[CSCI 566] Discerning Deep Learning Final Report

Nicholas Klein nmklein@usc.edu
Advait Rane aprane@usc.edu
Yang Cheng ycheng04@usc.edu
Rajat Singh rajatsin@usc.edu

Abstract

Deep learning models are successful in solving complex tasks in domains like images, videos, and graphs. However, the black box nature of such models has prevented their adoption in critical and high-risk use cases. Inherently interpretable models provide explanations for their outcomes that are faithful to model computations. The Concept Bottleneck Model (CBM) and the Prototypical Parts Network (ProtoPNet) are two such models. We aim to improve the interpretability and data-efficiency of these models by augmenting them with decision trees, auxiliary concept datasets, and other interpretability techniques. Our methods result in a comparable classification performance with improved interpretability and data-efficiency while learning intuitive and faithful concepts.

1 Introduction

The black-box nature of Deep Learning models makes their adoption difficult and unsafe in high-risk scenarios with high cost-of-error or regulation. Consequently, there has been a surge in research on Explainable AI(XAI), focusing on explainability and algorithm transparency in Deep Learning. Existing research on explainability can be divided into post-hoc and ante-hoc methods. Post-hoc interpretability refers to the application of interpretation methods after model training, for example saliency maps (Simonyan et al., 2014), LIME (Ribeiro et al., 2016), and TCAV (Kim et al., 2018). However, post-hoc methods have been criticized for providing explanations that do not make sense or are not faithful to the model computations (Rudin, 2019).

Ante-hoc methods incorporate interpretability from training stages. Inherently interpretable models make model computations transparent while achieving comparable performance. They typically learn a latent representation that can be reasoned about by humans, often in the form of disentangled concepts as shown in figure 1.

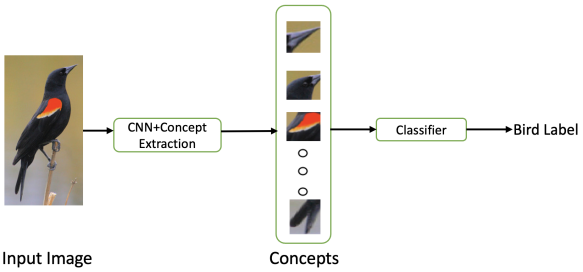

Figure 1: Concept-based model architecture

Concept Bottleneck Models (CBMs) (Koh et al., 2020) modify network architecture to learn concepts as binary labels at an intermediate layer in a supervised manner. This necessitates concept annotations for each training image, increasing annotation labor and cost. Additionally, CBM has been criticized because it often doesn't attend to the location of the concept in the input image when predicting it, and in fact may not even correspond to anything semantically meaningful (Margeloiu et al., 2021). The Prototypical Parts Network (ProtoPNet) (Chen et al., 2019) learns embeddings of class-specific image patches in an unsupervised manner by imposing clustering objectives in the latent space. These embeddings can be projected back to the input space to visualise the learned concepts.

We build on these two models to improve interpretability, data-efficiency, and faithfulness to concepts. We explore three strategies to achieve these goals-

- To increase model interpretability, we use decision trees which have more interpretative power than a neural linear layer.
- To improve the CBM's faithfulness to concepts, we add a layer of concept-specific chan-

nels to the bottleneck instead of using a single neuron to disentangle each concept.

- We combine auxiliary concept datasets with clustering objectives to build a model with better data-efficiency than the CBM and more control over concepts than the ProtoPNet.

Our methods achieve classification accuracy comparable to the original papers while offering improved interpretability and data-efficiency.

2 Related Work

We restrict our literature review to ante-hoc interpretability. We organize these methods by supervised, unsupervised, and hybird learning approaches.

2.1 Unsupervised Representation Learning

The work by (Li et al., 2018) learns prototypical images in the latent space for each class. These are compared with input images at inference time, and can be viewed through a decoder to yield insights into the model. (Chen et al., 2019) iterated upon this work with the Prototypical-Parts-Network (ProtoPNet) to learn more realistic prototypes. They use a pre-trained CNN and clustering objectives to learn embeddings for patches of training images to generate "prototypical parts" for each class. Self-Explaining-Neural-Nets (SENNs) (Alvarez-Melis and Jaakkola, 2018) simultaneously predict relevance scores for each concept to the input image and then aggregate the concepts and relevance scores to form a prediction. SENNs learn more general concepts rather than extracting prototypical features.

