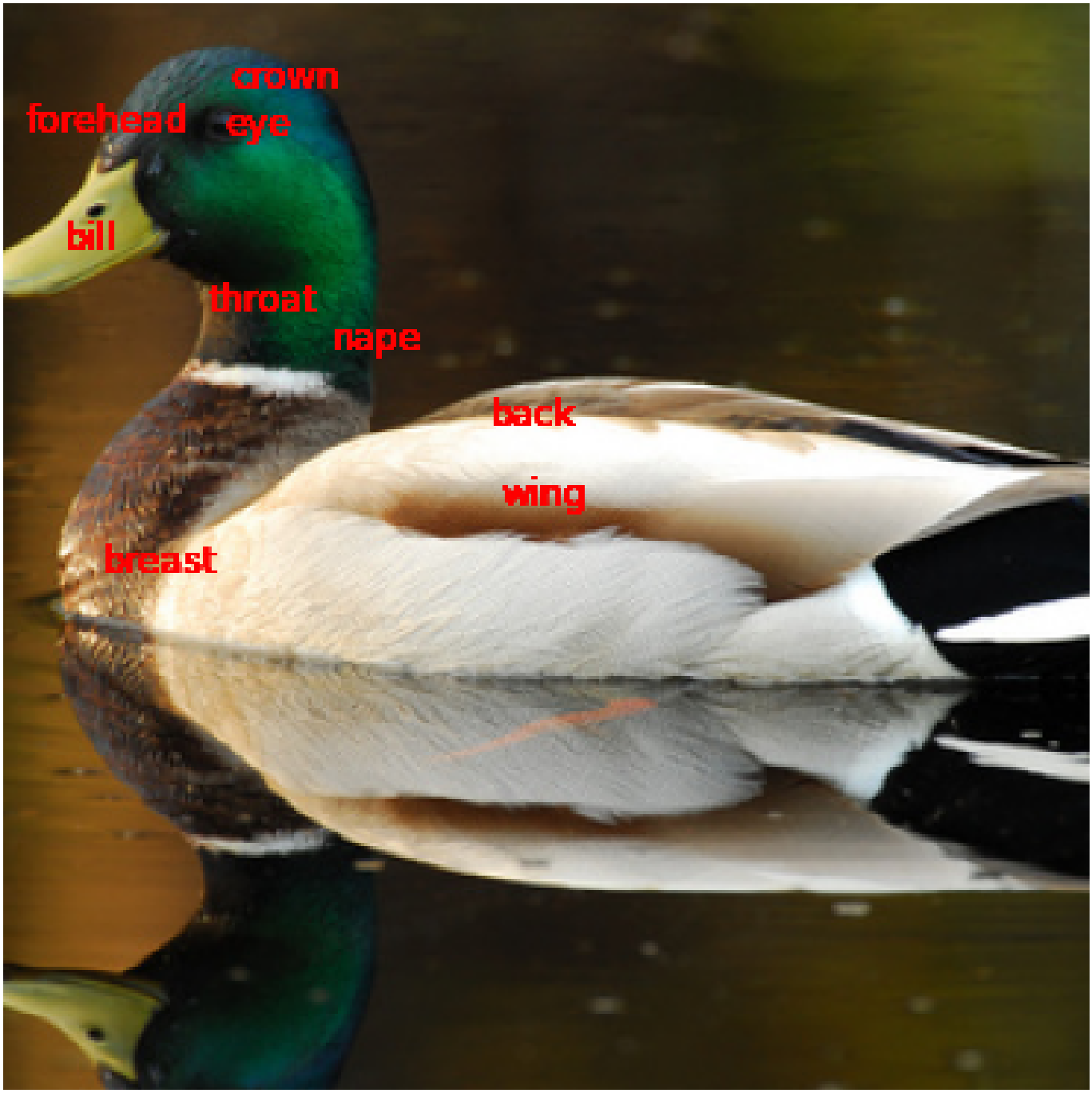
DTU Computer
Department of Applied Mathematics and Computer Science

Concept Bottleneck Models
Investigation of deceptive explainability

Master Thesis

Author: Andreas Raaskov

Advisor: Aasa Feragen



Acknowledgements

People who help with the creation of this thesis through academic sparing and feedback.

Manxi Lin
Nina Wang
Peter Johannes Tejlgaard Kampen
Andreas Abilstrup Hansen
Alexandra Inselmann
Elle Summer McFarlane
Albert Garde
Karina Knudsen
Ulla Mergrate Hansen
Henrik Madsen

Concept Bottleneck Models
Investigation of deceptive explainability

Master Thesis
December, 2024

By
Andreas Raaskov

GitHub repository: https://github.com/AndreasRaaskov/concept_bottleneck

Weights and bias training run overview: https://wandb.ai/s183901/bird_classification

Published by: DTU, Department of Applied Mathematics and Computer Science, Richard Petersens Plads Building 324, 2800 Kgs. Lyngby Denmark
www.compute.dtu.dk/
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Abstract

Concept bottleneck models (CBMs) are considered partially explainable image recognition models because they make a concept representation that should be easy for humans to interpret. However, the explainability of CBMs has come into question. For example, saliency maps have shown that CBMs do not always focus on the expected parts of an image, and there is a theory that these models may leak uninterpretable information through their concept bottleneck

In this thesis, I reimplemented the parts of the original *Concept Bottleneck* paper that focus on the CUB dataset are reimplemented. I also replicate the saliency-based criticism of CBMs. Furthermore, I propose a saliency score to measure the explainability of CBMs based on how well a saliency map attends to annotated coordinates.

I formulate a hypothesis that CBMs, in some cases, first predict the class and then use the concept to provide a post-hoc explanation of this prediction. I then demonstrate a few cases that show that CBMs can predict concepts not present in the image supporting this hypothesis.

I highlight 3 major criticisms of the original *Concept Bottleneck* paper that may cause this deceptive explainability:

- The paper uses majority voting on the concept, thus giving false concept labels to the images that do not contain the same concepts as its archetype.
- The paper uses a RandomResizedCrop preprocessing step that cuts concepts out of the image but still labels them as present.
- The use of a pre-trained model that is trained with data overlapping with the CUB dataset.

I show that CBMs trained without majority voting. still produces deceptive explanations but with much less certainty. Removing majority voting also comes with a big accuracy trade-off. While there, I state some theoretical arguments against using a pre-trained model and show that a model trained from scratch does not improve explainability. Not using RandomResizedCrop does seem to improve explainability, but it also makes models so unreliable that it is hard to conclude anything.

However, all models trained in this thesis still display signs of deceptive explainability, meaning that they cannot be trusted to predict concepts faithfully and can hide harmful bias in their concept predictions.

Contents

1 Introduction

This chapter introduces the basic ideas behind Concept Bottleneck Models (CBMs) and why they may not be explainable. It also defines the scope of the project and introduces what other people are doing with CBMs.

CBMs were introduced by Koh et al. (2020). The core idea behind a concept bottleneck model is that it consists of two classifiers: the first classifier takes an input *x* and predicts a series of concepts *c*, and the second classifier then uses the concepts to predict the class label *y*.

While the original paper (Koh et al., 2020) contains experiments on both the Osteoarthritis Initiative (OAI) dataset (Nevitt et al.) and the Caltech US Birds (CUB) dataset (Wah et al., 2011), the main focus of this thesis is on the CUB dataset. This is because the CUB data set is the most replicated part of the paper, but it is also the part where I see the biggest reason to criticize the paper. So the main predictions challenge in this thesis will be to use Inception-V3 (Szegedy et al., 2015) a deep convolutional neural network (CNN) to predict concepts c such as *has bill shape::all-purpose*, *has size::medium (9 - 16 in)* or *has forehead color::brown*. A full list of all concepts can be found in appendix A. Based on *c*, a perceptron classifier will be used to classify a bird label *y* such as *Mallard*, *Orange Crowned Warbler* or *Painted Bunting*, with a total of 200 different species of birds.

In theory, CBMs would be partially explainable since the model explains what concepts are part of the final prediction: for example, a model may tell us that a bird is a mallard because it has a green head, a Spartial bill shape, and a ducklike shape. Another key benefit of CBMs is that an operator can intervene on concepts during test time. For example, if the concept wing color is predicted to be black but the operator believes it to be brown, they can change a wrongly predicted concept to a correct one and thereby improve the accuracy of the final model.

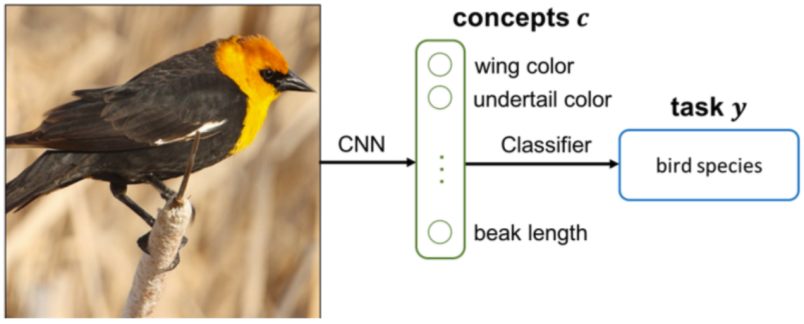


Figure 1.1: Figure by Koh et al. (2020): Illustration of how a concept bottleneck model classifies a bird by first classifying concepts and then use the concepts to predict what class the bird belongs to.

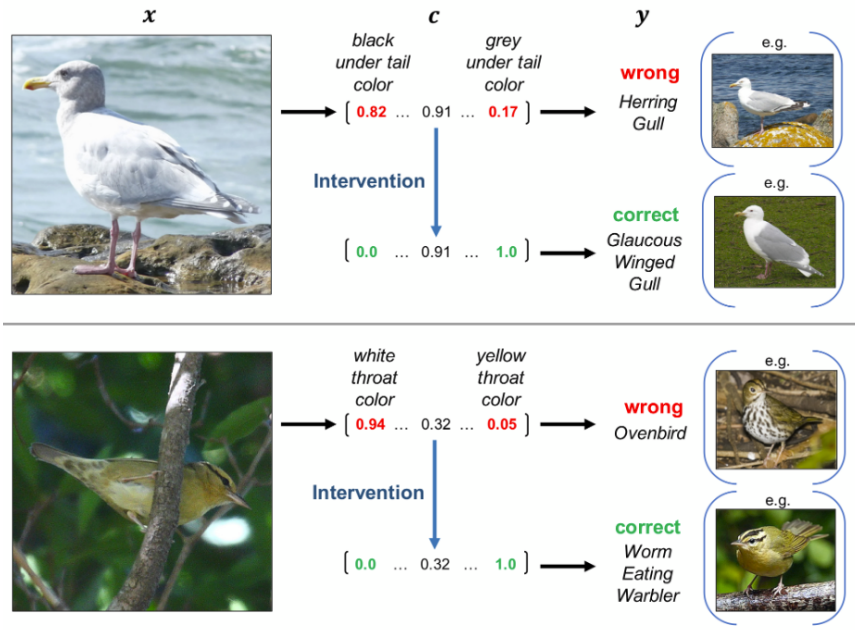


Figure 1.2: Figure by Koh et al. (2020) illustrating how intervention on concepts can provide better classification

While this kind of explainability seems intuitive, it still leaves open the problem that the model predicting the concepts is a black box. This means that we cannot know why the model thinks the wing is black or that the head is orange. Thus, in this thesis, I will explore the accuracy of the black box prediction and some examples of pitfalls that will cause the model to predict the wrong concepts in order to predict the correct bird, thus giving a deceptive explanation.

1.1 Motivation

The original paper (Koh et al., 2020) makes very few assumptions of concept bottleneck models (CBMs) being explainable: they only claim that the class predicting classifier is more explainable due to counterfactual explainability of performing interventions on concepts.
For example, if an operator changes the color, they can observe how this changes the type of bird the model predicts is in the image.
Despite the modest claims of the original paper, there seems to be a wide consensus in the community that CBMs are explainable. Therefore, this thesis aims to explore some of the fundamental assumptions made by Koh et al. (2020) to figure out how explainable CBMs are.

By pointing out potential pitfalls in the fundamentals of CBMs, we can prevent those mistakes from being replicated and, hopefully, develop CBMs that provide truthful explainability.

1.2 Hypothesis

The main hypothesis in this thesis is that deep convolutional neural networks (CNN) used to predict the concepts of Concept Bottleneck Models (CBMs) do not learn a latent representation of the concept. Instead, the CNN learns a latent representation of the class,

which is then mapped to the concept layer, as illustrated in Figure 1.3. If this is true, CBMs' explainability would be deceptive since they do not tell us what concepts are present in the input. Instead, they tell us the concept needed to justify a black box model's prediction.

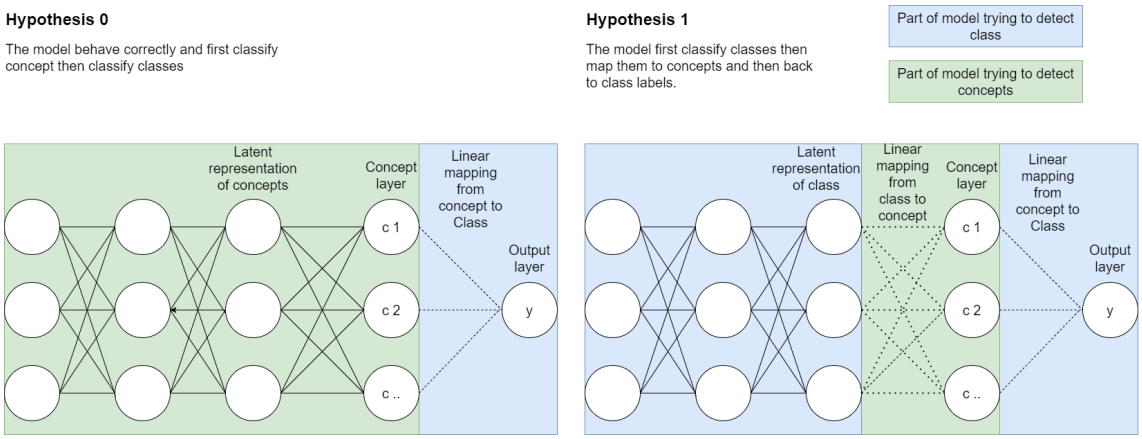


Figure 1.3

An example of hypothesis 1 being true is that if the CNN tries to predict whether or not the head is blue. Instead of looking at just the head, it will look at the entire bird to see if the bird is a type of bird with a blue head. This will happen even if the head is not visible or a minority of this class of birds have a different color head. If hypothesis 0 is true, the model would only look at the head and tell what color the head actually is; this would mean that if the head is not visible or if the head is another color, the head will not be predicted as blue.

1.3 Scope

The primary scope of this thesis is the implementation of the original Concept bottleneck paper Koh et al. (2020) to investigate the explainability of CBMs with respect to the stated hypothesis. I have reimplemented the part of the paper that focuses on the CUB-200-2011 dataset Wah et al. (2011).

While reimplementing the code, I found 3 major criticisms of Koh et al. methods that may cause the model to predict the class instead of concepts.

1.3.1 Majority voting

The criticism that led me to start this project was the use of majority voting in determining the concepts. Majority voting means that the concept labels were denoised by taking the setting of all labels of a concept to the majority of the class. Figure 1.4 illustrates an example of how majority voting is changing the concept labels to something not true. The details of how majority voting is performed can be seen in section 3.1.3.

(a) Painted Bunting male. (b) Painted Bunting female.

Figure 1.4: Painted Bunting: Male on the left and female on the right. Without majority voting, the crown and forehead of the female are labeled yellow; after majority voting, they are labeled blue.

1.3.2 Random Cropping

The Inception network used by Koh et al. is trained by using RandomResizedCrop (PyTorch Team, 2017) when training the model; this means that in many cases, the model is trained to recognize the concept that is not present in the part of the picture it is trained on. Cropping and preprocessing steps are described in more detail in section 3.1.4. examples of RandomResizedCrop can be seen in figure 1.5.

Figure 1.5: 4 samples of the Painted Bunting in figure 1.4a after RandomResizedCrop (PyTorch Team, 2017), all images would have the same concept label even if the concept is cropped out of the image

1.3.3 Data overlap

The CUB dataset overlaps with the ImageNet dataset (Wah et al., 2011). This means that when using a pretraining Inception network, the model is already trained to recognize the bird, and thus, it is theoretically easier for the model to map a latent layer already trained for bird classification to a concept layer than it is to learn the concept layer. Another problem is that there is a small overlap between the CUB dataset test set and the ImageNet training set. A blog post by Guo claims that 23 out of 5794 images in the CUB test set are also in the ImageNet training set, meaning that models using a pre-trained network have been trained on a tiny bit of their test set.

1.3.4 Objective explainability.

A problem when evaluating concept bottleneck models is the lack of a method to quantify how explainable in model is. This means that when dealing with a model with an accuracy

explainability tradeoff like CBMs, people are incentivized to maximize accuracy at the expense of explainability. I try to solve this by introducing a saliency score that, based on saliency maps, computes an objective score for how much attention the model pays to the part of the image it is supposed to attend. The details of how the saliency score is calculated can be seen in section 3.6.

1.4 Related work

1.4.1 Works on DTU

Part of the motivation for creating a better understanding of concept bottleneck models (CBMs) is that they are a key part of active DTU research into explainable fetal ultrasound quality assessment (Lin et al., 2022). This paper seems to avoid the problem of predicting class before concept by introducing a perceiving concept bottleneck model (PCBM) that first identifies a concept's location and produces a mask that makes sure the model only looks at the concept before predicting it, as seen in figure 1.6. However, they also test a CBM that shows that it classifies a concept despite this concept being manipulated out, which is a similar phenomenon to what I observe in section 5.3

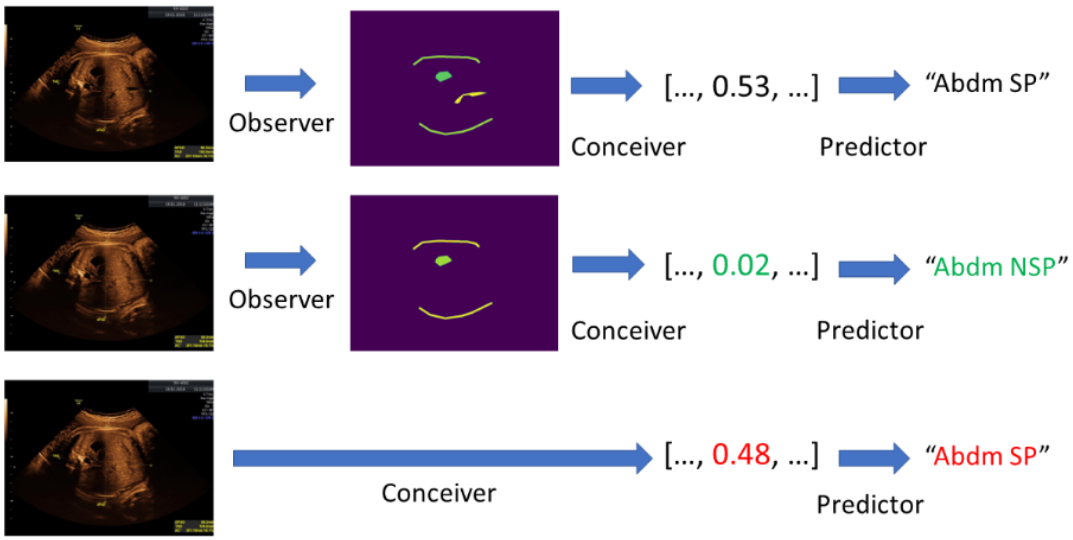


Figure 1.6: Figure by Lin et al. (2022) top picture shows a PCBM with the observer masking out concepts before the conceiver predicts it. The middle picture shows an input image where the input image has been manipulated to remove the umbilical vein. The mask from the umbilical vein is gone, and the concept related to the umbilical vein is predicted by the PCBM to be close to 0. The bottom image shows a classical CBM on the same manipulated image; the CBM predicts the umbilical vein to be present despite the manipulation, leading to a misclassification.

1.4.2 Concept bottlenecks on majority voted CUB dataset

The method used by Koh et al. (2020) has been used by many other papers to extend the work of concept bottleneck models. This means that criticism of the methods identified in this thesis also is to be found in those studies.

Barbiero et al. (2022) makes a model that can explain the classification in boolean logic. While they do use a ResNet10 is trained from scratch, they also use majority voting on the concept dataset. Ghosh et al. (2023) expand on this method by adding multiple logic-based experts using the same dataset as Barbiero et al. (2022). They only use 108 con-

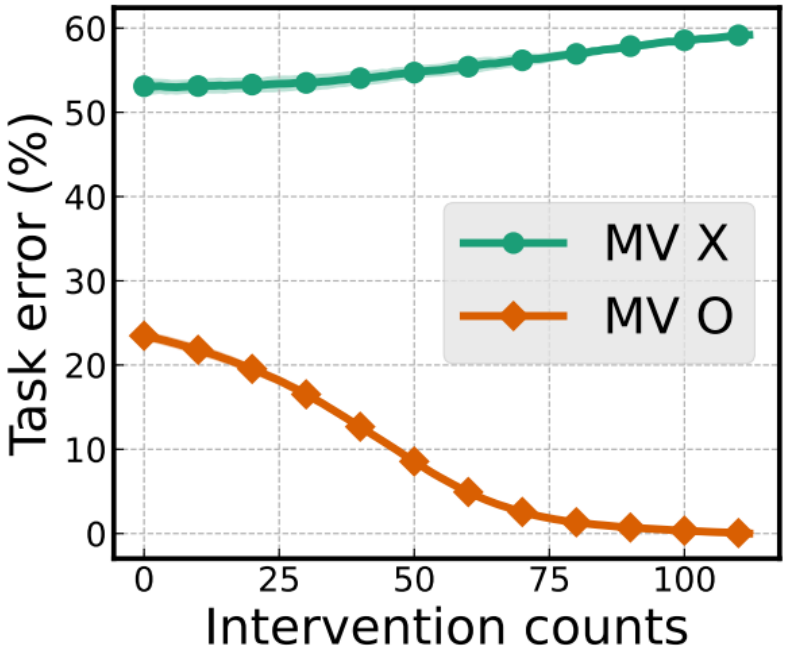


Figure 1.7: Figure by Shin et al. (2023) showing interventions on CBMs trained with majority voting (MV O) and without majority voting (MV X): What is worth noticing is that when intervening on concepts without majority voting, the error goes up.

cepts instead of the 112 used by Koh et al. (2020). This is most likely due to the majority voting and filtering being non-deterministic. I explore this problem in more detail in Section 3.1.1. Examples of problematic explainability from Ghosh et al. (2023) are provided in Appendix A.2."

Sheth and Kahou (2023) introduces new ways of training CBMs by improving the loss function and testing their results on several datasets. This includes CUB, where they do a majority voting but not the following filtering step that reduces the number of concepts as described in 3.1.3.

Shin et al. (2023) investigates new ways to intervene on concepts. Most of their paper are based on a majority-voted dataset. They do one small experiment at the bottom of their paper, where they train a concept bottleneck model without majority voting. The results of this experiment can be seen in figure 1.7:

Havasi et al. (2022b) proposes CBM that doesn't have leakage in the concept bottleneck layer as described in 2.3; they also use a majority-voted dataset.

