XAI Exam Project A - ABELE

Alessandro Trenta

March 29, 2022

1 Description

In this work I will be analyzing the ABELE[GMMP20] explanatory model on the CIFAR-10 dataset. ABELE is a black box local model, meaning that it produces local explanations on single images based on its neighborhood. ABELE uses an autoencoder (in this case an adversarial autoencoder[MSJG16]) and its learned latent space. This way the model produces rules and counterfactual which are not based on individual pixels but focuses instead on exploring the latent space neighborhoods of images giving us insights on the actual features and patterns of the images.
Recall that the CIFAR-10 dataset[Kri09] has 50000 train images and 10000 test images with 10 labels: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Images are 32×32 pixels and are colored. Example images and more information about the dataset can be found on the cifar-10 University of Toronto website.

2 Black box

The black box used in this experiment is a deep convolutional neural network made of 3 convolutional 2D layers, followed by a flattening layer and 2 more dense fully connected layers and finally an output layer. Below are the details for the parameters used.

Layer	**filters**	**kernel_size**	**stride**	**units**	**activation**
Conv2D	32	3x3	1x1	/	ReLU
MaxPooling2D	/	2x2	/	/	/
Conv2D	64	3x3	1x1	/	ReLU
MaxPooling2D	/	2x2	/	/	/
Conv2D	128	3x3	1x1	/	ReLU
Flatten	/	/	/	/	/
Dense	/	/	/	100	ReLU
Dense	/	/	/	50	ReLU
Output	/	/	/	10	/

This CNN is created and trained with the Tensorflow library for Python. It is trained for 50 epochs with a batch size of 128. The loss of the model is the Categorical Cross entropy and the metric used is accuracy. Here the detailed results after the training:

	Loss (CatCrossEnt)	**Accuracy**
Training	0.0680	0.9782
Test	2.4969	0.6975

The results on the test set are not great but this makes finding explanations for such predictions an interesting task as we could understand where this bad accuracy comes from.

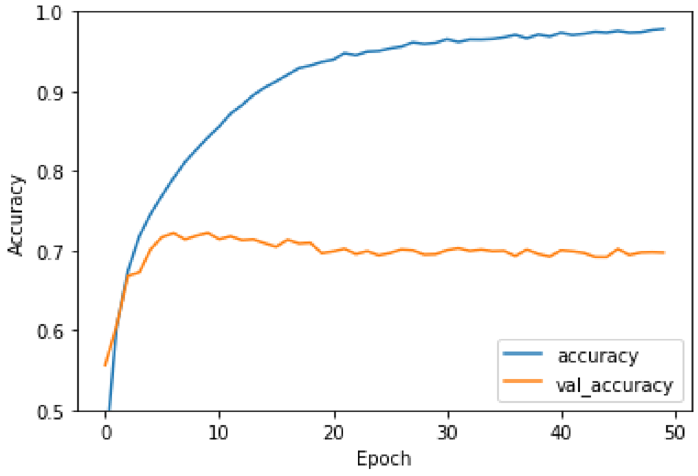

Figure 1: Training and validation accuracy with respect to the number of epochs of training. Unfortunately, after an initial fast increase in validation accuracy the value stabilizes around 70%.

3 Adversarial Autoencoder (AAE)

Adversarial autoencoders are a special kind of autoencoders. Given input x and its latent representation z they use a Generative Adversarial Network approach to match the latent distribution $p(z|x)$ with the prior selected, in this case the normal distribution $\mathcal{N}(0,1)$. AAEs are constituted by an encoder, a decoder and a discriminator. The encoder is also the generator for the GAN and the discriminator is trained to distinguish real samples from $\mathcal{N}(0,1)$ distribution and data generated from the encoder. Latent dimension of the autoencoder is 256. Many values were tried but for decent results it seemed that a value of 200 minimum was required.

3.1 Encoder and decoder

Here below a table describing the encoder network.

