# Interpretable & Actionable Models using Attribute & Uncertainty Information

Yew Siang Tang, Mikaela Angelina Uy, Olivia Hsu
Stanford University
{yewsiang, mikacuy, owhsu}@stanford.edu

## Abstract

*Deep-learning models can be difficult to understand and control intuitively due to the black-box nature of these models. However, such lack of interpretability and human actionability in the models' decision processes make it difficult to trust these models in critical applications that affect the lives of people. We propose to alleviate these problems through the use of attribute and uncertainty models in deep networks. Our contributions are showing that attribute models can improve the interpretability of models in a direct and causal manner, as opposed to post-hoc explanations. The models do not compromise and sometimes even improve performance on the CUB and OAI datasets. We show how users can have better control of our models through a test-time intervention procedure where they can make contributions to intermediate model outputs (attributes) and improve overall target performance. Finally, we show the usefulness of understanding uncertainty on attributes to enable users to achieve better intervention.*

## 1. Introduction

Current deep-learning models achieve superior performance over humans in many benchmarks, but suffer from poor *interpretability* and *actionability* due to their black-box nature and the lack of human control over their decisions. We define interpretability as the ability to understand why a model arrived at its decisions and actionability as the ability to intervene when we believe it has made mistakes.

Suppose that we are performing disease classification on medical images, a standard image classifier can only provide a probability distribution over disease classes through a black-box process. Ideally, a classification model that has understood what a disease means would tell us "I (model) believe that there is an 80% chance of disease *because* there might be cysts present and that there are large deformations around this area. However, I am only 60% certain about the presence of cysts and require *assistance* on this aspect." The user will then have a better *understanding* of the model's decisions and will therefore be able to *act* on this information by telling the model what it is uncertain about.

These aspects are important because of the increasing prevalence of deep learning and machine learning in many applications. These applications include credit risk assessment, facial recognition, medical diagnosis and more, which means the outputs of these algorithms can have a profound impact on the lives of many people. If we are to trust these algorithms, we need to have a better understanding and control of them.

To achieve the desired behaviors, we make use of attribute and uncertainty models. In this work, we show that:

- attribute models can improve the interpretability of our models in a direct and causal manner as opposed to post-hoc explanations,
- attribute models do not compromise and can even improve the performance of our models on the provided datasets,
- attribute models can also be used in a test-time intervention procedure that enables humans to make contributions to intermediate model outputs and improve overall target performance, and
- uncertainty modeling of attributes enables us to understand which attributes the model is having difficulty with and provide better human intervention, allowing us to achieve better test-time intervention results.

## 2. Related Work

**Interpretability** Work in improving the interpretability of neural networks is heavily focused on generation of explanations of model decisions that are post-hoc and not true explanations. For example, [10] uses image saliency maps as a heatmap for where the model focuses on, [5] answers questions and uses a separate model to point to the evidence, and [1] tries to find evidence for explanations.

**Attribute Models** The research focus in the use of attributes has generally revolved around zero-shot classification where the works aim to generalize beyond classes within the training set by defining attributes as the representation for classes. [9] uses the training set classes to learn attribute classifiers that allow prediction on disjoint test set classes. [6] extends the expressiveness of attributes by making use of attribute correlations. Not much work has been done to explore the use and limitations of attributes in the standard supervised learning classification models due to the effectiveness of end-to-end deep learning models.