1.5 Chat-GPT generated concepts

Instead of relying on human-annotated concepts Oikarinen et al. (2023) and Yang et al. (2023) use Chat-GPT3 to generate concept labels. While this avoids problems of majority voting, it doesn't necessarily make the concepts more explainable. Examples of the problematic explainability of this method can be found in the appendix A.2.

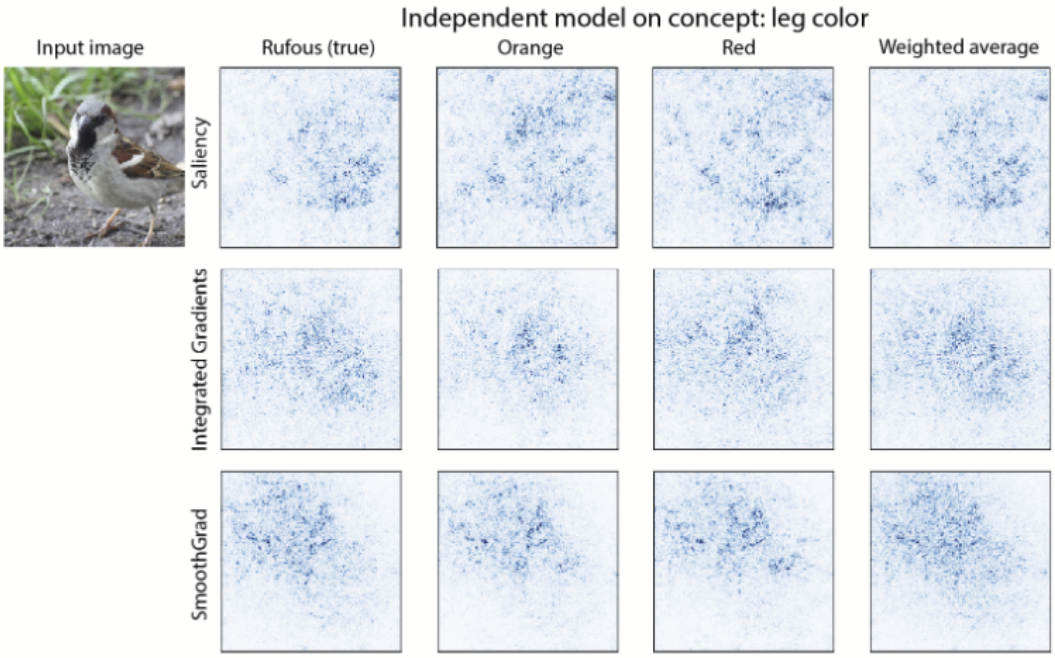


Figure 1.8: Figure by Margeloiu et al. (2021) illustrating three methods for calculating saliency maps for leg color concepts. Note that none of the maps focus on the legs.

1.6 Saliency maps for criticisms of concept bottleneck models

A major criticism of concept bottleneck models comes in the form of saliency maps showing that CBMs do not attend to the part of the image they are supposed to, as demonstrated by Margeloiu et al. (2021) in figure 1.8. A problem with saliency maps is that they only provide a qualitative description of a single image, not an objective number for the entire test set. however, because the CUB dataset contains coordinates for some concepts, it is possible to create an objective score based on comparing saliency maps with the actual locations of a concept. Huang et al. (2024) introduce a saliency score based on drawing a box around the most salient pixel. They then make a binary value based on whether the annotated coordinate is inside the box or not. How this is done is illustrated in figure 1.9: I introduce my own way of calculating a saliency score in section 3.6.

1.7 Toy and Simulated Datasets for Critiquing Concept Bottleneck Models

The paper *Promises and Pitfalls of Black-Box Concept Learning Models* by Mahinpei et al. (2021) contains several examples of simulated datasets and toy datasets to show how the concept layer is carrying more information than they are supposed to, also called leakage, and explores in detail in section 2.3. Shin et al. (2023) also presents a way to simulate a dataset for concept bottleneck models in order to investigate what happens when concepts become noisy and what happens if the dataset has a concept that is not used for training.

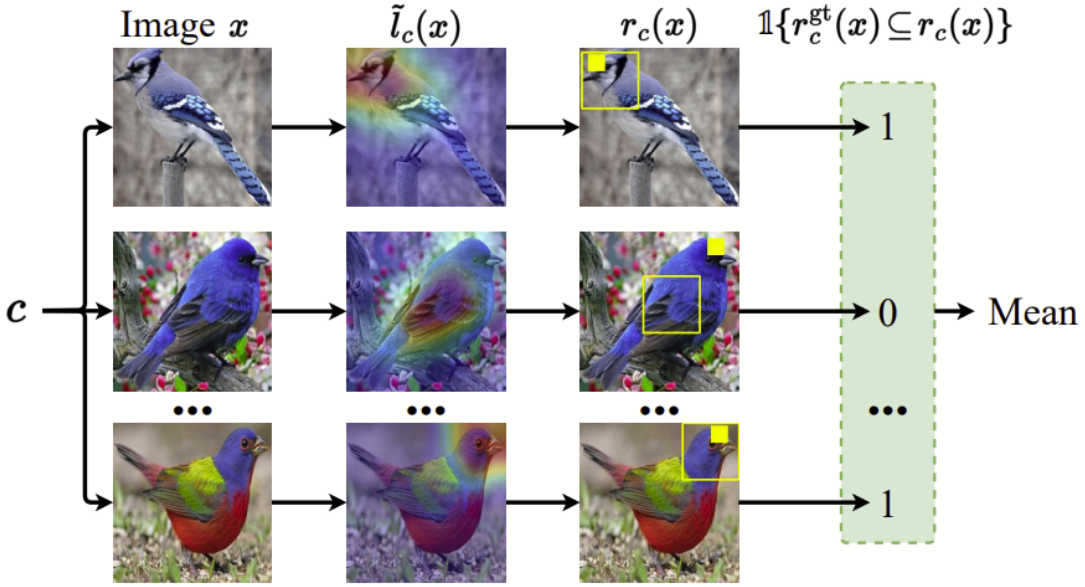


Figure 1.9: Figure by Huang et al. (2024): illustrating how they calculate their score. The yellow dot is the annotated coordinate, and the yellow box is centered around the most salient pixel. If the coordinate is inside the box, the score for that concept is 1. Else, it is 0.

2 Theory

This section seeks to give the reader a clear understanding of how concept bottleneck models work, and how they are trained. I also discuss the theory behind the Inception model used in this thesis, and how saliency maps can be used to critique explainable image models.

2.1 Concept bottleneck models

A concept bottleneck model can be described as a combination of two models that each perform their own prediction task. The first model $g()$ takes an input from the dataset $x \in R^d$ and transforms it to the concept space $c \in R^k$ such that $\hat{c} = g(x)$. The second model $f()$ uses $\hat{c}$ to predict $\hat{y}$, such that $\hat{y} = f(\hat{c})$. No matter how they were trained, the final concept bottleneck model would, at test time, compute:

$$\begin{aligned} \hat{c} &= \sigma(g(x)) \\ \hat{y} &= f(\hat{c}) \end{aligned} \tag{2.1}$$

Where σ can represent different functions, this function can either be a stepper function or a sigmoid function, but it can also be excluded. A theoretical argument for why these functions should be included can be found in section 2.3 a practical argument, as well as the implementation, can be found in section 3.2.2. In the original paper(Koh et al., 2020), this function is not used except for one small experiment that uses a sigmoid function all other CBMs in the original paper use $\hat{y} = f((\hat{c}))$

2.1.1 Different types of concept bottleneck models.

Koh et al. (2020) introduce 4 ways of training a concept bottleneck model: Joint, Sequential, Independent, and Standard.
All methods aim to learn the optimal $\hat{f}()$ and $\hat{g}()$ by minimizing a loss from the concepts and a loss from the class prediction.
The class loss can be described as L_Y and could be any loss function for a multiclass prediction.
The Concept loss can be described as L_{C_j} where j is the index of the concept such that each concept has its own loss function. This is because each concept can be treated as a binary classification independent of all other concepts. This also means that each concept can have its loss weighted depending on how important it is. Weighing of the concept loss is explored further in section 3.3.1.

2.1.1.1 Independent

The independent bottleneck model is based on training $f()$ and $g()$ independent of each other:

The concept predicting model $g()$ is trained by optimising equation 2.2

$$\hat{g} = \arg\min_g \sum_{i,j} L_{C_j} \left(g_j(x^{(i)}); c_j^{(i)} \right) \tag{2.2}$$

The class predicting model $f()$ is trained by optimizing equation 2.2 Note that the model is trained on the true concept provided by the dataset, and therefore this model can be trained independently of $f()$

$$\hat{f} = \arg\min_f \sum_i L_Y \left(f(c^{(i)}); y^{(i)} \right) \tag{2.3}$$

2.1.1.2 Sequential

In the sequential model, the concept predicting classifier is trained in the same way as in independent training as described in equation 2.2, the learned $\hat{G}$ is then according to Koh et al. (2020) used to train the class predicting model by optimizing

$$\hat{f} = \arg\min_{f} \sum_{i} L_Y \left(f \left(\hat{g}(x^{(i)}) \right); y^{(i)} \right) \tag{2.4}$$

In practice, both in my code and the original code, this is done by first creating a new concept dataset $\hat{c} = \hat{g}(x)$ and then training by optimizing equation 2.5:

$$\hat{g} = \arg\min_{g} \sum_{i,j} L_{C_j} \left(g_j(x^{(i)}); \hat{c}_j^{(i)} \right) \tag{2.5}$$

The difference between equation 2.4 and 2.5 illustrate that the only difference between the independent training and sequential training is that the independent concept classifier is training on the true concepts c while the sequential is trained on a generated concepts $\hat{c}$.

2.1.1.3 Joint

The joint models train both $f()$ and $g()$ by generating a loss function for the entire concept bottleneck model as:

$$\hat{f}, \hat{g} = \arg\min_{f,g} \sum_{i} \left[L_Y \left(f(g(x^{(i)}); y^{(i)} \right) + \sum_{j} \lambda L_{C_j} \left(g(x^{(i)}); c^{(i)} \right) \right] \tag{2.6}$$

Where $\lambda > 0$ is a hyperparameter. Koh et al. (2020) finds lambda by making a hyper parameter opitimisation, while this method can find the best value for λ it is also expensive. I, therefore, suggest a theoretical argument for $\lambda = 1/J$ where J is the number of concepts. The argument is that without any other prior assumptions, we should expect the loss from classifying all the concepts wrong to be the same as the loss from classifying the class wrong. The loss function for a single datapoint in a joint bottleneck model can be written as:

$$L_{joint} = L_Y \left(\hat{y}; y \right) + \lambda \sum_{j}^{J} L_{C_j} \left(\hat{c_j}; c_j \right) \tag{2.7}$$

So if $L_Y()$ and $L_{C_j}()$ are both loss functions with the same loss for a misclassification such that $L_Y(1,0) = L_C(1,0)$, then in order to for them contribute equal to the final loss we can write:

$$\lambda \sum_{j}^{J} L_{C_j} \left(1; 0 \right) = L_Y \left(1; 0 \right) \tag{2.8}$$

$$\lambda \cdot J \cdot L_{C_j} \left(1; 0 \right) = L_Y (1; 0) \tag{2.9}$$

$$\lambda = \frac{L_Y (1; 0)}{J \cdot L_{C_j} \left(1; 0 \right)} \tag{2.10}$$

$$\lambda = \frac{\cancel{L_{C_j}(1,0)}}{J \cdot \cancel{L_{C_j}(1,0)}} \tag{2.11}$$

$$\lambda = \frac{1}{J} \tag{2.12}$$

Note that this value λ should be used if class and concept prediction is weighted equally; in theory, if $\lambda < \frac{1}{J}$ the model would achieve better accuracy on the final class prediction but may allow more leakage thus being less explainable.

2.2 Standard

Standard training ignores the concept and only trains on the class label.

$$\hat{f}, \hat{g} = \arg\min_{f,g} \sum_{i} L_Y \left(f \left(g \left(x^{(i)} \right) \right) ; y^{(i)} \right) \tag{2.13}$$

The standard model is not a concept bottleneck but uses the same structure; thus, it is a useful baseline.

2.3 leakage

Leakage refers to a phenomenon where the learned concept representations encode information unrelated to the concept. For example, consider a scenario where a model predicts the concept *has bill shape hooked seabird*. If $\hat{c} = g(x)$ also encodes information about the presence of water in the image into $\hat{c}$, the final classifier might use that water-related information to associate the concept with birds that live near water—even if such information is irrelevant or misleading. This may lead to wrong predictions, but even worse, this model would not be explainable since we do not know if the concept *has bill shape hooked seabird* predicts the shape of the bill or just the presence of water. Since the model uses a soft concept, the information about water can be encoded by just changing a lot of concepts a little bit.

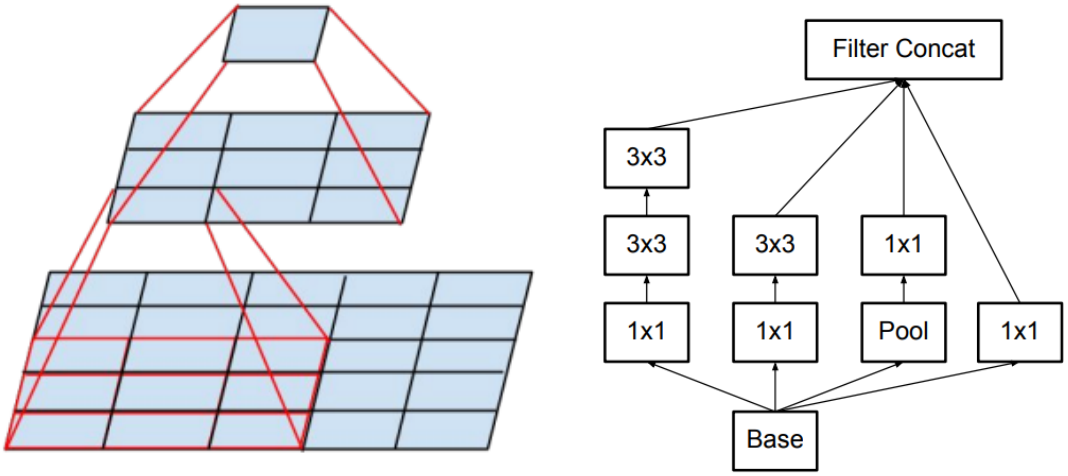
The theory of leakage was introduced by Mahinpei et al. (2021), who shows that leakage occurs in concept bottleneck models for both sequential training and joint training. They also show that concept bottleneck models can use leakage to predict the correct class even when concept labels are randomly generated. Havasi et al. (2022a) expands on the theory of leakage and defines the Markovian assumption as the label y being conditionally independent of the input x given the concepts c ($y \perp x \mid c$). Leakage then occurs when the Markovian assumption is not fulfilled, and x is carrying information not present in c. If the Markovian assumption is not fulfilled, the model could try to include more information into c than what can be supported by the true concepts in order to improve accuracy.

Both Mahinpei et al. (2021) and Havasi et al. (2022a) suggest addressing leakage by using hard concepts, meaning that each concept is predicted as either true or false prior to training $f()$. In section 3.2, I discuss the possibility of using hard concepts in section 3.2.2.

While leakage is a valid criticism of CBMs, my criticism of using majority-voted concepts goes beyond leakage, since using majority voting allows c to contain all the information needed to predict y (this is shown in section 4.1.1). This means that the Markovian assumption is fulfilled for the majority-voted dataset. The downside, however, is that in order for y to be predicted correctly, $\hat{c}$ must contain the majority-voted concept even if they are not present in x.

2.4 Inception-V3 architecture

The Convolutional Neural Network (CNN) used in this thesis is the Inception-V3 model (Szegedy et al., 2015). The key innovation in the Inception models compared to other



(a) Example of how two 3x3 convolutions can work as a 5x5 convelution

(b) An inception module replacing a traditional CNN layer

Figure 2.1: Two example from the Inception V3 paper Szegedy et al. (2015) on how an Inception network differentiates its architecture from a standard CNN

CNNs is that some convolutional layers are split up into modules that perform multiple smaller convolutions. Some convolutions are done in parallel to get the advantages provided by different-size convolutional filters and then concatenated at the end of the module as seen in figure 2.1b. Inside a module, they also stack multiple smaller convolutions in order to stack multiple smaller convolutions to approximate a larger convolution: an example theorized by Szegedy et al. (2015) is that two 3x3 convolutions could replace a 5x5 convolution with minimal performance loss, but using less computation power as seen in figure 2.1a. The inception network also uses an auxiliary that trains an auxiliary classifier on a layer midway through the model. This makes it possible to construct an auxiliary loss function that can give a stronger gradient to the early layers of the network.
A possible downside of using auxiliary loss is that when training Joint concept bottleneck models, a gradient for the final class bypasses the concept bottleneck layer, further incentivizing the model to learn class representations instead of concept representations.

2.5 Saliency maps

Saliency maps are a method to interpret the prediction of a neural net qualitatively. The saliency maps for convolutional neural nets were introduced by Simonyan et al. (2014) as a visual illustration of what part of images is most salient for making the prediction of a certain class.

Saliency maps are based on calculating the gradient with respect to a target class; this gives a tensor with the same dimensions as the input. By taking the maximum absolute gradient value across all color channels for each pixel, a map highlighting the most influential areas of the image can be generated. Thus, the saliency maps tell us which pixel would cause the biggest change in the final prediction if that pixel were changed just a little bit.

I used this standard type of silence map to calculate the saliency score described in section 3.6

2.5.1 noisetunnel

Saliency maps can be very noisy due to individual pixels having a relatively strong influence on the final prediction; thus Smilkov et al. (2017) introduce SmoothGrad or noisetunneling, whereby adding a small noise to the image and sample multiple saliency maps can create an image more interpretable to the human eye.

In this thesis, I have used noise tunneling for the saliency maps shown; the plots are all made with a standard deviation of 0.2 and sampled 50 times.

3 Method

In this chapter, I explain some of the most fundamental functions for how I implemented the paper and the new methods I have developed to test concept bottleneck models (CBMs). Most of my time in this project has been spent replicating the work of Koh et al. (2020): My code was originally based on the code provided with the paper [1]

To improve code quality and really understand the code, I have been rewriting almost all functions to the point where very little remains of the original code base.
All the code used in this thesis can be found on GitHub:
https://github.com/AndreasRaaskov/concept_bottleneck

3.1 Datasets

The Caltech UCSD birds-200-2011 dataset Wah et al. (2011) consists of 11788 images of 200 different bird species, that is the clas label y. Each image is annotated with 312 different attributes. To maintain consistency, I will, in this thesis, refer to those 312 attributes as concepts or c. All concepts were annotated using Amazon Mechanical Turk (A crowd-sourced service). A full list of all concepts can be found in appendix A

The dataset also contained the location of each part, which was annotated as the average of 5 Turck clicking on the same image.

3.1.1 Train val split of CUB dataset

The train validation split was defined by the CUB in the dataset (Wah et al., 2011). Later Koh et al. (2020)created a test set by splitting the training set using a random seed. This test set is used to find the best hyperparameters, and I refer to it for a few mine experiments in this chapter to argue for my hyperparameter selection

Since the majority voting and filtering process described in section 3.1.3 is only performed on the training set, this split made the concepts annotation and a number of majority-voted concepts partially random. This happens because removing random data points changes the distribution of concepts a little bit, leading to some concepts being the majority in some splits and not in others.

Thus, in order to get the same train test split as Koh et al. (2020), I had to download their dataset[2] and then extract the train test split from it.

I then rewrote the data processing function to take in the train test split and provide the same split as used for the original experiments.

3.1.2 Renaming of attributes

The terms concepts and attributes have been used interchangeably to describe the classification of the concept layer, also referred to as c. In the original dataset and code, the word attributes is used; however, concepts are used in the paper. In this thesis and my code, I have made a commitment to the name Concepts and have used this name both in the code and the report.

[1] The code from the original paper can be found on: https://github.com/yewsiang/ConceptBottleneck

[2] The dataset made by Koh et al. (2020) can be found on https://worksheets.codalab.org/worksheets/0x362911581fcd4e048ddfd84f47203fd2

3.1.3 Majority voting and filtering

The use of majority voting was my first main critique of Koh et al. implementations of CBM in the CUB dataset.

Majority voting is based on counting the number of concepts labeled true or false for each type of concept for a class; each type will of concept would then be changed to the majority for this type of bird such that all birds of this type would have the same set of concepts. This also means that all birds of the same class will have the same concept labels independent of what they look like or if the concept is present is visible in the picture.

In the CUB dataset (Wah et al., 2011), annotators were asked to state how certain they were in their prediction; this included stating if a concept is visible. When majority voting is performed, only concepts that are visible are counted. In the new majority-voted dataset, all concept labels would then be set to the majority label.

For example, if a class of birds has yellow legs that are true in 40% of the pictures, yellow legs are false in 60% of the pictures, but 30% of the legs are false because the legs are not visible. All birds in this class would be annotated to have yellow legs no matter what their legs look like or if the legs are visible.

Another example is that if 30% of a class has a size medium, 40% has a size small, and 30% has a size very small, all have size labels would be false because none of them has over 50%.

The filter step is meant to remove sparse concepts; a threshold is selected; Koh et al. (2020) use a threshold of 10. If a concept is not true for more than the threshold classes, then the concept is removed.

For example, if less than 10 types of birds have a green head after majority voting, the concept of green head will be removed.

This means that the number of concepts is reduced from 312 to 112 when majority voting and filtering are applied.

When I refer to models with majority voting, I mean models that went through those two steps. When I refer to models without majority voting, I mean models that did not go through those steps and thus still predicted all 312 concepts.

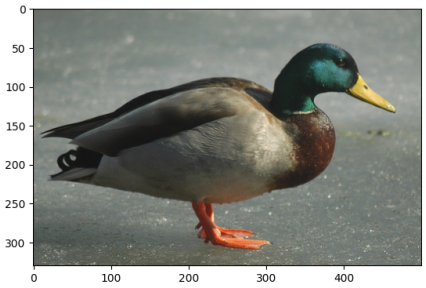
3.1.4 Cropping of images

The second critique I had of Koh et al. is the use of the RandomResizeCrop function PyTorch Team (2017) in the data preprocessing step that crops out a random portion of the image and resizes it to the 299 x 299 dimension that the Inception-V3 model takes as input. This method was also suggested by Szegedy et al. (2015) and Cui et al. (2018) as the correct for retraining inception networks.

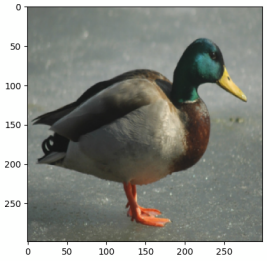
For RandomResizedCrop, I use the standard setting of the PyTorch implementations PyTorch Team (2017), meaning that up to 92% of the image can be removed before the picture is interpolated to 299 x 299 dimensions.