2.2 Supervised representation learning

Concept Bottleneck Models (CBMs) (Koh et al., 2020) add an additional loss on an intermediate concept-layer learning human-readable concepts. However, it requires significantly more annotation for the concept labels of each training image and restricts model interpretation to pre-trained concepts.

Neural backed decision trees (NBDTs) (Alvin Wan, 2021) replace a neural network's final linear layer with a differentiable sequence of decisions and a surrogate loss. NBDTs learn interpretable high-level concepts-to-label mappings, but the high-level concepts output by the black-box neural network remain non-interpretable.

Concept Whitening (Zhi Chen, 2020) performs whitening normalization and orthogonal transformation to align a model's latent space along concepts learned through auxiliary datasets. CW completely de-correlates outputs of all filters which might lead to prediction losses when dealing with highly correlated concepts.

2.3 Hybrid

(Bel'em et al., 2021) address the CBM's data inefficiency by supplementing human annotations with noisy, generated concept labels and weakly supervised training methods. They found improved classification performance with close to no reduction in interpretability. (Sawada and Nakamura, 2022) address this issue with their CBM+AUC model by supplementing the CBM's supervised concepts with the SENN's unsupervised concepts. The CBM+AUC perfoms better than both CBM and SENN individually and also has an improved concept accuracy.

3 Problem Statement

Consider a set of inputs x $\in \mathbb{R}^d$, targets y $\in \mathbb{R}$ and concepts c $\in \mathbb{R}^k$. If x is an image, concepts can be visual properties like shape or color in the image.

We first build on the CBM to improve its interpretability and concept faithfulness. Each input in the training set is annotated with the concepts and labels, $(x_n,y_n,c_n)_{n=1}^{n=N}$, where N is the number of training samples. The model attempts to learn the mappings $g : \mathbb{R}^d \rightarrow \mathbb{R}^k$, i.e. from input images to a subset of interpretable concepts, and $f : \mathbb{R}^k \rightarrow \mathbb{R}$, i.e. from concepts to the target labels. Target predictions take the form of $f(g(x))$. Concept accuracy is the accuracy with which $g(x)$ predicts concepts and classification accuracy is the accuracy of $f(g(x))$ with the labels. The CBM constructs an interpretable deep neural network which maximizes classification accuracy and concept accuracy by restricting an intermediate layer to predict the presence of concepts as binary labels.

We experiment with two modifications to the vanilla CBM. We aim to increase the interpretability of f, for which we employ interpretable models like decision trees rather than neural layers. We also aim to increase the faithfulness of the concepts learned by g using concept-specific channels.

We further aim to reduce the amount of annotated data required for the CBM by using small auxiliary concept datasets. Here a training set

$(x_n, y_n)_{n=1}^{n=N}$ is augmented with k small datasets of N_k image patch examples for each concept, where $N_k << N$. These datasets are used to learn embeddings for each concept $\{z_c\}$ by modifying g to optimise clustering objectives in the latent space. For an input image x_i, g outputs similarity scores to every concept by comparing x_i with the learned concept embeddings z_c. These similarity scores are passed to f to make class predictions y_i.

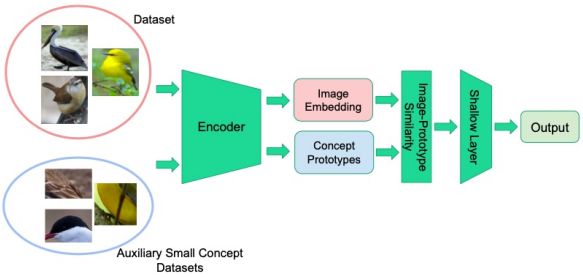

Figure 2: Prototype Bottleneck Model architecture

4 Methods

4.1 CBM with Interpretable Concepts to Labels Model

Traditional interpretable models can operate on tabular data, but image data has higher complexity and dimensionality. The nature of the CBM allows us to treat the bottleneck concepts as tabular data. These can be used as input to interpretable models like decision trees, deep neural decision trees (Yang et al., 2018) and Neural additive models(Agarwal et al., 2020).

Decision trees are a commonly used traditional machine learning technique which have the advantages of interpretation. Deep neural decision tree is a model which constructs a learnable function to decide how to split on features. The number of splits and the threshold of the split can be learnt with Gradient Descent. It can also split one node to multiple children which yields more flexibility than traditional decision trees.