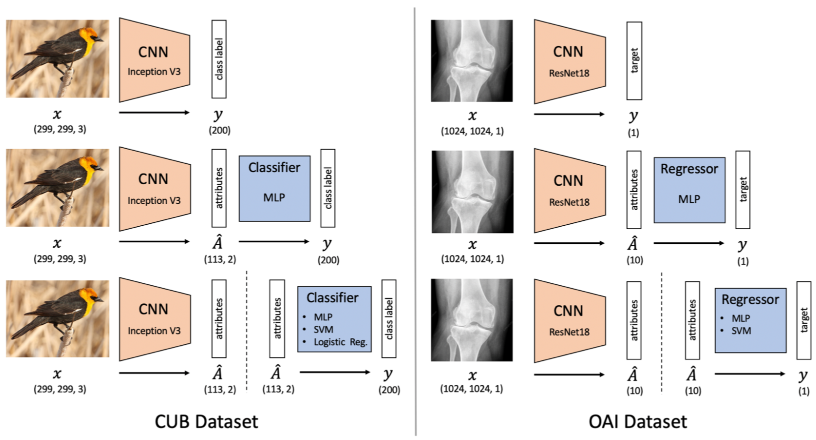

Figure 1. Model Diagram for OAI and CUB datasets

**Uncertainty Models** Bayesian Deep Learning is receiving increased research interest due to the effectiveness of deep learning and the usefulness of uncertainty estimates in downstream tasks such as Active Learning and Bayesian Optimization. [2] explores a general method for placing distributions on the weights of neural networks (NNs) instead of getting point estimates through Maximum Likelihood Estimation, thereafter allowing us to sample multiple model predictions per input data point. [7] discusses the breakdown of uncertainty into aleatoric uncertainty and epistemic uncertainty in a Computer Vision context. [3] proposes Bayesian CNNs and discusses the link between dropout and Bayesian learning in neural networks. We make use of dropout to implement Bayesian NNs that enable us to understand the uncertainty in attribute predictions.

## 3. Method

### 3.1. Problem Statement

We assume that we have a dataset $\mathcal{D} = \{(X_1, A_1, y_1), (X_2, A_2, y_2), ..., (X_n, A_n, y_n)\}$ which is composed of $n$ triplets, where $X_i \in \mathbb{R}^D$ is the $i$th input data, $A_i \in \mathbb{R}^K, K \ll D$ is the semantically meaningful attributes of the $i$th example and $y_i$ is either a class category if it is a classification task or a continuous value if it is a regression task for the $i$th example. We are interested in achieving 1) high performance for our predictions $\hat{y}$ and 2) good interpretability and actionability of any models.

### 3.2. Algorithms

Standard black-box models predict the target $\hat{y} = f(X)$ with a deep learning model $f$ that uses $X$ directly. To make our predictions interpretable and actionable, we want to make use of semantically meaningful attributes as a bottleneck layer such that the final output prediction model only uses these attributes as inputs. Additionally, we want to model the uncertainty of the attributes so humans can intervene when uncertainty is high.

Towards this end, we will learn an attribute model $\hat{A},\ \sigma_{\hat{A}} = g(X)$, where $\sigma_{\hat{A}} \in [0, 1]^K$ is the uncertainty in the attributes prediction and $g$ is a Bayesian neural network model. If the input dimensionality is low, $g$ can also represent other machine learning models that give uncertainty estimates. Finally, we will make our final prediction $\hat{y} = h(\hat{A})$, where $h$ can be a deep learning model or any other machine learning model.

#### 3.2.1 Interpretability through Attribute Model

Given a test data point $X_{test}$, we are able to see the predictions of attributes $\hat{A}_{test}, \sigma_{\hat{A}_{test}} = g(X_{test})$ before it is used to make the prediction $\hat{y} = h(\hat{A}_{test})$. If the data point has high error, we can check if the model has misunderstood the input by checking $\hat{A}_{test}$, allowing us to learn how to improve our model and debug individual predictions.

#### 3.2.2 Uncertainty Modelling of Attributes

Given a data point $X$, we are able to get the predictions and the corresponding uncertainty of attributes $\hat{A}, \sigma_{\hat{A}} = g(X)$.

We can obtain uncertainty estimates of the attributes through different methods. While the final softmax operation commonly used in deep classification models provides a probability distribution over classes, it does not provide us with a good estimate of uncertainty of our outputs. This is because the parameters are obtained through Maximum Likelihood training, which does not maintain multiple hypotheses and will not capture uncertainty inherent in the task and parameters. Additionally, if we have a regression task, there will be not be a softmax distribution that we can use to compute uncertainty.

To obtain better uncertainty estimates and to have a method that generalizes to continuous outputs, we implemented Bayesian neural networks by making use of [3]. In this algorithm, we will train a standard neural network with dropout. During inference, instead of turning off dropout (and keeping all the weights), we will keep dropout and run multiple forward passes to get multiple samples of the output. The inference procedure for a dropout-trained CNN is

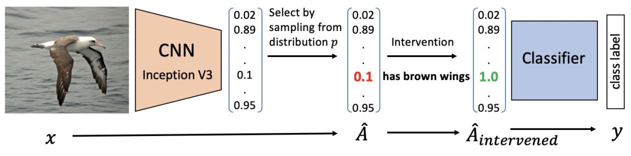

Figure 2. Test-time Intervention Algorithm Diagram

described in Algorithm 1.

