LioNets: Local Interpretation of Neural Networks through Penultimate Layer Decoding

Ioannis Mollas, Nikolaos Bassiliades, and Grigorios Tsoumakas

Aristotle University of Thessaloniki, Thessaloniki 54124, Greece
{iamollas,nbassili,greg}@csd.auth.gr

Abstract. Technological breakthroughs on smart homes, self-driving cars, health care and robotic assistants, in addition to reinforced law regulations, have critically influenced academic research on explainable machine learning. A sufficient number of researchers have implemented ways to explain indifferently any black box model for classification tasks. A drawback of building agnostic explanators is that the neighbourhood generation process is universal and consequently does not guarantee true adjacency between the generated neighbours and the instance. This paper explores a methodology on providing explanations for a neural network's decisions, in a local scope, through a process that actively takes into consideration the neural network's architecture on creating an instance's neighbourhood, that assures the adjacency among the generated neighbours and the instance.

Keywords: Explainable · Interpretable · Machine Learning · Neural Networks · Autoencoders.

1 Introduction

Explainable artificial intelligence is a fast-rising area of computer science. Most of the research in this area is currently focused on developing methodologies and libraries for interpreting machine learning models for two main reasons: a) increased use of black box machine learning models, such as deep neural networks, in safety critical applications, such as self-driving cars, health care and robotic assistants, and b) radical law changes empowering ethics and human rights, which introduced the right of users to an explanation of machine learning models' decisions that concern them.

Methods aiming to explain individual predictions of a particular model are called local explanators. LIME [11] is a state-of-the-art methodology that first constructs a local neighbourhood around a given new unlabeled instance, by perturbing the instance's features, and then trains a simpler transparent decision model in order to extract the features' importance. Subsequent model agnostic methods like Anchors [12] and LORE [5] focused on generating better neighbourhoods.

This paper is concerned with generating better neighborhoods too. However, it focuses on neural network models in particular, in contrast to the *model agnostic* local explanators mentioned in the previous paragraph that can work with

any type of machine learning model. Our approach is inspired by the following observation: small changes at the input layer, might lead to large changes at the penultimate layer of a (deep) neural network, based on which the final decision of the network is taken. We hypothesize that creating neighbourhoods at the penultimate layer of the neural network instead, could lead to better explanations.

To investigate this intuitive research hypothesis, we introduce our approach, dubbed LioNets (Local Interpretation Of Neural nETworkS through penultimate layer decoding). LioNets constructs a local neighborhood at the penultimate layer of the neural network and records the network's decisions for this neighborhood. However, in order to build a transparent local explanator, we need to have input representations at the original input space. To achieve this, LioNets trains a decoder that learns to reconstruct the input examples from their representations at the penultimate layer of the neural network. Taking together, the neural network model and the decoder resemble an autoencoder.

For the evaluation of LioNets, a set of experiments have been conducted , whose code is available at GitHub repository "LioNets"[1]. The results show that LioNets can lead to better explanations than LIME.

2 Background and Related Work

In order to be able to present LioNets architecture, this section will provide a sequence of definitions concerning the matter of explainable machine learning, autoencoders and knowledge distillation.

2.1 Explainable Machine Learning

Explainable artificial intelligence is a broad and fast-rising field in computer science. Recent works focus on ways to interpret machine learning models. Thus, this paper will focus on explainable machine learning. An accurate definition is the following:

> "An interpretable system is a system where a user cannot only see but also study and understand how inputs are mathematically mapped to outputs. This term is favoured over "explainable" in the ML context where it refers to the capability of understanding the work logic in ML algorithms"[1].

There are several dimensions that can define an interpretable system according to [6]. One interesting dimension is the *scope* of interpretability. There are two different scopes. An interpretable system can provide global or/and local explanations for its predictions. Global explanations can present the structure of the whole system, while local explanations are focused on particular instances.