The Neural Additive model (Agarwal et al., 2020) gives each feature a separate small neural network and does prediction individually. All prediction results are then added together to get the final predictions. Since all features are processed in their own network, we can easily interpret their influence on different prediction results.

We aim to improve the interpretability of the CBM by replacing the c-to-y model (f) with a vanilla Decision tree, a Deep Neural Decision Tree, and a Neural Additive model. As in the original CBM model, there are three training protocols we can use: *independent* trains the c-to-y model on the ground truth concepts (independent from the x-to-c model g); *sequential* trains the c-to-y model on the outputs of the already-trained x-to-c model; and *joint* trains the x-to-c and c-to-y models simultaneously, end-to-end. Note that because traditional decision trees are not differentiable, they cannot be evaluated with the joint training protocol.

4.2 Concept-Channels Bottleneck Model

To improve the faithfulness of concepts learned by the CBM, our model needs to focus on concept related areas (Margeloiu et al., 2021). We hypothesize that the CBM failed to achieve this because the fully connected layer for concept prediction looks at the correlation of all pixels in the picture. We additionally hypothesize that the parameters of the weights and bias corresponding to one node are not enough to encode information of a complete concept. Inspired by the structure of class activation mapping (Zhou et al., 2015), we proposed adding a new Concept-Channel layer to the CBM.

The concept layer in the CBM is a fully connected layer with number of nodes restricted to be the number of concepts. In our Concept-Channel layer, we construct a convolution layer to generate the same number of channels as the number of concepts. Each output channel is average-pooled to give one value for the score of a concept. The convolution layer retains spatial information, forcing the concept layer to learn more spatial patterns. At the same time, more parameters are learned in the convolution filters which may increase the information capacity of each concept node.

Since the only difference between the CBM and the CCBM is the x-to-c model, we only train the model to predict concepts from the input images. Then, as in (Margeloiu et al., 2021), we use saliency maps to find which part of the image the model is attending to. We also use Activation Maximization to maximize the concept filter to visualize what the filter is looking at.

4.3 Prototype Bottleneck Model

We modified the ProtoPNet (Chen et al., 2019) to learn concepts in a supervised manner like the CBM but with lesser annotated data. The ProtoPNet learns embeddings as model parameters which are optimised to minimize the clustering objectives.

We use smaller auxiliary concept datasets with image patches of concepts to provide supervision for concept learning. Figure 2 illustrates the pipeline of our model. We obtain concept embeddings by encoding the concept patches with the same encoder as the image encoder. We then optimise the clustering and separation losses described in (Chen et al., 2019) with these concept embeddings.

The learned embeddings are used to calculate similarity scores with input image embeddings obtained from the image encoder to determine whether a concept is present or absent. To train the model when calculating similarity, we assume that a class-to-concept mapping is given, i.e., we know which concepts are generally associated with a class but we don't have annotations for concepts in each image of that class. This assumption is similar to the assumption made in CBM data preprocessing when de-noising the training data. Furthermore, the ProtoPNet also assumes that each learned concept is associated with a specific class which makes our assumption permissible in the ProtoPNet architecture.

Thus, we use distances in the embedding space between image embeddings and concept patch embeddings to calculate similarity scores for each concept. These similarity scores are passed to a single dense layer to make class predictions, just like the CBM. Since the bottleneck in this model is characterised by similarity to prototypes, we call this model the Prototype Bottleneck Model (PBM).

5 Experiments

5.1 Dataset

We perform bird classification on the Caltech-UCSD Birds-200-2011 (Wah et al., 2011) dataset which comprises 11,788 bird images. The dataset has labels for 200 birds categories and 312 labelled attributes for each bird. These attributes are treated as concepts. The attributes describe different bird parts, for example the beak shape or the wing color.

For experiments with the CBM with interpretable c to y model and the Concept-Channels Bottleneck Model, we follow the data preprocessing steps described by (Koh et al., 2020) which include augmentation of input images as well as de-noising and filtering of the labeled concepts. Concepts are denoised by majority voting so that all instances of a class are labeled with the same concepts, and concepts that apply to <10 classes are filtered out, leaving us with 112 concepts. We use the same train, validation, and test sets as (Koh et al., 2020) which include 4,796, 1,198, and 5,794 bird images respectively.