**Algorithm 1** Predicting uncertainty $\sigma_i$ of attribute $a_i$ of given input $x$

```
1: Generated samples $G = \emptyset$
2: for j in range(num_samples) do
3:     $\hat{a}_i = CNN(x)$                ▷ dropout turned on
4:     Append $\hat{a}_i$ to $G$
5: $\sigma_i = std(G)$
```

### 3.2.3 Actionability with Test-time Intervention

Suppose the model is uncertain about any attribute or we are dissatisfied with the attribute value, we can help the model by setting the attribute to the correct value as shown in Fig. 2. With the new $\hat{A}_{intervened}$, we use our $\hat{A} \rightarrow y$ part of the model to produce the final $\hat{y}$. We term this procedure as test-time intervention.

The intuition behind why this procedure makes sense is because our targets $y$ can be difficult to predict by humans while the attributes $A$ are simple. For example, CUB is a difficult 200-way classification task but the attributes of CUB can be as simple as the color of a bird's chest. If humans are able to intervene in intermediate model representations (attributes) to provide alternative model outputs, it demonstrates the interpretability and actionability of the model.

$$p_{random}(a_i) = \frac{1}{K} \tag{1}$$

$$p_{softmax}(a_i) = \frac{|\hat{a}_i - 0.5|}{\sum_{j=1}^{K} |\hat{a}_j - 0.5|} \tag{2}$$

$$p_{dropout}(a_i) = \frac{(\sigma_i)^S}{\sum_{j=1}^{K} (\sigma_j)^S} \tag{3}$$

There are several ways in which we can select $\hat{A}$ for intervention. The simplest and most naive intervention scheme is a random selection (Eq. 1) of the $K$ different attributes. If we are classifying the attributes, a reasonable but poor performing scheme is selection according to how uncertain the softmax distribution is (Eq. 2). We do this by measuring the absolute distance of the probability distribution from 0.5 for the CUB dataset since attributes are binary and there is maximum entropy at 0.5. Finally, we make use of the uncertainty values returned by the Bayesian neural networks as shown in Eq. 3. $S$ is a scaling factor that allows us to interpolate between random ($S = 0$) and greedy ($S \rightarrow \infty$) selection. For our experiments, we simulate different levels of human-intervention by providing different amounts of oracle $A$ information that will override $\hat{A}$.

## 4. Experiments

### 4.1. Datasets

We will be using the public Caltech-UCSD Birds-200-2011 (CUB) [12] dataset, which is an image classification problem with 200 different bird species. The bird images are also annotated with their ground-truth attributes such as colors and patterns. Additionally, we will also be using a private medical image dataset (OAI) consisting of knee X-rays, where the task is to predict the Kellgren & Lawrence Grade (KLG) [8] which is an ordinal classification/regression that indicates the severity of osteoarthritis, a pervasive ailment affecting millions worldwide. There are 18 attributes, which are clinical annotations such as the presence of cysts.

### 4.2. Oracle Results

**Oracle A $\rightarrow$ y** To understand if attributes $A$ (clinical annotations) are useful for the prediction of target $y$ (KLG), we first construct an experiment where we allow a machine learning model to access the ground truth $A$ and assess its prediction performance on $y$. As seen in Tab. 2, the machine learning models have good performances when given oracle $A$. This provides an upper-bound for our models.

### 4.3. Model Setups

There are many different ways to integrate attributes $A$ into a deep learning model, we implemented a $X \rightarrow y$ model where we do not use $A$ at all. This model is an important baseline since it tells us what is the effect of the introduction of $A$ into our model. Then, we experiment on two possible $A$-based setups: $X \rightarrow \hat{A} \rightarrow y$ and $X \rightarrow \hat{A}, \quad \hat{A} \rightarrow y$. Let ResNet18$(X)$ be the function that uses a pre-trained 18-layer ResNet (with first 6 layers frozen) without the fully-connected layers to map an image input $X$ to an intermediate representation vector, FC$_M(X)$ be a fully connected layer that maps the input to $M$ outputs, and Inception$_{V3}(X)$ be version 3 of Inception model [4, 11].

