ITERATIVE BINARY DECISIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

The complexity of functions a neural network approximates make it hard to explain what the classification decision is based on. In this work, we present a framework that exposes more information about this decision-making process. Instead of producing a classification in a single step, our model iteratively makes binary sub-decisions which, when combined as a whole, ultimately produces the same classification result while revealing a decision tree as thought process. While there is generally a trade-off between interpretability and accuracy, the insights our model generates come at a negligible loss in accuracy. The decision tree resulting from the sequence of binary decisions of our model reveal a hierarchical clustering of the data and can be used as learned attributes in zero-shot learning.

1 INTRODUCTION

A classification decision made by a convolutional neural network (CNN) such as ResNet (He et al., 2016) is hard to interpret on its own. However, to spread adoption and create widespread acceptance of neural networks, it is important that the predictions of a neural network are explainable. An explanation can help an end-user to establish trust or help a machine learning practitioner to understand and debug deep models. In general, models which are easier to interpret, e.g. simple linear models, come at the cost of prediction performance. There is, however, growing work on explainability methods that do not interfere or have minimal impact on the training and inference procedures of the original network. Such methods include the visualization of features (Springenberg et al., 2014; Zhou et al., 2016; Selvaraju et al., 2017a), the creation of saliecy maps (Simonyan et al., 2013) to show attribution of input regions of an image and using an invertible mapping between the latent space of a generative model and interpretable features (Adel et al., 2018).

In this work, we present a framework that reveals more explicitly a decision structure in the form of a decision tree, by breaking down the single decision of the network into many smaller decisions without sacrificing prediction accuracy. Our model learns to figure out the most important bit of information at a time to communicate to a second agent that can, by combining all single bits of information, solve highly non-linear tasks such as determining the class of an image. Figure 1 illustrates the two-agent setup of this work. The classifier is tasked to solve a problem but starts with no information about the data and the observer has exclusive access to the data without necessarily having knowledge about the task at hand. Hence, our framework can deal with any type of data (e.g., images, text, videos) and tasks, for this work, however, we focus on image classification. After receiving a query message from the classifier, the observer is allowed to only communicate one bit of information about the data at a time to the classifier. This is equivalent to the classifier first formulating a questing and then the observer answering this question with yes or no. With each new bit of information that the classifier receives from the observer it learns more about the data at hand and, bit by bit, comes closer to its objective.

Through this setup we construct a binary decision tree though process, that we can use to understand which decisions lead to the final prediction of the network and, thus, make the inference process more transparent.

Our contributions are as follows. 1) We propose a two-agent model framework that learns to make iterative binary decision to solve a task (Section 2) maintaining classification accuracy compared to state-of-the-art CNNs (Section 3.2); 2) We showcase how the decision tree can provide additional information about the decision process (Section 3.3); 3) We propose a variant of the model (Section 2.1) that takes advantage of the structure of the decision tree to extract learned attributes that, when used for zero-shot learning, outperform features generated from Word2Vec (Section 3.4).

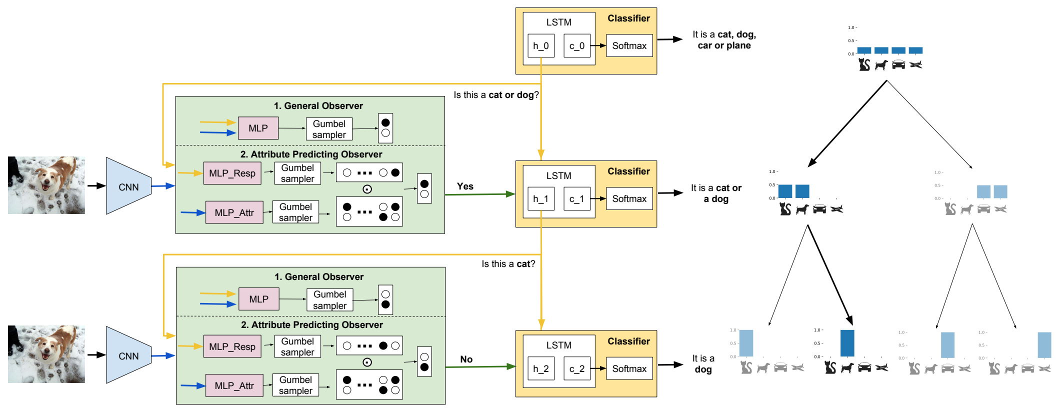

Figure 1: Our observer-classifier framework. This illustration shows an example task of classifying an image into one of four categories (cat, dog, man, woman). After extracting image features, the observer combines the image features with the query message of the classifier at each time step, feeds them to an MLP to create a binary response. The classifier consists of an LSTM that uses its hidden state as a query message. At every step, the state of the LSTM is updated with the binary response such that classification accuracy is improved. In this example task, two binary messages are needed to traverse the binary decision tree and solve the task.

