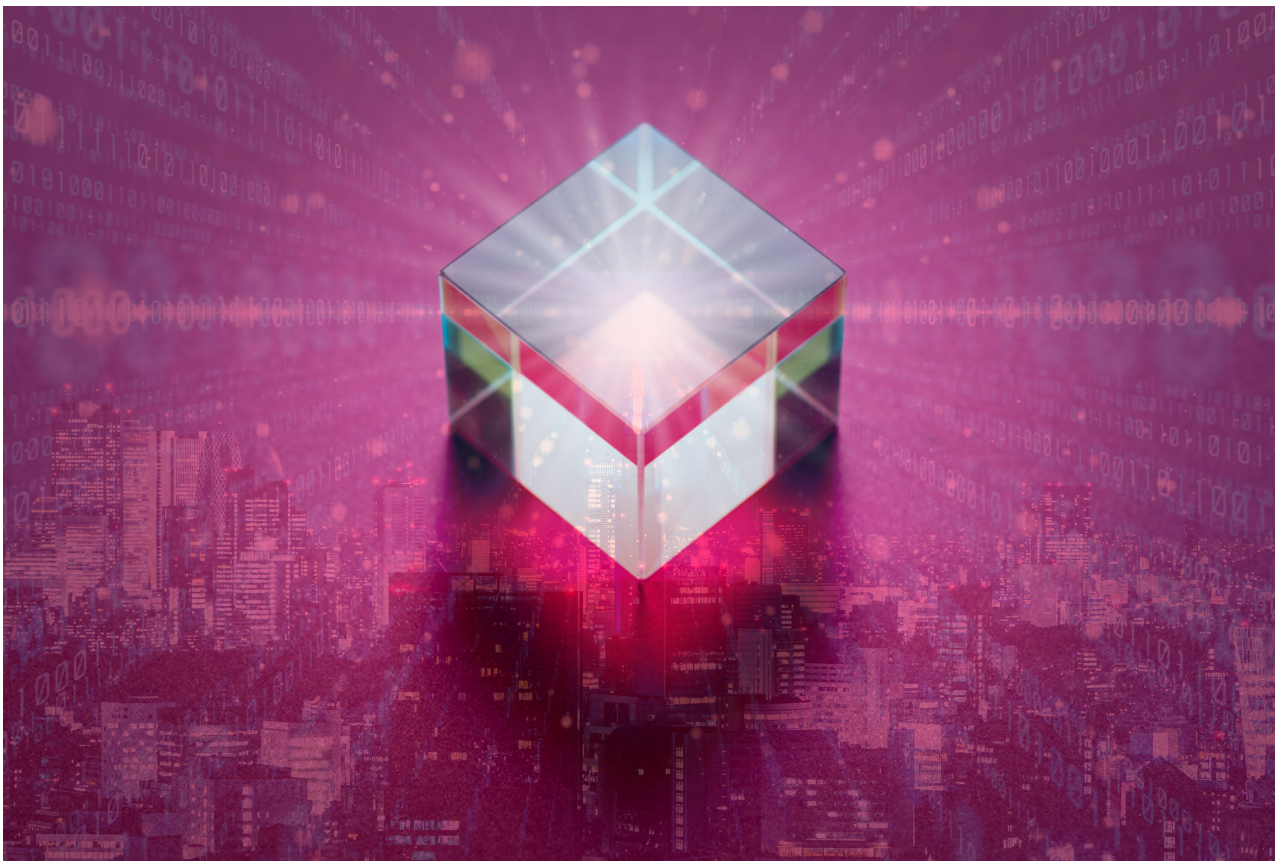


Arya-xAI: Accelerating the path to ML transparency



Introduction

AI and ML technologies have found their way into core processes of industries like financial services, healthcare, education, etc. Even with multiple use cases already in play, the opportunities with AI are unparalleled and its potential is far from exhausted.

