Explainable AI Through Linear Concept Layer: Advancing Deep Concept Reasoner

Ali Al Housseini
Politecnico di Torino
s333940
Turin, Italy

Roberto Pulvirenti
Politecnico di Torino
s317704
Turin, Italy

Enrico Giacalone
Politecnico di Torino
s320138
Turin, Italy

ABSTRACT

The complexity and opacity of AI systems pose significant challenges in ensuring their trustworthiness and interpretability. Concept-Based Explainable AI (C-XAI) seeks to address these challenges by leveraging human-like concepts to bridge the gap between machine learning models and human understanding. This paper introduces an innovative variant of the Concept Embedding Model (CEM) that incorporates a novel linear layer, aiming to balance accuracy and interpretability as an alternative to Deep Concept Reasoner (DCR). Through comprehensive evaluations on multiple datasets, our approach demonstrates superior precision performance, while simultaneously addressing computational complexity, offering new benchmarks in concept-based XAI.

1 INTRODUCTION

The rapid advancement of artificial intelligence (AI) technologies has led to their widespread adoption across various domains, including healthcare, finance, and autonomous systems. However, the complexity and opacity of these AI systems have raised significant concerns regarding their trustworthiness and interpretability. To address these issues, the field of Explainable AI (XAI) has emerged, focusing on creating AI systems that are not only accurate but also transparent and understandable to human users.

Concept-Based Explainable AI (C-XAI) is a subfield of XAI that aims to bridge the gap between human understanding and machine learning models by leveraging high-level human-like concepts. These concepts serve as intermediaries between the input data and the final predictions, providing a more intuitive explanation of how the model arrives at its decisions. Among the various approaches within C-XAI, significant progress has been made through the development of several models, each evolving to address the limitations of its predecessors.

One of the pioneering models in this field is the Concept Bottleneck Model (CBM) [3]. CBMs are designed to first predict a set of human-specified concepts from the input data and then use these predicted concepts to make the final prediction in a fully-supervised approach. However, CBMs face challenges such as the dependency on a substantial amount of annotated concept data and potential trade-offs between accuracy and interoperability. To overcome some of these limitations, Concept Embedding Models (CEM) [1] were introduced, the goal is to learn high-dimensional concept representations that capture meaningful semantics beyond their ground truth labels. This method aims to improve the model's overall accuracy by offering more complex and adaptable representations of concepts. However, this comes with a trade-off: it reduces interpretability. This is because the individual dimensions of the concept embeddings are not immediately understandable to us.

Building on the foundation of CEMs, Deep Concept Reasoner (DCR) [2] extends the idea of concept-based models by incorporating deeper reasoning capabilities. DCR model enables a more sophisticated understanding and manipulation of concepts to formulate a logical rule. The final prediction is derived by assessing these rules based on the truth values of the concepts, rather than their embeddings. This approach preserves clear semantics and offers a completely interpretable decision-making process. While DCR model boasts superior reasoning abilities and enhanced robustness, it also incurs a higher computational cost. Additionally, its interpretability is limited to a single method.

In this work, we propose a novel variant of the Concept Embedding Models (CEM), which enhances baseline results by means of linear equations. Our investigation includes a comprehensive evaluation of two different models, highlighting their strengths and weaknesses, and providing new benchmark results in the literature. This new variant aims to offer a superior balance between accuracy, interpretability and usability, setting a new standard in concept-based XAI.

2 RELATED WORK

2.1 Concept-Based explainable models

2.1.1 ***Concept Bottleneck Models (CBM).*** CBMs represent an early attempt to make deep learning models more interpretable by structuring them around human-understandable concepts. The core idea is to create an intermediate layer of concepts that the model must predict before making its final decision. This architecture allows for interventions at the concept level, enabling users to correct mispredicted concepts and potentially improve the model's overall accuracy. Pioneering works in this area demonstrated the feasibility of CBMs in applications such as medical diagnosis and fine-grained image classification. However, CBMs often struggled with maintaining high predictive accuracy, particularly when the set of concepts did not capture all the necessary information for the task.

In more detail, a CBM consists of two main components: a concept encoder and a task predictor. Given an input x, the model first predicts a set of concepts c using the concept encoder $g(x)$. These concepts are then used by the task predictor$f(c)$ to generate the final output y. Mathematically, this can be expressed as:

$$\begin{cases} c = g(x) \\ y = f(c) \end{cases} \tag{1}$$

The overall objective of the CBM is to minimize the concept prediction loss L_c and the task loss L_y:

$$\mathbf{L} = L_c(c, \hat{c}) + L_y(y, \hat{y}) \tag{2}$$

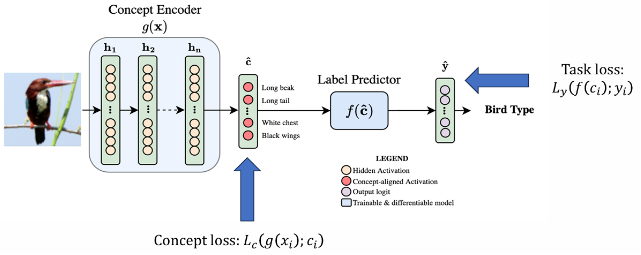

Figure 1: An example of CBM architecture

where $\hat{c}$ and $\hat{y}$ are the ground truth concepts and labels, respectively. This dual-objective optimization ensures that the model learns to accurately predict both the intermediate concepts and the final output.