However, this method should, in theory, not be suitable for concept bottleneck models since it would cut out concepts but still label them as present. An example of this can

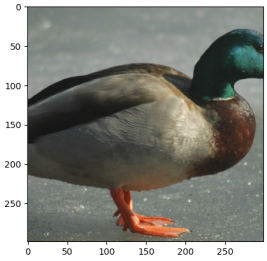
be seen in figure 3.1d, where the random resized crop removes concepts from the image but still labels them as true. Thus, I ran an experiment where I used a torch to resize the image to 299 x 299 without cropping anything. The result of the can be seen in section 4.3.
For the test and validation set, images were center-cropped to 299 x 299 since this is the method to evaluate the models used by Koh et al.. An example of center cropping can be seen in figure 3.1c. Note that concepts can also be cropped off when using center cropping, which I explore further in section 5.3



(a) Original Image



(b) Resized Image



(c) Center-Cropped Image

(d) 4 examples Randomly Resized crop done by torch Image

Figure 3.1: Demonstration of different image transformation techniques used in this thesis

3.1.5 Other data preprocessing

A few other pre-processing steps were used on the training set for data augmentation following the Koh et al. (2020) implementation; this includes Color Jitter, Random Horizontal Flip, and Normalization to mean 0.5 and standard deviation 2.

3.2 Models Architecture

In this section, I describe my implementation of the concept bottleneck models; since this is a practical implementation, I will change the notation from the abstract math notation in the theory section 2.1 to the model names used in the code. So what the theory section refers to as the $\hat{c} = g(x)$ will forward be referred to as the X-to-C model, and $\hat{y} = f(c)$ will forward be referred to as the C to Y model.

3.2.0.1 X to C model

The X to C model could, in theory, be any CNN architecture, but for the CUB dataset Koh et al. (2020) used the Inception-V3 architecture as described by Szegedy et al. (2015) therefore, I also use a standard inception model from pytorch (PyTorch Team, 2015) with the only chance that the output layer is reinitialized to fit the output dimensions of the number.

3.2.0.2 C to Y model

The C to Y model is a multi-class perceptron (single-layer neural network).

While concept bottleneck models can have more complicated C to Y models, the linear perceptron offers extra explainability since all concepts map directly to a class. And thus, we can inspect the weights.
The C to Y model can also be used as a baseline when tested on the true concepts. The idea is to test the predictive power of the annotated concepts to determine whether they are meaningful for the final classification.

3.2.0.3 X to Y model

The X to Y model is a standard inception like the X to C model, with the only difference being that the output layer is 200, matching the number of classes.
Since the standard model doesn't use concepts, it can be considered a baseline telling us how much accuracy we should expect from a model without a concept bottleneck.

3.2.1 Pretraining

My criticism of the Koh et al. (2020) is that they use the pre-trained weight from python PyTorch Team (2015) pre-train on the image net. I do the same in all my models but run one experiment on weight randomly initialized.

3.2.2 Augment against sigmoid and for hard concepts

I apply a sigmoid function to the output of the X-to-C model before passing it to the C-to-Y model, so that each concept can be interpreted as a probability. For joint models, this means that the model applies a sigmoid function between the final layer and the layer before.

For sequential and independent models, I create a new dataset using the X to C model and then apply an activation function before using the data to train the C to Y model. A minor experiment was run to determine the effect of applying sigmoid between the X to C model and the C to Y model compared to a model in which the raw logits of the X to C model were fed directly to the C to Y model.

As seen in the table 3.1 and 3.2, the model with sigmoid is slightly better than the raw logits. Thus, I chose to keep sigmoid between the two models. However, it is worth noticing that Koh et al. (2020) finds that the raw logits give better accuracy, and thus they use raw logits; they also show that while using logits gives better performance, it makes it harder to intervene on concepts, which is one of the core features of a concept bottleneck model.

Model	logits	Sigmoid
With majority voting	0.743	0.725
Without majority voting	0.688	0.687

Table 3.1: Model accuracy on class prediction on validation set for joint models

The hard function refer in table 3.2 refers to hard concepts, which are binary concepts determined by predicting the concept to True or False and then returning 0 or 1 instead of a sigmoid probability or logits. The fact that this is performing well may be relevant for experiments trying to limit leakage, as discussed in section 2.3.

Model	logits	Sigmoid	Hard
With majority voting	0.778	0.798	0.813
Without majority voting	0.217	0.274	0.251

Table 3.2: Model accuracy on class prediction on the test set for sequential models

3.3 Hyperparameter

Finding the correct hyperparameters has been quite challenging since the Koh et al. (2020) had done a hyperparameter search for all their models. However, I have more models and fewer computing resources; therefore, I only ran a few experiments until I found a hyperparameter configuration that seemed to work for most models. I then ran all models with the same hyperparameters except for λ in order to ensure a fair comparison. I ended up with the following hyperparameters: A batch size of 32 with an Adam optimizer (Kingma and Ba, 2017) and an initial learning rate of 0.001 with a decay of 0.94 every 20 epochs.

For joint models I use the λ value of $\lambda = 1/J$ as descubed in section the theory section 2.1.

For the $J = 112$ concept that is left after majority voting and filtering, this means that $\lambda = 1/112 \approx 0.008928$, which corresponds fine with Koh et al. (2020) concluding that the optimal $\lambda = 0.01$. Without majority voting, there are $J = 312$ original concepts; thus, the optimal value should be $\lambda = 1/312 \approx 0.003205$.

3.3.1 Concepts weighting

The concept dataset is quite sparse, with most labels being False.Koh et al. (2020) tries to deal with this problem by introducing a class imbalance weight that is calculated as:

$$\textit{imbalance ratio} = \frac{total}{positive} - 1$$

This formulation seemed a bit strange to me at first, but it is equal to the official formulation in **?**, which is:

$$\textit{imbalance ratio} = \frac{negative}{positive}$$

prof

$$\frac{total}{positive} - 1 = \frac{positive + negative}{positive} - 1 = \frac{negative}{positive} + \frac{positive}{positive} - 1 = \frac{negative}{positive}$$

The imbalance is given to the Binary cross entropy loss function. However, upon carefully reading the documentation, I discovered that the imbalance should be given as the argument *pos weight* where the original code uses *weight*

A short experiment on the test set that can be seen in the table, however, revealed that doing concept weighting increases the accuracy of the majority-voted Sequential model while making the non-majority-voted sequential model a lot worse. This meant that I chose not to use concept weighting in any of the models presented in the results.

Model	Whith weighting	Without weighting
Joint With majority voting	0.74	0.72
Sequential with majority voting	0.99	0.8
Joint Without majority voting	0.45	0.68
Sequential Without majority voting	0.05	0.2

Table 3.3: Model accuracy on class prediction on validation set for joint models

3.4 Training of the models

Training of the models follows the theory in section 3.1.1. For joint models, I use a Binary cross-entropy loss L_{C_j} for concepts and a cross-entropy loss for the class predictions L_Y. The final loss is then calculated as:

$$L_{joint} = L_Y + \lambda \cdot \sum_j L_{C_j}$$

Since the Inception-V3 produces an auxiliary output, it also needs an auxiliary loss that is calculated the same way as the main loss.

For the sequential trained models, the X to C classifier is first trained by minimizing L_{C_j}. Once the classifier is trained, it is used to make a new dataset of concept $\hat{c}$. The new dataset is then used to train a standard scikit-learn perceptron Pedregosa et al. (2011) as the C to Y model.

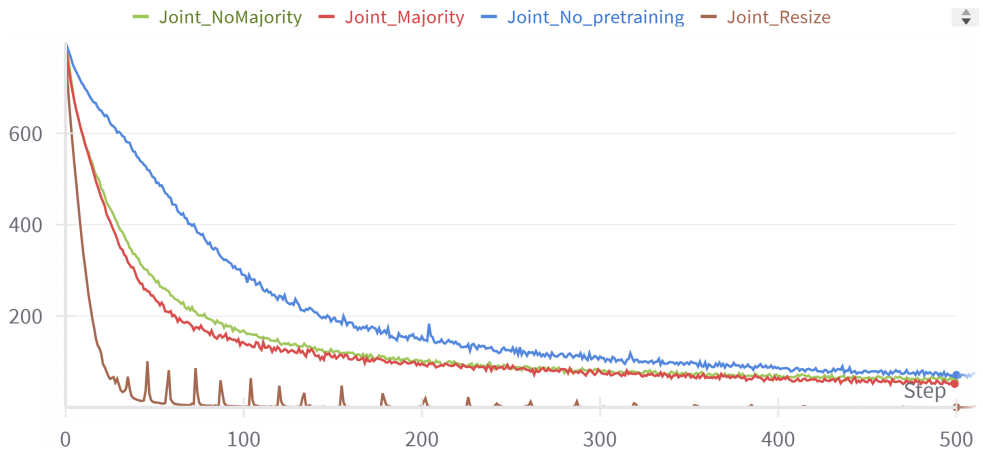
For independent models, a standard perceptron is also trained, but on the true concepts c, the X to C classifier is the same as that for the sequential model and is reused.

The standard model is trained to minimize L_y, in the same way, you would train any convolutional neural network.
All models were first trained on the trainset and evaluated on the test set, then the best epoche of based on the test was chosen, and the final models that were reported in the chapter 4 were trained by combining both the train and the test set and training until the best epoch in order to replicate the method used by Koh et al. (2020).

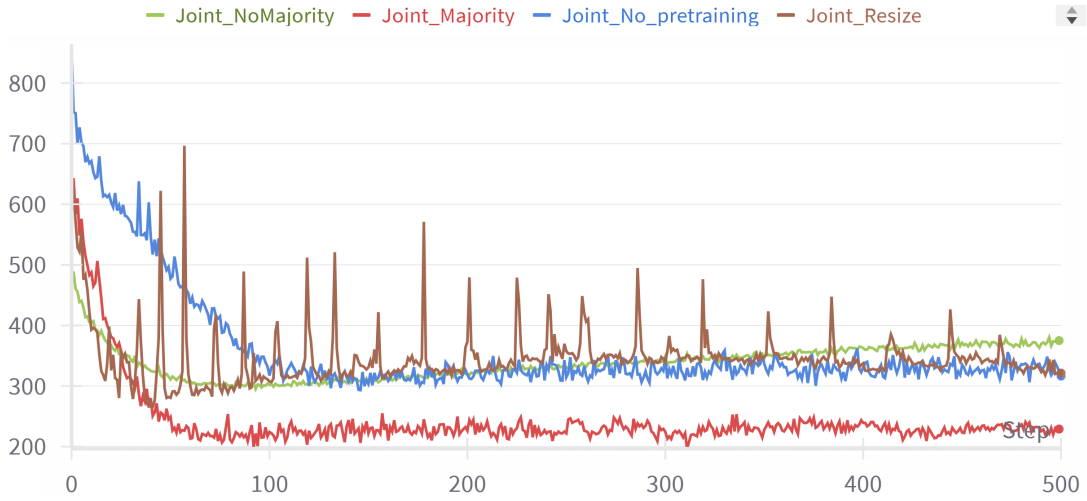
First, an experiment comparing majority-voted and non-majority-voted models was done, and then an experiment without pretraining and using a resize preprocessing step instead of RandomReziseCrop was done. The experiments without pretraining and resizing were also done on a dataset without majority voting.

3.5 Evaluation Criteria

The original paper Koh et al. (2020) only uses prediction error to evaluate both their concept and class; I made the choice to report accuracy instead because I find it more intuitive. For class predictions, I chose to include the accuracy pf top 5 predictions (are the correct class in the 5 classes with highest certainty) to see if the models can guess the type of birds. For example a model could have a very hard time differentiate an American crow from a fish crow, however it should idealy have all types of crows and Ravens in its top 5 predictions.
For concept prediction, most concepts appear very sparsely, meaning a model can get very good accuracy by just guessing all concepts as false. To combat this, I use precision, recall, and F1 scores to evaluate the models concept-predicting abilities. I also evaluate all models on both majority-voted concepts and non-majority-voted concepts.

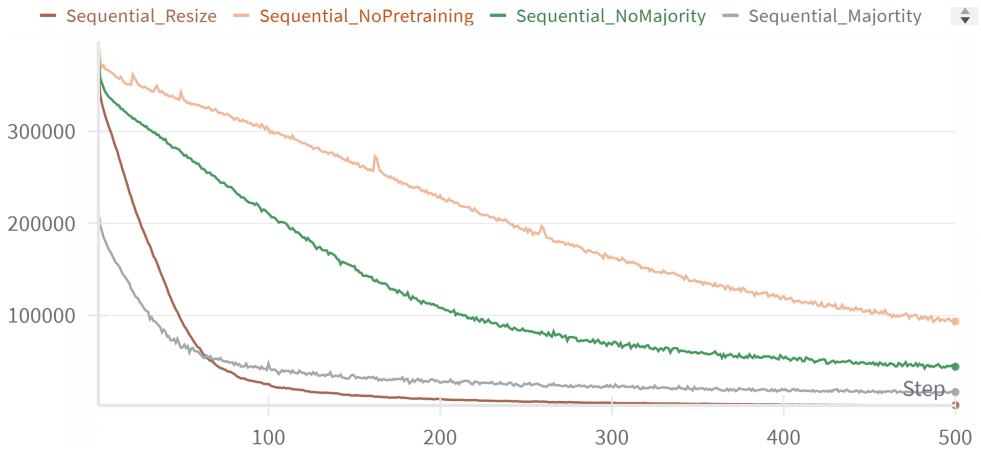


(a) Training loss $L_{joint} = L_Y \cdot \lambda \cdot \sum_j L_{C_j}$ for different joint models.

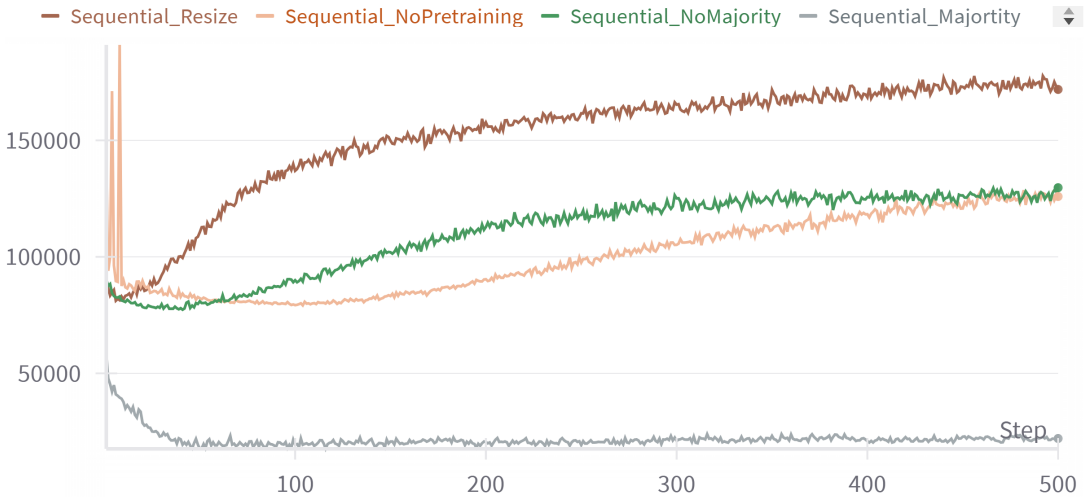


(b) Test loss $L_{joint} = L_Y \cdot \lambda \cdot \sum_j L_{C_j}$ for different joint models.

Figure 3.2: Training and test loss for joint models, note that majority voted model(red) trained more stable and almost doesn't overfit. The model trained using resizes and without majority voting(Brown) seems to memorize the training set without generalizing. The models trained without majority voting(Green) seems to overfit a little bit. The model trained without majority voting and without using a pre-trained model(blue) takes longer to learn but almost catch up to the pre-trained model.



(a) Training loss $\sum_j L_{C_j}$ for X to C models used for Sequential and Independent training.



(b) Test loss $\sum_j L_{C_j}$ for X to C models used for Sequential and Independent training. The majority-voted model (Gray) seems to be training quite well. The models without majority voting seem to overfit a lot, especially the model trained with Resized preprocessing (Brown), which seems to just try to memorize the train set. The model without pretraining(Orange) has some problems converging in the start but catches up to the model just trained without majority voting(Green)

Figure 3.3: Traing and test loss for X to C, The m

Precision tells us the proportion of predicted positive concepts that are actually positive according to the annotated labels:

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Recalls tell how many of the concepts the models were supposed to predict as true were actually predicted as true:

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

The F1 score is meant to balance precision and Recall, thus giving us an idea of how well a model predicts the relevant concepts.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

3.6 Salience score

I implemented a function for making saliency maps using the Captum package Simonyan et al. (2014) in order to replicate the method used by Margeloiu et al. (2021) to illustrate that concept bottleneck models do not focus on the part of the bird where the concept is.

Since I wanted an objective measure of explainability instead of just a few examples, I developed a method to calculate a saliency score.

The score is based on the locations of each part annotated in the CUB dataset (Wah et al., 2011), Parts are linked to concepts in such a way that one part can be the location of multiple concepts; for example, both *has bill color::white* and *has bill shape::dagger* belong to the beak location. Some concepts may also have multiple parts for example, *has wing color::blue* belong to both *Left Wing* and *Right Wing*; a full list of concepts and parts can be found in appendix A. Some parts only have one coordinate (such as only one bill), while other parts have two coordinates (such as left wing and right wing). Thus, let i, j be the index of each pixel and x,y be the coordinates associated with a part.

Calculating the saliency score starts by calculating a distance matrix M that calculates the Manhattan distance to the nearest coordinate of a part.

$$M_{i,j} = \begin{cases} |i - y| + |j - x| & \text{if one coordinate} \\ \min_c \left(|i - y_c| + |j - x_c|\right) & \text{if two coordinate} \end{cases}$$

A normalization value is calculated as the mean of M, where N is the total amount of pixels.

$$m = \frac{\sum_{i,j} M_{i,j}}{N}$$

let S be a normalized saliency map such that the sum of all values is 1 generated based on the gradient of a concept.

$$\sum_{i,j} S_{i,j} = 1$$

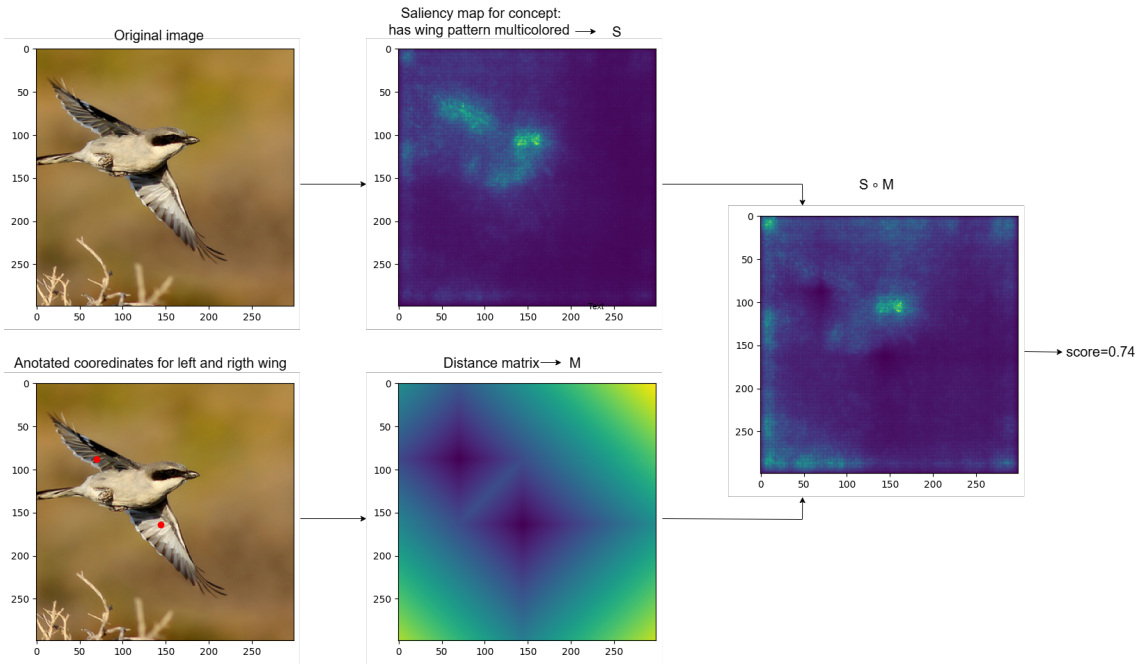


Figure 3.4: Visual example of how saliency score is calculated for the concept: has wing pattern multicolored. Note that the part of the saliency map that attends to the wing doesn't increase the score, while the part that attends to the background increases the score a lot.

The final saliency score can then be calculated as Hadamard's product of the distance matrix and the saliency map and normalized using the normalization values.

$$score = \frac{\sum_{i,j}(M \circ S)_{ij}}{m}$$

A low saliency score indicates that the saliency map is concentrated around the annotated parts. The ideal case is a saliency score of 0, which means that the model focuses solely on the annotated pixels.

A saliency score of 1 corresponds to an evenly distributed saliency map, suggesting that the model does not preferentially focus on any specific part.

A saliency score greater than 1 implies that the model's attention is focused on regions that are, on average, farther from the annotated parts.

The saliency score is highly dependent on the picture and annotation and can thus not be used to compare pictures or tell how well a model performs on a picture; however, for two models evaluated on the same picture, we should assume that the one with the lowest saliency score is the most explainable model.

Only concepts with an annotated part were used to calculate the saliency score, while concepts such as shape or size were ignored. The full list of concepts with a related part can be found in appendix A

4 Results Quantitative.

This Chapter is meant to provide a quantitative evaluation of what models are best in both accuracy and explainability (Saliency score) when evaluated on the entire validation dataset. This chapter also contains evaluations of the original models trained by Koh et al. (2020) for comparison.

4.1 Majority voting VS no majority voting

A central claim in this thesis is that using majority voting is a shortcut that may improve prediction but makes the model more unexplainable. To back this up, this section will test the difference between models trained on a majority-voted dataset and non-majority-voted data.

4.1.1 End classifier baseline

First, a baseline based on the C to Y classifier is established to see how well a class can be predicted, assuming a perfect X to C classifier that predicts all concepts as labeled. The model is trained like the C to Y classifier used for the independent model as described in section 3.4 and evaluated using the true concepts from the validation set the results of this can be seen in table 4.1.

	Majority voted	Non majority voted
accuracy	100%	37%
top 5 accuracy	100%	62%

Table 4.1: Results of a perceptron trained ether a majority voted dataset or a non-majority voted dataset and then evaluated on the same type of dataset

This baseline also indicates the potential impact of performing interventions on all concepts following the method proposed by Koh et al. (2020). It also tell us why accuracy goes down when Shin et al. (2023) performs interventions on a model trained without majority voting.

4.1.2 Test Results for Concept Bottleneck Models Trained with and without Majority Voting

In this section, I present the test results for models trained with and without majority voting. Performance was evaluated using concept accuracy, precision, recall, and F1 score on two validation sets: The validations set annotated (MV) in table 4.2 was subjected to the majority voting and filtering process based on the train set as described in section 3.1.3. The validation annotated (non-MV) in the table 4.2 was not subjected to majority voting and contained the original annotations; however, only the same 112 concepts left after filtering were used for both datasets in order to ensure a fair comparison.
A saliency score was calculated as described in section 3.6 based on a vanilla saliency map and the 112 concepts left after filtering.