Layer	**filters**	**kernel_size**	**strides**	**units**	**activation**	**padding**
Conv2D*	64	4	2	/	LeakyReLU ($\alpha = 0.2$)	same
Conv2D*	128	4	2	/	LeakyReLU ($\alpha = 0.2$)	same
Conv2D*	256	3	2	/	LeakyReLU ($\alpha = 0.2$)	same
Conv2D*	512	3	2	/	LeakyReLU ($\alpha = 0.2$)	same
Flatten	/	/	/	/	/	/
Dense*	/	/	/	1000	ReLU	/
Dense (output)	/	/	/	512	/	/

Table 1: * Layers with L2 kernel regularization and batch normalization

Recall that the output of the encoder is a vector of length 512 of which the first 256 values represent the mean μ of the latent distribution $p(z|x)$ and the remaining values represent the log variance σ. To complete the encoding we apply the so-called "reparameterization trick" that consists in sampling $\epsilon \sim \mathcal{N}(0,1)$ and applying the following operation to all the 256 components of the latent space, obtaining the final encoding z of the input x

$$z = \mu + e^{\frac{\sigma}{2}} \cdot \epsilon \tag{1}$$

The decoder is just a network symmetric to the encoder with Conv2D layers replaced by Conv2DTranspose layers.

3.2 Discriminator

The discriminator network takes in input a vector of length 256 (the latent dimension) and is constituted just by two dense layers with 200 units, ReLU activation and L2 kernel regularization, followed by a dense output layer of 1 unit, with no activation function. This is the output that controls the validity of the input. Values towards 1 are considered true.

3.3 Training and results

The AAE was trained for 149 epochs with a batch size of 256. The loss for the reconstruction error of the autoencoder is the mean squared loss, while the loss for both the discriminator and generator is the Binary Cross Entropy.
The training step is done in three parts:

- First we encode and reconstruct the image, calculate the reconstruction error and update the autoencoder weights.
- Second we calculate the discriminator error. The "real" inputs are sampled at each step with a $\mathcal{N}(0,1)$ distribution while the "fake" inputs are the encoded images. We then update the weights of the discriminator.
- Last we calculate the generator error and update its weights (recall that the generator is the encoder network).

The learning rate schedule for the autoencoder and discriminator parts of the training is the following:

Epoch	**Learning Rate**
0-50	0.0001
50-75	0.00005
75-100	0.00002
100-149	0.00001

The learning rate values for the generator are the same above multiplied by a 1.5 factor as it seemed like this part of the training needed more help. Validation was done using the test dataset.
These are the final metrics after the training:

	Training	**Validation**
Autoencoder loss	0.0033	0.0033
Discriminator loss	0.7001	0.6936
Discriminator accuracy	0.4998	0.4972
Generator Loss	0.6660	0.6487

The image below shows that the AAE managed to make the encoded points match a $\mathcal{N}(0,1)$ distribution on the latent space (for simplicity we plot only the first two dimensions.

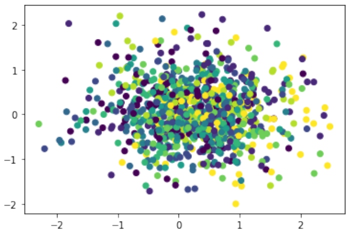

Figure 2: Scattering of the first 1000 test images on the first two latent dimensions