**1) X $\rightarrow$ y** This model simply takes the input data $X$ and directly maps it to the target $y$ with a deep network. **OAI Implementation:** $f^{OAI}(X) = \text{FC}_1(\text{ResNet18}(X))$, Training loss = $\mathcal{L}^{OAI}_{x\rightarrow y}$ where

$$\mathcal{L}^{OAI}_{x\rightarrow y} = L^{OAI}_{A\rightarrow y} = \frac{1}{n}\sum_{i=1}^{n} MSE(y^{(i)}, \hat{y}^{(i)})$$

**CUB Implementation:** $f^{CUB}(X) = \text{FC}_{200}(\text{Inception}_{V3}(X))$, training loss = $\mathcal{L}^{CUB}_{x\rightarrow y}$ where

$$\mathcal{L}^{CUB}_{x\rightarrow y} = \mathcal{L}^{CUB}_{A\rightarrow y} = \frac{1}{n}\sum_{i=1}^{n} CE(y^{(i)}, \hat{y}^{(i)})$$

| Algorithm | OAI RMSE of $y$ | OAI RMSE of $A$ | CUB Acc of $y$ | CUB Acc of $A$ |
|---|---|---|---|---|
| $X \rightarrow y$ | 0.428 | - | 74.64 | - |
| $X \rightarrow \hat{A} \rightarrow y$ | 0.408 | 0.718 | 73.14 | 83.24 |
| $X \rightarrow \hat{A},\ \ \hat{A} \rightarrow y$ with MLP | 0.429 | 0.737 | 63.93 | 94.92 |
| $X \rightarrow \hat{A},\ \ \hat{A} \rightarrow y$ with RBF SVM | - | - | 64.43 | 94.92 |
| $X \rightarrow \hat{A},\ \ \hat{A} \rightarrow y$ with LR | - | - | 65.67 | 94.92 |

Table 1. Prediction performances on $y$ and $A$ for different model setups on OAI and CUB dataset.

**2) $\mathbf{X \rightarrow \hat{A},\ \ \hat{A} \rightarrow y}$** In this model, we train the deep network $X \rightarrow \hat{A}$ first before we use the predicted attributes $\hat{A}$ to train general machine learning models $\hat{A} \rightarrow y$. This model is different from $X \rightarrow \hat{A} \rightarrow y$ because it is not trained end-to-end. Therefore, loss from $y$ is not back-propagated to the $X \rightarrow \hat{A}$ model.

**OAI Implementation:** $g^{OAI}(X) = \text{FC}_K(\text{ResNet18}(X))$, $f^{OAI}(X) = \text{RBFSVM}(g^{OAI}(X))$. RBFSVM is the $\epsilon$-Support Vector Classifier ($\epsilon$-SVC) with a radial basis function (RBF) kernel. Training loss on $g$ is $\mathcal{L}^{OAI}_{x\rightarrow A}$ and training loss on $f$ is $|\xi|_\epsilon$ the $\epsilon$-insensitive RBFSVM loss.

$$L^{OAI}_{x\rightarrow A} = \frac{1}{n}\sum_{i=1}^{n} MSE(A^{(i)}, \hat{A}^{(i)})$$

**CUB Implementation:** $g^{CUB}(X) = \text{FC}_K(\text{Inception}_{V3}(X))$, $f_1^{CUB}(X) = \text{RBFSVM}(g^{CUB}(X))$, $f_2^{CUB}(X) = \text{LR}(g^{CUB}(X))$ (One-vs-All multiclass logistic regression classification), $f_3^{CUB}(X) = \text{MLP}_{50,50}(g^{CUB}(X))$ (4-layer perceptron). Training loss on $g$ is $\mathcal{L}^{CUB}_{x\rightarrow A}$, on $f_1, f_3$ is $\mathcal{L}^{CUB}_{A\rightarrow y}$, and on $f_2$ is the multiclass squared hinge loss.

$$L^{CUB}_{x\rightarrow A} = \frac{1}{n}\sum_{i=1}^{n} CE(A^{(i)}, \hat{A}^{(i)})$$

**3) $\mathbf{X \rightarrow \hat{A} \rightarrow y}$** This model forces the intermediate feature representation to be mapped to $\hat{A}$ first, before the $y$ value. We used $g(X) = \text{FC}_K(\text{ResNet18}(X)),\ \ f(X) = \text{FC}_o(\text{FC}_{50}(\text{FC}_{50}(g(X))))$ for both datasets.

**OAI Implementation:** $K = 10$, $o = 1$, and we found $\lambda = 1$. Training loss $\mathcal{L}^{OAI}_{x\rightarrow A\rightarrow y} = \mathcal{L}^{OAI}_{A\rightarrow y} + \lambda\mathcal{L}^{OAI}_{x\rightarrow A}$, where $\lambda$ is a weighting factor on the loss on $A$.