One of the notable strengths of CBMs is their ability to support human interventions at the concept level. During deployment, if a user identifies a mispredicted concept, they can correct it manually, and the corrected concept values can be fed into the task predictor to improve the final prediction. For instance, in a medical diagnosis application, if a model incorrectly predicts the absence of a certain symptom, a doctor can intervene and correct this prediction, which can subsequently lead to a more accurate diagnosis. This intervention mechanism enhances the model's interpretability and reliability.

Despite their advantages, CBMs have certain limitations. One major drawback is the reliance on the availability of annotated concept data, which can be expensive and time-consuming to obtain. Additionally, the fixed set of predefined concepts might not fully capture all the nuances necessary for accurate prediction, leading to potential gaps in performance. Furthermore, the need to balance the dual-objective optimization can sometimes result in suboptimal performance on the primary task, especially if the concepts are not perfectly predictive of the target labels.

2.1.2 *Concept Embedding Models (CEM).*

To address the accuracy limitations of CBMs, CEMs were developed. CEMs enhance CBMs by embedding concepts into high-dimensional spaces, allowing for richer and more nuanced representations of the input data. This approach improves task accuracy and still supports effective human interventions, differently from hybrid Concept-based models [7]. CEMs have been shown to perform competitively with standard neural models while providing the added benefits of interpretability and intervention capabilities . However, the increased complexity of concept embeddings can sometimes obscure the interpretability benefits, as the relationship between the embedded concepts and the final prediction becomes less transparent.

CEMs introduce a mixture of two embeddings for each concept to capture both the presence and absence of the concept. Specifically, for each concept c_i, two embeddings c_i^+ and c_i^- are learned, representing the active and inactive states of the concept, respectively. The final concept embedding c_i is then a weighted mixture of these two embeddings:

$$c_i = p_i c_i^+ + (1 - p_i) c_i^- \tag{3}$$

where p_i is the predicted probability of the concept being active. This mechanism allows for clear and effective interventions, as correcting a mispredicted concept simply involves switching to the appropriate embedding.

Interventions in CEMs are more robust and effective compared to traditional CBMs. When an expert identifies a mispredicted concept, they can intervene by setting the concept probability p_i to the correct value, thus updating the concept embedding to reflect the true state. This correction propagates through the model, improving the final prediction. Additionally, the use of high-dimensional embeddings ensures that these interventions have a more significant and nuanced impact on the model's output.

However, the complexity of CEMs' high-dimensional embeddings can pose challenges for interpretability. The relationship between the embedded concepts and the final predictions can become less direct, making it harder for users to understand how specific concepts influence the model's decisions. To mitigate this, visualization techniques and regularization strategies can be employed to maintain a degree of interpretability while leveraging the rich representations the embeddings provide.

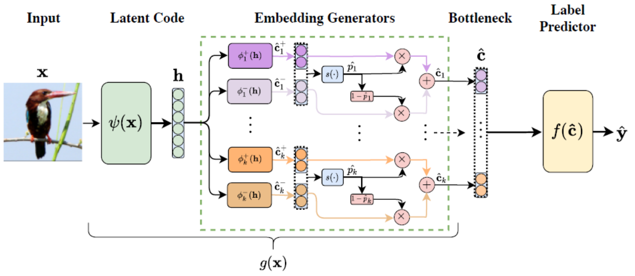

Figure 2: An example of CEM architecture

2.1.3 *Deep Concept Reasoner (DCR).*

By integrating symbolic reasoning with deep learning, DCR aims to combine the strengths of both approaches. The original DCR model introduced a neural-symbolic framework that allowed for logical reasoning over learned concept embeddings, thereby enhancing both the interpretability and robustness of the model. However, the initial implementations of DCR faced challenges related to scalability and the complexity of the reasoning processes.

DCR operates by employing differentiable and learnable modules that apply fuzzy logic rules on concept embeddings. This approach is designed to retain the semantic meaning of concepts while enabling sophisticated reasoning capabilities. The main advantage of DCR lies in its ability to generate interpretable rules from high-dimensional concept embeddings, which are used for the final prediction. These rules can be directly executed on the truth values of concepts, thus providing a fully interpretable decision process.

Mathematically, DCR constructs logic rules by generating and executing fuzzy rules on concept embeddings. Given an input x, the concept predictor $g(x)$ produces concept embeddings c. The fuzzy logic rules are then generated by two neural modules ϕ and ψ. The module ϕ determines the role (positive/negative) of each concept, while ψ assesses the relevance of each concept. These rules are expressed as:

$$y_j \Leftrightarrow \bigwedge_{i:r_{ji}=1} l_{ji} \tag{4}$$

where l_{ji} represented the literal of concept c_i (either c_i or $\neg c_i$), and r_{ji} indicates the relevance of c_i for predicting y_j. The truth degree l_{ji} is computed as:

$$\ell_{ji} = (\phi_j(c_i) \Leftrightarrow c_i) = (\neg\phi_j(c_i) \vee c_i) \wedge (\neg c_i \vee \phi_j(c_i)) \tag{5}$$

The task prediction y_j is then obtained by combining the relevant literals.

$$y_j = \bigwedge_{i=1}^{k} \ell_{r_{ji}} \tag{6}$$

One of the key strengths of DCR is its ability to provide stable and precise explanations, particularly in identifying and correcting prediction errors.