For the PBM, we use the train and test sets created for the ProtoPNet experiments (Chen et al., 2019). These sets each contain approximately 30 images for each bird class. The train set is further augmented with translation, rotation and shear image augmentations to get 30 images for each original image. We use 112 sets of image patches, each containing about 10-20 examples of concepts. The patches were resized to one-fourth the size of train images. We leverage the pre-processed concept annotations used in the CBM experiments to create these. For each concept, we sample one image from every class which exhibits that concept. We then pick 10-20 of the sampled images which have the concept clearly present and crop them to fit around the concept.

Notably, the per-image concept annotations are only used to aid us in creating these concept patches and are not utilized directly in this experiment. This concept annotation process was completed by two non-expert team members in about 1 week, which we believe is a significant reduction in annotation labor as compared to that required for annotating concepts in the CUB dataset.

5.2 Metrics

We evaluate our methods based on their class accuracy. Accuracy of the predicted concepts will also be used to evaluate the CCBM as this method directly modifies how the concepts are predicted. For CCBM we also used saliency maps (Morch et al., 1995) and activation maximization (Erhan et al., 2009) to visualize the results.

5.3 Baselines

We compare our methods against three baselines: (1, 2) the standard Concept Bottleneck Model with independent and sequential training protocols, and (3) the Prototypical Parts Network trained with automatically generated concept patches.

5.4 Implementation Details

CBM baseline We implement the independent and sequential training protocols. We use the tuned hyperparameter settings from the CBM paper: batch size of 64; SGD with momentum of .9 for the optimizer, using a weight decay of 5e-5 for *independent* and 4e-5 for *sequential*; cross-entropy loss; learning rates of .01 for the x-to-c model and .001 for

the c-to-y model; and a sigmoid activation of the concept logits before being fed to the c-to-y model when performing inference with the *independent* model.

ProtoPNet baseline We train the ProtoPNet with an augmented dataset which uses 10 times the number of images. We use 10 prototypes per output so that each class has a high probability of having a prototype associated with it. We tuned the hyperparameters to have a clustering coefficient of 0.8, separation coefficient of -.08, and training batch size of 124. Prototypes are projectected every 20 epochs along with a convex optimization of the last layer.

CBM with Decision Tree The c-to-y neural model is replaced with a vanilla decision tree that uses Gini Impurity as the criterion for splitting and is not depth-limited. We found that the learned decision tree splits are more intepretable if the inputs it is trained on are binarized. Therefore, the output logits of the x-to-c model are passed through a sigmoid and binarized before being given as input to the decision tree during training of the sequential model and inference of both the independent and sequential models. We use the same settings as the CBM baseline for all other hyperparameters.

CBM with Differentiable interpretable Models The DNDT is trained with ground truth concepts as input. We assume that the tree is split 5 times into at most 3 parts. Thus, there are at most 249 leaves, which can handle the 200 different classes. The model is trained for 1000 epochs with the Adam optimizer with learning rate 0.01 and a cross entropy loss. The training parameters are the cutting points of each node. The class label is converted to one-hot encoder to fit the tree prediction structure.

For the Neural Additive model, each feature net has three hidden layers (64, 64, 32 units) with ReLU activation. The prediction results of the different feature networks were summed together to generate the final predictions.

Concept-Channel Bottleneck Model The Concept-Channel Bottleneck Model uses the same training settings as the CBM baseline. For the concept filters included before the concept layer, we use a kernel size of 3 and average pooling to generate concept scores. The activation maximization is achieved by using flashtorch package in python.

Prototype Bottleneck Model We implement the PBM with reference to the ProtoPNet implementation[1]. We use a VGG-19 base with two additional convolutional layers and a sigmoid activation. This encodes images to a 128-channel output and the patches to 128-dimensional vectors. The prototype embeddings are re-calculated every iteration while training. A logarithmic function given by -

$$f(x) = log(\frac{x+1}{x}) \tag{1}$$

converts distances to similarity. The similarity scores for all prototypes are passed to the final layer, which is a linear layer with no bias term.

The CUB dataset provides a class-to-label mapping stating the percentage of images of each class positive for a concept. We binarize this by considering a value greater than 50% as 1 and others as 0. This assumption is similar to the preprocessing done in (Koh et al., 2020). This mapping is used as the prototype-to-class mapping required for training our model.

We experiment with different hyperparameters to improve performance. The hyperparameters include the coefficients for the different loss terms and the frequency of updating the last layer weights. The hyperparameters that give us the best performance are given in table 1. We trained the model on the USC CARC facility with NVIDIA Tesla V100 GPUs. One epoch (where all layers were trained) took approximately 80-90 minutes. Our source code can be found on GitHub[2].