However, with increasing use of AI and ML among AI-driven organizations, ML engineers and decision makers who rely on AI outcomes, are now faced with explaining and justifying the decisions by AI models. Decisions have already been made, with the formation of various regulatory compliance and accountability systems, legal frameworks, requirements of Ethics and Trustworthiness. Ultimately, an AI model will be deemed trustworthy only if its decisions are explainable, comprehensible and reliable.

Today, multiple methods make it possible to understand these complex systems, but they come with several challenges to be considered.

While 'intelligence' is the primary deliverable of AI, 'Explainability' has become the fundamental need of a product. It helps to serve important purposes like:

- 1.Accountability
- 2.Trust and transparency
- 3.Better Model Pruning
- 4.Better AI controls

Arya.ai has innovated a state-of-the-art framework, 'Arya-xAI' to offer transparency, control and Interpretability on Deep learning models. This whitepaper explores the explainability imperative, tangible business benefits of XAI, overview of current XAI methods, their challenges and details on the functioning of Arya-XAI framework.

Why is XAI needed?

Explainable AI has been taking a lot of attention in recent times, while AI adoption is increasing. The sophistication and complexity of AI systems has evolved to an extent that they are difficult for humans to comprehend. There is a commonality between the complexity of the model and complexity of explanations - the more complex a model is, the more tough it is to explain.

From a regulatory standpoint, it becomes imperative to understand how the model reached a particular decision, and whether or not the decision was a result of any bias. It's not just from a regulation standpoint, but also from a fundamental model building standpoint, especially for the ML team- if one can understand how the model is functioning and how it is working behind the scenes, it gets easier to find ways for improving the model.

Whereas from a business user standpoint, there needs to be confidence in the system that can be built by providing a clear understanding of how the model is working, and what would be the scope boundaries for the model. There is also a need to validate the product before it can be used in production.

Let's consider the example of financial services. Typical stakeholders in an underwriting process are underwriter, a risk manager, or sales expert, or an actuary. They need to have a comfort that the product is serving their purpose, and it's serving it rightly. To validate that the product is working accurately, and be confident on how the system is functioning on those use cases, they will have a lot of questions on how it works, how did it arrive at the decision, or if they trust the AI decision?

The risk managers will play a critical role in defining whether or not the AI solution can be productionized. Their questions will not only be around system performance, but also system fairness and regulatory compliance - like data bias, any unforeseen

circumstances, undue fairness to certain classes of users because of the input data definition.

The auditor will have questions around tracking the progress and consistency in the system when it is being productionized - when the risk metrics are not consistently maintained, they do not have a consistent risk function, which is again a high risky proposition.

Customers again, will have the right to know the reason for the transaction. Let's say as part of underwriting, the system is predicting that the case cannot be accepted, then the customer has the right to know the reasons behind this decision.

If these questions are extrapolated for any vertical, not just financial services, like healthcare, retail or manufacturing use cases, the questions are very similar - how can I trust the AI decision? Is it consistent?

While we see more use cases where AI is being used as the core driver, be it the end user or the primary builder themselves, both are now seeing the importance of having transparency in the system. The inability to see inside these 'opaque' systems hinders AI adoption and scalability, and hence there are increasing requests around Explainable AI.

Current Methods of XAI in Deep Learning

Current methods can be categorized into **Visualization based methods** and **Distillation methods**.

Visualization Methods

Distillation Methods

Backpropagation-based	Perturbation-based	Local Approximation	Model Translation
Activation maximization	Occlusion Sensitivity	LIME (Local Interpretable Model-agnostic Explanations)	Tree Based
Deconvolution	Representation Erasure	Anchor-LIME	FSA Based
CAM and Grad-CAM	Meaningful Perturbation	STREAK	Graph Based
Layer-Wise Relevance Propagation	Prediction Difference Analysis		Rule Based
DeepLIFT	SHAP		
Integrated Gradients	CEM		

Let's take a look at these methods:

Visualization based methods

a. Backpropagation-based methods:

Backpropagation-based methods highlight parts of your data, which are strongly inferential towards the output. They can either do this by considering the network structure or using different components within the network itself.