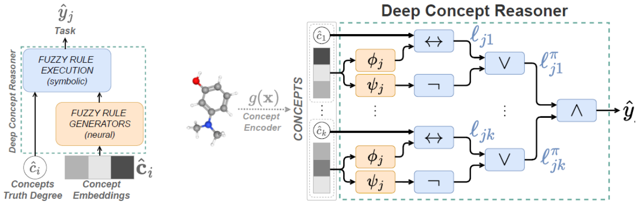

Figure 3: DCR model illustration

2.2 Datasets

2.2.1 ***XOR.*** The first dataset adapted to the experiments is inspired by the exclusive-OR (XOR) problem proposed in [8], a classic example used to demonstrate non-linear separability.

We generate input samples from a uniform distribution within the unit square $\mathbf{x} \in [0, 1]^2$ and define two binary concepts $\{c_1, c_2\}$ by using the Boolean (discrete) transformation to the input features, where $c_i = 1_{x_i > 0.5}$.

Finally, we construct a downstream task label using the XOR operation between the two concepts: y = $c_1 \oplus c_2$.

2.2.2 ***XNOR.*** The XNOR (exclusive NOR) dataset is the negated version to the XOR dataset.

Similar to the XOR dataset it has binary inputs, but the XNOR operation determines the output: y = $c_1 \odot c_2$ (1 if the inputs are equal).

2.2.3 ***DOT.*** Tests the model's ability to generalize and interpret high-dimensional concept spaces. Input samples consist of vectors classified by their alignment with reference vectors. Specifically, two reference vectors w_1 and w_2 are defined, and the dataset includes vectors v_i drawn from a normal distribution. Each vector is classified according to its alignment with w_1 or w_2, challenging the model to identify and separate high-dimensional patterns.

2.2.4 ***Trigonometry.*** Inspired by that proposed by [5], it aims to illustrate the advantages of embedding concept representations over fuzzy concept representations. Samples are generated from three independent latent normal random variables $h_i \sim N(0, 2)$. Seven features for each sample are constructed using non-invertible transformations of these latent factors: three features are of the form $sin(h_i) + h_i$, three are $cos(h_i) + h_i$, and one is $h_1^2 + h_1^2 + h_3^2$. Binary concepts are assigned based on the signs of the latent variables $c_i = (h_i \geq 0)$. The task label y is determined by whether the sum $h_1 + h_2$ is positive, making this a complex, non-linear classification problem.

2.2.5 ***MNIST-Addition.*** Is a standard benchmark for neural-symbolic systems, derived from the well-known MNIST dataset [4], which contains images of handwritten digits from 0 to 9. In particular, in MNIST-Addition [6] each sample is made of a pair of MNIST digit images, to predict the sum of these two digits.

3 RESEARCH GAPS

While the Deep Concept Reasoner (DCR) combines the strengths of Concept Bottleneck Models (CBMs) and Concept Embedding Models (CEMs), it still has several limitations:

- Computational Cost: Training DCR is more computationally intensive than training CBMs and CEMs due to the incorporation of fuzzy logic.
- Dependency on Logic Rules: Although DCR relies on the explainability paradigm of logic rules, there are several issues:
  - Global rules may not perfectly reflect the exact reasoning process of the model.
  - The complexity of these rules can increase significantly and may become contradictory, particularly when dealing with a large number of concepts.
  - Even with a small number of concepts, logic rules can sometimes be challenging for users to interpret.

To address these limitations, this work proposes a new alternative to DCR that maintain the generalization capability and complexity of a CEM while offering interpretable predictions. Specifically, we propose changing the interpretability paradigm of DCR by providing interpretable predictions not through logic rules but through weights associated with the corresponding concepts, akin to a linear equation. This approach aims to make the model's reasoning more technical and precise, thereby enhancing the granularity of explanations.

4 METHODOLOGY

In this section, we introduce two distinct implementations of our Interpretable Linear Layer. While we have previously highlighted the effectiveness of Deep Concept Reasoner (DCR) and its limitations, we adopted a similar design structure by creating a task predictor that processes concept embeddings and concept truth degrees separately. However, unlike DCR, which generates logic rules, our architecture assigns a weight to each concept. This change shifts the paradigm of interpretability, bringing with it various advantages and disadvantages that will be discussed in detail in the following section.

4.1 Enhancing interpretability through linearity

In our first attempt to enhance interpretability using a linear layer, we define the task predictor as two independent Neural Networks:

$\phi_1(x)$, that assigns a weight for each concept, and $\phi_2(x)$ that computes a bias term for each concept.
More precisely $\phi_1, \phi_2 : \mathbb{R}^m \rightarrow \mathbb{R}^n$ where m is the embedding size of a concept, and n is the number of output classes. Additionally, we define $\hat{b}_i$ as the mean of all biases computed by ϕ_2 as:

$$\hat{b}_i = \frac{\sum_{j=1}^{n_c} b_j}{n_c} \qquad \forall \quad i \in \{n\} \tag{7}$$

with n_c represents number of concepts.
Finally, the logits of the architecture are computed as follows:

$$y_i = \sum_{j=1}^{n_c} c_j \cdot w_j + \hat{b}_i \qquad \forall \quad i \in \{n\} \tag{8}$$

4.1.1 ***End-to-End Training.*** To further enhance the interpretability and efficiency of our model, we propose an end-to-end trainable architecture. This model is designed to integrate the advantages of both Concept Bottleneck Models (CBMs) and Concept Embedding Models (CEMs) while addressing the limitations identified in DCR. The fundamental concept is to substitute complex logic rules with a linear blend of concept weights, thereby enhancing the model's transparency and precision in reasoning. As in DCR, the model is composed of two sequential modules: the first creates an embedding representation of the concept, while the second applies ϕ_1 and ϕ_2 accordingly to get the final logits as in 8.