2 Observer-classifier framework

The framework is set up as a sequential interaction between two agents with a goal to solve a supervised learning task through communication. We demonstrate the capabilities of this framework on the example of image classification. The first agent, referred to as the observer, holds information about the image $x \in \mathbb{R}^D$ to be classified, either by having access to the actual image or by provided pre-extracted image features $z \in \mathbb{R}^Z$. The second agent, the classifier, is tasked to predict the associated ground-truth class $y \in \mathcal{Y}$ without having direct access to any data about the image apart from the messages that the observer broadcasts.

The classification task of a single image is constructed as follows. The classifier initializes its initial state corresponding to its prior belief of the class distribution $\hat{y}_0 \in \mathbb{R}^C$. It then creates a query message $h_t \in \mathbb{R}^H$ in latent space sent to the observer requesting information about the image. The observer processes the query together with the input image to construct a binary response $d_t \in \{0, 1\}$. The classifier uses this one bit of information to update its state such that the classification error is reduced. This concludes one iteration of the observer-classifier communication. The interaction repeats until the maximum number of steps or until the classifier is confident in making a correct classification.

We deliberately limit the observer's messages to be binary in order to study the iterative binary decision process that emerges. Classification loss is minimized when two agents jointly learn to communicate the most important bit of information about the image at each time step that improves the classifier's belief over the class distribution the best.

2.1 Observer: Binary decisions

In this section, we present two variations of our observer model. We refer to the first variant as the General Observer (see Fig. 1). In this setting, the observer consists of a CNN that provides image features and a multilayer perceptron (MLP) that produces the binary decision d_t with $MLP_g : \mathbb{R}^{Z+H} \to \{0, 1\}$. Input to the MLP is the concatenation of both the image features and the latent query message from the classifier $[z, h_t]$. The binary output is computed using the Gumbel-softmax estimator (Jang et al., 2017; Maddison et al., 2017), which allows sampling from a discrete categorical distribution by first sampling from a Gumbel distribution $g_i = -\log -(\log(u_i)$ with $u_i \sim Uniform(0, 1)$ coming from a uniform distribution. Subsequently, we can calculate a continuous relaxation of the categorical distribution

$$d_i = \frac{\exp((\log o_i + g_i)/\tau)}{\sum_{j=1}^{K} \exp((\log o_j + g_j)/\tau)} \tag{1}$$

where $\log o$ is the unnormalized output of our MLP and τ is the temperature that parameterizes the discrete approximation. When τ approaches 0, the output becomes a one-hot vector (binary when $K = 2$). If τ is too big, the signal becomes continuous and breaks the restriction of the binary response d_t. To overcome this issue, we augment the Gumbel-softmax with an $\arg\max$ function that discretizes the activation in the forward pass and a straight-through identity function in the backward pass. Doing so guarantees that the response from the observer is always binary.

We refer to the second variant of our observer as the Attribute Predicting Observer (see Fig 1). The CNN used for retrieving image features is extended with an *Attribute MLP, i.e. MLP_Attr* that outputs a set of binary attributes $a \in \{0, 1\}^A$ with the dimensionality A chosen in advance. Each of these attributes uses the Gumbel-softmax estimator to sample and discretize them as binary features. Once the binary attribute vector is obtained for an image, a *Response MLP, i.e. MLP_Resp* takes the query message as input and produces a probability distribution $\pi \in \mathbb{R}^A$ over which of the binary attributes should be chosen as a response to the classifier. The choice distribution over the attributes is again discretized using the Gumbel-softmax estimator resulting in a one-hot vector of size A. The final binary decision $d_t = a \odot \pi_{onehot}$ is computed by selecting the position encoded by π_{onehot} from a denoted by $\odot$. Since the attribute-based observer is limited to a fixed number of binary attributes to choose from as a response, its performance depends upon the hyperparameter A.