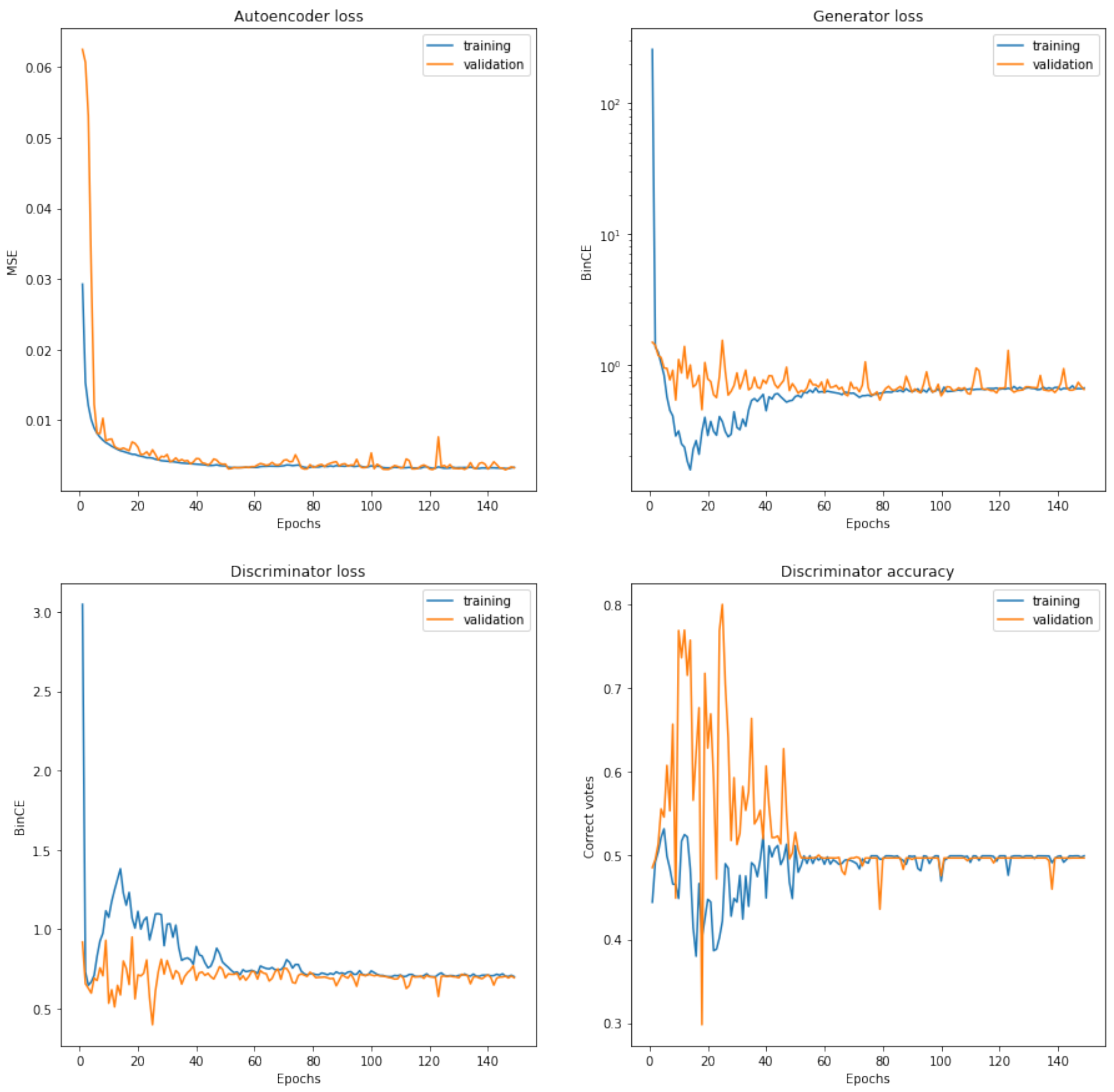

Figure 3: Training and loss validation plots for the AAE training

Unfortunately, given the local and on-line resources available to me (mainly my PC and the web servers offered by Google Colab and Gradient) I wasn't able to try more complex networks with a lower dimensional latent space but results were not bad in the final run. Training time for the autoencoder was around $5-6$ hours on Gradient web servers.

4 ABELE

ABELE is a local Black Box explainer based on the latent space of the autoencoder. ABELE first encodes the image to be explained, then generates a neighborhood of the image in the latent space through a genetic algorithm. In the next step, the model generates a decision tree on the latent space to understand the topology and the boundaries of the various image classes. ABELE finally generates the decision rules and three kind of images: a saliency map, prototypes and counterexemplars.
Now we analyze the most significant outputs of the ABELE explainer:

- **Explanatory decision tree rules**: ABELE generates a decision tree for predicting the class based on the neighborhood of our image in the latent space, outputting the rules. Here below an example:

  $$\begin{aligned} e = \{ & \\ & r = \{205 \leq 1.76, 39 > 2.60, 171 \leq -0.17, 220 > -0.04, 211 \leq 0.38, 239 \leq 0.50\} \rightarrow \{\text{class} : 7\} \\ & c = \{\{39 \leq 2.60\} \rightarrow \{\text{class} : 4\}, \{171 > -0.17\} \rightarrow \{\text{class} : 3\}, \{220 \leq -0.04\} \rightarrow \{\text{class} : 3\}, \\ & \{239 > 0.50\} \rightarrow \{\text{class} : 4\}, \{205 > 1.76\} \rightarrow \{\text{class} : 4\}, \{211 > 0.38\} \rightarrow \{\text{class} : 4\}\} \\ \} & \end{aligned}$$

  The explanation e is given by the rules r for the predicted class and some counterexamples rules c. The output is in the form $\{x_d \leq a\} \rightarrow \{\text{class} : b\}$ meaning that if the latent space component x_d is changed to a value less or equal than a then the predicted class is b. Unfortunately, unless the latent space is low dimensional or has a clear interpretation this output is not much human-readable or understandable.

