XAI Exam Project A - ABELE

Alessandro Trenta

March 19, 2022

Abstract

Your abstract.

1 Description

2 Blackbox

The blackbox 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 an output layer. Below are the details for the layers.

This CNN is created and trained with the Tensorflow library for Python. It is trained for 50 epochs

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	/

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 are: The results on the test set are not great but

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

this makes finding explanations for such predictions an interesting task as we could understand where this bad accuracy comes from.

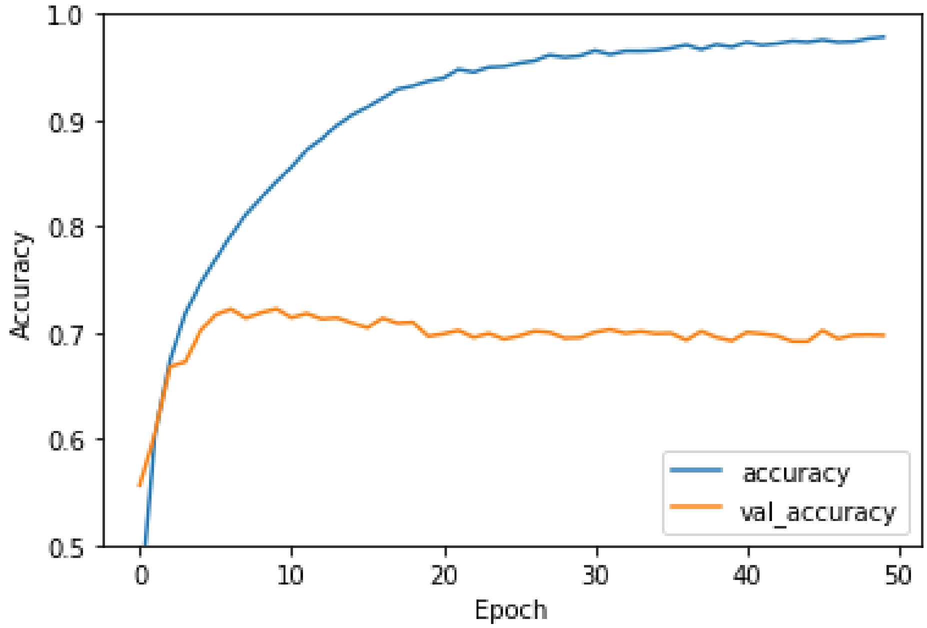

Figure 1: Training and validation accuracy with respect to the number of epochs of training. Unfortunately, after an initial fast increase in validation accuracy the value stabilizes around 70%

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 minumum was required.

3.1 Encoder and decoder

Here below a table describing the encoder network. Recall that the output of the encoder is a vector

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

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 "reparametrization 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 the symmetric of 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 or not of the input.

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 of the encoder. We then update the weights of the discriminator.
- Last we calculate the generator error and update its weights (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.
Finally here 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