The results can be found in table 4.2; the data shows that the models trained on majority-voted data perform better than those trained without it on predicting the class label (Bird prediction), especially for sequential and independent models.

Method	Concept Prediction Metrics					Class Metrics		Saliency Score
	Validation Dataset	Performance Metrics						
		Accuracy	Precision	Recall	F1	Final Acc	Top 5 Acc	
Joint with MV	MV	0.9643	0.9219	0.9023	0.9120	0.7470	0.8997	0.7575
	Non-MV	0.8128	0.6001	0.5295	0.5626			
Joint without MV	MV	0.8827	0.7411	0.6570	0.6965	0.6857	0.9006	0.7814
	Non-MV	0.8242	0.6419	0.5129	0.5701			
Sequential with MV	MV	0.9533	0.8911	0.8799	0.8855	0.6446	0.7607	0.71968
	Non-MV	0.8108	0.5944	0.5291	0.5599			
Sequential without MV	MV	0.8714	0.7312	0.5890	0.6525	0.3795	0.6201	0.7097
	Non-MV	0.8222	0.6501	0.4721	0.5470			
Independent with MV	MV	0.9533	0.8911	0.8799	0.8855	0.6196	0.7273	0.71968
	Non-MV	0.8108	0.5944	0.5291	0.5599			
Independent without MV	MV	0.8714	0.7312	0.5890	0.6524	0.3309	0.6060	0.7098
	Non-MV	0.8222	0.6502	0.4721	0.5470			
Standard	-	-	-	-	-	0.7056	0.9099	-

Table 4.2: Result of models trained with majority voting and without majority voting, and tested set both a majority-voted and non-majority voted validation set. MV indicates majority voting. Due to the filtering of concepts, the concept accuracy and saliency score are only calculated for the 112 concepts left after filtering for all models. For the Saliency Score, lower is better.

For concept prediction, models trained on a majority-voted training set performed significantly better than those not trained on a majority-voted training set when evaluated on a majority-voted validation set. However, models trained without majority voting perform better on the majority-voted validation set than they do on the original validation set. The saliency score indicates that Sequential models may be more explainable than joint models but seem inconclusive regarding how majority voting impacts explainability.

4.2 Results of original models

Since the models trained by Koh et al. (2020) is available online, I downloaded [1] some of their models in order to compare my results to the results. Only models with random seed 1 were tested on my test set. The results of their models are in table 4.3, the method used for testing the models is the same as the method in section 4.1.2, this also means that the results of table 4.2 and 4.3 is comparable.

The results show that all models trained by Koh et al. (2020) perform better than mine. This difference is discussed in section 6.1 however, the original models have higher saliency scores, indicating that they may be less explainable.

4.3 No pre-training and resize transformation

To address the criticism of using RandomResizedCrop and using pre-trained network models trained Other variables include training the network without loading a pre-trained checkpoint and training on a resize transformation instead of a RandomReziseChop transformation. The results of those models can be seen in table 4.4. The results in this section are validated on datasets with all 312 concepts, meaning that the dataset annotated MV in table 4.4 is only majority voted but did not have the filtering step described in section 3.1.3, the dataset annotated Non-MV is the original CUB Validation set. A saliency score was

[1] worksheets.codalab.org/worksheets/0x362911581fcd4e048ddfd84f47203fd2

Method	Concept Prediction Metrics					Class Metrics		Saliency Score
	Validation Dataset	Performance Metrics						
		Accuracy	Precision	Recall	F1	Final Acc	Top 5 Acc	
Joint Model	MV	0.8588	0.7290	0.4953	0.5899	0.8305	0.9615	0.7957
	Non-MV	0.7892	0.5596	0.3428	0.4251			
Sequential Model	MV	0.9677	0.9405	0.8993	0.9194	0.7573	0.8832	0.8771
	Non-MV	0.8144	0.6065	0.5228	0.5615			
Independent Model	MV	0.9677	0.9405	0.8993	0.9194	0.7428	0.8745	0.8771
	Non-MV	0.8144	0.6065	0.5228	0.5615			
Standard Model	MV	-	-	-	-	0.8243	0.9553	-
	Non-MV	-	-	-	-			

Table 4.3: Results of models downloaded from Codalab. MV indicates a majority-voted test set, and Non-MV indicates a non-majority-voted test set. For the Saliency Score, lower is better.

calculated as described in section 3.6 based on a vanilla saliency and all concepts with annotated coordinates. Note that since saliency score and Concept prediction metrics were calculated using different numbers of concepts, the results in table 4.2 and table 4.4 is not directly comparable, this is also why models trained without majority voting (but no other intervention) is present in both table despite being the same model that is evaluated.

Method	Validation Dataset	Concept Prediction Metrics				Class Metrics		Saliency Score
		Performance Metrics						
		Accuracy	Precision	Recall	F1	Final Acc	Top 5 Acc	
Joint without MV	MV	0.9508	0.7040	0.6561	0.6792	0.6857	0.9006	0.7772
	Non-MV	0.9169	0.6214	0.4547	0.5251			
Joint No pretraining	MV	0.9506	0.7012	0.6571	0.6785	0.6590	0.8828	0.7233
	Non-MV	0.9165	0.6181	0.4548	0.5240			
Joint Resize	MV	0.9422	0.6437	0.6082	0.6254	0.5167	0.7915	0.7113
	Non-MV	0.9124	0.5897	0.4375	0.5023			
Sequential without Majority	MV	0.9476	0.7065	0.5819	0.6382	0.3795	0.6201	0.7033
	Non-MV	0.9167	0.6357	0.4110	0.4993			
Sequential No pretraining	MV	0.9448	0.6960	0.5409	0.6087	0.34915	0.5635	0.7534
	Non-MV	0.9155	0.6340	0.3869	0.4805			
Sequential Resize	MV	0.9430	0.7275	0.4513	0.5571	0.1964	0.4418	0.6604
	Non-MV	0.9133	0.6455	0.3144	0.4229			
Independent without MV	MV	0.9476	0.7065	0.5819	0.6382	0.3309	0.6060	0.7033
	Non-MV	0.9167	0.6357	0.4110	0.4993			
Independent No Pretraining	MV	0.9448	0.6960	0.5409	0.6087	0.3288	0.5739	0.7534
	Non-MV	0.9155	0.6340	0.3869	0.4805			
Independent Resize	MV	0.9430	0.7275	0.4513	0.5571	0.1255	0.3101	0.6604
	Non-MV	0.9133	0.6455	0.3144	0.4229			

Table 4.4: Results of models trained with resize transformation and without pretraining. MV indicates a majority-voted test set, and Non-MV indicates a non-majority-voted test set. For the Saliency Score, lower is better.

5 Qualitative results

In this section, I select a few images and evaluate a few models on them. I also present saliency maps and individual concept predictions. The purpose of the section is to demonstrate examples of deceptive explainability and analyze how models use their concepts to make predictions
The examples presented in this chapter are chosen to demonstrate some points and are not necessarily representative of the entire dataset. A more representative evaluation can be found in appendix A.3, where I present randomly chosen images classified by randomly chosen models and present 3 randomly chosen saliency maps and 30 random concept predictions for each image.

5.1 Effect on easy prediction

The first qualitative test involves an archetypical male Mallard (validation set ID 2463) The image can be seen in figure 5.1. This example is chosen because it is so straightforward that all models can predict the right class with good confidence. All concepts are also very visible except the legs and belly.

In table 5.1, the true concepts for both the majority-voted and the original datasets and the prediction made by the X to C models are listed for a few selected concepts. The first thing worth noticing is that the models trained on the majority-voted dataset are extremely confident in their prediction, with all concepts being either almost 1 or almost 0. The model's prediction also precisely matches the Majority-voted labels, strongly supporting the hypothesis that the model first predicts the class and then maps the predictions to the concepts. This is further supported by the model being willing to make predictions about belly patterns as solid despite not being able to see the belly.

In table 5.2, the model is trained using a Resize transformation, and the model does not use a pre-trained network. We see that both models are far less certain than the majority-voted models. For the models trained on the non-majority-voted dataset, the predictions are more balanced for cases where there may be doubt about the true label.

The saliency map for the X to C models used in sequential and independent training can be seen in figure 5.1. The saliency maps were made by Noisetunnel with 50 samples and 0.2 std as described in section 2.5.1. The saliency score was calculated as described in section 3.6. What is worth noticing is that in all models, the saliency does not focus on the annotated spot. We also see that no matter what concept, the saliency maps produced by a model seem very similar, indicating that the model is predicting the bird first. This also matches the finding by Margeloiu et al. (2021) that concept bottleneck models don't learn as intended. While the model trained without majority voting arguably is a little better, it does not seem to solve the problem of training without majority voting.

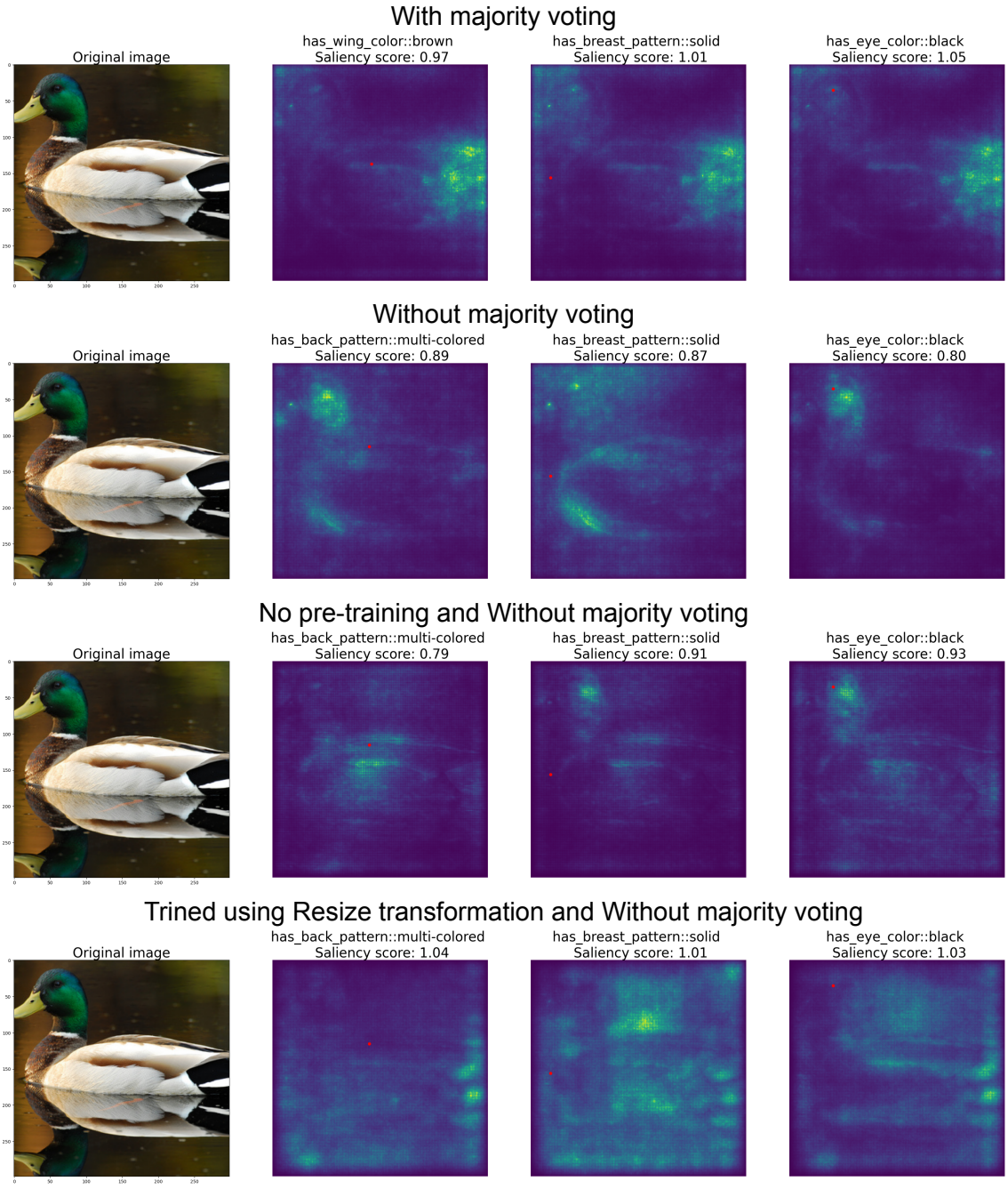


Figure 5.1: Saliency maps for some selected concepts saliency map on image 2463. The top row is for an X to C model (Sequential). The concept predicted is that it has breast color: Brown, Wing color: white, and bill length: about the same as the head. The prediction for all those concepts can be found in table 5.1. The red dot indicates the annotated coordinate of the part associated with the concept.

Concept	True Label		Joint Model		Sequential Model		Distribution	
	Non-MV	MV	MV	Non MV	MV	Non MV	Non MV	MV
wing color: brown	T	F	0.000	0.620	0.000	0.570	0.270	0.245
wing color: buff	T	F	0.000	0.450	0.000	0.470	0.182	0.130
upperparts color: brown	F	T	1.000	0.640	1.000	0.670	0.243	0.230
upperparts color: black	T	F	0.000	0.540	0.000	0.630	0.391	0.410
breast pattern: solid	F	T	1.000	0.490	1.000	0.380	0.548	0.645
breast pattern: striped	T	F	0.000	0.010	0.000	0.050	0.117	0.075
back color: white	T	F	0.000	0.680	0.000	0.280	0.154	0.120
back color: buff	T	F	0.000	0.530	0.000	0.520	0.146	0.110
upper tail color: white	T	F	0.000	0.230	0.000	0.580	0.124	0.120
eye color: black	F	T	1.000	0.980	1.000	0.960	0.837	0.960
forehead color: blue	T	F	0.000	0.010	0.000	0.010	0.051	0.050
under tail color: white	F	T	1.000	0.740	1.000	0.650	0.179	0.160
back pattern: multi-colored	F	T	1.000	0.740	1.000	0.410	0.195	0.070
belly pattern: solid	F	T	1.000	0.170	1.000	0.350	0.573	0.745
primary color: white	T	F	0.000	0.670	0.000	0.550	0.290	0.205
primary color: buff	T	F	0.000	0.130	0.000	0.190	0.192	0.090
crown color: blue	T	F	0.000	0.020	0.000	0.050	0.055	0.050

Table 5.1: Some selected concepts prediction for majority voted and non-majority voted models on image 2463 (see the image in figure 5.1) concepts are selected based on the place where the annotated label disagree with the majority label. T and F stand for True False or a label of 0 and 1. NoN-MV is the non-majority-voted dataset, and MV is the voted dataset for true labels and label distribution, while for models, MV and NoN-MV refer to whether a model was trained on a majority-voted dataset or not. The distribution is how big a percentage of the data is true for each concept(baseline for a model that always guesses true)

Concept	True Label		Joint Model		Sequential Model		Distribution
	Original	Majority	Resize	NoPretraining	Resize	NoPretraining	
has wing color: brown	True	False	0.22	0.70	0.15	0.68	0.27
has wing color: buff	True	False	0.26	0.59	0.13	0.60	0.18
has upperparts color: brown	False	True	0.25	0.72	0.13	0.63	0.24
has upperparts color: black	True	False	0.33	0.63	0.27	0.74	0.39
has breast pattern: solid	False	True	0.42	0.37	0.57	0.49	0.55
has breast pattern: striped	True	False	0.28	0.11	0.07	0.06	0.12
has back color: white	True	False	0.39	0.37	0.16	0.82	0.15
has back color: buff	True	False	0.25	0.58	0.08	0.58	0.15
has tail shape: fan-shaped tail	True	False	0.21	0.26	0.16	0.13	0.08
has tail shape: squared tail	False	True	0.33	0.39	0.11	0.52	0.09
has upper tail color: white	True	False	0.31	0.29	0.12	0.59	0.12
has eye color: brown	True	False	0.03	0.01	0.04	0.00	0.03
has eye color: black	False	True	0.86	0.80	0.83	0.94	0.84
has forehead color: blue	True	False	0.52	0.11	0.10	0.02	0.05
has under tail color: white	False	True	0.49	0.52	0.25	0.76	0.18
has nape color: blue	True	False	0.58	0.14	0.09	0.07	0.05
has nape color: green	True	False	0.81	0.60	0.13	0.78	0.01
has back pattern: multi-colored	False	True	0.48	0.84	0.32	0.80	0.19
has belly pattern: solid	False	True	0.28	0.36	0.56	0.34	0.57
has primary color: white	True	False	0.30	0.47	0.39	0.66	0.29
has primary color: buff	True	False	0.13	0.20	0.13	0.15	0.19
has leg color: orange	False	True	0.40	0.63	0.13	0.77	0.08
has crown color: blue	True	False	0.62	0.21	0.10	0.17	0.05

Table 5.2: Some selected concepts prediction for models trained with resize transformation or without pertaining on image 2463 (see the image in figure 5.1) concepts are selected based on the place where the annotated label disagree with the majority label. T and F stand for True False or a label of 0 and 1. The distribution is the percentage of the data that is true for each concept.

5.2 Effect on prediction on Manipulated image

In this section, a heavily manipulated image of a Mallard (validation set ID 2463)(Same image as in section 5.1 on a heavily manipulated image. What is done is that most of the image was cropped out and replaced with a black background, and the eyes were colors yellow, the results of this manipulation can be seen in figure: 5.2. After the manipulation, all models still predict the correct class except the Sequential Resized model that predicts the class to be a Rufous Hummingbird. A few selected concepts predicted by each model can be found in table 5.3 and 5.4. Here, the models trained with majority voting again overconfidently predict concepts that have been deleted; however, the models trained without majority voting also predict concepts that have been deleted. We also see that all models still predict the Eye color to be black instead of red; however, 84% birds have black eyes, indicating that none of the models have learn this concept but instead just guess on the baseline.

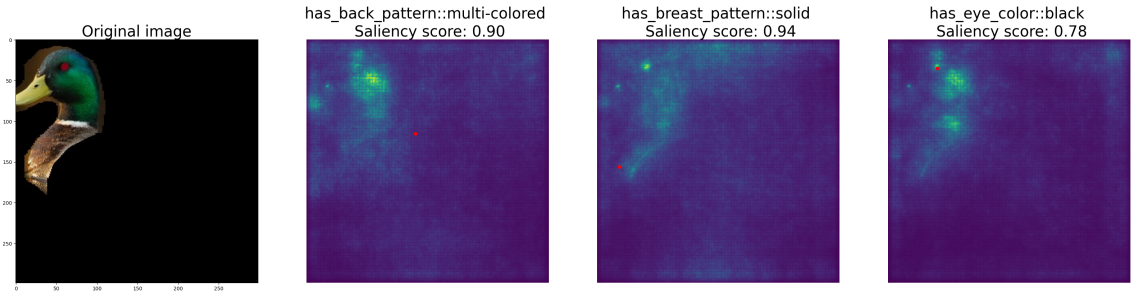


Figure 5.2: The manipulated image of a Mallard and saliency maps for a model trained without majority voted concepts.

Concept	True Label		Joint Model		Sequential Model		Distribution	
	Non-MV	MV	MV	Non MV	MV	Non MV	Non MV	MV
has back pattern: multi-colored	False	True	1.00	0.47	1.00	0.39	0.19	0.07
has wing color: white	True	True	1.00	0.60	1.00	0.36	0.26	0.20
has breast pattern: solid	False	True	1.00	0.52	1.00	0.43	0.55	0.65
has wing pattern: multi-colored	True	True	1.00	0.64	1.00	0.62	0.30	0.22
has eye color: black	False	True	1.00	0.88	1.00	0.92	0.84	0.96
has under tail color: white	False	True	1.00	0.40	0.99	0.38	0.18	0.16

Table 5.3: Some selected concept predictions for models trained with and without majority voting on a manipulated image in figure 5.2. MV stands for majority voting.

Concept	True Label		Joint Model		Sequential Model		Distribution
	Original	Majority	Resize	NoPretraining	Resize	NoPretraining	
has back pattern: multi-colored	False	True	0.46	0.47	0.18	0.50	0.19
has wing color: white	True	True	0.34	0.18	0.14	0.40	0.26
has breast pattern: solid	False	True	0.44	0.51	0.49	0.43	0.55
has wing pattern: multi-colored	True	True	0.51	0.62	0.23	0.62	0.30
has eye color: black	False	True	0.77	0.72	0.84	0.84	0.84
has eye color: red	False	False	0.04	0.05	0.02	0.01	0.02
has nape color:green	True	False	0.54	0.47	0.05	0.76	0.01

Table 5.4: Some selected concept prediction for models trained with resize transformation or without pertaining on a manipulated image in figure. 5.2

5.3 Learning wrong concepts

The second qualitative test shows a mallard whose bill is being chopped off in the centre chop preprocessing step for the validation set, and the picture can be seen in figure 5.3. Yet, in table 5.5, both a joint and an independent model trained without majority voting will predict the bill shape to be spatulate with fairly high certainty. furthermore, we see that the model is almost just as certain of other concepts related to a Mallard, such as duck-like shapes, supporting the hypothesis that even models trained without majority voting still make the class prediction before the concept prediction. However, what counts against this hypothesis is that only the joint model predicts the bill colour to be yellow (with low certainty), and both models fail to predict the leg colour orange despite the leg colour being just as important for classifying Mallards according to the weights of the C to Y model. Another theory could be that only water birds have a spatial-shaped bill, and thus, this concept indicates the presence of water instead of the actual bill shape. The saliency maps in figure 5.3 may partially support this theory, but fundamentally, these are just speculations that indicate that it is next to impossible to explain why the model predicted this concept or how much leakage is in this concept, bringing the explainability of concept bottleneck models into question.