Within back-propagation methods where network structures are used, there are methods such as Activation maximization, Deconvolution, Class activation maps (CAM) and Grad-CAM, which utilize either the pooling layers or convolution layers activation function to propagate back which input was

responsible for that particular output. So, these methods heavily rely on convolution layers to be present in the network.

Layer wise relevance propagation method uses scalar expansion to propagate relevance from the output towards the input layers. **DeepLIFT** and **Integrated Gradients** use a reference sample to calculate the impact based on the change between the samples. DeepLIFT directly takes the differences between different components of the network, and the Integrated Gradients method integrates the gradients along the complete path (from output to input) between the sample and reference to calculate the influence of a particular feature on the output, which can be both positive or negative.

b. Perturbation-based methods:

These methods treat the network as a complete black box. They try to remove or alter information to see what are the changes in the network output. Based on those changes, they can either highlight the change that was selecting more, or calculate marginal modular contributions based on each part of the data set. It can be a different feature in case of tabular data, or a word in case of subtext in GIF images, or they can be parts of images or a couple of pixels.

In the **Occlusion Sensitivity** technique, parts of the ocular sensitivity are removed to understand a network's sensitivity in different regions of the data using small perturbations of the data. Whereas, **Representation Erasure** is more data agnostic - it can be applied to any kind of data. For example, the technique can be used on textual data to move words, or for tabular data to move one or more features, and in case of image data, to erase parts of images. The eraser can be in any form, and can either use a constant value or average values.

Essentially the input is altered to see how the output is being affected - this requires multiple iterations to get an explanation.

SHAP (SHapley Additive exPlanations) is a game theoretic approach. SHAP considers marginal contribution for every feature. In case of previous metrics, it is easier to calculate marginal contribution in tree based methods, because the features are in a specific order, and based on that order, different orders can be constructed, or different types of levels can be created within the same sample. This technique also makes it possible to see average values of every node before branching. But this order doesn't exist in deep learning. For a given sample data set, SHAP permutes the values for each feature, to see how much the particular feature is affecting your output, and that marginal contribution will be your estimation for that particular feature.

Distillation based methods

Distillation based methods build a white box model in conjunction with your current model. For a trained neural network, distillation methods will build a separate, less complex model to explain what is happening within the network. This happens through two methods - Local approximation and Model translation.

a. Local approximation:

These methods use a small range of data, or a small data set, concentrated around some particular score to approximate the behavior of the neural network. And within that range, they explain what should be the reason, or what inputs were more responsible for that particular score. Analogous to Taylor expansion, (which converts any complex, non-linear function into a linear function) local approximation method builds a white box model, although complex, but on a smaller range.

LIME (Local Interpretable Model Agnostic Explanations) puts together a surrogate model on a small dataset. It is generally useful for random forests, but can also be used for any particular algorithm.

b. Model translation:

These methods use the complete data to try to map the input to the network predictions. The input can be mapped to the predictions with the white box model, and then using the white box model, it is possible to explain how decisions are being made. So in this case, a surrogate model is used instead of the original model to explain results.

In model translation, there can be Tree based models, graphs, or even rule based models. So basically, for a completed neural network, a neural network with large depth, each node can be converted into a rule, and these techniques try to explain how the model works. But this has its own caveats, since a continuous system is being converted into a discrete system.

Challenges with existing methods

Most of the algorithms currently being used provide only a positive or a negative inference for every feature. This does not hold true in case of deep learning, since every node is contributing to each node above it.