The General Observer is more flexible in constructing a binary response conditioned on both image features and query message and does not need the additional hyperparameter. The predefined number of binary attributes features in the Attribute Predicting Observer can, however, be easily extracted as learned shared features for zero-shot learning tasks which are discussed in Section 3.4.

2.2 CLASSIFIER

The classifier mainly consists of an LSTM (Hochreiter & Schmidhuber, 1997). We use the hidden state $h_t \in \mathbb{R}^H$ of the LSTM as a query message for the observer. Each iteration the classifier receives the observer's binary decision together with the previous query message $[d_t, h_{t-1}]$ to update its state and, thus, produce the following message h_t. The LSTM's cell state c_t, which corresponds to h_t before applying the non-linearity and output gate mask, is used to get the class prediction using an affine transformation: $\hat{y}_t = \mathrm{softmax}(Wc_t + b)$ with $W \in \mathbb{R}^{C \times H}, b \in \mathbb{R}^C$.

Since the primary objective of the classifier is to maximize the classification performance, we want to minimize the cross-entropy loss of the predicted class probabilities $\hat{y}_t$ and the true class probabilities y. Whereas decision trees usually place a classification loss on the leaf nodes, we place the cross-entropy loss also after every binary decision. This results in the loss function:

$$\mathcal{L} = -\frac{1}{T} \sum_{t=1}^{T} \sum_{i} y_i \log \hat{y}_{t,i} \tag{2}$$

where we take the cross-entropy loss on the current classification distribution after each iteration of the observer-classifier communication loop. This encourages the network to predict the correct class in as few communication steps as possible.

3 EXPERIMENTS

In this section, we first describe our experimental setup, i.e. datasets, then provide quantitative and qualitative results demonstrating the performance of our model. We finalize with experiments that demonstrate the attributes learned by our model in a different task, i.e. zero-shot learning.

3.1 DATASETS AND IMPLEMENTATION DETAILS

We evaluate our model on two benchmark classification datasets (MNIST, CIFAR10) and a zero-shot learning dataset (AWA2). MNIST consists of grey-scale images of the size 28×28 depicting

Model	MNIST	CIFAR10	AWA2
CNN + Softmax Classifier	99.28	93.02	96.04
CNN + Our Observer Classifier Model	99.06	92.46	95.34

Table 1: Top-1 accuracy (in %) on MNIST, CIFAR10 and AWA2 in percent. The image features are the same for both models. For training our model we use a maximum of 4, 6 and 28 binary decision for MNIST, CIFAR10 and AWA2 respectively.

handwritten digits. 10 classes are balanced across the 60K training and 10K test examples. Similarly, CIFAR10 also consists of 10 classes and is a balanced dataset with 50K training and 10K test data points. Its color images are of the size $3 \times 32 \times 32$ and show centered instances of the categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. These two datasets serve to validate whether our model can be successfully applied to the image classification tasks while maintaining accuracy as compared to CNNs.

On the other hand, Animals with Attributes 2 (AWA2) provides high-resolution color images but has an imbalanced class distribution. It contains $37,322$ images from 50 classes. AWA2 does not come with a predefined train-test split. Thus, we randomly assign 20% of each class as a test data for the image classification. In all the experiments across the datasets, we randomly separate 10% of the training data as a validation set.

In AWA2, every class is annotated with 85 attributes encoding the relevance of an attribute with a class. The attribute quality directly effects the zero-shot learning performance. Hence, evaluating our model on this task shows the effectiveness of our Attribute Predicting Controller model in attribute prediction. We use the proposed split and the image features from Xian et al. (2017) to do the zero-shot learning experiments. We compare the predicted attributes by our model with Word2Vec feature vectors (Mikolov et al., 2013) extracted from Wikipedia articles (Akata et al., 2015).

The image features are extracted once per image and then accessed by the observer at each iteration. The CNN can be either jointly trained with the observer-classifier framework or use a pretrained model. The MLPs consist of two layers with a ReLU non-linearity in between. It is beneficial to learn the temperature hyperparameter τ of the Gumbel-softmax estimator jointly with the network.