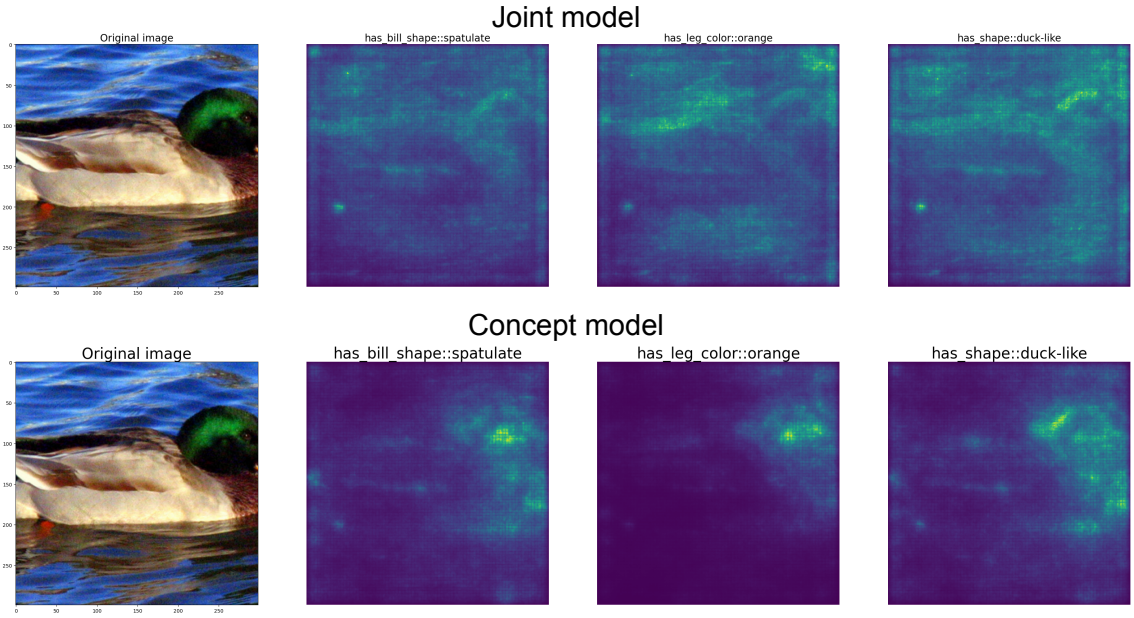


Figure 5.3: Mallard, whose bill was cut out of the image due to center cropping on the test set. The picture ID is 2455. The top row is saliency maps, the joint model trained without majority voting, and the bottom row is the concept model (X to C model) trained without majority voting. The saliency maps are made using a noise tunnel with 50 samples and a 0.2 standard deviation. The concepts chosen have bill shape spatulate, leg color orange, and shape duck-like. The prediction made by those models can be seen in table 5.5.

Concept	True Label		Joint		Independent		Distribution	
	Non-MV	MV	Prediction	Weight	Prediction	Weight	Non MV	MV
bill shape: spatulate	T	T	0.960	1.391	0.630	7	0.040	0.020
bill color: yellow	T	T	0.510	1.190	0.170	6	0.057	0.015
bill length: same as head	T	T	0.680	0.337	0.450	-1	0.362	0.340
forehead color: green	T	F	0.250	0.700	0.060	1	0.009	0.000
size: medium (9 - 16 in)	T	T	0.900	0.654	0.630	0	0.196	0.185
shape: duck-like	T	T	0.930	1.096	0.690	2	0.061	0.050
leg color: orange	F	T	0.330	1.112	0.110	10	0.083	0.045
breast color: brown	T	T	0.440	0.736	0.370	8	0.114	0.065

Table 5.5: Table of results from selected concepts predicted on the mallard seen in figure 5.1. The model tested was the joint and independent model trained without majority voting and tested on all 312 concepts. The prediction is what the X to C part of the model predicts. The weights are the weights of the C to Y model from the concepts to the true class (Mallard), telling how important the concept is for the model in predicting the true class. The true label is True or False, with Non-MV being the annotated label and MV being the label after majority voting.

5.4 Misclassification for female bird

A major critique of majority voting is that it sets the true concepts for female birds to the true concepts of male birds (if male birds are the majority). In this section, two examples of when this happens are investigated. The first example is a female painted bunting as seen in figure 5.4b, table 5.7 shows how different models misclassify this Painted Bunting as an Orange Crowned Warbler.

(a) Painted Bunting male (b) Painted Bunting female (c) Orrange Crowned Warbler

Figure 5.4: Example of how models can be confused by concepts when the female painted Bunting looks more like an Orange Crowned Warbler than the male painted bunting that makes up the majority of the dataset. The Painted Bunting in the middle is the bird classified in this section.

Prediction	Joint MV	Joint No MV	Sequential MV	Sequential No MV
Painted Bunting	0.0101	0.0108	$4 \cdot 10^{-9}$	10^{-6}
Orange Crowned Warbler	0.0101	0.0108	0.99	0.86

Table 5.6: Model prediction of Painted Bunting (True class) and Orange Crowned Warbler (Wrong class)

Table 5.7 shows how each model is predicting concepts for image 391 seen in 5.4b. All models predict that the bird has a yellow breast and yellow belly, even though this is false according to the majority voting, however, the Joint model trained on the majority-voted dataset gives at least some probability that the head and crown may be blue, which can be interpreted as an example of leakage.

Concept	True Label	Painted Bunting	Orrange Warbler	Joint Model		Sequential Model		Distribution	
				MV	Non MV	MV	Non MV	Non MV	MV
has bill shape all-purpose	False	False	True	0.570	0.620	0.940	0.690	0.392	0.405
has bill shape cone	True	True	False	0.720	0.230	0.020	0.180	0.261	0.245
has breast color yellow	True	False	True	0.890	0.790	1.000	0.830	0.137	0.145
has fore-head color blue	False	True	False	0.400	0.000	0.010	0.000	0.051	0.050
has belly color yellow	True	False	True	0.900	0.620	1.000	0.770	0.144	0.145
has back pattern solid	False	False	True	0.010	0.450	0.630	0.310	0.365	0.495
has back pattern multi-colored	True	True	False	1.000	0.320	0.400	0.250	0.195	0.070
has tail pattern multi-colored	False	False	True	0.020	0.200	0.590	0.260	0.228	0.130
has leg color grey	True	True	False	0.780	0.290	0.050	0.310	0.256	0.225
has crown color blue	False	True	False	0.400	0.010	0.010	0.000	0.055	0.050
has wing pattern multi-colored	True	True	False	1.000	0.270	0.360	0.330	0.297	0.220

Table 5.7: Selected concept predicted for image 391 by different models. The Painted Bunting column is the majority-voted label of a Painted Bunting, and the Orange Warbler column is the majority-voted concept for Orange Crowned Warbler; concepts are selected where the two columns are not the same. MV and NoN-MV indicate that a model was trained on a majority-voted dataset.

6 Discussion

6.1 Lack of accuracy

The biggest frustration working on this thesis is that I was unable to replicate the accuracy Koh et al. achieved on the class predictions, with my models consistently performing 10% worse than the original models on the same task (training with majority voting).

The choice of hyperparameter is probably the most influential factor in this effect. My hyperparameters were primarily chosen after what worked on the Non-majority-voted dataset trained using Resized instead of RandomResizedCrop. The hyperparameter I chose thus makes all models converge even though some are not very good, especially training with resized images, which proved very challenging since the model always overfitted without sufficient data augmentation.

Furthermore, Koh et al. (2020) makes a hyperparameter search for each type of model; this was not realistic for me to do since I had more models and less access to GPU resources. Thus, I ended up finding one setting for all models, this may not have been the optimal strategy, for example, when tested on the test set using Stochastic gradient descent seemed to work better than ADAM on the majority voted models but worse on non-majority voted models, furthermore in section 3.3.1 I show that performance of the Majority voted models can be boosted using concept weighting and may even give better results than the original paper if that was the goal of the thesis.

6.2 Consequences of Majority voting

The C to Y model baseline in 4.1 suggests that Majority-voted concepts are a shortcut that improves accuracy by creating an artificially easy classification problem for the C to Y classifier which is why the baseline accuracy for the majority-voted dataset is 100%
For joint models, the loss of accuracy due to not using majority voting is not that bad (especially for top-five accuracy). The joint model also performs a lot better than the baseline of 37% accuracy baseline for Majority models, indicating a lot of leakages; this would also support the finding by Shin et al. (2023) that performing an intervention on Joint models trained without majority voting would decrease accuracy.

For sequential and independent models, not doing majority voting brings a model down to the baseline of an independent model on the true concepts of 37% accuracy.

Interestingly, even models trained without majority voting predict the majority-voted concept better than the original concepts when evaluated on a majority-voted and non-majority-voted validation set, indicating that all models perform some form of internal majority voting or noise reduction.

A problem with the dataset is that a model in the concepts has too much noise to be truly useful for a concept bottleneck model, but by doing majority voting, we are both removing the annotation noise and the real-world noise (often, the concepts are not visible). In the further work section 6.7, I discuss other preprocessing steps that may reduce and make the concepts better for predicting without giving more information than what is visible in the image.

6.3 Consquences of pre-training

Not using a pre-trained network makes the model a little less accurate in all prediction matrices but not much compared to the model trained with a pre-trained network. In the training section 3.4, the model without pre-training takes longer to converge but ends up around the same place as the other model trained on a majority-voted dataset and pre-trained on Imagenet.
The saliency score is better for the joint models and worse for the sequential and independent ones; thus, while there may be theoretical arguments against using a pre-trained network, I don't see any data that suggests that not using a pre-trained network makes the model more explainable.

6.4 Consequences of RandomResizeCrop

While the slightly lower saliency score may indicate that a model train without RandomResizeCrop may be more explainable than the model trained on a Resized image, the Resizede models also performed far worse than the non-Resized models. Furthermore, the loss curves in section 3.4 show that it is hard to train a model without overfitting in the current training setup.
This also means that it is hard to tell if the explanations are deceptive or just badly predicted, making it hard to make any conclusion on the explainability of models trained without RandomResizeCrop.

6.5 Soft labels

My results indicate that a joint model is better than a standard model when trained with majority-voted data. While Koh et al. (2020) don't claim this, it is worth noticing that they include a figure that indicates that for the right value of lambda, Joint models outperform standard models as seen in figure 6.1. This goes against the theoretical argument that concept bottleneck models have trade-offs between being explainable or accurate.

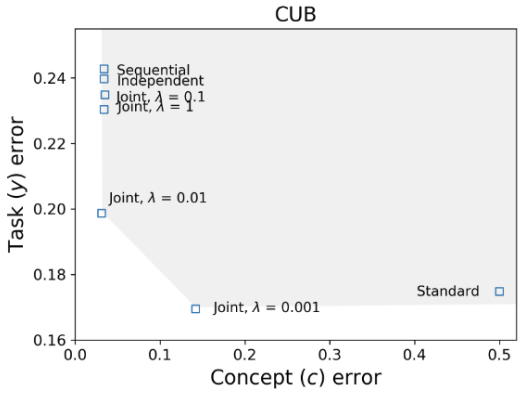


Figure 6.1: Figure from Koh et al. (2020) telling the concept error versus the task error for different values of λ note that for $\lambda = 0.001$ concept task error is lower than the task error for the standard model

I would explain this phenomenon by pointing out that a concept layer can be seen as a soft label. Soft labels are a way to increase a model's robustness and accuracy by using labels given as a percentage instead of binary. A way to make a soft label is to give the label the probability of how many annotators believe the label to be of that class if multiple annotators are used (Peterson et al., 2019) or a soft label can be generated by asking an annotator how certain they are (Nguyen et al., 2014). **?** also suggests a version of

soft labels called label smoothing, where adding a bit of random noise to the labels can improve accuracy.
For concept bottleneck models, the concept vector is a soft label, and the similarity between two concept vectors is a probability.

For example, a standard model would get the same loss for misclassifying an American crow as a Mallard as it would get for misclassifying it as a fish crow. However, for a model that includes the majority of concepts in its loss function, the fish crow misclassification would only get 2 concepts wrong. In comparison, the mallard misclassification would get 28 concepts wrong, leading to the mallard misclassification getting a much higher loss. Thus, by training on an American crow, the model can learn features that may help it classify all crow-like birds.

Thus, the concept layer acts as soft labels, making the model more robust. This would explain why concept-based models perform better when validated on a dataset with a shifting background or when trained on less data, as demonstrated by Koh et al. (2020).

6.6 Fairness of Concept bottleneck models

In this section, I discuss the fairness of CBMs. In their discussion Shin et al. (2023) points out that majority voting discriminates against minorities since concepts unique to a minority inside a class are removed, for example, in the CUB dataset, male birds are often more colourful than females making them more favoured by photographers and thus overrepresented in the dataset, this means that female birds are often annotated wrong.
In this thesis, I think there is some evidence that even without majority voting, models still discriminate against female birds. The first way a model may discriminate is by guessing the correct bird but then telling the concepts related to the male bird. However, I did not find any evidence for this. The second way a model may discriminate is by classifying the female bird as a different type of bird and then coming up with the concepts to justify this decision as seen in section 5.4 where the models classify the female Painted Bunting as an Orange Warbler and then say that the Bunting has all-purpose bill shape of the warbler despite this being false.
In other hypothetical applications of concept bottleneck models, their ability to appear explainable may also be used to justify discrimination. A thought example could be a hiring algorithm that maps a CV stating that the applicant was a member of the Boy Scouts to the concept of leadership experience. In contrast, an otherwise equal applicant who was a member of the Girl Scouts would get a lower leadership score, leading to the company hiring the male applicant. If the woman then seeks insight into why she wasn't hired, the company can excuse itself by stating that she didn't have the leadership experience they were looking for, thus avoiding a lawsuit despite discriminating against females.

Another example could be a member of a minority group receiving the wrong medical treatment as a result of a CBM misdiagnosis. If the model is a black box, the patient would have a good legal case against the hospital. However, if the patient receives 310 predictions of Latin-named concepts that all may and may not contain racial bias, it would be much harder (and more expensive) for the patient to prove which concepts were predicted wrong.

6.7 Future work

6.7.0.1 An explainable model on the CUB dataset

While none of the models I trained seem to be truthfully explainable or very accurate, it may still be possible to train a concept bottleneck model on the CUB dataset (Wah et al.,

2011) by using some of the other features of the CUB dataset.
One feature is that the annotator was asked to state how certain they were in their concepts; if this certainty was turned into a soft label, it should be possible to make a concept space that is better for predicting birds.
Another problem is that RandomResizeCrop is the correct way to train an Inception network (PyTorch Team, 2015)(Cui et al., 2018) and without it the model overfits a lot, the easiest way to get around this would be to use a different CNN to predict concepts, ResNet seems most obvious as it is used by Koh et al. (2020) on the OAI dataset. The coordinates of each concept could also be used to make sure that the model is only trained to predict the concept actually present in the cropped image; this may allow RandomResizedCropping to learn all concept representations faithfully. This should have a similar effect as masking the concepts as shown by Lin et al. (2022) and discussed in section 1.4.1. The provided mask of the entire bird by the CUB dataset may also be used to avoid model shortcuts based on the background. A perceiving model could also be used to mask concepts.

Using a more complicated C-to-Y model than a perception may also improve the accuracy of CBMs. I did make a mini experiment on the test set where I set up a small neural network (two layers of each 200 neurons) following the same methods as Shin et al. (2023) on the baseline prediction in section 4.1.1 and only got 42% indicating an improvement over the 37% baseline provided by the perceptron but not much.

6.7.0.2 Interpretability score

What seems to be a problem across the literature on concept bottleneck models is that there is no objective measurement of how explainable a concept bottleneck model is. I attempt to solve this by implementing a distance-based saliency score, and Huang et al. (2024) makes a score based on a bounding box and saliency map.
Both those ideas would probably need more development in order to be a reliable measurement; for example, for a saliency score, it would be worth establishing if a Euclidean distance would be better than a Manhattan distance, using a Grad Cam saliency (Selvaraju et al., 2019) map may also give better results.

6.7.0.3 Simulated data

A big problem with real-life datasets is the concept noise; this means that we can not be sure if the label is wrong when evaluating models quantitatively; simulating data thus may be a way to test the effect of majority voting, especially to detect if models learn concept representation or class representation.
Shin et al. (2023) provides an example of simulating a concept annotated dataset exploring adding noise and how a model performs if relevant concepts are not annotated. This could further be expanded upon by simulating a dataset with different correlations between x, c, and y to show when/ if a CBM would predict a concept not present in x.

7 Conclusion

In this thesis, I present evidence for Hypothesis 1, which states that the model predicts class instead of concept. One example of this can be section 5.1 where I show that if a model is certain of the final label it is overconfident in predicting concepts I also show that models trained with majority voting will predict the majority voted concepts even if they are not visible or the concept (as annotated) is different than the majority voted.

I confirm the first of my criticisms that majority voting makes the model unexplainable by predicting concepts with high certainty even when they are not present in the data.

When majority voting is not applied in the training models, the predictions seem less certain but still predict concepts that are not present in the data.

My criticism of using a pre-trained model remains theoretical since I could not find any evidence that a model trained without pre-training is any more explainable than an otherwise equal model that is pre-trained on ImageNet.

I try to address the problem of using RandomResizeCrop and find some evidence that a model trained using Resize instead is more explainable; however, training a model without RandomResizeCrop makes the model overfit easily in the current setup and perform so poorly that it is hard to conclude anything.

Overall, I show that in some cases concept bottleneck models don't learn as intended but instead learn a class representation and use that to predict concepts. This makes their explainability deceptive since concepts are just used to justify a prediction made by a black box model.

Bibliography

Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Pietro Lió, Marco Gori, and Stefano Melacci. Entropy-based logic explanations of neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6046–6054, June 2022. ISSN 2159-5399. doi: 10.1609/aaai.v36i6.20551. URL http://dx.doi.org/10.1609/aaai.v36i6.20551.

Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning, 2018. URL https://arxiv.org/abs/1806.06193.

Shantanu Ghosh, Ke Yu, Forough Arabshahi, and Kayhan Batmanghelich. Dividing and conquering a blackbox to a mixture of interpretable models: Route, interpret, repeat, 2023. URL https://arxiv.org/abs/2307.05350.

Pei Guo. Overlap between imagenet and cub, 2016. URL https://guopei.github.io/2016/Overlap-Between-Imagenet-And-CUB/.

Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022a. URL https://openreview.net/forum?id=tglniD_fn9.

Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022b. URL https://openreview.net/forum?id=tglniD_fn9.

Qihan Huang, Jie Song, Jingwen Hu, Haofei Zhang, Yong Wang, and Mingli Song. On the concept trustworthiness in concept bottleneck models, 2024. URL https://arxiv.org/abs/2403.14349.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models, 2020. URL https://arxiv.org/abs/2007.04612.

Manxi Lin, Aasa Feragen, Zahra Bashir, Martin Grønnebæk Tolsgaard, and Anders Nymark Christensen. I saw, i conceived, i concluded: Progressive concepts as bottlenecks, 2022. URL https://arxiv.org/abs/2211.10630.

Anita Mahinpei, Justin Clark, Isaac Lage, Finale Doshi-Velez, and Weiwei Pan. Promises and pitfalls of black-box concept learning models, 2021. URL https://arxiv.org/abs/2106.13314.

Andrei Margeloiu, Matthew Ashman, Umang Bhatt, Yanzhi Chen, Mateja Jamnik, and Adrian Weller. Do Concept Bottleneck Models Learn as Intended?, 2021. URL http://arxiv.org/abs/2105.04289.

Authors C Michael Nevitt, David T Felson, and Gayle Lester. Oai protocol the osteoarthritis initiative protocol for the cohort study.

Quoc Nguyen, Hamed Valizadegan, and Milos Hauskrecht. Learning classification models with soft-label information. *Journal of the American Medical Informatics Association : JAMIA*, 21(3):501–508, 2014. doi: 10.1136/amiajnl-2013-001964. URL https://doi.org/10.1136/amiajnl-2013-001964.

Tuomas Oikarinen, Subhro Das, Lam M. Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models, 2023. URL https://arxiv.org/abs/2304.06129.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Joshua C. Peterson, Ruairidh M. Battleday, Thomas L. Griffiths, and Olga Russakovsky. Human uncertainty makes classification more robust, 2019. URL https://arxiv.org/abs/1908.07086.

PyTorch Team. Inception_v3, 2015. URL https://pytorch.org/hub/pytorch_vision_inception_v3/. Accessed: 2025-01-11.

PyTorch Team. *RandomResizedCrop — Torchvision main documentation*, 2017. URL https://pytorch.org/vision/main/generated/torchvision.transforms.RandomResizedCrop.html. Accessed: 2025-01-14.

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. *International Journal of Computer Vision*, 128(2):336–359, October 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL http://dx.doi.org/10.1007/s11263-019-01228-7.

Ivaxi Sheth and Samira Ebrahimi Kahou. Auxiliary losses for learning generalizable concept-based models, 2023. URL https://arxiv.org/abs/2311.11108.

Sungbin Shin, Yohan Jo, Sungsoo Ahn, and Namhoon Lee. A closer look at the intervention procedure of concept bottleneck models, 2023. URL https://arxiv.org/abs/2302.14260.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014. URL https://arxiv.org/abs/1312.6034.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise, 2017. URL https://arxiv.org/abs/1706.03825.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015. URL https://arxiv.org/abs/1512.00567.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. Caltech-ucsd birds-200-2011. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Yue Yang, Artemis Panagopoulou, Shenghao Zhou, Daniel Jin, Chris Callison-Burch, and Mark Yatskar. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification, 2023. URL https://arxiv.org/abs/2211.11158.

A List of parts and concepts

Table of all concepts and what part they belong to; concepts in bold are concepts still present after the filter step, while concepts in italics are not present in the dataset after majority voting and filtering.