- **Dependency on network architecture:** The above mentioned algorithms are heavily dependent on the component being used. One needs to define the architecture based on the algorithm being used for estimation .
- **Output from the algorithms:** The outputs from the algorithms require a separate reference sample or sample data set. Ideally, since the model and sample both are present, so it is possible to explain how the material sample was being used in such a network. A second set of data or a data sample will not be required to explain the same.
- **Independent functioning of algorithms irrespective of data type:** There are limitations in the existing methods based on the type of input data being used. Ideally, an algorithm should be able to work with all types of data - digital, image, text, or tabular data, there can also be a mix of all three and the algorithm should be able to handle all of them.
- **Explanation of each part of the network:** Current methods do not provide explanation for each part of the network. The contribution of any singular unit to all connected units might not be considered

Ideology behind creating a new XAI Approach

Providing explanations is only part of the reason why these outcomes are used, the other part is understanding how the neural network is working and what it is seeing, improvements can be made accordingly. So if there is an explanation of each part of the network, each change within the neural network can be quantified - how is it being fine tuned, which parts should be pruned in case an alternate output is needed, which features have less or more influence so as to assign different weights for each of them, and also see if those weights are compliant with the business processes.

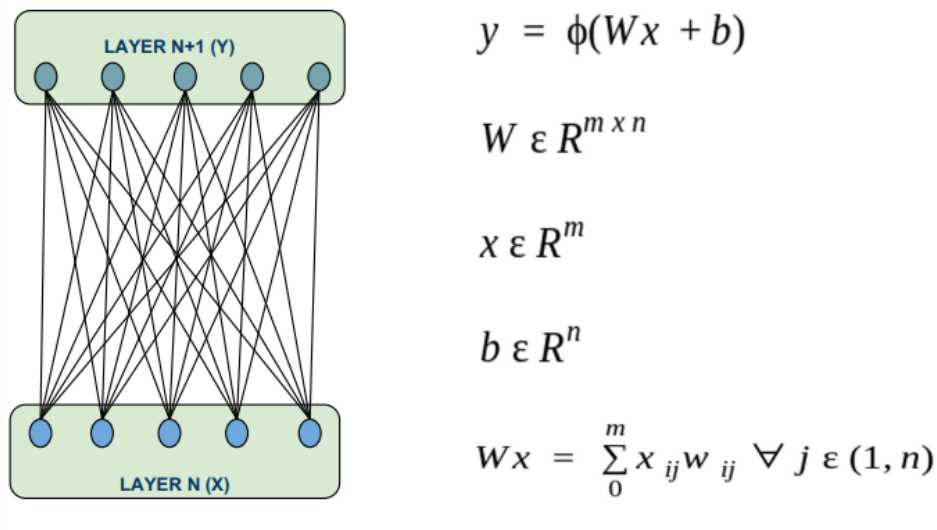
Essentially, each neural network is built to predict certain variables in the process. For example, while automating a particular process in financial services, the data will have different ordering of each feature, some more important and some less important than others. Now for the model to make a decision, the same ordering should be reflected to prove that it is able to directly replicate this particular business process. Our aim was to have a model that can address the challenges briefly mentioned earlier:

- The algorithm should function independent of network architecture
- The output of the algorithm should be only based on the model and input data
- Algorithm should function independently with respect to the input data type
- The algorithm should consider the contribution of any singular unit to all connected units

Backtrace: The core part of Arya-xAI

We introduced a new patent pending approach called 'Back-trace' to explain Deep Learning systems. It can generate true to model explanations Local/Global by assessing the model directly instead of approximating it from input and outputs.

The fundamental idea was using the most basic structure we had, which was a fully connected perceptron model



As presented above, there is one layer below layer x, which is layer y. Each node is connected to every other node on the above layer, and is defined by a basic equation:

$y = \Phi(Wx + b)$ where,

Φ = activation function

W = weight matrix of the layer

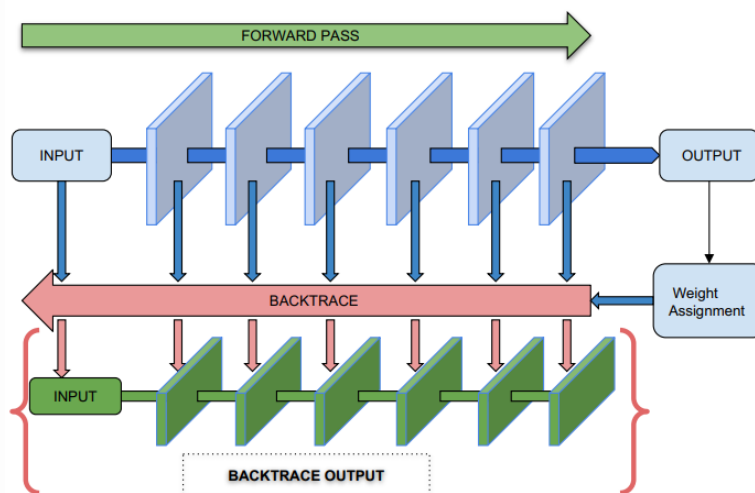
b = bias

x = input

y = output

Since each node is connected to the node below, the contributing node should have both kinds of influence, both positive and negative. Considering that each node in each unit is contributing, there should be post inferences allocated to the node which is connected to the output node. This was the starting point, based on which we wrote the initial algorithm that used this provision, which was then scaled up to reflect the probability of each layer type, commonly used in different neural networks. We are still expanding that to cover additional layer types. This is an overview of the operation which happens.

Let's understand how this connects for a complete network:

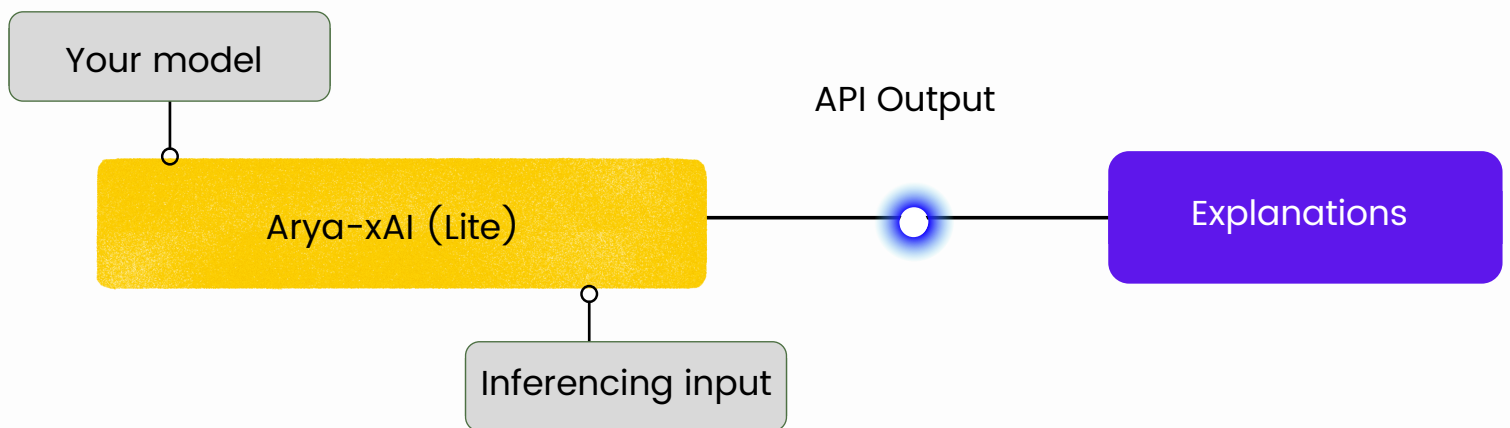


1. *For any sample, the forward pass is done and output from all layers is stored*
2. *The network output is assigned initial weightage based on the scope of intended usage*
3. *This initial weightage and the layer outputs are used to allocate weightage to each subsequent layer starting from final layer*
4. *The algorithm ends when all the input layers are allotted their respective weightage*

So currently, we are automatically assigning for classifications and immigration cases. We also have a functionality where the user can put in their own weight maps. Users can use tabular data, or use multiple images in the same network, and each of them will be assigned relevance weights based on the output.

Arya-xAI API

Arya-xAI API offers back-trace as a simple **plug and play XAI API**. The API adds explainability to your models in just a few lines of code.