Cross-entropy loss coefficient	1
Clustering loss cefficient	0.0008
Separation loss coefficient	-0.008
L1 regularisation λ for last layer	0.0001
Last layer is trained every x epochs	2
Last layer is trained for x epochs	4

Table 1: PBM hyperparameters that give us the best performance

6 Results

6.1 CBM with Interpretable Concepts to Labels Model

Performance As can be see in table 2, the class accuracy of the CBM + Decision Tree model is lower than that of the standard CBM for both the independent and sequential training strategies. This said,

[1] https://github.com/cfchen-duke/ProtoPNet

[2] https://github.com/advaitrane/AuxProtoPNet

Model	Class Accuracy
CBM (Independent)	72.3%
CBM (Sequential)	76.0%
ProtoPNet	71.8%
CBM + DT (Independent)	65.7%
CBM + DT (Sequential)	69.0%
PBM	**69.9%**

Table 2: Model classification accuracy results

Model	Concept Accuracy
CBM	97.1%
CCBM	96.7%

Table 3: Concept accuracy of the Concept-Channels Bottleneck model

both of these models have performance comparable with our baselines, and the *CBM + DT (Sequential)* model reaches 69.0% which is competitive with the *CBM (Independent)* and *ProtoPNet* baselines. We did not include results for the DNDT and Neural Additive Model in this report since the accuracy is extremely low and training does not converge.

Interpretability The main benefit of the CBM + DT model is its interpretability. Like the original CBM, this model provides predicted concepts that are relevant to the class prediction, allowing for better understanding of the model's decision. A *new* benefit of the CBM + DT model over the original CBM comes from the ability to easily inspect the learned c-to-y model via traversal of the decision tree. By doing so, we enable users of this model to directly inquire about why the predicted class was chosen over another for some image. For example, figure 3 shows an image of a Black-footed Albatross that the *CBM + DT (Sequential)* model incorrectly predicts as a Northern Fulmar. If a bird watcher was using this model and wondered why we did not predict "Black-footed Albatross" for this bird, a traversal of the learned decision tree could automatically explain that it is because we noticed white by its head and upper body, which is uncommon for the Black-footed Albatross, and we didn't notice grey on its underside, which makes sense since the underside is not visible in this image. Inquiring about this same explanation with the original CBM would require repeatedly guessing different combinations of concepts to see what causes the target class to be predicted, and it would be even more challenging to find the simplest way of modifying the concepts that would result in this change.

Figure 3: Image of a Black-footed Albatross that the CBM+DT model predicts as a Northern Fulmar. The decision tree explains that Black-fotted Albatross was not predicted because of the white crown and upperparts concepts being present, and the grey underparts concept not being present.

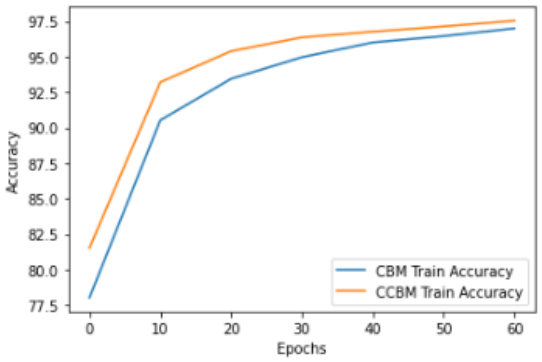

Figure 4: line graph for converge speed, y-axis represents validation accuracy and x-axis represent epochs.

6.2 Concept-Channels Bottleneck Model

Performance As can be seen in table 3, the Concept-Channels Bottleneck Model achieves similar concept accuracy as the original CBM baseline. Notably, the hyperparameters for the CCBM model were not tuned, but instead set to the same settings as the tuned CBM model. With proper tuning, we may achieve better results. An interesting finding that can be seen in figure 4 is that the convergence speed has slightly improved in CCBM model.