- **Black box prediction, decision tree prediction and fidelity**: ABELE then outputs the black box prediction of the image, the decision tree prediction, which is based on the local decision tree generated by the genetic algorithm and a fidelity metrics. This metric in particular express how much the decision tree matches the predictions of the black box. The higher the value towards 1, the better our decision tree is. This is helpful in the sense that it tells us if the rules and the next explanations are coherent with the black box model.

- **Saliency map**: it is a reproduction of the image, constituted by the borders of the objects in the image and colored pixels: a green-blue colored pixel is a pixel that is relevant for the predicted class (it characterizes it in some way). Yellow-brown pixels are those one that if changed in some way can lead to a different predicted class. White pixels are considered to be not important.

- **Prototypes and counterexamples**: images generated from the neighborhood that are examples from the same class of the image to be explained or representative of other classes. As we will see the prototypes are very confused as our latent space is really complex and images generated from it are not really human understandable in most cases, while the counterexamples are usually very similar to the original images and show the differences needed to obtain a different class.

In the next section we are going to look at some relevant examples and explanation I found.

4.1 Relevant examples

4.1.1 Image 1

Let's start by having a look at the image and its saliency map:

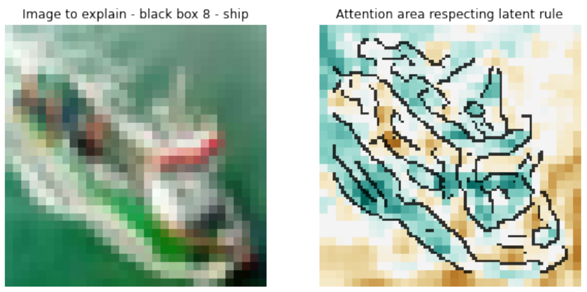
Image to explain - black box 8 - ship
Attention area respecting latent rule

Figure 4: Image 1 - ship and its saliency map

The image gets labeled correctly by the black box. From the saliency map we can see that the pixel that are important for the prediction are the ones that correspond to the actual body of the ship, meaning that in this case the black box is making a correct prediction by actually recognizing it from its features (as we will see later on this doesn't happen in all cases).
The decision tree also classifies correctly the image but the fidelity in the neighborhood is around 64%, meaning that there is a discrepancy between decision tree and black box predictions in the genetic neighborhood of the image.

Figure 5: Image 1 - prototypes

In this case just one prototype was found and as we can see it is not very much human understandable or interesting for our analysis (from now on, we will show the prototypes only when they give relevant insights).

Figure 6: Image 1 - Counterexemplars

Two counterexemplars were found: one predicted to be an airplane and one predicted to be a ship by the back box. The second is more similar to the original one and is somehow similar to a ship. The first one, on the contrary, has a different background color (close to the color of the sky) which may have favored an airplane prediction. It's interesting to see that the predicted labels for the counterexemplars are still vehicles or machines instead of animals.

4.1.2 Image 2

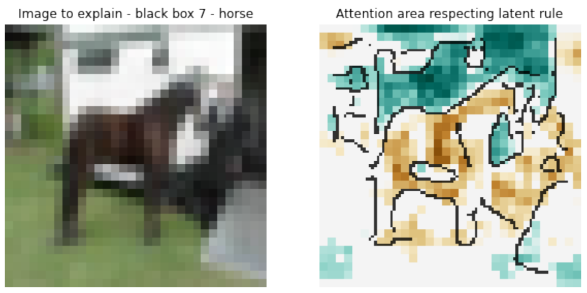
Image to explain - black box 7 - horse
Attention area respecting latent rule

Figure 7: Image 2 - horse with its saliency map

In this next example the image gets predicted correctly by both the black box and the decision tree with a much higher fidelity around 90%. But as we can see from the saliency map here the situation is much different:

- The pixels relevant for the prediction are not those representing the horse but the ones on the background, specially the house. This may indicate that our black box is biased towards recognizing a horse not by its features but from what is around it, leading to an incorrect reasoning and possible errors.
- The pixels of the horse are actually those that may affect the prediction indeed (the counterexemplars are in this case helpful). Recall that there exist also a deer class and with this image resolutions the two animals may look similar. In general, this is a behaviour we may want to avoid for our black box.