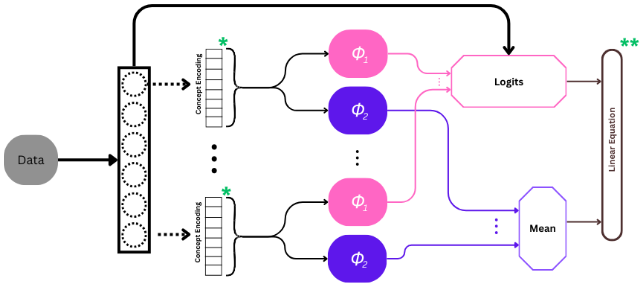

**Figure 4: The figure illustrates the architecture of the proposed end-to-end trainable model. Elements highlighted with a green asterisk are used in loss computation.
(*) Concept Loss (**) Task Loss**

4.1.2 ***Loss Computation and Regularization.*** The loss function is described as:

$$\mathbf{L} = L_C + \alpha L_T \tag{9}$$

where L_C is the concept loss, L_T is the task loss, and α is an hyper-parameter, with $\alpha \in]0; 0.5]$.

We performed a series of comprehensive experiments to find the best value for α; however, despite these efforts, the concept loss results were not satisfactory. Therefore, we added a regularization term associated with the computed bias to further refine the loss function. This term aims to enhance the model's performance by addressing the shortcomings observed in the concept loss. Table 1 displays the changes in accuracy metrics before and after the regularization.

After incorporating the regularization term related to the bias, the validation results improved. This is because, regularization helps prevent overfitting by constraining the model's complexity. This leads to a more generalizable model that performs better on unseen data.

	Regularization	
Dataset	Before	After
	Concept Validation Accuracy (%)	
XOR	74.7	**77.3**
XNOR	72.7	**80.3**
DOT	72.1	**79.3**
Trigonometry	64.0	**81.6**
MNIST-Addition	33.8	**47.6**
	Task Validation Accuracy (%)	
XOR	98.9	**99.4**
XNOR	98.7	**99.4**
DOT	96.9	**97.6**
Trigonometry	97.4	**98.3**
MNIST-Addition	63.2	**86.0**

Table 1: Impact of Regularization: Improved validation accuracy of the linear layer.

The updated loss function is:

$$\mathbf{L} = L_C + \alpha L_T + \lambda ||b||^2 \tag{10}$$

with $\lambda \in [0.1; 0.0001]$

4.1.3 ***Comparison and Limitations.*** Compared to the Deep Concept Reasoner (DCR), the LLR model offers several advantages:

- **Simplicity and efficiency**: It is computationally less intensive than DCR, which involves fuzzy logic rules. Additionally, the model achieved higher scores for assessing prediction abilities, such as precision, recall and F1 scores, demonstrating its effectiveness in various tasks. (See Appendix A.1).
- **Training Speed**: LLR demonstrates faster training times compared to DCR, (see Table 7) making it more efficient for large-scale or time-sensitive applications.
- **Interpretability**: By using linear weights, it provides more straightforward interpretation of how each concept contributes to the final prediction, making it easier to understand the model's reasoning.

The model offers significant advantages in terms of explainability, leveraging the simplicity of linear relationships between concepts and predictions. This transparency allows for straightforward local explanations, where the contribution of each concept to a specific prediction can be easily understood and quantified. For instance, Figure 8 shows how LLR is able to quantify the importance of each concept by assigning a higher weight to the corresponding concept (More in Appendix A.3). Furthermore, the model offers local explainability through weights, which proves highly effective in elucidating model behavior. Table 2 displays the average weight assigned to all input samples for the XOR dataset.

It is evident that for $x_0x_1 = 00$ or $x_0x_1 = 11$, the weights w_0 and w_1 for class $y = 0$ are higher than the corresponding weights for

$c0$	$c1$	$w0\_y0$	$w1\_y0$	$w0\_y1$	$w1\_y1$	b_y0	b_y1	$y0$	$y1$
0	0	0.972	0.977	0.025	0.017	3.210	-3.260	1.0	0.0
0	1	0.669	0.048	0.149	0.925	-3.162	2.552	0.0	1.0
1	0	0.046	0.570	0.934	0.252	-3.237	2.493	0.0	1.0
1	1	0.904	0.930	0.054	0.037	2.062	-3.443	1.0	0.0

Table 2: Average weights computed from LLR v1 on XOR

$y = 1$. This indicates that the model assigns greater importance to both concepts when they are either both "inactive" or both "active".

Conversely, for $x_0x_1 = 01$ or $x_0x_1 = 10$, the weight assignment is less clear, raising concerns about the model's ability to provide a transparent local explanation for all input combinations. The current approach allows the model to represent each concept with a *single positive weight*, which may not sufficiently capture the complexity of the concept even in a local instance.