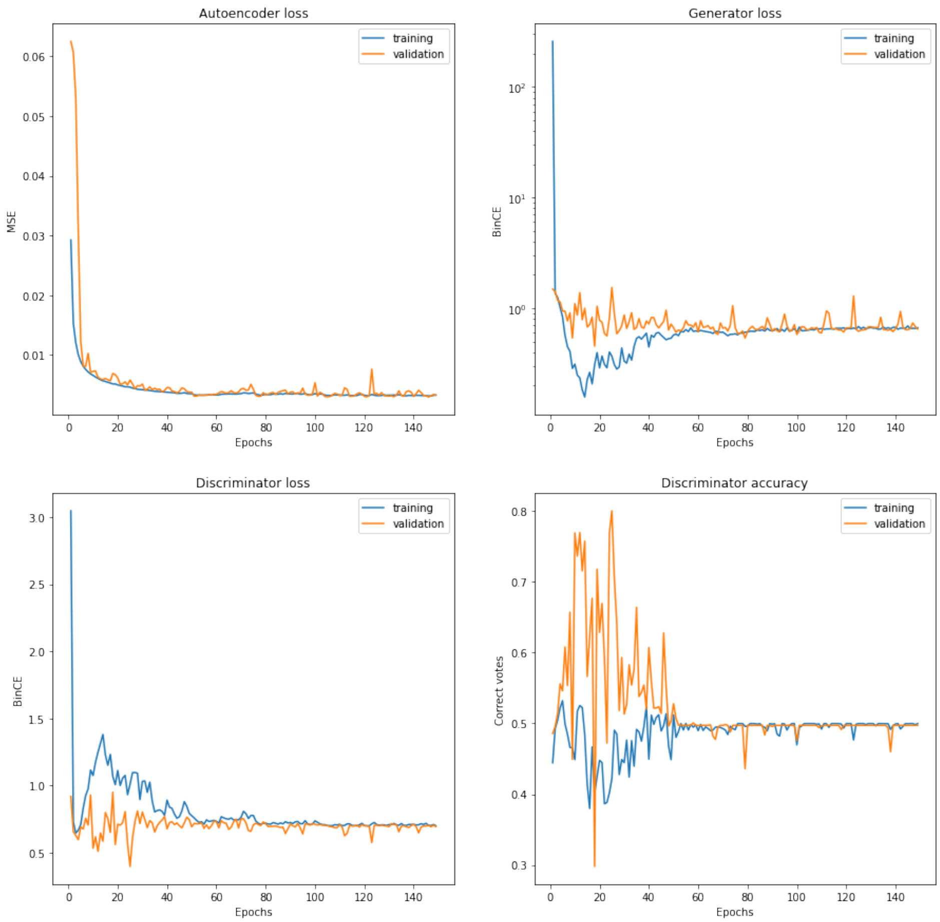

Figure 2: Training and loss validation plots for the AAE training

Lastly, as you can see the AAE managed to make the encoded points match a $\mathcal{N}(0, 1)$.

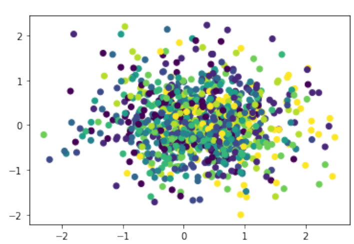

Figure 3: Scattering of the first 1000 test images on the first two latent dimensions

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 class boundaries of the various classes. ABELE finally generates the decision rules and three kind of images:

- A saliency map, representing the importance of each pixel and its effects.
- Some prototypes: images generated from the latent space that are in the same class of the one to be explained.
- Some counterexeplars: images similar to the one to be explained but predicted to be in other classes, to better understand what needs to be modified to obtain a different prediction.

Now we analyze the most significant outputs of the ABELE explainer:

- **Explainatory 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 only the latent space component x_d is changed to a value less or equal than a then the 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 for the image, 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 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-blu colored pixel is a pixel that is relevant and important for the predicted class (it characterizes it in some way). Yellow-brown pixels are those one that if changed in some way lead to a different predicted class.

- **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 can be really general and 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:

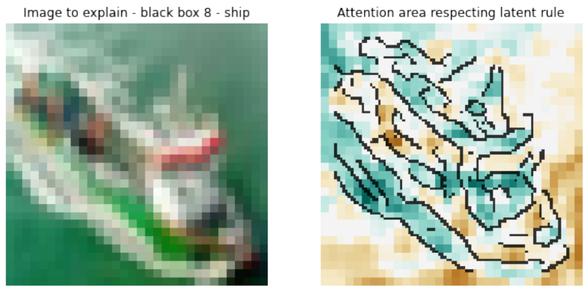

Figure 4: Image 1 - ship and its saliency map

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 explanation is relevant: the black box is predicting a ship and it is actually recognizing it from its feature (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 the decision tree doesn't respect a high level the black box predictions.

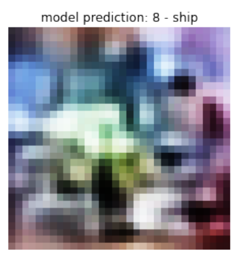

Figure 5: Image 1 - prototypes

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 still has resembles a ship. The first one, on the contrary, has a different background color (more similar to 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

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, expecially 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.
- The pixels of the horse are actually those that may affect the prediction indeed (the counterexemplars are in this case helpful).

Figure 8: Image 2 - prototypes

Write something about prototypes or erase

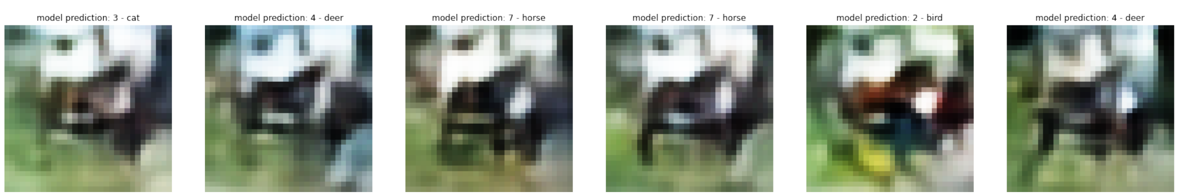

Figure 9: Image 2 - counterexemplars

It is interesting to see that all the counterexemplars contain clearly the same type of building in the back 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 animals. In my opinion this is what may be happening given this counterexemplars and the saliency map:

- 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. This is in general not very good for the black box usage
- The animal is then recognized using its actual features, diffent shapes and colors help the black box recognize the correct animal: this indeed is a good behaviour.