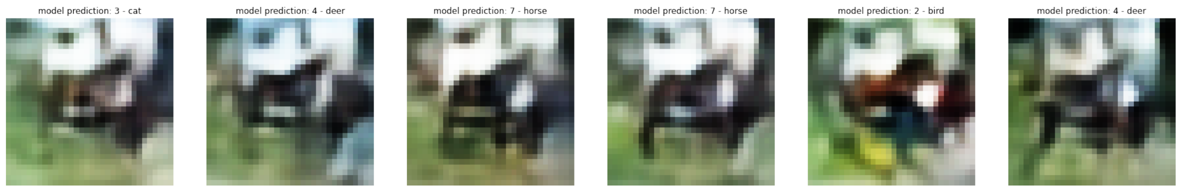
model prediction: 3 - cat
model prediction: 4 - deer
model prediction: 7 - horse
model prediction: 7 - horse
model prediction: 2 - bird
model prediction: 4 - deer

Figure 8: Image 2 - counterexemplars

It is interesting to see that all the counterexemplars contain clearly the same type of building in the background with the same white color, while the animal in front is the object that gets modified the most. All the counterexemplars are indeed classified as some kind of animal. A possible explanation of this behaviour given this counterexemplars and the saliency map may be:

- The building in the back is the one feature that the black box is using the most to identify that in the image an animal is present. In general we don't want this effect for our machine learning models.
- The animal is analyzed using its actual features, different shapes and colors help the black box recognize the correct animal class: this indeed is a good behaviour.

This has to be proven but may still be a description of how the black box is working and recognizing patterns.

4.1.3 Image 3

Let's now have a look at an image that gets predicted incorrectly:

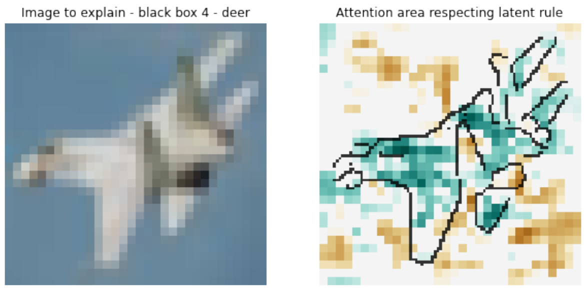
Image to explain - black box 4 - deer
Attention area respecting latent rule

Figure 9: Image 3 - Airplane and saliency map

The correct label for the image is 0 - airplane, while both the black box and the decision tree predict it to be a deer (fidelity is not so high: around 64%).
In this case the most important pixels for the prediction are indeed corresponding to some parts of the airplane itself, while the orange pixels are in the sky around it. In reality, from my own prospective the back tails of the airplane itself may contain some patterns matching the face or the legs of an animal (this may also be seen in the counterexamples below).
Finally let's have a look at the counterexamples:

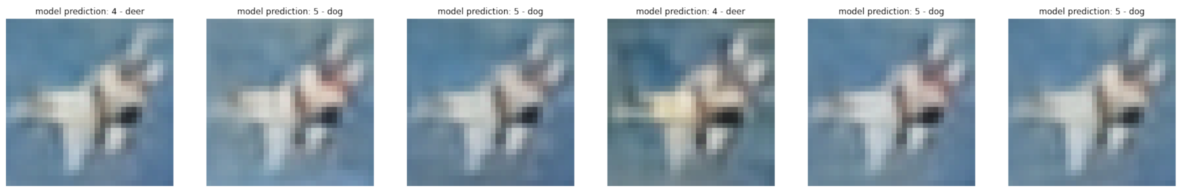
model prediction: 4 - deer
model prediction: 5 - dog
model prediction: 5 - dog
model prediction: 4 - deer
model prediction: 5 - dog
model prediction: 5 - dog

Figure 10: image 3 - counterexamples

From this counterexamples the observation pointed out above are even clearer. The black box is also classifying all these images with an animal label.
It is interesting to see that even if the background is all blue the black box is not biased towards an incorrect use of this feature of the image.