Table 3 also shows the weights for the XNOR dataset, by looking specifically at the initial condition case $x_0x_1 = 00$ we can notice that weights for $y = 1$ are $w_0 = 0.661$ and $w_1 = 0.664$ where these values do not reflect any kind of explaining the relevance, or role of the concepts. In addition, both biases for $y = 0$ and $y = 1$ are positive, which poses a question about the ability of this model to explain such cases. The following section will explore an enhanced version of the LLR model to address these issues.

$c0$	$c1$	$w0\_y0$	$w1\_y0$	$w0\_y1$	$w1\_y1$	b_y0	b_y1	$y0$	$y1$
0	0	0.385	0.366	0.661	0.664	-3.180	3.133	0.0	1.0
0	1	0.965	0.950	0.039	0.040	2.462	-3.199	1.0	0.0
1	0	0.941	0.932	0.046	0.072	2.525	-3.318	1.0	0.0
1	1	0.027	0.017	0.961	0.975	-3.477	2.119	0.0	1.0

Table 3: Average weights computed from LLR v1 on XNOR

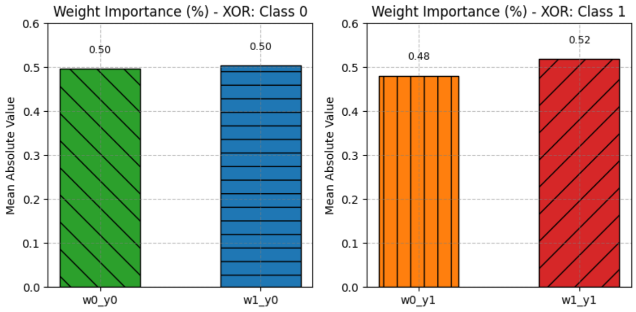

Figure 5: Weight Importance on predicting both classes (0 and 1) on XOR dataset

4.2 Enhancing Interpretability of Linearity

As previously discussed, the linear layer introduced in the preceding section can quantify the impact of a specific concept, while lacks showing the sign of that effect (positive or negative).

For instance, in the XNOR dataset, when $x_0x_1 = 00$, both concepts are inactive, yet the corresponding weights are relatively high. This raises concerns about the interpretability framework's ability to adequately explain this behavior.

In this version, we introduce first, two neural networks $\phi_1^+(x)$ and $\phi_1^-(x)$, where the former returns a weight representing the *positive* contribution of the concept to the prediction. In contrast, the latter instead returns the *negative* contribution weight.
Given an input x_i: $\phi_1^+(x_i) = w_i^+$ and $\phi_1^-(x_i) = w_i^-$. We define w_i as the weight associated with the concept c_i as the difference between the positive and negative weights:

$$w_i = w_i^+ - w_i^-$$

By building weights in this way, the model will adequately assign weights to each concept so that the difference between the positive and negative weight represent the *sign* of contribution of the corresponding concept towards the prediction. Same as for weights, the biases are computed as:

$$b_i = b_i^+ - b_i^-$$

where $\phi_2^+(x_i) = b_i^+$ and $\phi_2^-(x_i) = b_i^-$. We denote this extension as *LLR v2* (See Fig 6). Subsequently, the logits are computed following the same formula as in 8.

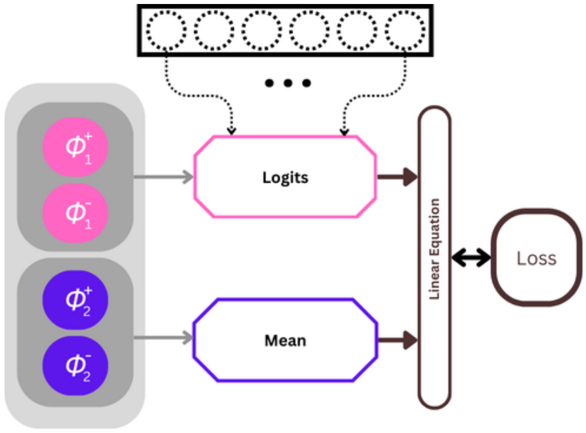

Figure 6: LLR v2 Architecture

5 EXPERIMENTAL INTERPRETABILITY AND ANALYSIS

The LLR model introduces a novel interpretability paradigm that enhances both local and global explainability. Additionally, Table 6 and Table 7 demonstrate that LLR v2 surpasses DCR in terms of both accuracy and computational efficiency. While the inclusion of additional DNNs slightly reduces these metrics, this trade-off results in improved explainability. *This concept is known as the Accuracy-Explainability Trade-Off*

The primary objective of developing an extended version of LLR is to address the former version's limited interpretability on certain benchmarks due to constrained weights. Here, we revisit the XOR and XNOR datasets to illustrate these improvements.

$c0$	$c1$	$w0\_y0$	$w1\_y0$	$w0\_y1$	$w1\_y1$	b_y0	b_y1	$y0$	$y1$
0	0	0.031	-0.005	-0.059	-0.003	3.238	-3.237	1.0	0.0
0	1	-0.999	-0.986	0.999	0.986	-2.521	2.541	0.0	1.0
1	0	-0.926	-0.950	0.920	0.952	-2.514	2.521	0.0	1.0
1	1	0.938	0.981	-0.940	-0.979	2.135	-2.107	1.0	0.0

Table 4: Average weights computed from LLR v2 on XOR

In the XOR dataset, the weights range between $[-1, 1]$. By analyzing each input combination, we can discern how the inputs influence the outputs through weight assignments. For instance, $y_0 = 1$ (indicating the output is 0) can be achieved either by both concepts being "inactive" or both being "active." This is evident from the weights assigned to these concepts, where both w_0 and w_1 are either high or low in these scenarios. The bias term plays a crucial role in determining how the linear decision boundary is positioned in the embedding space to ensure separation between the two distinct classes.

