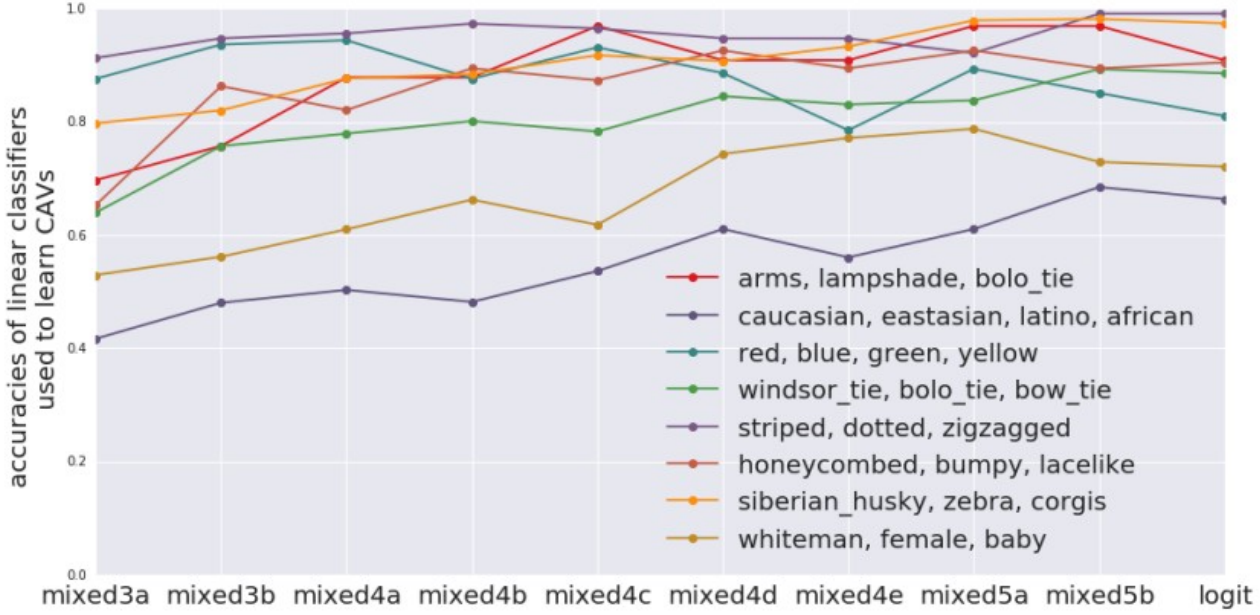

Abbildung: Aktivierungsschichten und Genauigkeit des linearen Klassifikators. Quelle: [3].

Code

https://github.com/tensorflow/tcav/ created by James Wexler, Been Kim and others the authors of the paper

Verwendete Libraries: tensorflow, sklearn

Repo Bestandteile

- model.py → activation_generator.py → tcav.py → cav.py → utils_plot.py →

Code::breakdown(model.py)

1. Model wrapper – kann Modelle Laden (`Checkpoints/SavedModel/Frozen graph`) die mit (tf.Session) erstellt worden sind. Beinhaltet Model wrappers die ermöglichen Operation auf das Modell auszuführen.

   Interesannte Methoden in `ModelWrapper`:
   `get_gradient() → runs backpropagation to get gradient regarding activations, class`
   `_find_ends_and_bottleneck_tensors() – finds all bottleneck tensors`
   `_make_gradient_tensors() – makes gradients for bottleneck layers`

```python
class GoogleNetWrapper_public(PublicImageModelWrapper):

  def __init__(self, sess, model_saved_path, labels_path):
    image_shape_v1 = [224, 224, 3]
    self.image_value_range = (-117, 255 - 117)
    endpoints_v1 = dict(
        input='input:0',
        logit='softmax2_pre_activation:0',
        prediction='output2:0',
        pre_avgpool='mixed5b:0',
        logit_weight='softmax2_w:0',
        logit_bias='softmax2_b:0',
    )
    self.sess = sess
```

Inheritance

`ImageModelWrapper(ModelWrapper)`
`PublicImageModelWrapper(ImageModelWrapper)`
`GoogleNetWrapper_public(PublicImageModelWrapper) → our example model wrapper`

Code::breakdown (activation_generation.py)

2. Activation(Generator) – beinhaltet die Methoden, um die Aktivierungen von dem gewünschten Layer in Bezug auf Konzept zu erhalten auf tf.Graph.