Benefits:

- User friendly
- Preserves multivariate relationships
- Works without reference or sample data
- Analyzes actual final model instead of refitted or proxy
- Analyzes inner functioning of model rather just inputs and outputs
- Provides both positive and negative influences from data

Back-Trace: comparison and benefits

As mentioned earlier, our goal was to make the algorithm data type agnostic.

Multiple types of data - tabular, text, images, characters, or a combination of these can be used to feed data into the network. The algorithm works for both classification integration use cases, without requiring any separate data set, sample or a surrogate monitor to get the explanations. Below is a comparison of Back-trace with other techniques:

Method	Models	Explanations	Classification	Regression	Tabular	Text	Images	Categorical features	Train set required
ALE	BB	global	✓	✓	✓				✓
Anchors	BB	local	✓		✓	✓	✓	✓	For Tabular
CEM	BB* TF/Keras	local	✓		✓		✓		Optional
Counterfactuals	BB* TF/Keras	local	✓		✓		✓		No
Prototype Counterfactuals	BB* TF/Keras	local	✓		✓		✓	✓	Optional
Integrated Gradients	TF/Keras	local	✓	✓	✓	✓	✓	✓	Optional
Kernel SHAP	BB	Local/global	✓	✓	✓			✓	✓
Back-Trace	Keras	Local/Global	✓	✓	✓	✓	✓	✓	No

Benefits:

- Decodes 'true-to-model' feature weightage and node level weightages for any neural network
- Generates explanations at local level and global level
- Offers contrastive explanations on how the variations of the features can impact the final verdict
- Provides feature importance that's comparable and combinable across the dataset, and provides justifications for prediction accuracy
- Opens up the network and offers node and layer level explanations
- Monitors the model in production and informs about the model drift or data drift
- Provides more advanced controls for Audit/Risk/Quality control teams to ensure that the AI solution adheres to business guidelines

Conclusion

Often, organizations focus on the predictive skill of an AI/ ML model rather than explainability, and predictive models have streamlined a majority of the core processes across industries - be it Financial services, Healthcare, Manufacturing, etc. However, when it comes to trust, responsible deployment, or model governance, these models fall short.

The tradeoff between accuracy and explainability has always created inhibitions for stakeholders to either adopt for scale AI and ML technologies within their organizations. This is where Explainable AI (XAI) comes in - to eliminate the tradeoff and make outcomes and activities of AI explainable, transparent and reliable.

Since AI explainability is becoming a norm, organizations have begun to move beyond treating these models as 'blackboxes', using a multitude of methods available to explain even the complex AI systems. Data teams can start using XAI to monitor KPIs and improve outcomes, whereas business users can explain the actions and insights of the systems, creating an ecosystem where organizations can take accountability for their AI systems.



Resources

- Know more about Arya-xAI: <https://xai.arya.ai/>
- Explore Arya-xAI documentation: <https://arya-xai.readthedocs.io/en/latest/>
- Arya-xAI workshop: <https://resources.arya.ai/arya-xai-explainable-ai-workshop-video/>
- Notebook: <https://github.com/Arya-ai/XAI>

About Arya.ai

Arya.ai offers a low-code 'AI Operating Platform' for financial institutions to deploy, scale and maintain responsible self-learning AI engines for core business functions like Underwriting, Risk Monitoring, Claims Processing etc. all on the same platform. The modular construct of the platform allows financial institutions to scale one function at a time and offers utmost flexibility to centralize the control on AI assets like - Models, Data pipelines, APIs, Security Guidelines etc. This makes it the only AI platform required to achieve organization wide adoption of autonomous AI. Arya brings in the best of both worlds - products & platforms onto a single unified technology stack.

Arya.ai was one of the first startups to deploy deep learning in Financial Institutions since 2013! Arya.ai founders named in Forbes Asia 30 under 30 under 'Technology' category. Arya.ai is also named in 'Top 61 AI Startups global' list by CB Insights.



Contact:

hello@arya.ai



Lithasa Technologies Private Limited

3A 103, WeWork,

Jogeshwari-Vikhroli Link Road,

Andheri East Mumbai,

India - 400076