3.2 Comparing Classification Performance With The Baseline

In this section, we compare the classifier of our Observer Classifier model with the baseline softmax classifier. Note that, we use the same CNN architecture for the baseline model and our observer. On MNIST, we use two convolutional layers followed by ReLU and Max-Pooling and finally a fully connected layer. ResNet18 for CIFAR10 and ResNet152 for AWA2 are finetuned after initialization with pretrained weights learned on ImageNet.

Table 1 shows that our model reaches the same classification accuracy across datasets as in softmax. Since our model uses the same image features and relies on a discrete binary decision, i.e. the binary message being the only means of communication between the observer and the classifier, it is expected that we do not surpass the state of the art performance. Indeed these results (99.06% vs 99.28% on MNIST, 92.46% vs 93.02% on CIFAR10 and 95.34% vs 96.04% on AWA2 constitute the state of the art on these datasets) demonstrate that our model can maintain the same performance while being more transparent than a softmax classifier as it structures the decision-making process as a binary decision tree.

During training we fix the maximum number of communication loops between observer and classifier. We further analyze how this affects classification results in Figure 2. The minimum number of binary decisions depends on the number of classes that have to be discriminated. On MNIST and CIFAR10 we start with four binary decisions. While on MNIST, our model can consistently achieve a high accuracy, CIFAR10 benefits from 2 extra decisions. Analogously, the model can trained on AWA2 with six binary decisions, but achieves the best results when trained on 28 decisions.

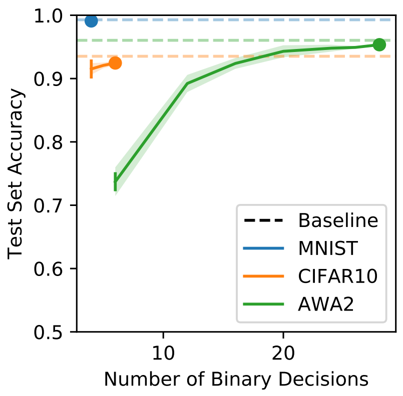

Figure 2: Test-set accuracy of proposed model trained with increasing number of maximum number of binary decisions in the communication loop between observer and classifier.

Side information	Dimensionality	Supervision	SJE
Attributes	85	Human	62.18%
Attribute Predicting Observer	40	None	46.32%
Word2Vec	400	None	38.78%

Table 2: Zero-shot learning accuracy on AWA2. We compare the accuracy of different side information, i.e. human annotated attributes and learned attributes (with our model or with Word2Vec). We also compare the dimensionality of the features.

3.3 Introspective Explanations: Visualizing Decisions

The main advantage of our model is that it is fully transparent. In other words, our decision tree based decision maker reveals its decision process by pointing to which tree branch a certain image would fall. We inspect the learned structure of the decision tree by illustrating a tree up to depth four from our model on CIFAR10 and AWA in Figure 3 and Figure 4 respectively.

Since CIFAR10 has only 10 classes, we can easily visualize the complete tree and the clustering that the model learned to discriminate between the classes. On each edge we show the highest predicted probabilities of classes after taking that particular decision. Interestingly, we observe that the first decision of the tree seperates vehicles (plane, car, ship, truck) from animals (bird, cat, deer, dog, frog, horse). On CIFAR10, we get strict separations between classes. For instance, after the second decision, among the vehicles "plane" gets separated into one tree branch while other vehicles that are all road vehicles, i.e. "car, ship, truck" get clustered into one. Similarly, "frog" and "bird" being animals with the most distinct features get separated from highly confusing classes early on in the decision tree.

On AWA2, even though the network cannot yet reliably predict the correct class after four decisions, because there is a total of 50 classes, we can still observe a reasonable separation of classes: we find a group of small mammals (rabbit, hamster, squirrel, weasel, fox, rat and mouse); a grouping of wolf, collie, sheep, and german shepherd; another consisting of raccoon, otter and beaver; and seal and walrus forming a group, to name a few. These clusters reveal an expectation for the individual decisions of our model.

Another property of explanations is that they are contrastive. In addition to justifying a positive decision, our model can reason about negative decisions. When images are misclassified, we can inspect at which point in the tree the error occurred exposing a more fine-grained information of when features are mistaken to be from another class or group of classes.