In the same paper, they are also presenting the desired features of any interpretable system. Those are:

[1] `https://github.com/iamollas/LioNets`

- **Interpretability**: Interpretability measures how much comprehensible is an explanation. In fact there is not a formal metric because for every problem different attributes get measured.
- **Accuracy**: The accuracy, and probably other metrics, of the original model and the accuracy of the explanator.
- **Fidelity**: Fidelity describes the mimic ability of the explanator.

2.2 Autoencoders

utoencoders is a growing area within deep learning [8]. An autoencoder is an unsupervised learning architecture and can be expressed as a function

$$f : X \to X. \tag{1}$$

Autoencoder networks are widely used for reducing the dimensionality of the input data. They initially encode the original data into some latent representation, and subsequently reconstruct the original data by decoding this representation to the original dimensions. The most common varieties of autoencoders are the four following:

- **Vanilla**: A three-layered neural network with one hidden layer.
- **Multilayer**: A deeper neural network with more than one hidden or recurrent layers.
- **Convolutional**: Used for image or textual data. In practice, the hidden layers are not fully connected layers, but convolutional layers.

2.3 Related Work

As already mentioned, LIME [11] is a state-of-the-art method for explaining predictions. It follows a simple pipeline. It generates a neighbourhood of a specific size for an instance by choosing randomly to put a zero value in one or more features of every neighbour. Then the cosine similarity of each neighbour with the original instance is measured and multiplied by one hundred. This constitutes the weight on which the simple linear model will depend on for its training. Thus, the most similar neighbours will have more impact on the training process of the linear model. A disadvantage of LIME is in sparse data. Due to the perturbation method that takes place on the original space, LIME can only generate 2^n different neighbours, where n the number of non-zero values. For example, in textual data, in a sentence of six words, which is represented as a vector of four thousand words, the non-zero features are only six. Hence, only $2^6 = 64$ different neighbours can be generated. However, LIME will create a neighbourhood of five thousand instances by randomly sampling through the 64 unique neighbours.

Another set of methodologies in explaining decision systems, and specifically neural networks, are using Knowledge Distillation [7,4]. Those methods are trying to explain globally the whole structure and the predictions of a deep neural

network, by distilling its knowledge to a transparent system. This idea originates by the Dark Knowledge Distillation [13], which is trying to enhance the performance of a shallow network (the student) through the knowledge of a deeper and more complex network (the teacher).

3 LioNets

This section presents the full methodology and architecture of LioNets. LioNets consist of four fundamental sub-architectures. The main part of such system is the neural network, which will work as the predictor. A decoder based on the predictor is the second part. Finally, a neighbourhood generation process and a transparent predictor are the last two mechanisms. Hence, in order to e following process should be executed.

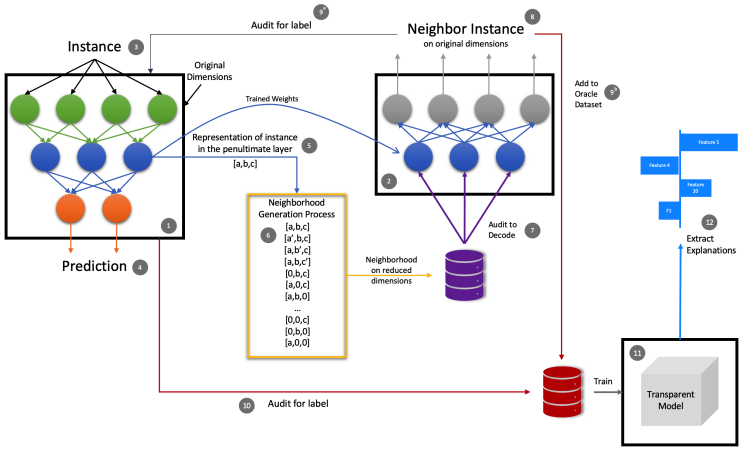


Fig. 1: LioNets' architecture

3.1 Neural Network Predictor

For a given dataset, a neural network with a suitable fine-tuned architecture is being trained on this dataset. The output layer is by design in the same length as the number of classes of the classification problem. This process is similar to other supervised methodologies of building and training a neural network for classification tasks, which defines a function $f : X \rightarrow Y$.