Faithfulness The major achievement of the CCBM is its improved faithfulness to the concepts that can be observed by visualizing the attended

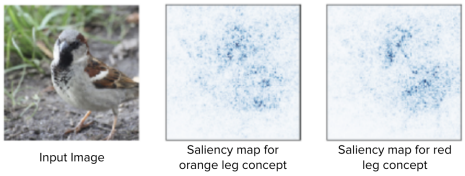

Figure 5: Saliency maps for different leg-color concepts in the original CBM. Taken from (Margeloiu et al., 2021)

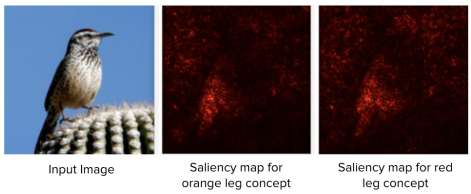

Figure 6: Saliency maps for different leg-color concepts in CCBM

areas in the input image. The CBM fails to attend to the correct areas of the input images when predicting concepts.

One possible reason is the fully connected layer learns relationships of all pixels in the image and captures some hidden correlations such as leg color and feather color, which cause the model to attend to all area of birds instead of only legs. Another possibility is that every concept has only one dedicated node which doesn't have enough capacity to disentangle a concept. As can be seen in figure 5 from (Margeloiu et al., 2021) in appendix, the CBM does not attend to the region of the image that includes the legs when predicting leg color related concepts. Comparing this with saliency maps for the same concepts in the CCBM, seen in figure 6, the CCBM focuses on lower body regions instead which contains the legs. We also have the visualization of activation maximization in appendix.

6.3 Prototype Bottleneck Model

Performance The PBM gives a competitive classification accuracy when compared with our baseline experiments. Table 2 shows that PBM has a class accuracy of 69.9%, ProtoPNet with the same model architecture has 71.8%, and the CBM has 76.0%. The PBM may even surpass the ProtoPNet with further hyperparameter optimization. Although the performance is lower than the CBM, it must be noted that the CBM has as many annotations per concept as the number of images in the full dataset whereas we have only 10-20 sample patches per concept. Thus, we achieve a comparable performance with much less annotated data.

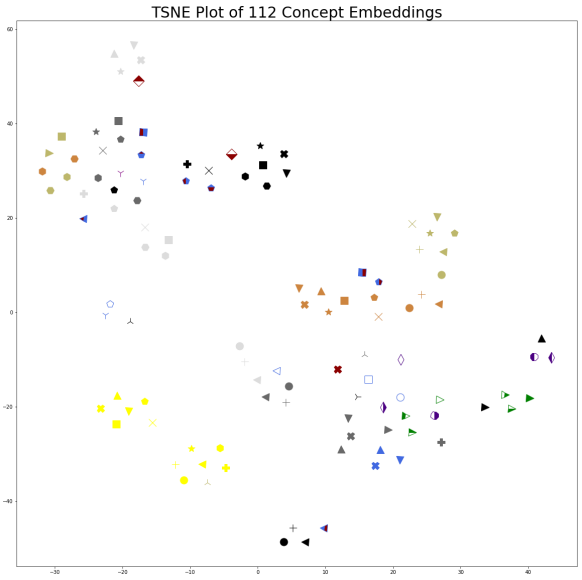

Figure 7: TSNE plot of the 112 concept prototype embeddings learned by the PBM

Symbol	Part	Symbol	Part	Symbol	Part	Symbol	Part	Symbol	Part
■	Back	Y	Tail	⬟	Wing	⬟	Multi-color	▷	All-purpose
●	Belly	⅄	Leg	⬢	Underta il	⬟	Striped	▷	Shorter than head
▶	Bill	⊱	Eye	⬢	Upperta il	⬠	Solid	▷	Same as head
◀	Breast	+	Underp arts	★	Primary color	⬟	Pointed	▷	Cone
▲	Crown	×	Upperp arts	◆	Plain	⬟	Round	▷	Dagger
▼	Nape	◆	Head	◆	Capped	◊	Medium	▶	Hooked seabird
✖	Forehea d	◆	Size	◐	Duck	◊	Small		
✚	Throat	●	Shape	◑	Perchin g	◊	Very small		

Figure 8: Legend for the TSNE plot

Learned Prototype Embeddings Figure 7 shows a TSNE plot of the 112 prototype embeddings learned by the PBM. This plot indicates that the latent space learned by the PBM is meaningful in terms of identifying different concepts and encoding their similarities and co-occurrence. For example, all the yellow points represent concepts relating to the color yellow (e.g. wing color yellow, belly color yellow). These points are all clustered together near the bottom left. The brown and buff points, which represent similar colors, are also clustered in the right of the image. A similar observation can be made for other colors. When the color is not the most significant factor in a concept, the embeddings for the same part are clustered together. This can be seen in the green rightward triangle points which represent different bill shapes. Figure

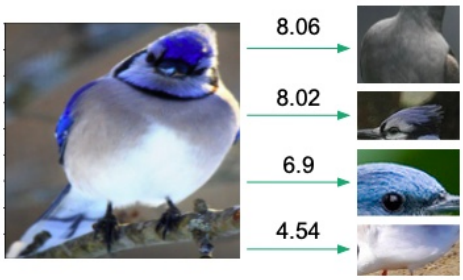

Figure 9: The bird in the input image gets highest similarity scores for the concepts grey breast, blue crown, blue forehead, and white belly.