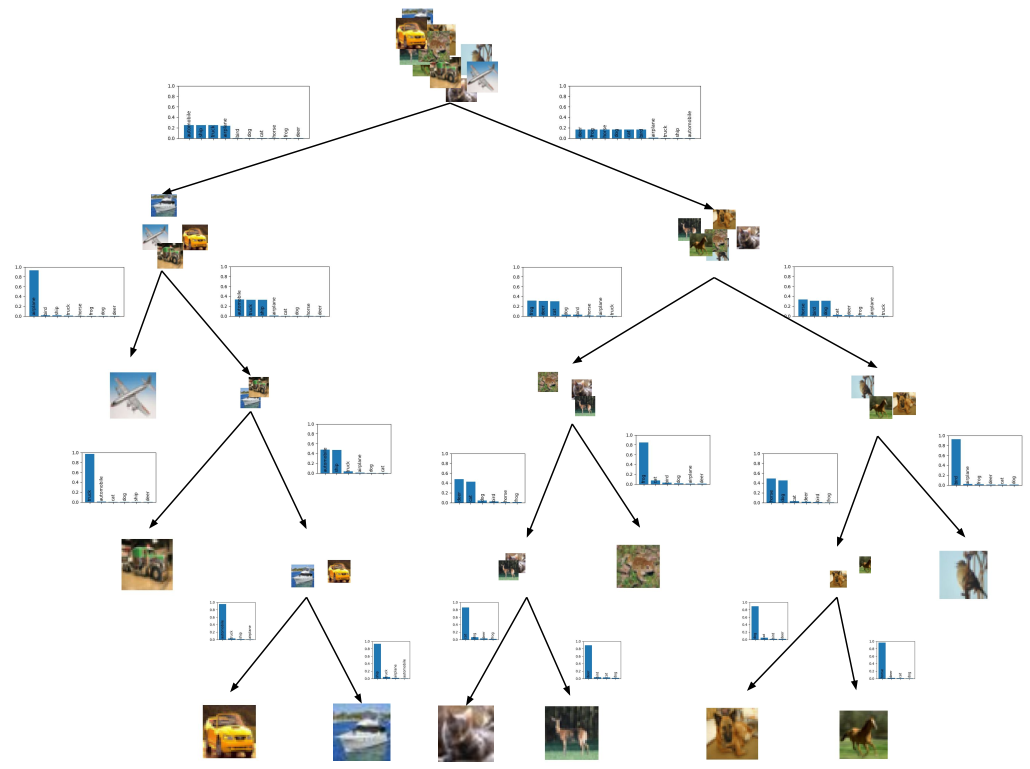

Figure 3: Decision tree of the observer-classifier framework on CIFAR10. Images of classes indicate where the majority of data example are routed to. Each edge shows the predicted probability distribution over classes after that decision has been made.