3.2 Encoder and Decoder

When the training process of the neural network is over, a duplicate it is created. Then removing the last layer of this copy model and labelling every other layer

as untrainable, the foundations for the autoencoder have been defined. Actually, this foundations would be the encoder, the first half of the autoencoder, thus only the decoder part is missing. By building successfully the decoder part and training it, the first two stages for Lionets' completion are achieved. Although, this is the most difficult stage to complete due to the fact that autoencoders are trained very hard successfully, especially when the first half of the autoencoder is untrainable, for example when training only the decoder. Another approach is to build the autoencoder firstly and afterwards to extract the layers in order to create the encoder, decoder and predictor networks.

Mathematically those neural networks can be expressed via these functions:

$$Encoder\colon X \to Z, \tag{2}$$

$$Decoder\colon Z \to X, \tag{3}$$

$$Autoencoder\colon X \to X, \tag{4}$$

$$Predictor\colon X \to Y. \tag{5}$$

By keeping the encoder part untrainable with stable weights, it guarantees that the generated neighbourhood is transforming from the reduced dimensions to the original dimensions with a decoder, which was trained with the original architecture of the neural network. That process will produce a more representative neighbourhood for the instance, without any semantic meaning to humans.

The academic community has extensively explored ways to create better neighbourhoods for an instance, but every methodology was focused on generating new instances in the level of the input. In this work, the generation processes take place to the latent representation of the encoded input.

3.3 Neighbourhood Generation Process

The neighbourhood generation process takes place after the training of the neural network, the encoder and the decoder. This process could be a genetic algorithm or even another neural network, but simpler solutions are preferred. Then for an instance, that is desirable to get explanations, after encoding it via the encoder neural network, extracting its new representation form from the penultimate level of the neural network, the neighbourhood generation process begins with input the instance with reduced dimensions. By making small changes in the reduced space it could affect more than one dimensions of the original space. Thus, the simple feature perturbation methods on low dimensions will lead to a complex generated neighbour, which most probably would have no semantic meaning for humans.

At that point in time, a specific number of neighbours is generated through a selected generation process and that set of neighbours is given to the decoder, in order to be reversed to the original dimensions.

3.4 Transparent Predictor

By the end of the neighbourhood generation stage, the neighbourhood dataset is almost complete. The only missing part is the neighbours' labels. Thus, the neural network is audited one last time by the neighbourhood dataset in order to assign labels to every neighbour, in the form of probabilities. Afterwards, the final dataset with the neighbours and their labels are given as training data to any transparent regression model. The ultimate goal is to overfit that model to the training data by keeping the comprehensibility of the model at low levels.

4 Evaluation

The following section is presenting the setup for the experiments. The data pre-processing methods for two different datasets are described, alongside with the neural network models preparation and the neighbourhood generation process. Finally, there is a discussion about the results of the experiments.

4.1 Setup

Our experiments involve two textual binary classification datasets. The first one concerns the detection of hateful YouTube comments[2] [3] and the second one the detection of spam SMS messages [2]. The pre-processing of these datasets consists of the following steps for each document:

- Lowercasing.
- Stemming and Lemmatisation through WordNet lemmatizer[9] and Snowball stemmer[10].
- Phrases transformations [Table 1].
- Removal of punctuation marks.
- Once again, Stemming and Lemmatisation.

Then, for transforming the textual data to vectors a simple term frequency-inverse document frequency[14] (TF-IDF) vectorization technique is taking place.

Afterwards, the neural network predictor for these experiments consists of six layers Fig 2a. The encoder has five layers and the decoder has four layers as well Fig 2b. The autoencoder is the combination of the encoder and the decoder.