Part	Attributes / Concept name
Back	*has_back_color::blue* **has_back_color::brown** *has_back_color::iridescent* *has_back_color::purple* *has_back_color::rufous* **has_back_color::grey** **has_back_color::yellow** *has_back_color::olive* *has_back_color::green* *has_back_color::pink* *has_back_color::orange* **has_back_color::black** **has_back_color::white** *has_back_color::red* **has_back_color::buff** **has_back_pattern::solid** **has_back_pattern::spotted** **has_back_pattern::striped** **has_back_pattern::multi-colored**
Bill	*has_bill_color::blue* *has_bill_color::brown* *has_bill_color::iridescent* *has_bill_color::purple* *has_bill_color::rufous* **has_bill_color::grey** **has_bill_color::yellow** *has_bill_color::olive* *has_bill_color::green* *has_bill_color::pink* *has_bill_color::orange* **has_bill_color::black** **has_bill_color::white** *has_bill_color::red* **has_bill_color::buff** *has_bill_shape::curved_(up_or_down)* **has_bill_shape::dagger** *has_bill_shape::hooked* *has_bill_shape::needle* **has_bill_shape::hooked_seabird** *has_bill_shape::spatulate* **has_bill_shape::all-purpose** **has_bill_shape::cone** *has_bill_shape::specialized* **has_bill_length::about_the_same_as_head** **has_bill_length::longer_than_head** **has_bill_length::shorter_than_head**
Belly	*has_belly_color::blue* **has_belly_color::brown** *has_belly_color::iridescent* *has_belly_color::purple* *has_belly_color::rufous* **has_belly_color::grey** **has_belly_color::yellow** *has_belly_color::olive* *has_belly_color::green* *has_belly_color::pink* *has_belly_color::orange* **has_belly_color::black** **has_belly_color::white** *has_belly_color::red* **has_belly_color::buff** **has_belly_pattern::solid** *has_belly_pattern::spotted* *has_belly_pattern::striped* **has_belly_pattern::multi-colored**
Breast	**has_breast_color::brown** **has_breast_color::grey** **has_breast_color::yellow** **has_breast_color::black** **has_breast_color::white** **has_breast_color::buff** **has_breast_pattern::solid** *has_breast_pattern::spotted* **has_breast_pattern::striped** **has_breast_pattern::multi-colored**
Crown	**has_crown_color::blue** **has_crown_color::brown** **has_crown_color::grey** **has_crown_color::yellow** **has_crown_color::black** **has_crown_color::white**

Forehead	**has_forehead_color::blue** **has_forehead_color::brown** **has_forehead_color::grey** **has_forehead_color::yellow** **has_forehead_color::black** **has_forehead_color::white**
Left and Right Eye	*has_eye_color::blue* *has_eye_color::brown* *has_eye_color::purple* *has_eye_color::rufous* *has_eye_color::grey* **has_eye_color::yellow** *has_eye_color::olive* *has_eye_color::green* *has_eye_color::pink* *has_eye_color::orange* **has_eye_color::black** **has_eye_color::white** *has_eye_color::red* *has_eye_color::buff*
Left and Right Leg	*has_leg_color::blue* *has_leg_color::brown* *has_leg_color::iridescent* *has_leg_color::purple* *has_leg_color::rufous* **has_leg_color::grey** **has_leg_color::yellow** *has_leg_color::olive* *has_leg_color::green* *has_leg_color::pink* *has_leg_color::orange* **has_leg_color::black** **has_leg_color::white** *has_leg_color::red* *has_leg_color::buff*
Left and Right Wing	**has_wing_color::blue** **has_wing_color::brown** *has_wing_color::iridescent* *has_wing_color::purple* *has_wing_color::rufous* **has_wing_color::grey** **has_wing_color::yellow** *has_wing_color::olive* *has_wing_color::green* *has_wing_color::pink* *has_wing_color::orange* **has_wing_color::black** **has_wing_color::white** *has_wing_color::red* **has_wing_color::buff** **has_wing_pattern::solid** **has_wing_pattern::spotted** **has_wing_pattern::striped** **has_wing_pattern::multi-colored** **has_wing_shape::rounded-wings** **has_wing_shape::pointed-wings** *has_wing_shape::broad-wings* *has_wing_shape::tapered-wings* *has_wing_shape::long-wings*
Nape	**has_nape_color::brown** **has_nape_color::grey** **has_nape_color::yellow** **has_nape_color::black** **has_nape_color::white** **has_nape_color::buff**
Tail	*has_upper_tail_color::blue* **has_upper_tail_color::brown** *has_upper_tail_color::iridescent* *has_upper_tail_color::purple* *has_upper_tail_color::rufous* **has_upper_tail_color::grey** **has_upper_tail_color::yellow** *has_upper_tail_color::olive* *has_upper_tail_color::green* *has_upper_tail_color::pink* *has_upper_tail_color::orange* **has_upper_tail_color::black** **has_upper_tail_color::white** *has_upper_tail_color::red* **has_upper_tail_color::buff** **has_under_tail_color::blue** **has_under_tail_color::brown** *has_under_tail_color::iridescent* *has_under_tail_color::purple* *has_under_tail_color::rufous* **has_under_tail_color::grey** **has_under_tail_color::yellow** *has_under_tail_color::olive* *has_under_tail_color::green* *has_under_tail_color::pink* *has_under_tail_color::orange* **has_under_tail_color::black** **has_under_tail_color::white** *has_under_tail_color::red* **has_under_tail_color::buff** **has_tail_pattern::solid**

	has_tail_pattern::spotted **has_tail_pattern::striped** **has_tail_pattern::multi-colored** *has_tail_shape::forked_tail* **has_tail_shape::notched_tail** *has_tail_shape::rounded_tail* *has_tail_shape::fan-shaped_tail* *has_tail_shape::pointed_tail* *has_tail_shape::squared_tail*
Throat	*has_throat_color::blue* *has_throat_color::brown* **has_throat_color::grey** **has_throat_color::yellow** **has_throat_color::black** **has_throat_color::white** **has_throat_color::buff**
No Specific Part	*has_size::large_(16_-_32_in)* *has_size::small_(5_-_9_in)* *has_size::very_large_(32_-_72_in)* *has_size::medium_(9_-_16_in)* *has_size::very_small_(3_-_5_in)* *has_shape::upright-perching_water-like* *has_shape::chicken-like-marsh* *has_shape::long-legged-like* **has_shape::duck-like** *has_shape::owl-like* *has_shape::gull-like* *has_shape::hummingbird-like* *has_shape::pigeon-like* *has_shape::tree-clinging-like* *has_shape::hawk-like* *has_shape::sandpiper-like* *has_shape::upland-ground-like* **has_shape::swallow-like** **has_shape::perching-like** *has_head_pattern::spotted* **has_head_pattern::malar** *has_head_pattern::crested* *has_head_pattern::masked* *has_head_pattern::unique_pattern* **has_head_pattern::eyebrow** *has_head_pattern::eyering* **has_head_pattern::plain** *has_head_pattern::eyeline* *has_head_pattern::striped* *has_head_pattern::capped* **has_primary_color::blue** **has_primary_color::brown** **has_primary_color::iridescent** **has_primary_color::purple** **has_primary_color::rufous** **has_primary_color::grey** **has_primary_color::yellow** **has_primary_color::olive** **has_primary_color::green** **has_primary_color::pink** **has_primary_color::orange** **has_primary_color::black** **has_primary_color::white** **has_primary_color::red** **has_primary_color::buff**

A.1 Selceted Confusions matrix

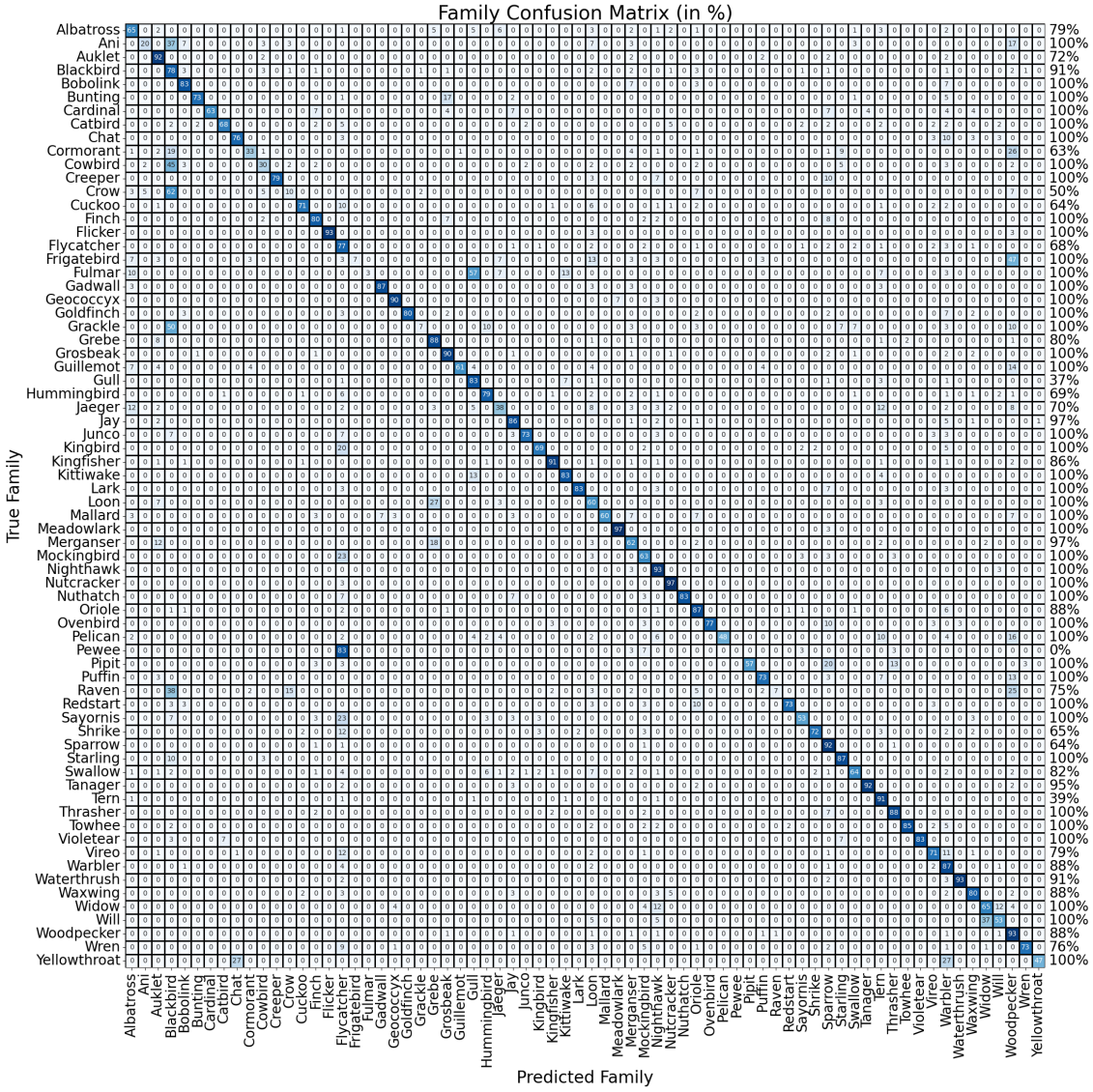


Figure A.1: Confision matrix of the independent Majority voted model. The number at the right side is within family accuracy, fx as all the Warblers that were correctly classified as a Warbler only 51% was classified as the correct type of Warbler.

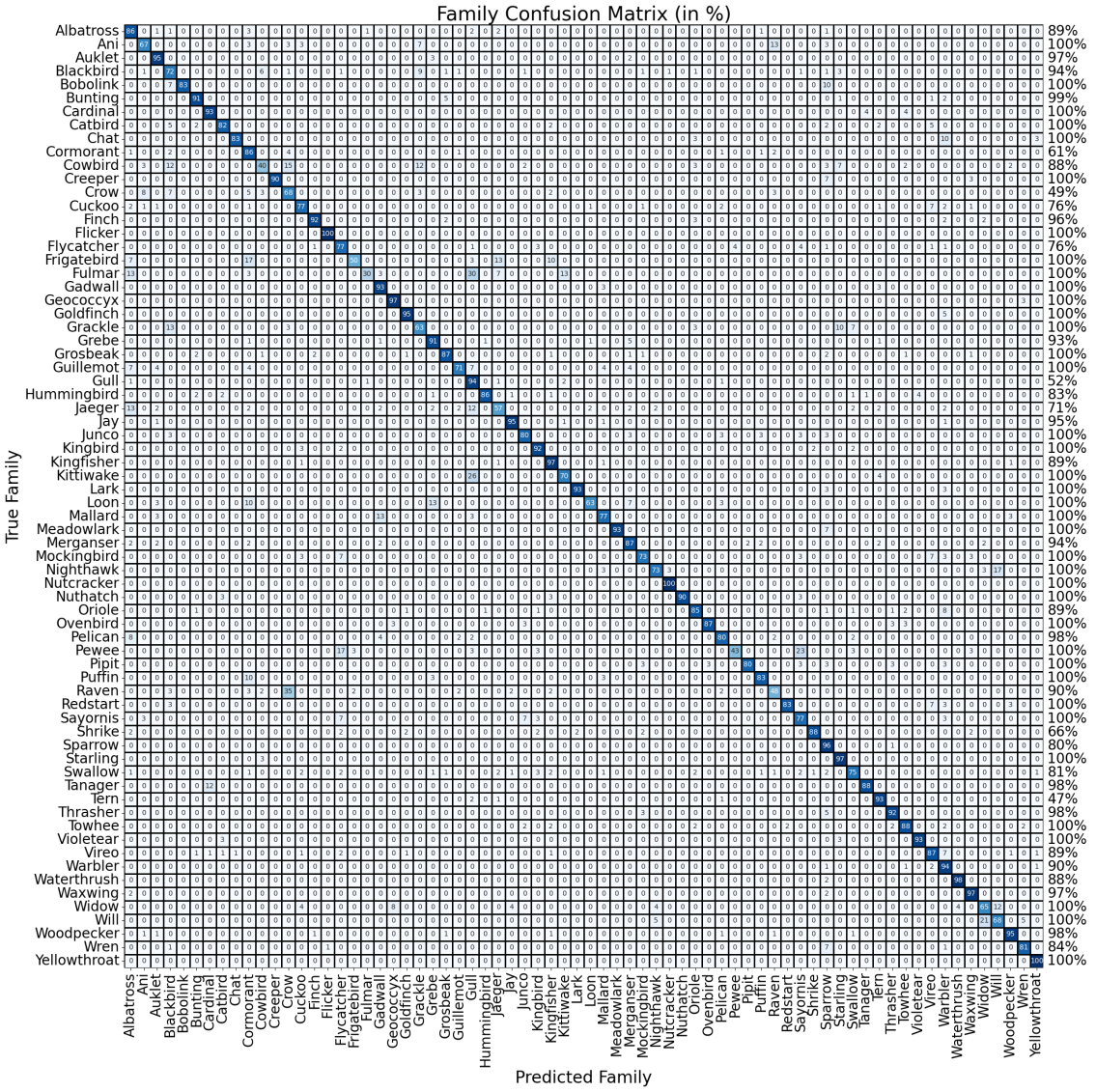


Figure A.2: Confusion matrix of the Joint Majority voted model. The number at the right side is within family accuracy, fx as all the Warblers that were correctly classified as a Warbler only 51% was classified as the correct type of Warbler.

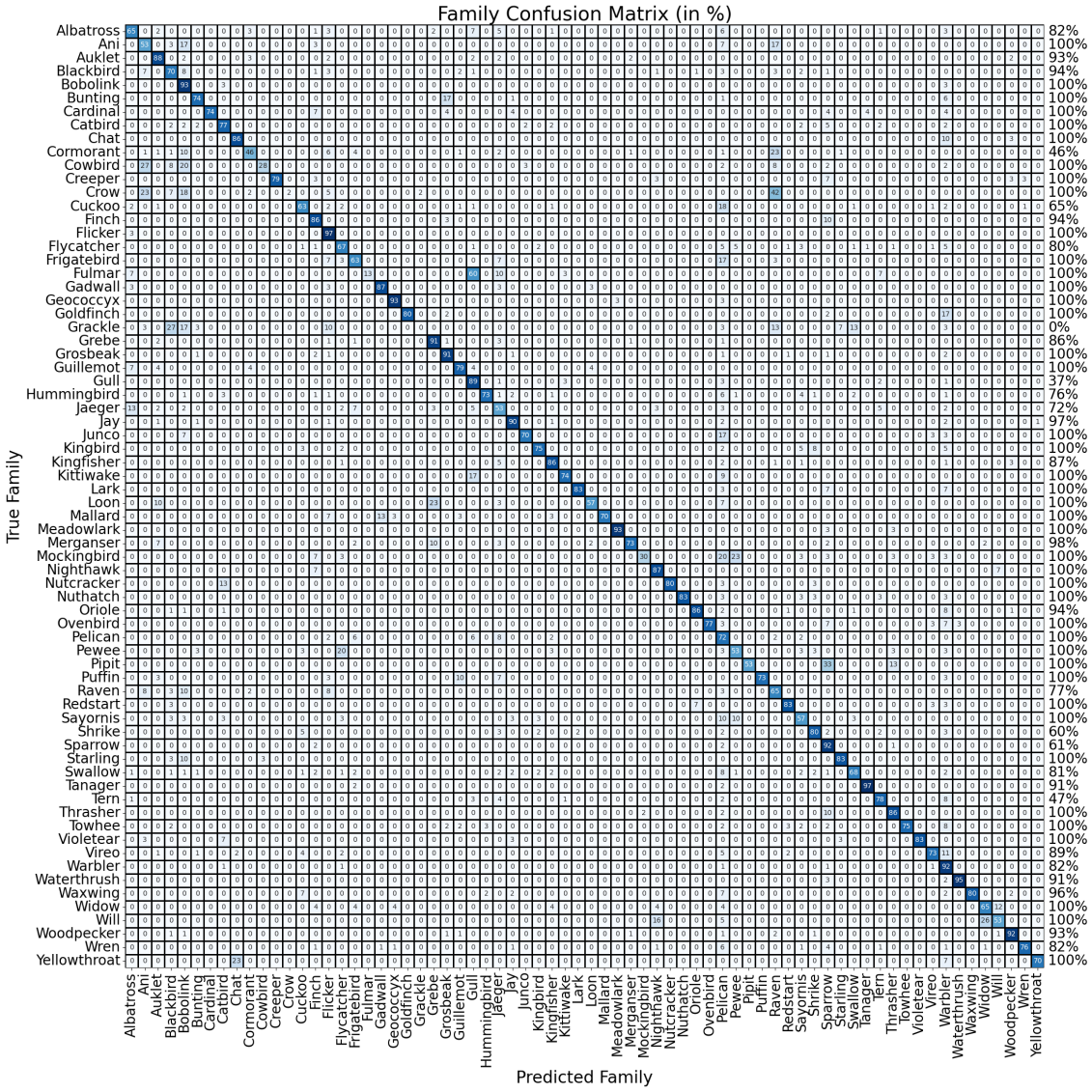


Figure A.3: Confision matrix of the Sequential Majority voted model. The number at the right side is within family accuracy, fx as all the Warblers that were correctly classified as a Warbler only 51% was classified as the correct type of Warbler.

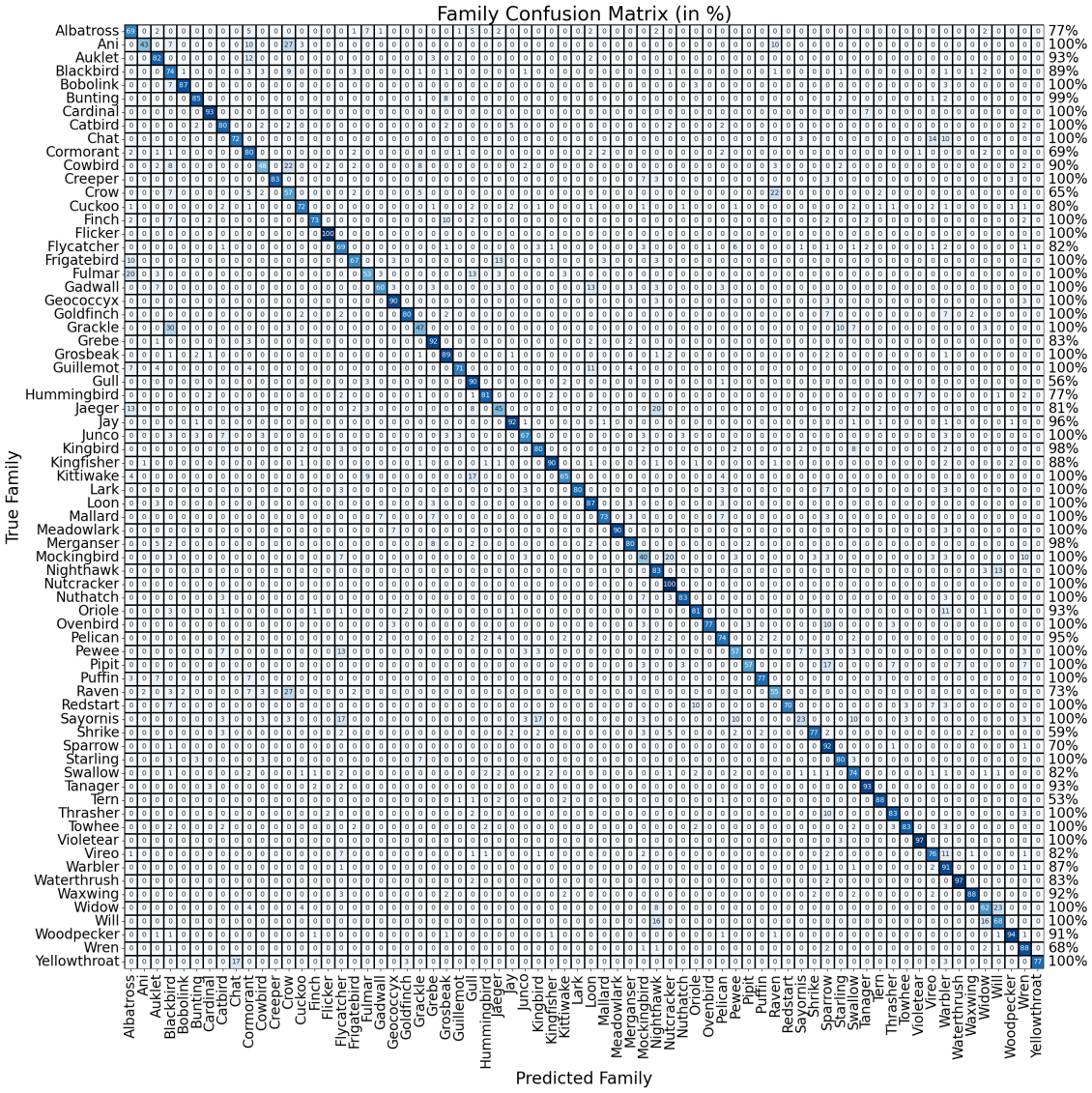


Figure A.4: Confusion matrix of the Joint NoNMajority voted model. The number at the right side is within family accuracy, fx as all the Warblers that were correctly classified as a Warbler only 51% was classified as the correct type of Warbler.

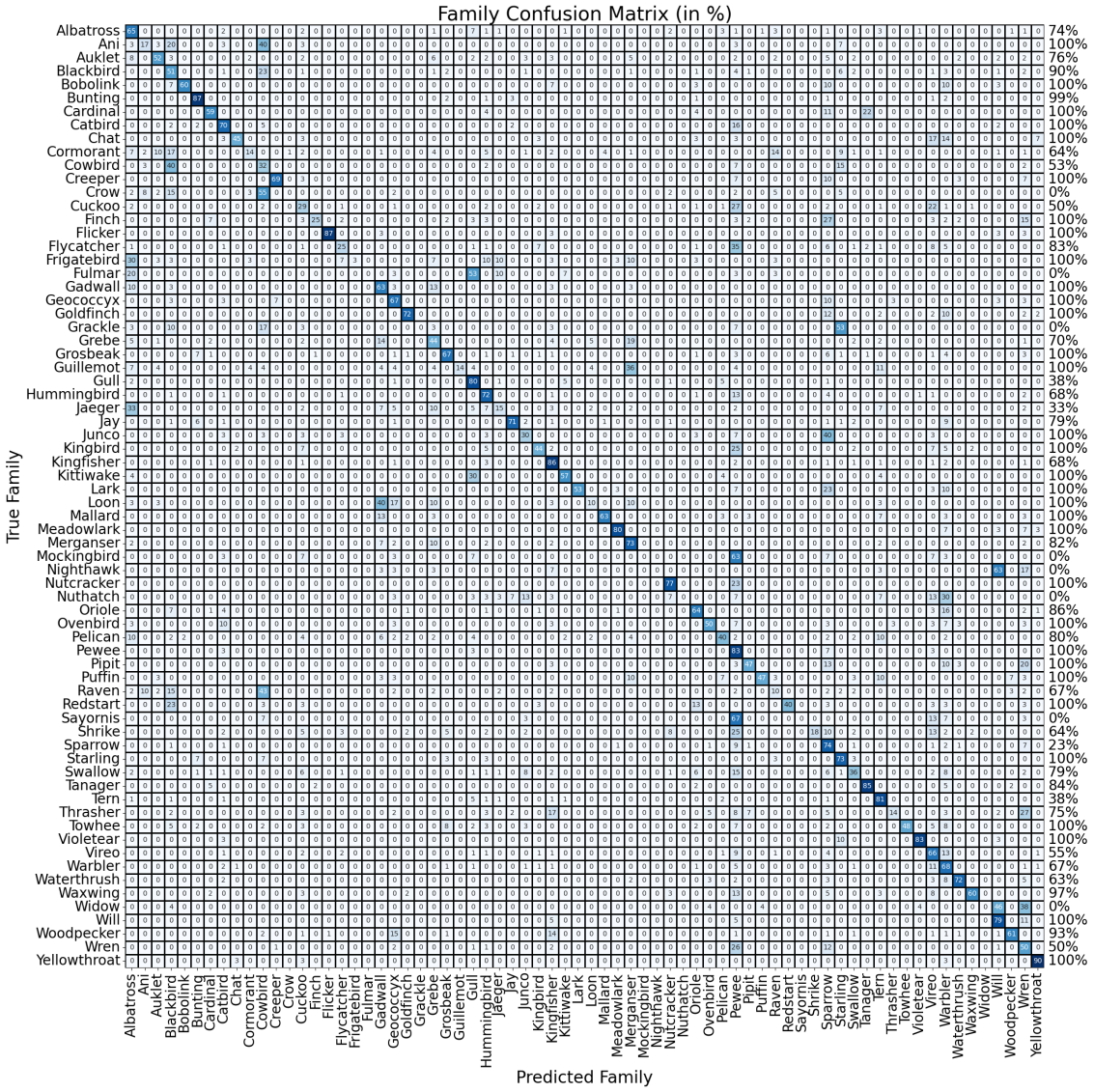


Figure A.5: Confusion matrix of the Sequential NoN-Majority voted model. The number at the right side is within family accuracy, fx as all the Warblers that were correctly classified as a Warbler only 51% was classified as the correct type of Warbler.