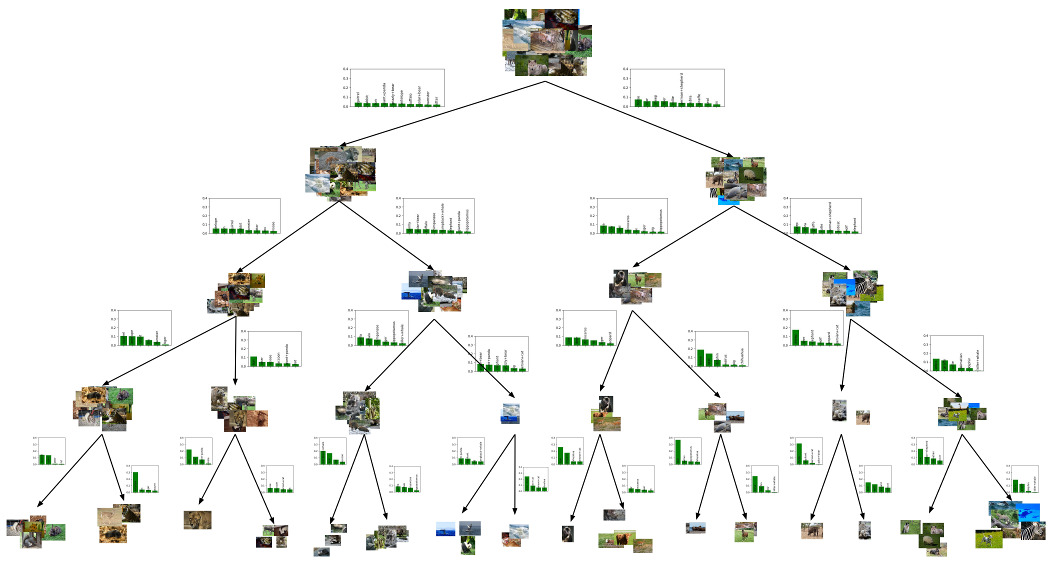

Figure 4: Learned decision tree on AWA2. We show the first four decisions of the most likely path for each class. Each edge shows the predicted class distribution after the decision was made.

3.4 Transfering Representations to Other Tasks: Zero-Shot Learning

In Lombrozo (2012) it is argued that explanations are useful when they enable solving an independent task. As an independent task we choose zero-shot learning as solving this task requires interpretable representations to be used as side information.

In zero-shot learning, the classes used during test time is disjoint from the classes that are used at training time. In order to reason about which unseen class a query image belongs to, side information that associate different classes is necessary for information transfer. The most widely used side-information is attributes and they are often human annotated. Alternatively to human annotation, attributes can be learned using our Attribute Predicting Observer model. Since our network resembles a binary decision tree, classes share features in the form of the binary decision of mutual parent nodes. This enforced sharing of attributes is crucial in the zero-shot learning setting. Additionally, our Attribute Predicting Observer allows to define a fixed number of features which do not grow with the depth of the tree. We obtain the probabilities of each learned binary attribute via softmax. The per class attributes are then the attribute vector of every image belonging to a particular class scaled to lie between -1 and 1. We compare against Word2Vec (Mikolov et al., 2013) learned on Wikipedia articles which is an alternative widely used side-information for zero-shot learning.

Our zero-shot learning experiments follow the recommendations of Xian et al. (2017), i.e. we use the same image features which were obtained from a ResNet101 model trained on ImageNet and the proposed train-test split ensures that test classes do not co-occur in ImageNet. As the zero-shot learning model, we use Structured Joint Embedding (SJE) (Akata et al., 2015), i.e. the best zero-shot learning performance on AWA2 according to Xian et al. (2017). SJE uses a ranking-based objective with a bi-linear compatibility function

$$F(x, y, W) = \theta(x)^T W \phi(y) \tag{3}$$

to relate image features $\theta(x)$ with class embeddings $\phi(y)$, both of which are given in our case. W is the weight matrix that is learned minimizing the loss

$$\mathcal{L}_{SJE}(x_n, y_n) = \max_{y \in \mathcal{Y}^{tr}} (0, \Delta(y_n, y) + \theta(x_n)^T W[\phi(y) - \phi(y_n)]) \tag{4}$$

where $\Delta(y_n, y) = 0$ if $y_n = y$ and 1 otherwise. During test time, the test set class with highest compatibility value $F(\cdot)$ is the predicted category.

As indicated in Table 2, while human annotated attributes, i.e. upper bound, reaches 62.18% accuracy, our Attribute Predicting Observer achieves 46.32% accuracy significantly surpassing the accuracy obtained with Word2Vec, i.e. 38.78%. This result is encouraging as it demonstrates that our learned attributes lead to discriminative representations useful for tackling the challenging task of zero-shot learning. This shows that our model does not only reach the accuracy obtained with a state of the art deep network but also produces representations more interpretable than Word2Vec extracted from wikipedia articles. Another interesting observation is that while the dimensionality of Word2Vec is 400, our learned attributes reaches a significantly better accuracy of 46.32% using 10 times fewer attributes.

4 Related work

Our work is related to several different areas of neural networks research, namely the combination of deep learning and decision trees, multi-agent communication and interpretable machine learning.

Decision Trees with Neural Networks. Adaptive Neural Trees (Tanno et al., 2018) directly model the neural network as a decision tree where each node and edge correspond to one or more modules of the network. Our model is self-adapting through the use of a recurrent network in the classifier that makes a prediction at every node and can be easily rolled-out to greater depth without changing the architecture or number of weights. The prior work that is closest to ours is the Deep Neural Decision Forest (Kontschieder et al., 2016) which first uses a CNN to determine the routing probabilities on each node and then combines nodes to an ensemble of decision trees that jointly make the prediction. Similarly, in our Attribute Predicting Observer, we compute the binary decisions, i.e. router nodes of the tree, once before using them ad-hoc. Our method differs in that it explicitly only considers a hard binary decision at each node whereas the other methods use soft decisions making a large portion of the tree responsible for the predictions and, thus, harder to interpret.