In these set of experiments, a simple generation process via features perturbation methods is applied. Specifically, the creation of neighbours for an instance emerges by multiplying one feature value at a time with 2^z, $z \in \{-2, -1, 0, 1, 2\}$. Concisely, the above process generates instances which are different in only one dimension in their latent representation.

As soon as the neighbourhood is acquired, every neighbour is transformed via the decoder to the original dimensions. Then, the transformed neighbourhood is given as input to the predictor to predict the class probabilities. Finally,

[2] https://intelligence.csd.auth.gr/research/hate-speech-detection

“what’s”	to	“what is”
“don’t”	to	“do not”
“doesn’t”	to	“does not”
“that’s”	to	“that is”
“aren’t”	to	“are not”
“’s”	to	“ is”
“isn’t”	to	“is not”
“%”	to	“ percent”
“e-mail”	to	“email”
“i’m”	to	“i am”
“he’s”	to	“he is”
“she’s”	to	“she is”
“it’s”	to	“it is”
“’ve”	to	“ have”
“’re”	to	“ are”
“’d”	to	“ would”
“’ll”	to	“ will”

Table 1: Phrases and words transformations

combining the output of the predictor with the transformed neighbourhood a new oracle dataset has been created.

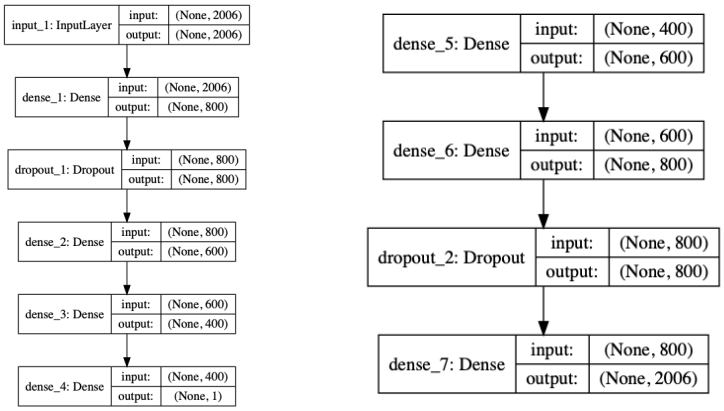
```
input: neighbourhood
output: X,y
transformed_neighbourhood = decoder.predict(neighbourhood)
class_probabilities = predictor.predict(transformed_neighbourhood)
X = transformed_neighbourhood
y = class_probabilities
```

Algorithm 1.1: Oracle dataset synthesis

The last step is to train a transparent model with this oracle dataset. It might be useful to check the distribution of probabilities of this dataset and if needed to transform it to have normal distribution. In these experiments, the transparent model chosen is a Ridge Regression model. By training this model, the coefficient of the features is extracted and transformed into explanations.

```
input: X,y,instance,feature_names
transparent_model = Ridge().fit(X,y)
coef = transparent_model.coef_
plot_explanation(coef*instance,feature_names)
```

Algorithm 1.2: Explaining an instance

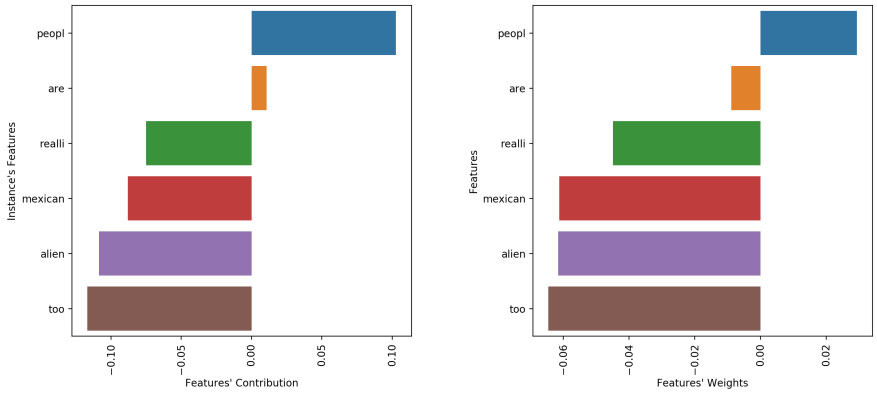


(a) (b)

Fig. 2: The predictor's architecture(a) and the decoder's architecture(b)

4.2 Results on the Hate Speech Dataset

We take the following YouTube comment from the hate speech dataset as example: "aliens really, Mexicans are people too". The true class of this comment is *no hate*. According to the neural network the probability of the *hate* class is approximately 0.00208.