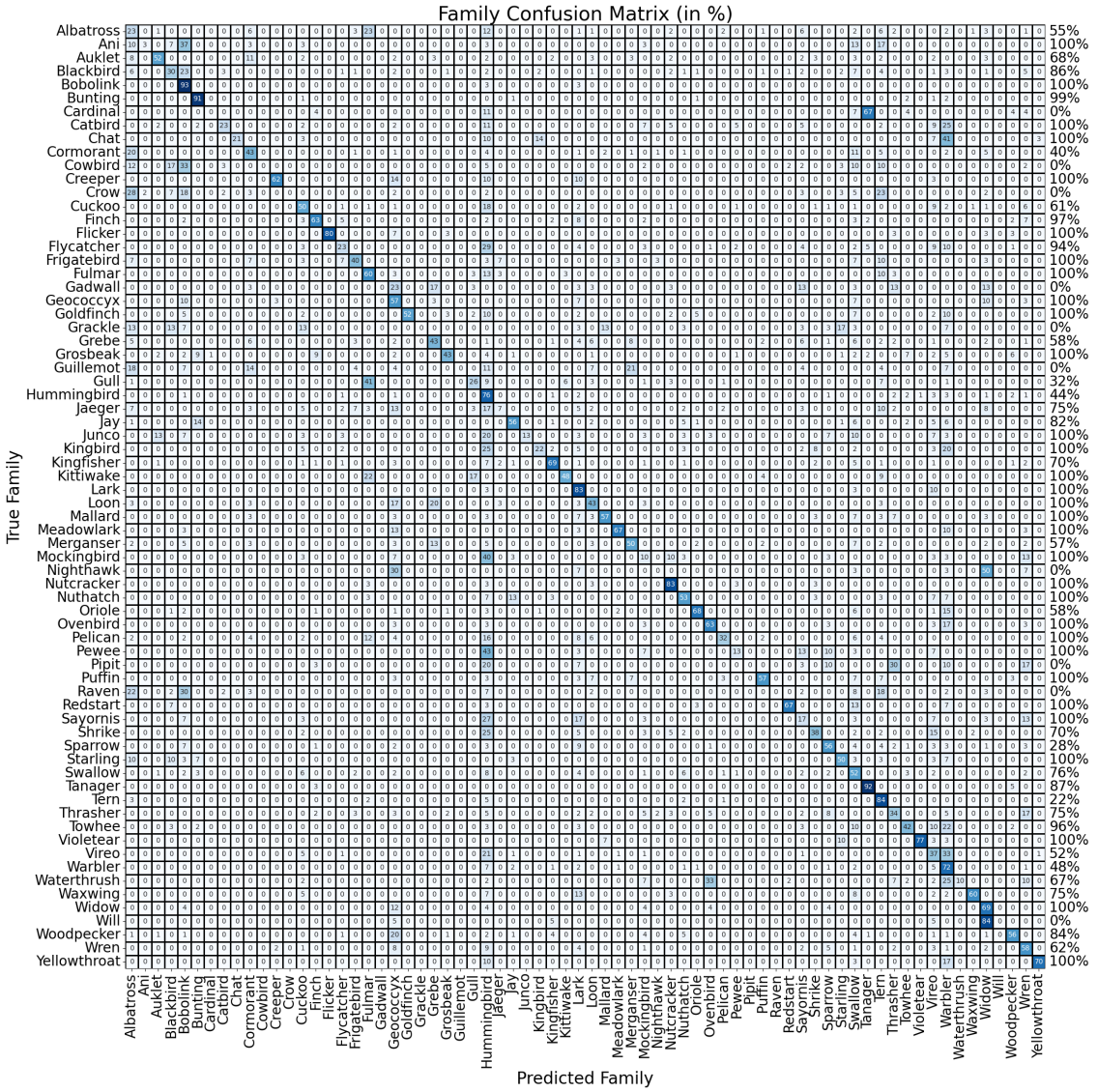


Figure A.6: Confusion matrix of the Sequential model trained without pretending and on a Non-Majority voted dataset. The number at the right side is within family accuracy, fx as all the Warblers that were correctly classified as a Warbler only 51% was classified as the correct type of Warbler.

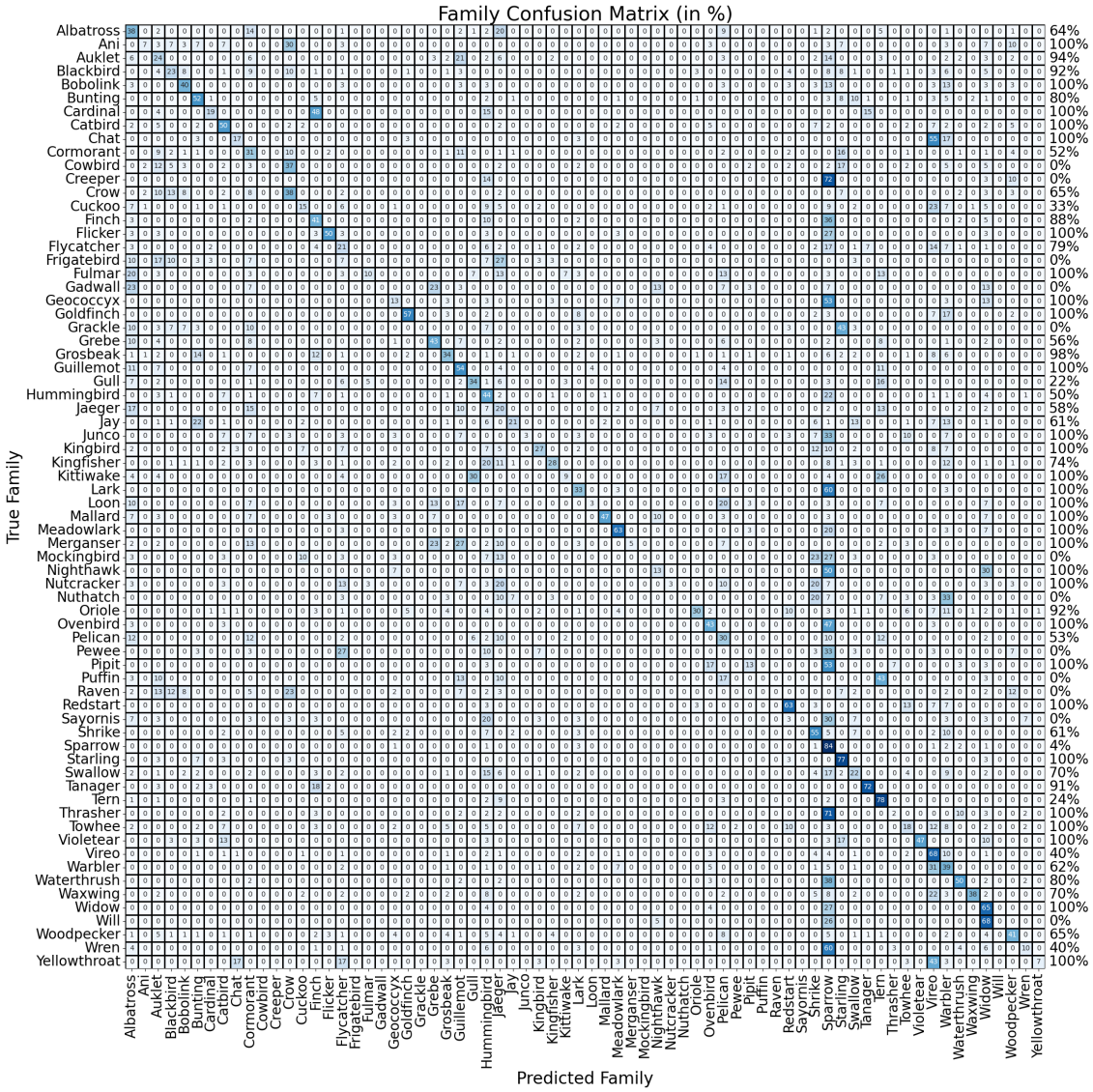


Figure A.7: Confusion matrix of the Sequential model trained on Resized transformation and without majority voting. The number at the right side is within family accuracy, fx as all the Warblers that were correctly classified as a Warbler only 51% was classified as the correct type of Warbler.

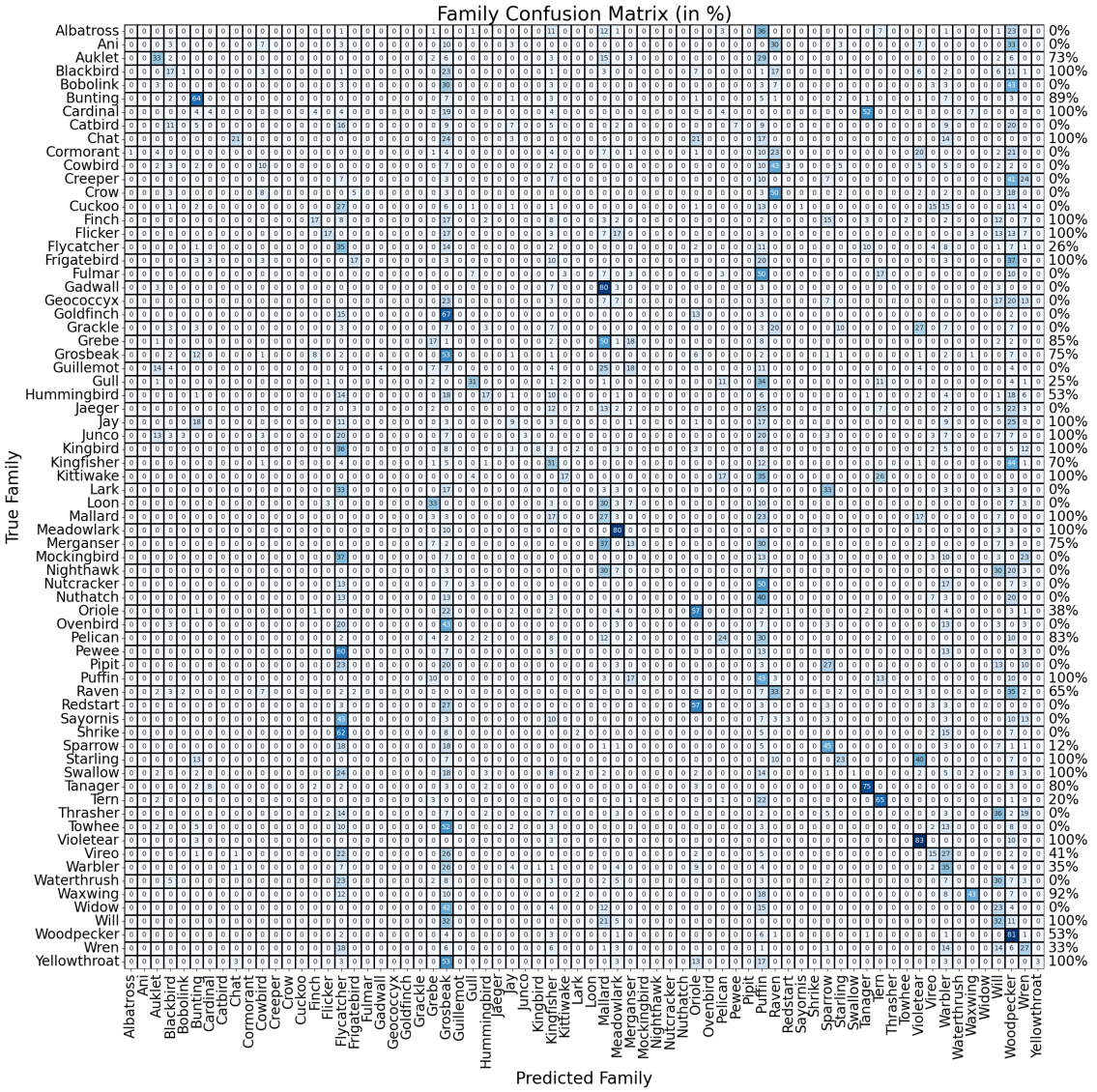


Figure A.8: Confusion matrix of the Independent model trained on Resized transformation and without majority voting. The number at the right side is within family accuracy, fx as all the Warblers that were correctly classified as a Warbler only 51% was classified as the correct type of Warbler.

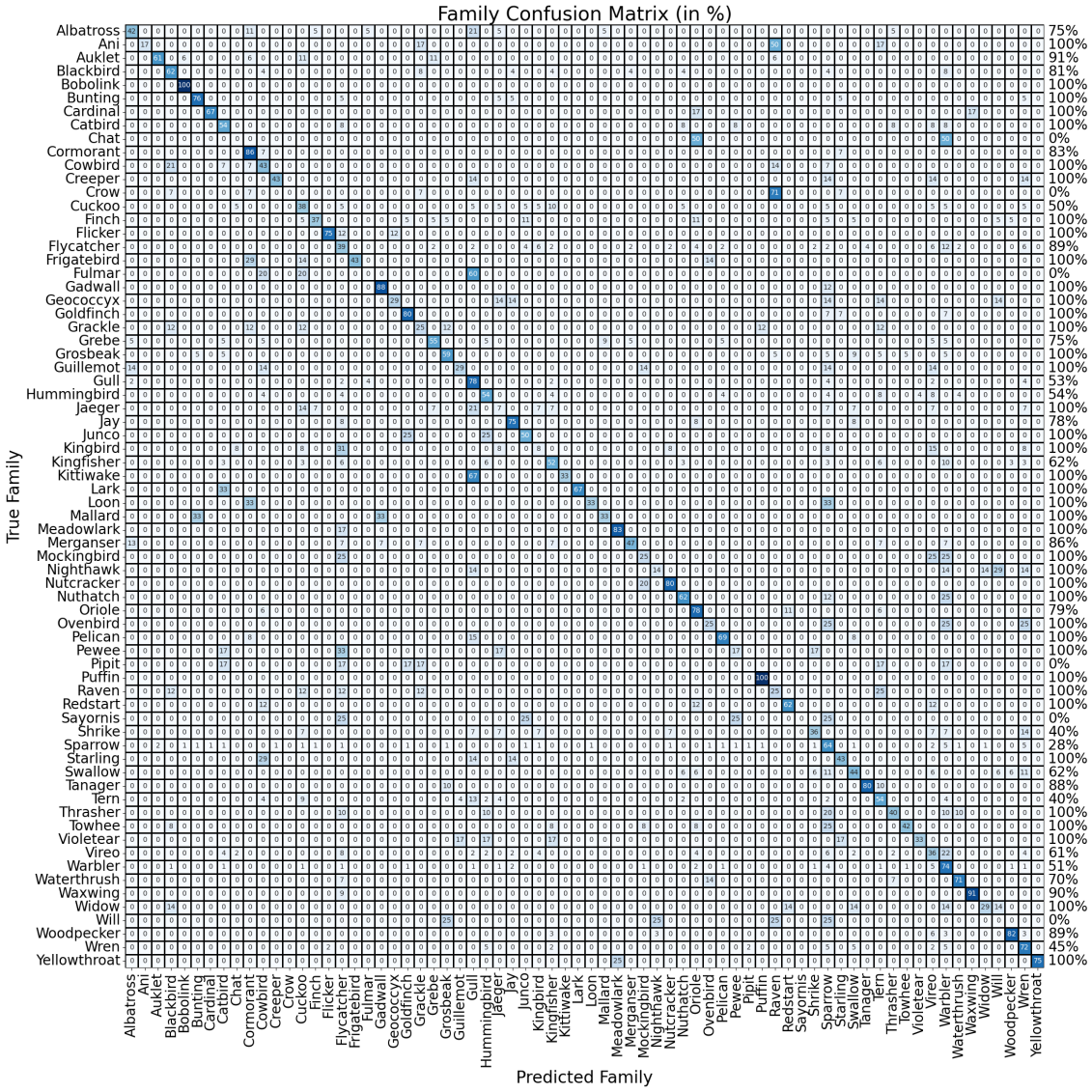


Figure A.9: Confision matrix of the baseline C to Y model evaluated on the anoteted concepts. The number at the right side is within family accuracy, fx as all the Warblers that were correctly classified as a Warbler only 51% was classified as the correct type of Warbler. It shows that the concepts are not annotated well enough to show the difference in classifying some families of birds while others can be classified correctly using the true concepts

A.2 Examples of problematic explainability

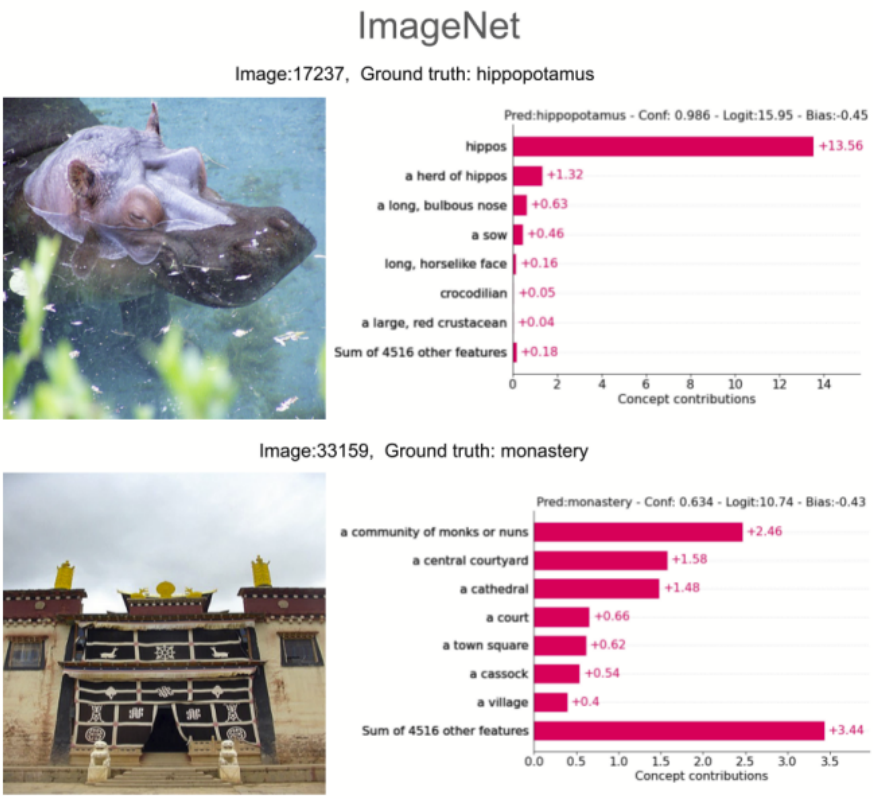


Figure A.10: Figure by Oikarinen et al. (2023): Example of concept made by ChatGPT on an imagenet dataset note that the most activated concept is just a synonym for the predicted class.

Figure A.11: This image was in the appendix of Ghosh et al. (2023) claiming to be an explainable model based on boolean logic; what is worth noticing is that defining concept for Expert 4 to classify an image of a horse is that it is smelly. This is an artifact of training a concept bottleneck model on the Animals with Attributes 2 dataset, which contains concepts not present in the image.

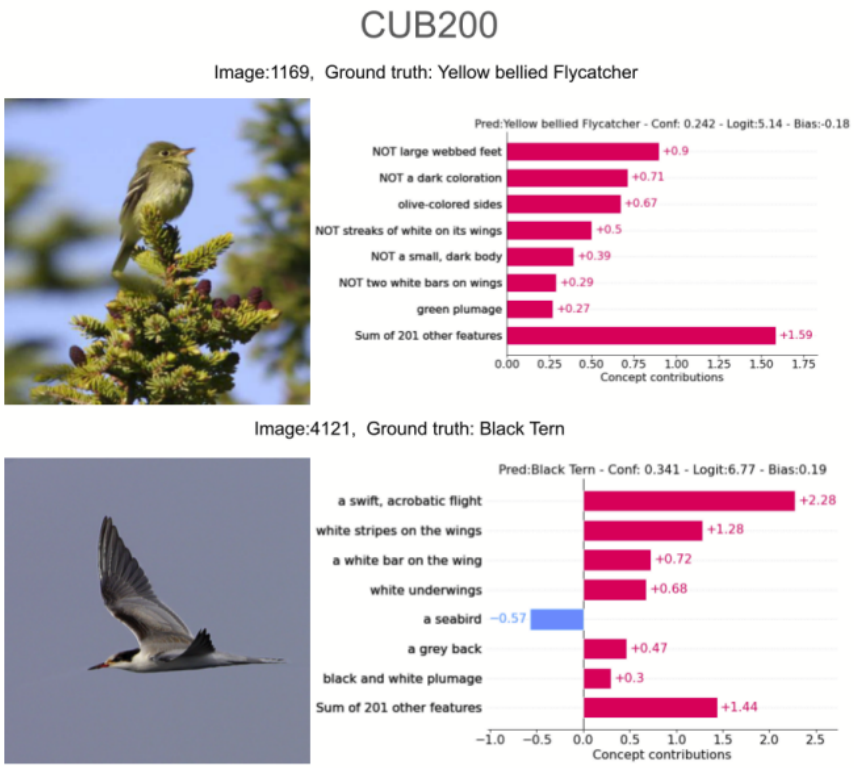
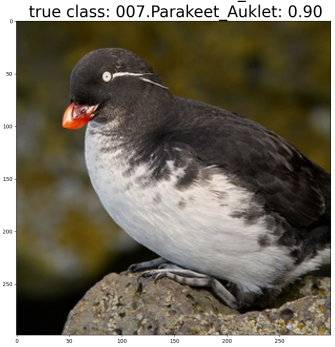


Figure A.12: Figure by Oikarinen et al. (2023) showing labels generated by ChatGPT on the CUB dataset, while the labels appear more descriptive, we still see that the model is willing to make predictions about feet and underwings despite those concepts not being present.