Multi-Agent Communication. Learning to communicate in a multi-agent setting has recently gained interest mostly due to the emergence of deep reinforcement learning (Foerster et al., 2016; Havrylov & Titov, 2017). Most related to our work, Foerster et al. (2016) is consists of an agent that composes a message of categorical symbols at once and another agent to use the information in these messages to solve a referential game. For discrete symbols, they also rely on the Gumbel-softmax estimator, but in contrast to our model, their communication is not iterative and concludes after one message. Additionally, our model explicitly allows to inspect the prediction after each bit of information is transmitted rather than only after the whole message was receive in their model.

Explainability. The importance of explanations for an end-user has been studied from the psychological perspective (Lombrozo, 2012), showing that humans use explanations as a guide for learning and understanding by building inferences and seeking propositions or judgments that enrich their prior knowledge. They usually seek for explanations to fill the requested gap depending on prior knowledge and goal in question.

In support of this trend, recently explainability has been growing as a field in computer vision and machine learning (Hendricks et al., 2016; Park et al., 2018; Andreas et al., 2016; Zintgraf et al., 2017). Following the convention in Park et al. (2018), our focus is on introspective explanations, where a deep network as the decision maker is trained to explain its own decision which is useful in increasing trust for the end user and provides a means to detect errors the model leads to.

Textual rationalizations are explored in Hendricks et al. (2016) which proposes a loss function based on sampling and reinforcement learning that learns to generate sentences that realize a global sentence property, such as class specificity. Andreas et al. (2016) composes collections of jointly-trained neural modules into deep networks for question answering by decomposing questions into their linguistic substructures, and using these structures to dynamically instantiate modular networks with reusable components.

As for visual rationalizations, (Zintgraf et al., 2017) proposes to apply prediction difference analysis to a specific input. (Park et al., 2018) utilizes a visual attention module that justifies the predictions of deep networks for visual question answering and activity recognition. Grad-CAM (Selvaraju et al., 2017b) uses the gradients of any target concept, e.g. predicted action, flowing into the a convolutional layer to produce a localization map highlighting the important regions in the image for predicting the concept. Interpretable CNNs (Zhang et al., 2018a) modifies the convolutional layer such that each filter map corresponds to an object part in the image and follow up work (Zhang et al., 2018b) uses a classical decision tree to explain the predictions based on the learned object part filters.

5 Conclusion

In this work, we have presented a two-agent framework that is able to solve an image classification task by conveying one bit of information at a time. Our model does not compromise on classification accuracy while revealing a binary decision tree for solving the task. The tree clusters the data and allows to better understand the decisions the network makes in an iterative manner. Promising results on zero-shot learning validate that the tree indeed learns transferable binary features across classes.

References

Tameem Adel, Zoubin Ghahramani, and Adrian Weller. Discovering interpretable representations for both deep generative and discriminative models. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Zeynep Akata, Scott E. Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, 2015.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 2016*, 2016.

Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Advances in Neural Information Processing Systems 2017*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, 2016.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *ECCV*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8), 1997.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulò. Deep neural decision forests. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, 2016.

T. Lombrozo. *Explanation and abductive inference.* The Oxford handbook of thinking and reasoning, 2012.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 2013*, 2013.

Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. Multimodal explanations: Justifying decisions and pointing to the evidence. In *IEEE CVPR*, 2018.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision, ICCV 2017*, 2017a.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE ICCV*, 2017b.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.

Ryutaro Tanno, Kai Arulkumaran, Daniel C. Alexander, Antonio Criminisi, and Aditya V. Nori. Adaptive neural trees. *CoRR*, abs/1807.06699, 2018.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning - the good, the bad and the ugly. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018a.

Quanshi Zhang, Yu Yang, Ying Nian Wu, and Song-Chun Zhu. Interpreting cnns via decision trees. *CoRR*, abs/1802.00121, 2018b.

Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, 2016.

Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *ICLR*, 2017.