8 provides a detailed legend for the plot.

Transparent and Faithful Concept Learning The concepts learned by the PBM are grounded in expert-chosen concepts meaningful to the task. This is an advantage over the ProtoPNet whose unsupervised mechanism can be confused by adversarial or compression noise (Hoffmann et al., 2021). The prototype similarity mechanism provides an intuitive way to visualise input image parts with respect to concept patches rather than binary labels given by the CBM. Visualising the input image and concept patches helps explain the similarity scores and interpret model decisions. A sample visualisation is shown in figure 9, which makes it obvious why the input image correctly gets a high similarity score for the corresponding concepts.

7 Discussion

Our results indicate that there are several avenues to make improvements to the current state of inherently interpretable models. We explore three such strategies and achieve encouraging results.

One avenue for improvement is increasing the interpretability of the concept-to-label model. Our results with the Decision Tree do provide better interpretability. However, it is not differentiable and cannot be trained end-to-end. We could not achieve good results with differentiable models due to the limitation of model structure. The DNDT does not perform well with a large number of features as it uses Kronecker product to exhaustively find all final nodes and prediction results. The Neural Additive Model can be used with a large number of features but can not deal with large number of classes. The summation structure used can only do binary classification. To our knowledge, there is no differentiable tree model that can perform as well as a decision tree when predicting among more than 100 classes. Further work in this area should aim to improve the performance of differentiable interpretable models to classify among a large number of classes.

We do not attempt to increase the interpretability of the input-to-concept model. Since this part is typically a large pretrained model, increasing the interpretability in this part is also important. The bottleneck only provides interpretability at one layer, whereas the ultimate goal for these models should be to make the entire model interpretable.

Our results with the CCBM highlight that a single neuron may be insufficient to encode a concept. More complex, composite, class-specific architectures can lead to better faithfulness to concepts. The critique about the CBM's attended area of has been mitigated by our CCBM as we observed from saliency maps. The activation maximization results are not human identifiable, which may be caused by the complexity of birds in CUB dataset. We may get better results with other methods, like Grad-CAM. More experiments can be done to prove the efficiency of the model. Thus, designing alternate architectures which encode useful inductive biases for this task is a fruitful area of research to improve model performance.

Lastly, the PBM results indicate that CBMs are highly data-inefficient and comparable performance can be achieved with much fewer labeled samples. Combining few-shot learning techniques with inherently interpretable models could further improve data-efficiency. Our method also makes it simple to collate auxiliary patch datasets. This model may be applicable to a host of tasks which require interpretable models with expert supervision in critical use cases. For example, the PBM can be used to detect ailments in medical imaging data where experts could provide small auxiliary datasets of the relevant concepts like tumors or inflammation.

8 Conclusion

Our results indicate that our strategies are successful in improving the interpretability, data-efficiency, and concept faithfulness of the CBM. We combine ideas from different interpretable models like the CBM, the ProtoPNet, and Decision Trees. The PBM has significant benefits over both the ProtoPNet and the CBM. Our project thus effectively contributes to research on inherently interpretable neural models.

References

Rishabh Agarwal, Nicholas Frosst, Xuezhou Zhang, Rich Caruana, and Geoffrey E. Hinton. 2020. Neural additive models: Interpretable machine learning with neural nets. *CoRR*, abs/2004.13912.

David Alvarez-Melis and T. Jaakkola. 2018. Towards robust interpretability with self-explaining neural networks. In *NeurIPS*.

Daniel Ho Jihan Yin1 Scott Lee1 Suzanne Petryk Sarah Adel Bargal Joseph E. Gonzalez Alvin Wan, Lisa Dunlap. 2021. Nbdt: Neural-backed decision tree. In *ICLR*.