This is still my supposition and has to be proven but given the outputs of ABELE this could be an explanation of this behaviour.

4.1.3 Image 3

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

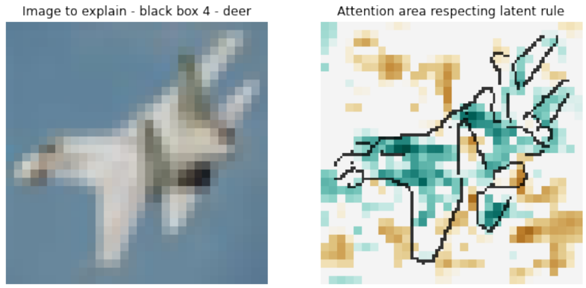

Figure 10: Image 3 - Airplane and saliency map

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 resemble the face or the legs of an animal (this may also be seen in the counterexamples below).

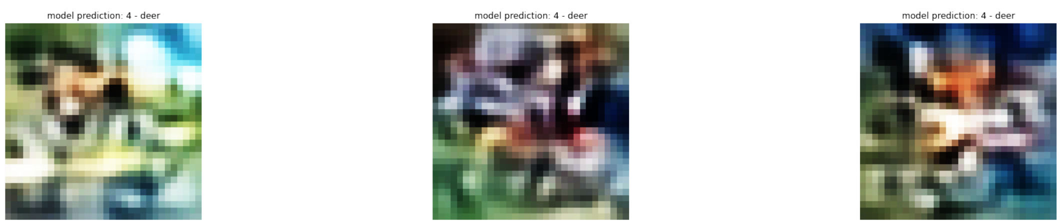

Figure 11: Image 3 - prototypes

Finally let's have a look at the counterexamples:

Figure 12: image 3 - counterexamples

From this counterexamples the observation pointed out above are even clearer. The black box is also classifying all these images into 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

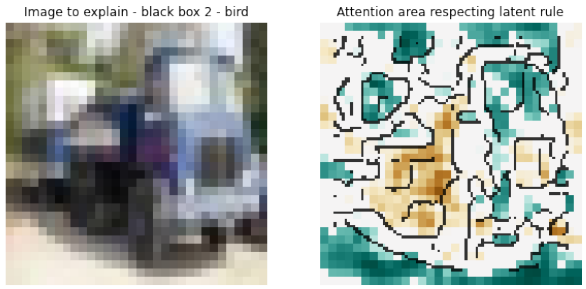

Figure 13: Image 4 - Truck and saliency map

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). This image present 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.
- The pixel relevant for label changing are again part of the truck, but the majority of pixels of the main object are colored white, considered to be non relevant.

In this case only one prototype was produced

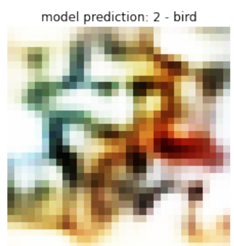

Figure 14: Image 4 - prototypes

Finally we have a look at the counterexamples:

Figure 15: 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.

4.1.5 Image 5

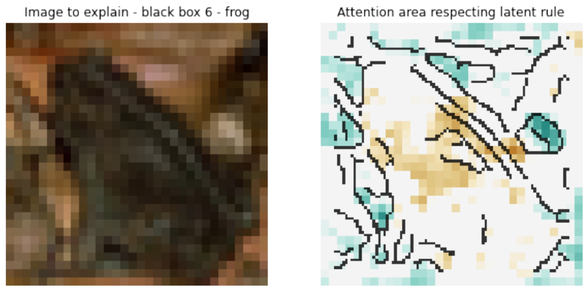

Figure 16: Image 5 - frog and saliency map

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.

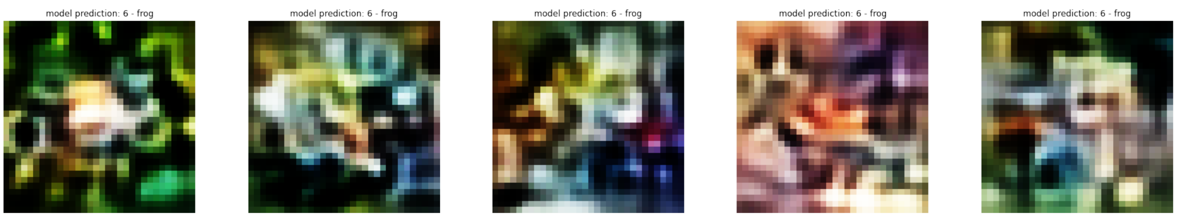

Figure 17: Image 4 - prototypes

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 all green. With just these pictures we do not understand if the all-green background is important for the prediction but this is surely one of the most interesting prototypes found.

Figure 18: Image 4 - counterexamples

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.

5 Final Results

References