(a) (b)

Fig. 3: Explanation plots of a hate speech instance via (a) LioNets and (b) LIME.

Fig. 3 visualizes the explanation of the neural network's decision via LioNets (3a) and LIME (3b). At first sight, they appear similar. Their main difference is

that they assign the feature's "are" contribution to different classes. By removing this word from the instance the neural network predicts 0.00197, which is a lower probability. Thus, it is clear that the feature "are" it was indeed contributing to the "Hate Speech" class for this specific instance as LioNets explained.

Although to support LioNets explanations, the generated neighbourhoods' distances from the original instance computed and presented in Table 2. As it seems the neighbours generated by LIME on original space, in this example, when are encoded to the reduced space are further to the neighbours generated by LioNets in the encoded space. However, when the LioNets' generated neighbours are transformed back to the original space, are more distant to the original instance in comparison to LIME's neighbours, but that is the assumption that has been made through the beginning of these experiments. It is critical to mention at this point, that these distances measured with neighbours generated by changing only one feature at a time.

	Euclidean distance
LIME: Generated on Original Space	0.3961
LIME: Encoded	0.9444
LioNets: Generated on Encoded Space	0.2163
LioNets: Decoded to Original Space	0.7635

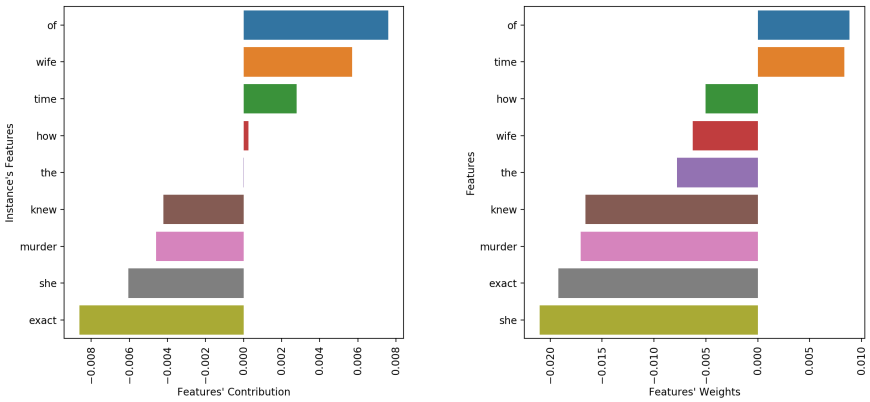
Table 2: Neighbourhood distances for instance of hate speech dataset

4.3 Results on SMS spam dataset

The second example which is going to be explained belongs to the SMS spam dataset. The text of the preprocessed instance is the following: "Wife.how she knew the time of murder exactly". This instance has true class "ham". The classifier predicted truthfully 0.00014 probability to be "spam".

In Fig. 4 there are the two different explanations for the classifier's predictions. Like before, in Fig. 4a is the explanation provided by LioNets and in Fig. 4b is LIME's explanation. The feature's "wife" contribution to the prediction is assigned to different classes on each explanation. To prove the LioNets stability and robustness, this feature is removed and by auditing again the neural network the new prediction is lower with a probability of 0.000095. Thus, it is clear that the feature "wife" it is indeed contributing to the "spam" class as LioNets explained.

Like before, the neighbourhoods' distances from the original instance are computed and presented in Table 3. As it seems the neighbours generated by LIME on original space, in this example, when are encoded are slightly further to the neighbours generated by LioNets in the encoded space too.