**CUB Implementation:** $K = 113$, $o = 200$, and we found $\lambda = 1$. Training loss $\mathcal{L}^{CUB}_{x\rightarrow A\rightarrow y} = \mathcal{L}^{CUB}_{A\rightarrow y} + \lambda\mathcal{L}^{CUB}_{x\rightarrow A}$, where $\lambda$ is a weighting factor on the loss on $A$.

### 4.3.1 Implementation Details

For the OAI dataset, we used the Adam optimizer with a batch size of 8 and learning rate of 0.0005, while for the $X \rightarrow \hat{A} \rightarrow y$ of the CUB dataset, we used the SGD optimizer with a batch size of 64 and learning rate of 0.01. Early stopping is done by selecting the model with the best validation performance. The hyperparameters are selected using random search to optimize $y$ on the validation set.

## 4.4. Results

| Algorithm | RMSE |
|---|---|
| Oracle $A \rightarrow y$ (MLP) | 0.191 |
| Oracle SVM $A \rightarrow y$ (RBF) | 0.120 |

Table 2. OAI performance on $y$ using Oracle $A$.

| | Macro | Micro |
|---|---|---|
| Precision | 0.753 | 0.731 |
| Recall | 0.734 | 0.731 |
| F1 | 0.736 | 0.731 |

Table 3. CUB performance on $y$ for $X \rightarrow \hat{A} \rightarrow Y$ model.

We use accuracy as the performance metric for the CUB dataset. The equation we use for accuracy is defined below.

$$accuracy^{CUB} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y^{(i)}_{pred} == y^{(i)}\} = \frac{tp}{n}$$

We use the root mean squared error (RMSE) as the performance metric for the OAI dataset, which is defined below.

$$RMSE^{OAI} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - y^{(i)}_{hat})^2}$$

From Tab. 1, we observe that $X \rightarrow \hat{A} \rightarrow y$ achieves an RMSE on $y$ of 0.425, which is better than the baseline $X \rightarrow y$'s 0.434 despite having an additional $\lambda L_A$ loss term in its training loss. This seems to suggest that modelling attributes $A$ in the intermediate layers not only allows us to achieve better interpretability, it also has the potential to enable us to perform better on our target $y$. Additionally, its surprising that $X \rightarrow \hat{A} \rightarrow y$ achieves an RMSE on $A$ of 0.718 while the $X \rightarrow \hat{A},\ \ \hat{A} \rightarrow y$ achieves 0.766 because there is an additional $L_y$ term in the former model's loss. In any case, we believe that there might be better performance for the $X \rightarrow \hat{A},\ \ \hat{A} \rightarrow y$ model if we used a more expressive $\hat{A} \rightarrow y$ model such as a deep network.

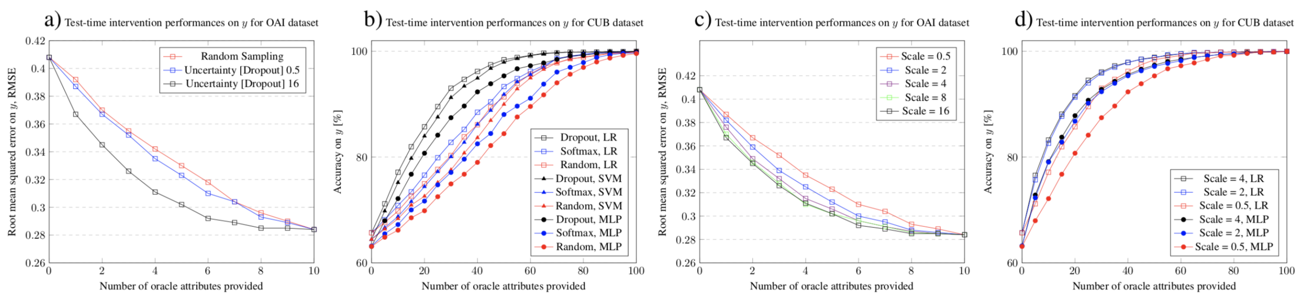

Figure 3. Test-time intervention performances o. $y$ for different sampling schemes on OAI and CUB dataset. The different attribute sampling and replacement schemes are random, dropout-based uncertainty, and softmax-based uncertainty.