   Letztlich führen die wichtigen Methoden auf tf.Session.run() um die Aktivierungen der Layer zu erhalten

   Interessante Methoden:

```
get_activations_for_examples(examples, bottleneck) ->
calls tf.Session.run() to evaluate inputs and get the
activations of the desired layer
```

- Die Aktivierungen werden auf der Festplatte zwischengespeichert um diese wiederzuverwenden
- Kann parallel Daten laden

Inheritance

```
ActivationGeneratorBase(ActivationG
eneratorInterface)
ImageActivationGenerator(Activation
GeneratorBase)
```

Code::breakdown(cav.py)

3. Train CAV – contains essential methods to train a linear model (cav) including data processing

```
lm.fit(x_train, y_train)
y_pred = lm.predict(x_test)
```

lm - linear model (`SGDClassifier/ LogisticRegression`)
x - activations [num_data, data_dim]
(for pattern & random data)
y - concepts [num_data]
acc = `float(num_correct) / float(len(y_test))`

Code::breakdown(tcav.py)

4. beinhaltet Routine für die Berechnung von TCAV score

```
# Grad points in the direction which DECREASES
probability of class
grad = np.reshape(mymodel.get_gradient(act,
[class_id],cav.bottleneck,example), -1)
dot_prod = np.dot(grad,cav.get_direction(concept))
```

- Die CAV's werden zwischen gespeichert

```
striped-random500_28-mixed4c-linear-0.1.pkl
striped-random500_29-mixed4c-linear-0.1.pkl
zigzagged-random500_0-mixed4c-linear-0.1.pkl
zigzagged-random500_1-mixed4c-linear-0.1.pkl
```

- Parallele Ausführung von CAV trainings

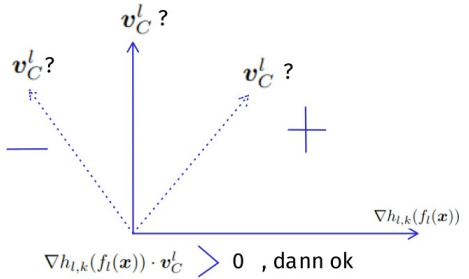

Abbildung: Sensitivität. Quelle: Eigene Darstellung.

Code::breakdown(utils_plot.py)

5. Führt ein Hypothesentest auf getesteten Konzepte gegen zufällige Konzepte

   mit p < 0.05

Bsp:

Koncept TCAV's	Zufällige TCAV'S
zigzag/random0:0.94	random1/random2:0.64
zigzag/random1:0.94	random1/random3:0.17
zigzag/random2:0.94	random1/random4:0.54

```
Calculate statistical significance
# i_ups - tcav score of concept CAV's
# random_i_ups[bottleneck] - tcav score of random
CAV's
_, p_val = ttest_ind(random_i_ups[bottleneck], i_ups)
```

Demo

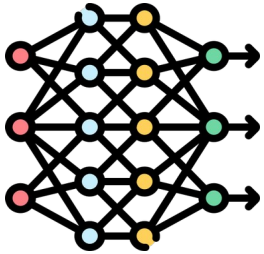

Abbildung: NN Model. Quelle: [4].

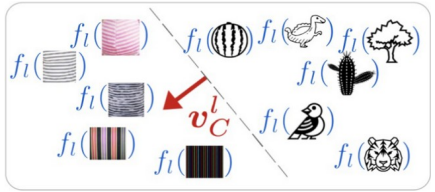

Abbildung: CAV. Quelle: [3].

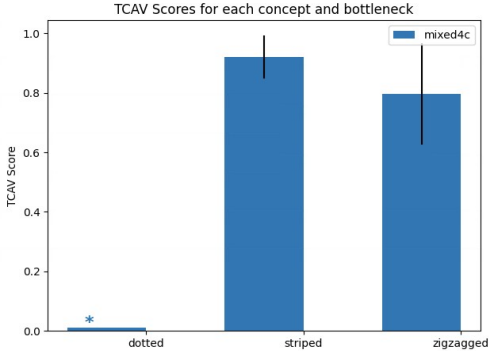

Abbildung: TCAV Score Quelle: Eigene Darstellung.

Fazit TCAV

- Gibt eine relativ gute Aussage, ob die jeweiligen Muster der Klasse richtig beschrieben sind.
- Könnte als NN debugging verwendet werden
- Methode ist quantitativ (im Gegensatz zu Feature Visualisierung)
- TCAV kann für andere Modelle eingesetzt werden wie NLP
- Kann Bias in Datensätzen aufdecken

Vielen Dank

Quellen

[1] https://commons.wikimedia.org/wiki/File:Osten_und_Hans.jpg. Abgerufen am: 01.05.2023
[2] Saliency Maps in Tensorflow 2.0, https://usmanr149.github.io/urmlblog/cnn/2020/05/01/Salincy-Maps.html. Abgerufen am: 01.05.2023
[3] Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F. und Rory, V. (2018). Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)
[4] https://www.flaticon.com/free-icon/deep-learning_8637101. Abgerufen am: 01.05.2023