4.1.4 Image 4

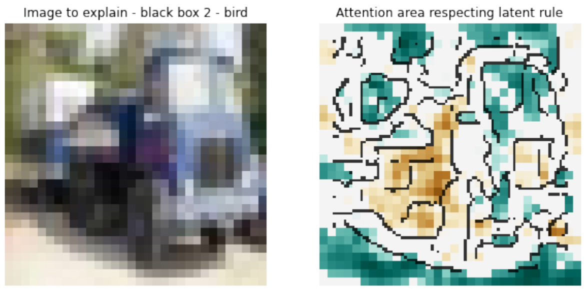
Image to explain - black box 2 - bird
Attention area respecting latent rule

Figure 11: Image 4 - Truck and saliency map

The correct label for the image is 9 - truck, while both the black box and the decision tree predict it to be a bird (label 2). The decision tree is coherent with the black box as it classifies the image as a bird and has fidelity around 95%. This image features the same problems of image 2 we have seen before:

- The important pixels for prediction are on the floor and in the higher part of the image, only a few of them are inside the actual truck.
- Yellow pixels are mostly part of the truck, but the majority of pixels of the truck itself are colored white and therefore are considered not important for the prediction.

Finally we have a look at the counterexamples:

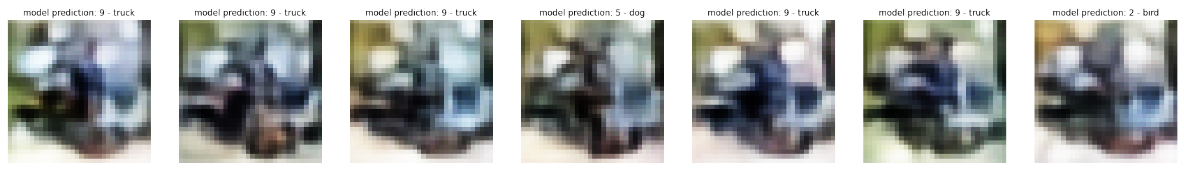
model prediction: 9 - truck
model prediction: 9 - truck
model prediction: 9 - truck
model prediction: 5 - dog
model prediction: 9 - truck
model prediction: 9 - truck
model prediction: 2 - bird

Figure 12: Image 4 - Counterexamples

It is interesting to see that most of the counterexamples generated are indeed similar to the original image but get predicted in the ”correct” truck class. On the other hand, 2 of this images are predicted to be either a dog or a bird. The neighborhood of this image may be really complex.

4.1.5 Image 5

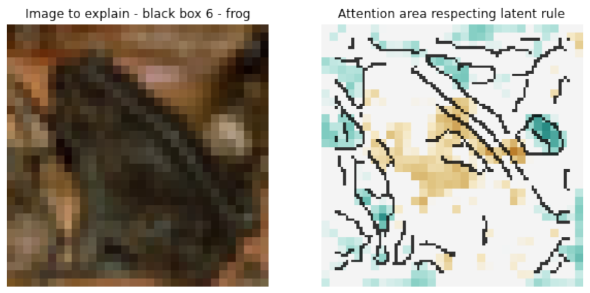
Image to explain - black box 6 - frog
Attention area respecting latent rule

Figure 13: Image 5 - frog and saliency map

The image gets predicted correctly by both the black box and the decision tree generated in the genetic neighborhood. On the other hand fidelity is around 70%.
Unfortunately the most important pixels for prediction are not in the frog itself but in the background, as we have seen from previous examples.

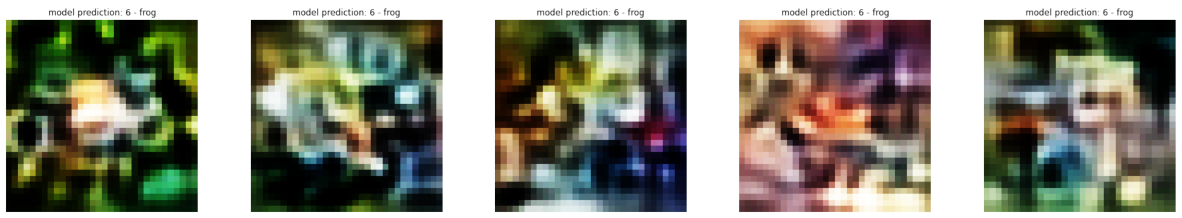
model prediction: 6 - frog
model prediction: 6 - frog
model prediction: 6 - frog
model prediction: 6 - frog
model prediction: 6 - frog