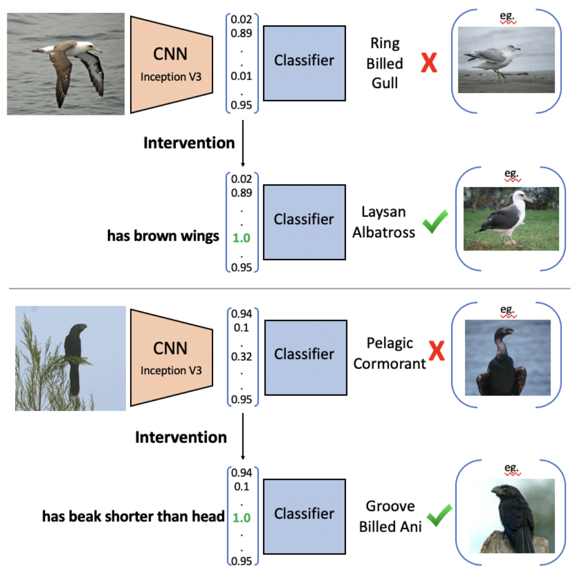

Figure 4. Successful test-time intervention examples.

## 4.5. Test-time Intervention Results

Fig. 3 shows performance improvements on both datasets after intervention, which supports the motivation of learning with attributes. Fig. 3-(a) and -(b) show that even with random selection of attributes to intervene, the performance still significantly improves. But more importantly, it shows that modelling and intervening with uncertainty is important as on all model variants in both datasets, our selection with dropout achieves the best performance, and softmax selection also improves over random. Fig. 3-(c) and -(d) show performances with different values of $S$ in $p_{dropout}$, showing that a good uncertainty model is important. Fig. 4 shows qualitative examples on the CUB dataset.

## 4.6. Limitations

Fig. 5-(ii) shows misclassified examples of our attribute model. The middle row shows the query of a Brandt Cormorant (top row) and a Frigatebird (bottom row). These show that there can be multiple possible labels for a given

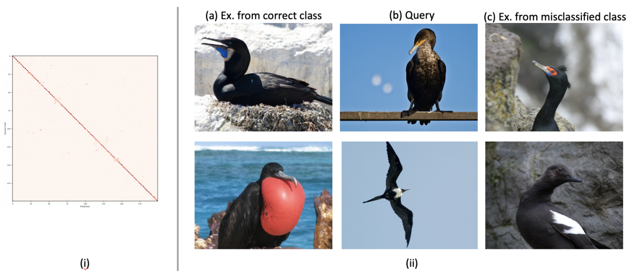

Figure 5. (i) shows the confusion matrix of our $X \rightarrow \hat{A} \rightarrow y$ model. (ii) shows examples of our limitations. (ii)-(b) shows our query bird, while (ii)-(a) and (ii)-(c) show an example from the correct and misclassified classes, respectively.

attribute in a bird class, e.g. the Brandt Cormorant's lower beak can be blue or orange, while a Frigatebird's belly can be red (males) or white (females). These make classification difficult even for humans as the examples from the misclassified class (ii)-(c) look visually more similar than a examples of the same species (ii)-(a). Our attribute model does not completely solve and struggles with this many-to-one mapping, but this phenomena further highlights the advantage of using attributes for interpretability. In the future, we hope to design better attribute learning models that can handle more complex hierarchies of attribute labels that would be able to handle the many-to-one mapping better.

## 5. Conclusion

Our work shows that attribute models allow us to achieve interpretability by allowing us to ask counterfactual questions and actionability by enabling human intervention. With our proposed uncertainty modelling of the attribute models, we further improved the effectiveness of human intervention by understanding where models require assistance. We provided extensive experiments on the CUB and OAI datasets to demonstrate our contributions. Through our framework, we hope to allow machine learning models to become more interpretable and actionable.

**Acknowledgements** We would like to thank Thao Ngyuen for providing the initial codebase and data preprocessing for the CUB dataset, and Pang Wei Koh and Percy Liang for their advice.

## 6. Contributions

Yew Siang: OAI dataset. Mikaela and Olivia: CUB dataset. All: Ideas and report writing.

Code link is found here: `https://drive.google.com/file/d/1yDQQfkqPXCiUEzkeQ3fUi3CkAj1mkHiR/view?usp=sharing`

## References

[1] Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. Grounding visual explanations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 264–279, 2018.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[3] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[5] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8779–8788, 2018.

[6] Huajie Jiang, Ruiping Wang, Shiguang Shan, Yi Yang, and Xilin Chen. Learning discriminative latent attributes for zero-shot classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4223–4232, 2017.

[7] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[8] Mark D Kohn, Adam A Sassoon, and Navin D Fernando. Classifications in brief: Kellgren-lawrence classification of osteoarthritis, 2016.

[9] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2013.

[10] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[12] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.