Examining the other cases, $x_0x_1 = 01$ or $x_0x_1 = 10$, reveals a more powerful interpretation. In both cases, the weights are close to 1 (for $y = 1$), indicating that both concepts are relevant and positively contribute to the output prediction.

This interpretation for XOR appears similar to the previous LLR v1 case. However, the XNOR dataset presents a different scenario. Here, when both concepts are "inactive," the output is HIGH. This was difficult to interpret using LLR v1, but with the relaxed constraint on weight signs in this version, the weights become interpretable. For $x_0x_1 = 00$ in XNOR, high positive weights for both concepts indicate that even when both concepts are "inactive," the model can still output a high value ($y = 1$) due to the positive bias pushing the output towards this value.

$c0$	$c1$	$w0\_y0$	$w1\_y0$	$w0\_y1$	$w1\_y1$	b_y0	b_y1	$y0$	$y1$
0	0	-0.833	-0.877	0.803	0.850	-3.010	3.009	0.0	1.0
0	1	0.775	0.934	-0.791	-0.937	2.612	-2.604	1.0	0.0
1	0	0.907	0.679	-0.908	-0.697	2.588	-2.593	1.0	0.0
1	1	-0.930	-0.918	0.926	0.910	-2.014	1.996	0.0	1.0

Table 5: Average weights computed from LLR v2 on XNOR

Overall, the LLR v2 model demonstrates a robust ability to provide clear and interpretable explanations across all benchmarking datasets. In the XOR and XNOR datasets, the model effectively distinguishes the importance of each concept through weight assignments, addressing previous limitations in interpretability. Notably, in the MNIST-Addition dataset, LLR v2 assigns high weights to concepts related to the added numbers while assigning very small weights to non-relevant concepts. This highlights the model's capability to handle highly computational datasets, effectively identifying and emphasizing the most important concepts for each local instance. For detailed tables on other datasets, please refer to Appendix A.2.

6 CONCLUSIONS

In this study, we introduced a novel variant of the Concept Embedding Models (CEMs) that integrates a linear layer to simultaneously enhance interpretability and accuracy. Our comprehensive evaluations across multiple datasets revealed that this approach effectively mitigates the limitations of previous models, including high computational costs and reliance on complex logic rules. The incorporation of a linear layer enables more transparent and interpretable predictions, establishing a new standard in explainable AI. The results demonstrate significant improvements in both concept and task accuracy, offering a new interpretability paradigm based on weights derived from linear equations. This research highlights the critical need for developing AI systems that are both accurate and transparent, fostering greater trust and reliability in AI applications.

Model	**Dataset**				
	XOR	XNOR	Dot	Trigonometry	MNIST-Addition
	Concept Validation Accuracy (%)				
DCR	**77.4**	76.6	79.0	76.9	**70.9**
LLR v1	77.3	**80.3**	**79.3**	**81.6**	47.6
LLR v2	73.4	73.6	78.7	75.6	44.6
	Task Validation Accuracy (%)				
DCR	99.3	99.3	96.3	98.3	60.7
LLR v1	**99.4**	**99.4**	97.6	98.3	**86.0**
LLR v2	**99.4**	98.9	**98.4**	**99.0**	82.3

Table 6: Validation accuracy results

Model	**Dataset**				
	XOR	XNOR	Dot	Trigonometry	MNIST-Addition
	Time (seconds)				
DCR	**4.853**	6.021	12.230	7.973	939.191
LLR v1	5.602	**4.677**	7.773	4.434	**266.971**
LLR v2	5.319	7.881	**6.377**	**1.980**	331.654
	Epoch (#)				
DCR	**11**	**13**	33	10	52
LLR v1	16	**13**	18	5	**16**
LLR v2	14	18	**15**	**4**	17

Table 7: Validation convergence results

REFERENCES

[1] Mateo Espinosa Zarlenga et al. 2022. Concept embedding models. In *NeurIPS 2022-36th Conference on Neural Information Processing Systems.*

[2] Pietro Barbiero et al. 2023. Interpretable neural-symbolic concept reasoning. In *International Conference on Machine Learning.*

[3] Pang Wei Koh et al. 2020. Concept bottleneck models. In *International conference on machine learning.*

[4] Y. LeCun et al. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE.*

[5] A. Mahinpei, J. Clark, I. Lage, F. Doshi-Velez, and W. Pan. 2021. Promises and pitfalls of black-box concept learning models. arXiv preprint.

[6] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt. 2018. Advances in Neural Information Processing Systems. Deepproblog: Neural probabilistic logic programming.

[7] Alvarez Melis, David, and Tommi Jaakkola. 2018. Towards robust interpretability with self-explaining neural networks. Neurips.

[8] M. Minsky and S. Papert. 1969. Perceptrons: An Introduction to Computational Geometry. MIT Press.

A APPENDIX

A.1 Results

All experiments were done using Nvidia A100 GPU, using the following parameters.