(a) (b)

Fig. 4: Explanation plots of SMS spam instance using LioNets(a) and LIME(b)

	Euclidean distance
LIME: Generated on Original Space	0.3184
LIME: Encoded	0.8068
LioNets: Generated on Encoded Space	0.3459
LioNets: Decoded to Original Space	0.7875

Table 3: Neighbourhood distances for instance of SMS spam collection

5 Conclusion

In summary, LioNets architecture seems to provide valid, and comparable to other state of the art techniques, explanations for the decisions of a neural network, while at the same time they guarantee better adjacency between the generated neighbours of an instance because the generation of the neighbours is performing on the penultimate layer of the network. Additionally, LioNets can create better and larger, and by extension more representative neighbourhoods, because the generation process takes place in the encoded space, where the instance has a dense representation. These are the main points of creating and using LioNets on decision systems like neural networks. One main disadvantage of LioNets is that they are focused only on explaining neural networks, thus they are not model agnostic models. Moreover, the overall process of building LioNets is more difficult than training neural network predictors. Future improvements will contain testing the LioNets methodology on different variations of encoders and decoders, implementing more complex neighbourhood generation processes and neighbours selection processes, as well as exploring different transparent models for explaining the instances like rule-based models, decision tree models or models based on argumentation.

Acknowledgment

This paper is supported by European Union's Horizon 2020 research and innovation programme under grant agreement No 825619. AI4EU Project[3].

References

1. Adadi, A., Berrada, M.: Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). IEEE Access **6**, 52138–52160 (2018). https://doi.org/10.1109/ACCESS.2018.2870052, https://ieeexplore.ieee.org/document/8466590/
2. Almeida, T.A., Hidalgo, J.M.G., Yamakami, A.: Contributions to the study of sms spam filtering: new collection and results. In: Proceedings of the 11th ACM symposium on Document engineering. pp. 259–262. ACM (2011)
3. Anagnostou, A., Mollas, I., Tsoumakas, G.: Hatebusters: A Web Application for Actively Reporting YouTube Hate Speech. In: IJCAI (2017), http://www.inach.net/index.php
4. Frosst, N., Hinton, G.: Distilling a Neural Network Into a Soft Decision Tree. arXiv preprint arXiv:1711.09784 (11 2017), http://arxiv.org/abs/1711.09784
5. Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., Giannotti, F.: Local Rule-Based Explanations of Black Box Decision Systems. arXiv preprint arXiv:1805.10820 (5 2018), http://arxiv.org/abs/1805.10820
6. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A Survey of Methods for Explaining Black Box Models. ACM Computing Surveys **51**(5), 1–42 (8 2018). https://doi.org/10.1145/3236009, http://dl.acm.org/citation.cfm?doid=3271482.3236009
7. Hinton, G., Vinyals, O., Dean, J.: Distilling the Knowledge in a Neural Network. arXiv preprint arXiv:1503.02531 (3 2015), http://arxiv.org/abs/1503.02531
8. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F.E.: A survey of deep neural network architectures and their applications. Neurocomputing **234**, 11–26 (4 2017). https://doi.org/10.1016/j.neucom.2016.12.038, https://linkinghub.elsevier.com/retrieve/pii/S0925231216315533
9. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)
10. Porter, M.F.: Snowball: A language for stemming algorithms. Published online (October 2001), http://snowball.tartarus.org/texts/introduction.html, accessed 11.03.2008, 15.00h
11. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16 (2 2016), https://arxiv.org/abs/1602.04938
12. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-Precision Model-Agnostic Explanations. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018), www.aaai.org
13. Sadowski, P., Collado, J., Whiteson, D., Baldi, P.: Deep Learning, Dark Knowledge, and Dark Matter (8 2015), http://proceedings.mlr.press/v42/sado14.html
14. Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of documentation **28**(1), 11–21 (1972)

[3] https://www.ai4eu.eu