Catarina Bel'em, Vladimir Balayan, Pedro Saleiro, and P. Bizarro. 2021. Weakly supervised multi-task learning for concept-based explainability. *ArXiv*, abs/2104.12459.

Chaofan Chen, Oscar Li, Alina Barnett, Jonathan Su, and Cynthia Rudin. 2019. This looks like that: deep learning for interpretable image recognition. In *NeurIPS*.

D. Erhan, Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2009. Visualizing higher-layer features of a deep network.

Adrian Hoffmann, Claudio Fanconi, Rahul Rade, and Jonas Kohler. 2021. This looks like that... does it? shortcomings of latent space prototype interpretability in deep networks. *arXiv preprint arXiv:2105.02968*.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *ICML*.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. 2020. Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR.

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. 2018. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI*.

Andrei Margeloiu, Matthew Ashman, Umang Bhatt, Yanzhi Chen, Mateja Jamnik, and Adrian Weller. 2021. Do concept bottleneck models learn as intended? *arXiv preprint arXiv:2105.04289*.

N.J.S. Morch, U. Kjems, L.K. Hansen, C. Svarer, Ian Law, Benny Lautrup, Stephen Strother, and K. Rehm. 1995. Visualization of neural networks using saliency maps. pages 2085 – 2090 vol.4.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Yoshihide Sawada and Keigo Nakamura. 2022. Concept bottleneck model with additional unsupervised concepts. *ArXiv*, abs/2202.01459.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. 2011. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Yongxin Yang, Irene Garcia Morillo, and Timothy M. Hospedales. 2018. Deep neural decision trees. *CoRR*, abs/1806.06988.

Cynthia Rudin Zhi Chen, Yijie Bei. 2020. Concept whitening for interpretable image recognition. In *Nature*.

Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. 2015. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150.

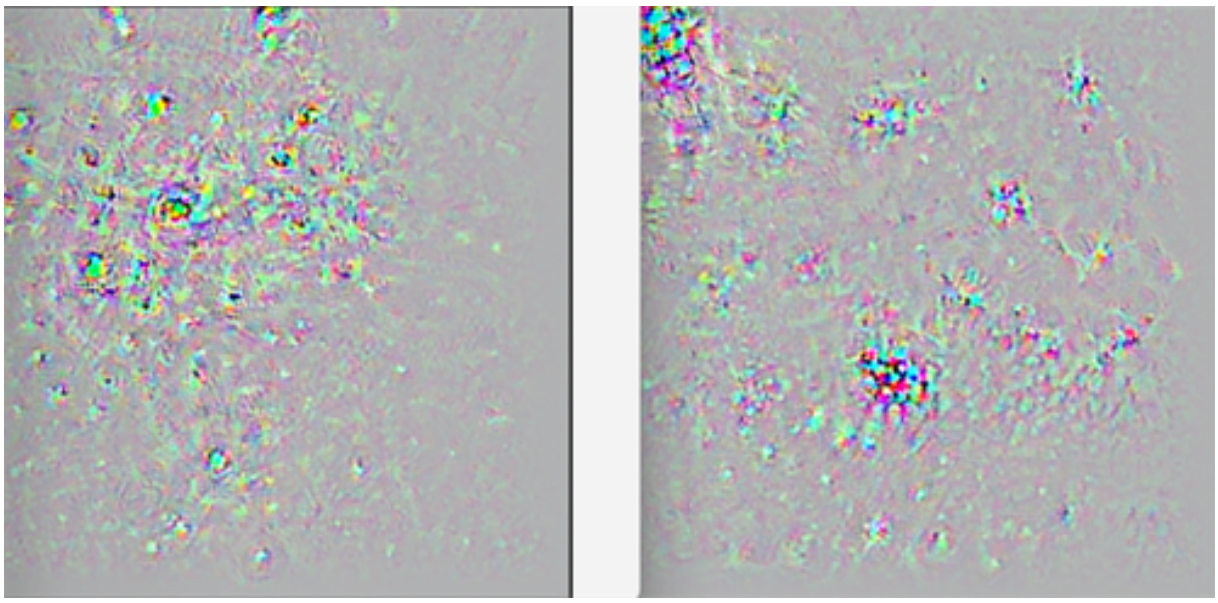

Figure 10: Left is the visualization of upper body color, right is the visualization of beak. We can tell that color block of larger body part is more concentrated. This may because the camera prefer to put main body in the middle of the camera

A Appendices

A.1 Visualization results for CCBM