- Batch size = 64
- Optimizer: *Adam* with learning rate **0.0005** and weight decay **0.01**
- Scheduler: *ReduceLROnPlateau*, with a factor of **0.0001** and patience (number of epochs) equal to **7**.
- Fixed Number of epochs **100**.

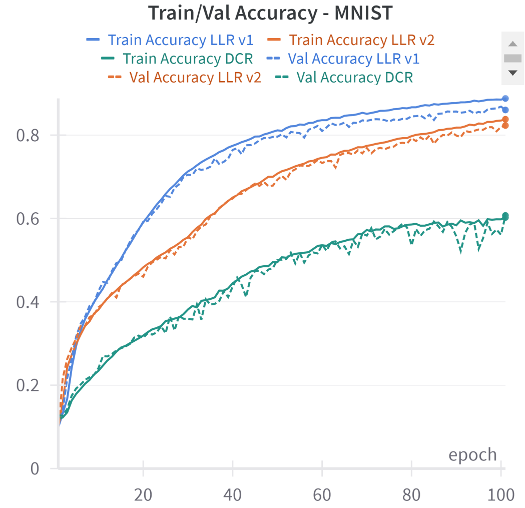

Figure 7: Train/Val Accuracy on MNIST-Addition dataset

Model	**Dataset**				
	XOR	XNOR	Dot	Trigonometry	MNIST-Addition
			Validation Precision (%)		
DCR	99.3	99.3	96.3	98.3	55.0
LLR v1	**99.4**	**99.4**	97.6	98.3	**86.1**
LLR v2	**99.4**	98.9	**98.4**	**99.0**	82.4
			Validation Recall (%)		
DCR	99.3	99.3	96.3	98.3	60.7
LLR v1	**99.4**	**99.4**	97.6	98.3	**86.0**
LLR v2	**99.4**	98.9	**98.4**	**99.0**	82.3
			Validation F1 score (%)		
DCR	99.3	99.3	96.3	98.3	57.5
LLR v1	**99.4**	**99.4**	97.6	98.3	**86.0**
LLR v2	**99.4**	98.9	**98.4**	**99.0**	82.3

Table 8: Validation Precision, Recall and F1 score results

A.2 Weights

$c0$	$c1$	$w0\_y0$	$w1\_y0$	$w0\_y1$	$w1\_y1$	b_y0	b_y1	$y0$	$y1$
0	0	0.781	0.711	0.215	0.263	1.618	-1.706	0.782	0.218
0	1	0.242	0.239	0.744	0.754	-2.248	1.470	0.266	0.734
1	0	0.235	0.211	0.755	0.781	-2.548	1.702	0.224	0.776
1	1	0.764	0.760	0.224	0.227	-2.741	-2.188	0.799	0.201

Table 9: Average weights computed from LLR v1 on Dot

$c0$	$c1$	$w0\_y0$	$w1\_y0$	$w0\_y1$	$w1\_y1$	b_y0	b_y1	$y0$	$y1$
0	0	0.687	0.464	-0.686	-0.443	1.787	-1.776	0.782	0.218
0	1	0.104	-0.500	-0.095	0.496	-1.488	1.490	0.266	0.734
1	0	-0.554	-0.468	0.556	0.484	-1.643	1.618	0.224	0.776
1	1	0.553	0.550	-0.552	-0.549	1.136	-1.125	0.799	0.201

Table 10: Average weights computed from LLR v2 on Dot

$c0$	$c1$	$c2$	$w0\_y0$	$w1\_y0$	$w2\_y0$	$w0\_y1$	$w1\_y1$	$w2\_y1$	b_y0	b_y1	$y0$	$y1$
0	0	0	0.858	0.734	0.734	0.130	0.156	0.156	3.124	-3.249	1.000	0.000
0	0	1	0.685	0.249	0.249	0.324	0.655	0.655	2.334	-3.016	0.923	0.077
0	1	0	0.331	0.918	0.918	0.650	0.081	0.081	2.400	-3.148	0.905	0.095
0	1	1	0.186	0.403	0.403	0.815	0.585	0.585	-1.315	-0.122	0.395	0.605
1	0	0	0.878	0.450	0.450	0.117	0.468	0.468	-2.053	-2.897	0.877	0.123
1	0	1	0.417	0.157	0.157	0.573	0.812	0.812	-1.228	-0.255	0.421	0.579
1	1	0	0.364	0.348	0.348	0.625	0.645	0.645	-1.548	0.097	0.382	0.618
1	1	1	0.009	0.008	0.008	0.989	0.991	0.991	-3.149	1.314	0.008	0.992

Table 11: Average weights computed from LLR v1 on Trigo

$c0$	$c1$	$c2$	$w0\_y0$	$w1\_y0$	$w2\_y0$	$w0\_y1$	$w1\_y1$	$w2\_y1$	b_y0	b_y1	$y0$	$y1$
0	0	0	0.999	0.989	0.989	-0.999	-0.993	-0.993	3.219	-3.205	1.000	0.000
0	0	1	0.835	0.654	0.654	-0.844	-0.668	-0.668	2.221	-2.229	0.923	0.077
0	1	0	0.557	0.822	0.822	-0.574	-0.821	-0.821	2.233	-2.224	0.905	0.095
0	1	1	-0.191	-0.189	-0.189	0.177	0.193	0.193	-0.412	-0.366	0.395	0.605
1	0	0	0.765	0.659	0.659	-0.759	-0.692	-0.692	-2.045	-2.036	0.877	0.123
1	0	1	-0.138	-0.194	-0.194	0.136	0.162	0.162	-0.475	-0.507	0.421	0.579
1	1	0	-0.277	-0.275	-0.275	0.279	0.266	0.266	-0.485	0.486	0.382	0.618
1	1	1	-0.981	-0.984	-0.984	0.981	0.984	0.984	-1.464	1.422	0.008	0.992