Figure 14: Image 4 - prototypes

I find very interesting the first prototype generated by the algorithm. As we can see the image contains a yellow-white figure in the center, and the background is mostly of a green color. With just these pictures we do not understand if this kind of background is relevant for the prediction but this is surely one of the most interesting prototypes found.
The counterexamples produced from this image are instead very confused. The predicted labels are still meaningful, as the images are all predicted to be images of animals.

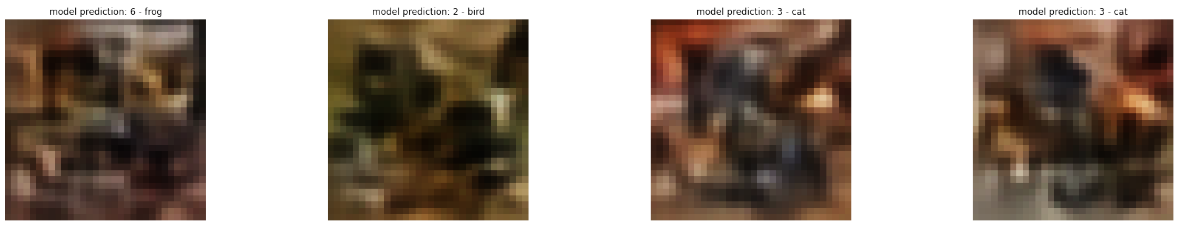
model prediction: 6 - frog
model prediction: 2 - bird
model prediction: 3 - cat
model prediction: 3 - cat

Figure 15: Image 4 - counterexamples

5 Conclusion

I analyzed the ABELE model for black box local explanations on the CIFAR-10 dataset.
Here are my comments and conclusion:

- The saliency map and the counterexemplars are the most meaningful outputs for this work: the saliency map gives an insight on which pixels are the most important for our black box prediction and which ones may instead lead to another label prediction. This is really useful given that in some cases ABELE showed that the black box was making predictions mostly based on the background rather than on the objects itself. In this sense, this method helps us understand if the black box is biased.
- The counterexemplars let us understand how an image can be modified to have a different black box prediction. The autoencoder is fundamental in modifying images respecting the features and patterns of the objects by using the learned latent space. Unfortunately the quality of this images is strictly related to the reconstruction error of the autoencoder itself which, in this case, was good but not perfect. Images appear slightly blurred.
- Given the high dimension and complexity of the latent space used for this dataset, the prototypes were confused and not really human-understandable. In simpler works (e.g. with the MNIST dataset) this indeed can give good results.
- The rules and counterfactual rules produced by ABELE are based on the latent space: if we don't have a clear understanding or interpretation of the latent space this rules are not human understandable but are fundamental to the method itself for producing the next outputs.
- The predictions and the fidelity of the decision tree in the neighborhood are really useful because they give us information about how the decision tree rules for prediction match the original black box prediction boundaries. A low fidelity can lead to bad explanations as ABELE might not be understanding correctly how the black box is reasoning in the neighborhood.
- Unfortunately ABELE is not fast at producing these outputs: an explanation on one test image required on average around 10 minutes with the parameters used.
- This model doesn't work every time: some images never led to any kind of explanation even with different parameters and in some cases more than one run was necessary to obtain the explanations. Sometimes prototypes couldn't be found.

In conclusion, the method is indeed very helpful for understanding black box predictions. Given that it is based on the latent space features of an autoencoder which, at least theoretically, represent intrinsically the features of the images instead of working merely on pixels means that explanations based on this method can be more interesting and meaningful with respect to the semantics and themes of images.

References

[GMMP20] Riccardo Guidotti, Anna Monreale, Stan Matwin, and Dino Pedreschi. Black box explanation by learning image exemplars in the latent feature space. In Ulf Brefeld, Elisa

Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet, editors, *Machine Learning and Knowledge Discovery in Databases*, Cham, 2020// 2020. Springer International Publishing, Springer International Publishing.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[MSJG16] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016.