Table 12: Average weights computed from LLR v2 on Trigo

A.3 Local Explanation Analysis

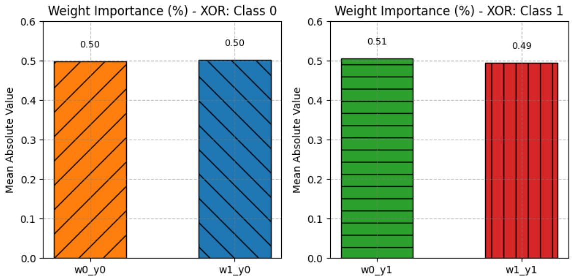

Figure 8: Weight Importance - LLR v1 - XOR dataset

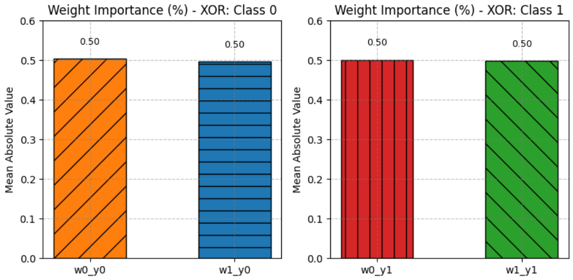

Figure 9: Weight Importance - LLR v2 - XOR dataset

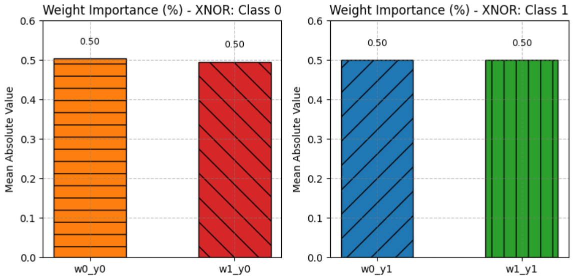

Figure 10: Weight Importance - LLR v1 - XNOR dataset

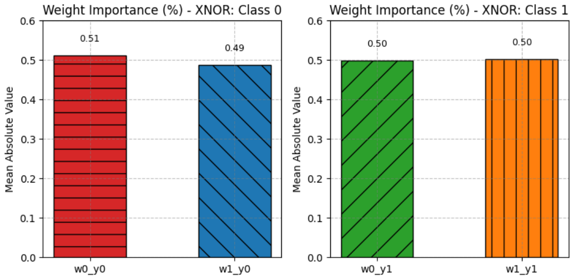

Figure 11: Weight Importance - LLR v2 - XNOR dataset

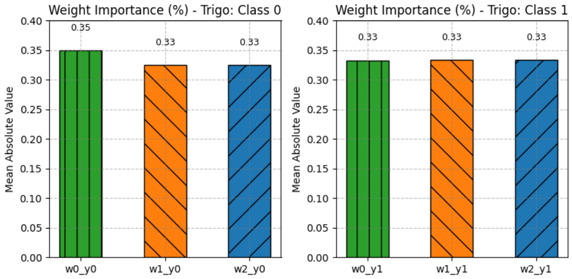

Figure 12: Weight Importance - LLR v1 - Trigo dataset

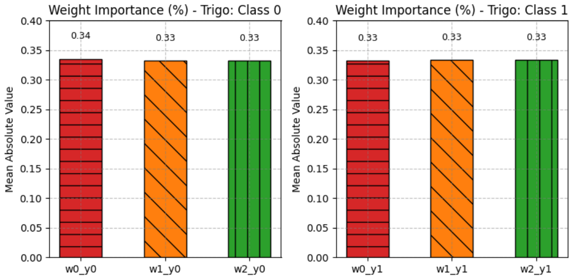

Figure 13: Weight Importance - LLR v2 - Trigo dataset

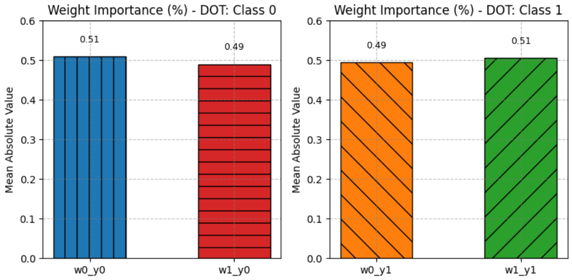

Figure 14: Weight Importance - LLR v1 - DOT dataset

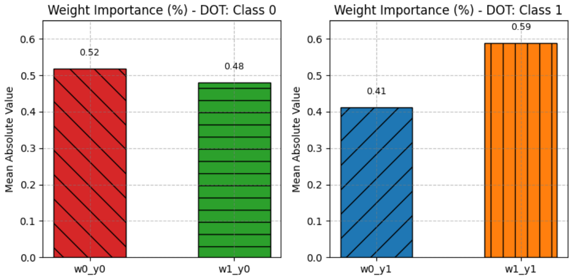

Figure 15: Weight Importance - LLR v2 - DOT dataset