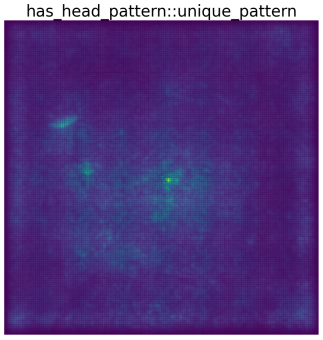
A.3 Random Examples

joint model trained on No-Pretaining model and NoN-Majority-voted concepts

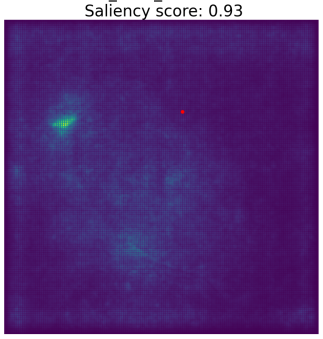
Image validation index 145
Predicted class: 007.Parakeet_Auklet: 0.90
true class: 007.Parakeet_Auklet: 0.90



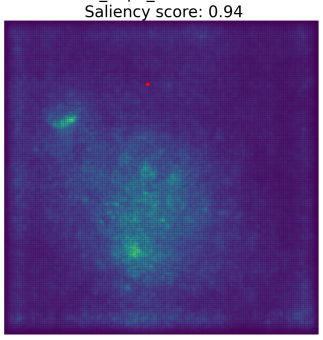
has_head_pattern::unique_pattern



has_back_color::red
Saliency score: 0.93



has_nape_color::white
Saliency score: 0.94

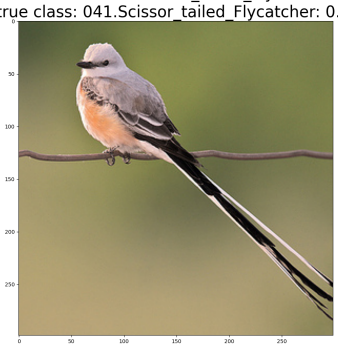


Concept	Orginal Lables	Majority-voted	Prediction	C to Y weight
has_head_pattern::unique_pattern	False	False	0.00	-0.42
has_back_color::red	False	False	0.00	-0.03
has_nape_color::white	False	False	0.06	-0.34
has_tail_shape::pointed_tail	False	False	0.15	0.07
has_underparts_color::white	True	True	0.97	0.59
has_primary_color::brown	False	False	0.01	-0.16
has_leg_color::grey	True	True	0.52	0.67
has_tail_shape::notched_tail	False	False	0.21	-0.35
has_wing_color::red	False	False	0.00	-0.16
has_leg_color::orange	False	False	0.00	-0.51
has_breast_color::rufous	False	False	0.00	-0.10
has_belly_color::white	True	True	0.96	0.62
has_throat_color::iridescent	False	False	0.00	-0.14
has_throat_color::grey	False	False	0.26	0.30
has_wing_shape::pointed-wings	False	False	0.12	0.02
has_size::very_small_(3_-_5_in)	False	False	0.05	-0.55
has_leg_color::black	False	False	0.48	-0.30
has_forehead_color::white	False	False	0.01	-0.38
has_throat_color::olive	False	False	0.00	-0.06
has_primary_color::blue	False	False	0.00	-0.37
has_head_pattern::plain	False	False	0.29	-0.07
has_throat_color::pink	False	False	0.00	-0.03
has_size::very_large_(32_-_72_in)	False	False	0.00	-0.22
has_primary_color::rufous	False	False	0.00	-0.09
has_primary_color::black	True	True	0.95	0.34
has_belly_color::green	False	False	0.00	-0.05
has_breast_color::orange	False	False	0.00	-0.12
has_throat_color::black	True	True	0.66	0.26
has_belly_color::grey	False	False	0.16	-0.00
has_primary_color::red	False	False	0.03	0.10
has_upper_tail_color::iridescent	False	False	0.00	-0.12
has_breast_color::blue	False	False	0.00	-0.14
has_shape::pigeon-like	False	False	0.12	0.24
has_leg_color::yellow	False	False	0.00	0.01
has_primary_color::purple	False	False	0.00	-0.09
has_under_tail_color::green	False	False	0.00	-0.03
has_underparts_color::iridescent	False	False	0.01	-0.13
has_belly_color::yellow	False	False	0.00	-0.31
has_upperparts_color::iridescent	False	False	0.01	-0.15
has_under_tail_color::purple	False	False	0.00	-0.03
has_wing_color::buff	False	False	0.00	-0.51
has_tail_pattern::solid	False	True	0.29	-0.04
has_shape::gull-like	False	False	0.04	-0.35
has_breast_color::black	True	False	0.35	0.25
has_bill_color::green	False	False	0.00	-0.01
has_bill_shape::dagger	False	False	0.00	-0.39
has_back_color::grey	False	False	0.07	-0.12
has_wing_color::yellow	False	False	0.00	-0.25
has_bill_color::buff	False	False	0.05	-0.16
has_back_color::buff	False	False	0.00	-0.46

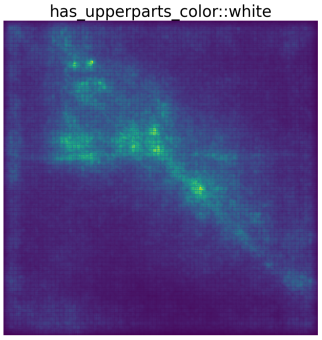
Figure A.13

Sequential model trained on NoN-Majority-voted concepts

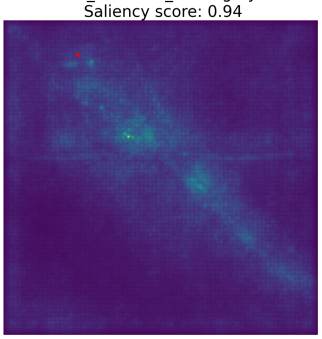
Image validation index 1115
Predicted class: 041.Scissor_tailed_Flycatcher: 0.97
true class: 041.Scissor_tailed_Flycatcher: 0.97



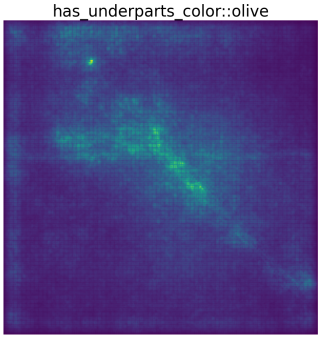
has_upperparts_color::white



has_forehead_color::grey
Saliency score: 0.94



has_underparts_color::olive

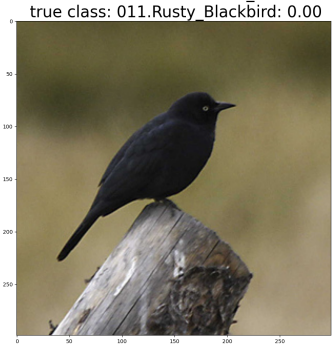


Concept	Orginal Lables	Majority-voted	Prediction	C to Y weight
has_upperparts_color::white	True	False	0.11	-2.43
has_forehead_color::grey	True	False	0.09	2.19
has_underparts_color::olive	False	False	0.00	-1.11
has_bill_length::about_the_same_as_head	False	False	0.17	-0.52
has_primary_color::purple	False	False	0.01	0.89
has_back_color::white	True	True	0.06	-3.46
has_head_pattern::eyeline	True	False	0.11	2.57
has_forehead_color::purple	False	False	0.00	0.56
has_bill_color::grey	False	False	0.33	-2.82
has_back_color::yellow	False	False	0.06	-0.98
has_tail_shape::notched_tail	False	False	0.46	1.76
has_upperparts_color::brown	False	False	0.26	2.34
has_bill_shape::spatulate	False	False	0.00	-0.87
has_upper_tail_color::black	True	True	0.33	3.93
has_breast_color::brown	False	False	0.04	0.74
has_under_tail_color::red	False	False	0.02	0.94
has_throat_color::brown	False	False	0.01	-0.48
has_crown_color::yellow	False	False	0.02	-3.83
has_under_tail_color::white	True	False	0.11	-2.12
has_bill_shape::needle	False	False	0.00	-1.69
has_primary_color::red	False	False	0.07	-2.46
has_breast_color::yellow	False	False	0.09	-1.77
has_nape_color::black	False	False	0.01	-5.04
has_back_color::green	False	False	0.00	-0.51
has_crown_color::purple	False	False	0.01	0.80
has_tail_shape::squared_tail	False	False	0.08	-0.27
has_belly_color::pink	False	False	0.01	0.45
has_throat_color::orange	False	False	0.07	-0.38
has_breast_color::buff	False	False	0.18	-0.80
has_nape_color::brown	False	False	0.07	-5.92
has_crown_color::white	True	True	0.09	7.22
has_wing_shape::tapered-wings	False	False	0.07	5.37
has_eye_color::rufous	False	False	0.00	1.04
has_underparts_color::purple	False	False	0.00	0.21
has_belly_color::yellow	False	False	0.20	-0.38
has_bill_shape::hooked_seabird	False	False	0.01	-13.43
has_head_pattern::crested	False	False	0.02	-0.75
has_primary_color::white	True	True	0.05	-2.07
has_tail_pattern::solid	False	False	0.45	-2.03
has_wing_color::olive	False	False	0.00	-2.95
has_wing_color::buff	True	False	0.20	0.58
has_belly_pattern::striped	False	False	0.01	-0.21
has_forehead_color::blue	False	False	0.00	-2.89
has_breast_pattern::solid	False	True	0.87	0.53
has_bill_shape::cone	True	False	0.19	-3.89
has_breast_color::purple	False	False	0.00	0.11
has_shape::duck-like	False	False	0.00	-1.13
has_back_color::grey	True	False	0.25	-1.28
has_nape_color::purple	False	False	0.00	0.37
has_throat_color::green	False	False	0.00	-0.14

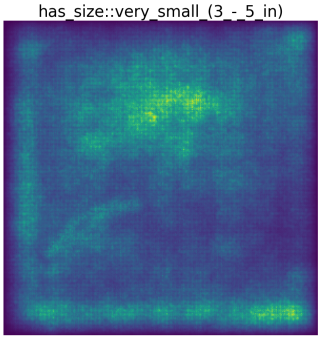
Figure A.14

Sequential model trained on Rezise transformation and NoN-Majority-voted concepts

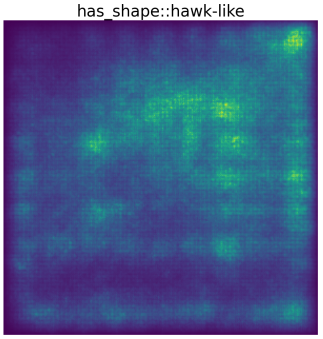
Image validation index 243
Predicted class: 029.American_Crow: 0.98
true class: 011.Rusty_Blackbird: 0.00



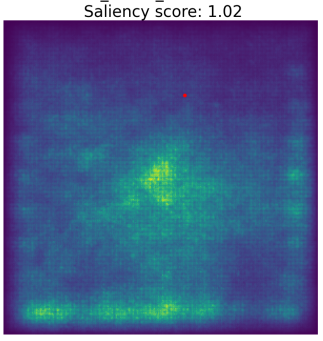
has_size::very_small_(3_-_5_in)



has_shape::hawk-like



has_crown_color::buff
Saliency score: 1.02



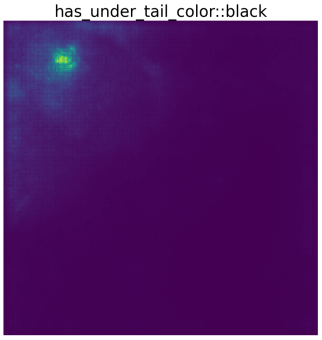
Concept	Orginal Lables	Majority-voted	Prediction	C to Y weight
has_size::very_small_(3_-_5_in)	True	False	0.10	-1.16
has_shape::hawk-like	False	False	0.06	0.53
has_crown_color::buff	False	False	0.00	1.37
has_eye_color::red	False	False	0.02	-0.38
has_underparts_color::yellow	False	False	0.01	-2.86
has_upper_tail_color::orange	False	False	0.01	-0.15
has_wing_pattern::spotted	False	False	0.00	0.30
has_forehead_color::black	True	True	0.93	-3.79
has_crown_color::brown	False	True	0.01	2.24
has_belly_color::purple	False	False	0.00	0.06
has_throat_color::pink	False	False	0.00	0.46
has_upperparts_color::buff	False	False	0.01	-2.41
has_shape::duck-like	False	False	0.01	0.25
has_back_pattern::striped	False	False	0.03	0.28
has_size::small_(5_-_9_in)	False	True	0.62	0.97
has_back_color::white	False	False	0.02	0.14
has_belly_color::grey	False	False	0.04	2.17
has_back_pattern::multi-colored	False	False	0.11	-0.21
has_throat_color::olive	False	False	0.00	0.02
has_eye_color::purple	False	False	0.00	0.17
has_upperparts_color::green	False	False	0.00	0.10
has_wing_shape::rounded-wings	True	False	0.25	-0.69
has_back_pattern::spotted	False	False	0.00	-0.90
has_breast_color::grey	False	False	0.05	2.08
has_tail_shape::pointed_tail	False	False	0.15	-0.47
has_forehead_color::red	False	False	0.00	-0.52
has_belly_color::iridescent	False	False	0.01	0.02
has_head_pattern::spotted	False	False	0.00	-1.75
has_upperparts_color::black	True	True	0.91	0.92
has_bill_color::red	False	False	0.01	-0.20
has_upper_tail_color::green	False	False	0.00	0.01
has_leg_color::black	True	True	0.75	4.27
has_throat_color::rufous	False	False	0.00	0.25
has_bill_length::about_the_same_as_head	False	False	0.37	-5.10
has_upperparts_color::brown	False	False	0.02	0.53
has_tail_shape::squared_tail	False	False	0.08	0.85
has_nape_color::white	False	False	0.04	1.53
has_upperparts_color::purple	False	False	0.01	0.07
has_upper_tail_color::yellow	False	False	0.00	-0.83
has_under_tail_color::buff	False	False	0.01	-2.67
has_forehead_color::brown	False	False	0.01	2.45
has_under_tail_color::red	False	False	0.00	0.08
has_breast_color::black	True	False	0.90	-1.00
has_belly_color::green	False	False	0.00	0.02
has_belly_pattern::spotted	False	False	0.01	-0.73
has_wing_shape::pointed-wings	False	False	0.16	1.41
has_primary_color::green	False	False	0.00	-0.03
has_nape_color::iridescent	False	False	0.01	-0.23
has_bill_color::purple	False	False	0.00	0.24
has_under_tail_color::olive	False	False	0.00	0.39

Figure A.15

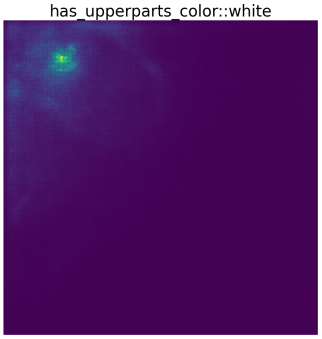
Sequential model trained on Majority-voted concepts

Image validation index 410
Predicted class: 016.Painted_Bunting: 1.00
true class: 016.Painted_Bunting: 1.00

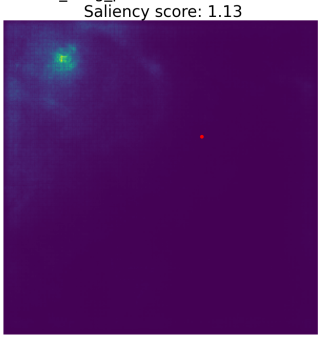
has_under_tail_color::black



has_upperparts_color::white



has_wing_pattern::multi-colored
Saliency score: 1.13



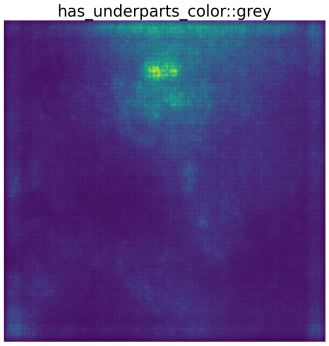
Concept	Orginal Lables	Majority-voted	Prediction	C to Y weight
has_under_tail_color::black	False	False	0.00	-3.60
has_upperparts_color::white	False	False	0.00	-1.01
has_wing_pattern::multi-colored	False	True	1.00	2.35
has_primary_color::yellow	False	False	0.00	-1.30
has_belly_color::yellow	False	False	0.00	0.71
has_belly_color::brown	False	False	0.00	-0.01
has_belly_color::white	False	False	0.00	-4.16
has_belly_color::grey	False	False	0.00	-1.98
has_primary_color::black	False	False	0.00	-1.06
has_leg_color::grey	True	True	1.00	4.43
has_size::very_small_(3_-_5_in)	False	False	0.00	-0.05
has_head_pattern::plain	False	False	0.00	-10.46
has_forehead_color::black	False	False	0.00	-1.11
has_breast_color::yellow	False	False	0.00	-1.16
has_wing_color::white	False	False	0.00	-2.00
has_throat_color::yellow	False	False	0.00	-2.58
has_primary_color::buff	False	False	0.00	-0.00
has_upperparts_color::brown	False	False	0.00	-2.03
has_upper_tail_color::black	False	False	0.00	-2.01
has_back_color::white	False	False	0.00	-0.01
has_wing_color::buff	False	False	0.00	-0.04
has_belly_pattern::solid	True	True	1.00	-0.72
has_breast_pattern::multi-colored	True	False	0.00	-0.02
has_wing_color::black	False	False	0.00	-1.13
has_breast_pattern::solid	False	True	1.00	1.44
has_wing_shape::pointed-wings	False	False	0.00	-0.02
has_crown_color::brown	False	False	0.00	-1.04
has_nape_color::black	False	False	0.00	-1.17
has_wing_pattern::spotted	False	False	0.00	-0.00
has_bill_shape::dagger	False	False	0.00	-0.01
has_underparts_color::black	False	False	0.00	-0.99
has_forehead_color::grey	False	False	0.00	-1.16
has_throat_color::grey	False	False	0.00	-1.00
has_back_color::buff	False	False	0.00	-0.02
has_wing_pattern::solid	True	False	0.00	-0.23
has_wing_color::grey	False	False	0.00	-2.04
has_eye_color::black	True	True	1.00	-4.03
has_head_pattern::eyebrow	False	False	0.00	-0.00
has_bill_shape::hooked_seabird	False	False	0.00	-0.00
has_tail_pattern::solid	False	False	0.00	-3.81
has_bill_color::grey	True	True	1.00	5.92
has_breast_pattern::striped	False	False	0.00	-1.00
has_under_tail_color::brown	True	False	0.00	-1.00
has_wing_pattern::striped	False	False	0.00	-1.97
has_back_color::brown	False	False	0.00	-1.04
has_upper_tail_color::white	False	False	0.00	-1.03
has_forehead_color::white	False	False	0.00	-0.00
has_back_color::grey	False	False	0.00	-1.04
has_tail_shape::notched_tail	False	True	1.00	5.12
has_bill_length::shorter_than_head	True	True	1.00	-3.98

Figure A.16

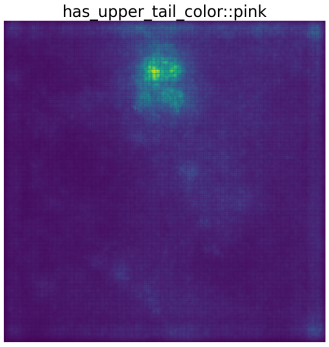
joint model trained on NoN-Majority-voted concepts

Image validation index 4778
Predicted class: 166.Golden_winged_Warbler: 0.99
true class: 166.Golden_winged_Warbler: 0.99

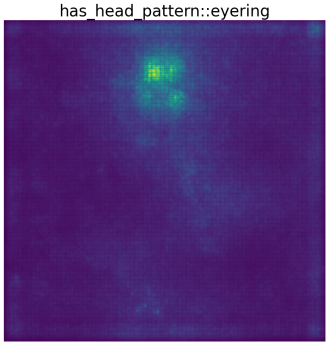
has_underparts_color::grey



has_upper_tail_color::pink



has_head_pattern::eyering

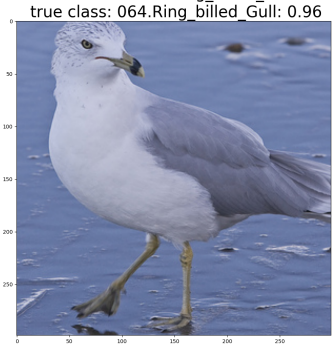


Concept	Orginal Lables	Majority-voted	Prediction	C to Y weight
has_underparts_color::grey	True	True	0.74	0.35
has_upper_tail_color::pink	False	False	0.00	-0.05
has_head_pattern::eyering	False	False	0.05	-0.65
has_leg_color::white	False	False	0.01	-0.04
has_nape_color::purple	False	False	0.00	-0.04
has_under_tail_color::buff	False	False	0.04	-0.18
has_eye_color::orange	False	False	0.00	-0.07
has_forehead_color::orange	False	False	0.00	-0.25
has_nape_color::iridescent	False	False	0.00	-0.19
has_belly_color::olive	False	False	0.00	-0.14
has_shape::upright-perching_water-like	False	False	0.00	-0.13
has_forehead_color::yellow	True	True	1.00	1.18
has_head_pattern::crested	False	False	0.01	-0.08
has_nape_color::orange	False	False	0.00	-0.18
has_breast_color::buff	False	False	0.01	-0.33
has_upperparts_color::buff	False	False	0.08	0.17
has_back_color::white	False	False	0.03	-0.76
has_crown_color::white	False	False	0.03	-0.41
has_upperparts_color::grey	True	True	0.79	0.28
has_throat_color::blue	False	False	0.00	-0.04
has_back_color::black	False	False	0.14	-0.41
has_head_pattern::eyeline	False	False	0.50	0.72
has_under_tail_color::olive	False	False	0.00	-0.23
has_size::medium_(9_-_16_in)	False	False	0.01	-0.57
has_breast_pattern::multi-colored	True	False	0.28	0.39
has_upper_tail_color::red	False	False	0.00	-0.07
has_belly_color::black	False	False	0.04	-0.58
has_bill_color::olive	False	False	0.00	-0.05
has_leg_color::iridescent	False	False	0.00	-0.05
has_breast_color::purple	False	False	0.00	-0.01
has_shape::chicken-like-marsh	False	False	0.01	-0.08
has_crown_color::blue	False	False	0.00	-0.12
has_forehead_color::grey	False	False	0.09	-0.56
has_primary_color::green	False	False	0.00	-0.11
has_belly_pattern::spotted	False	False	0.00	-0.24
has_eye_color::white	False	False	0.02	-0.33
has_wing_color::brown	False	False	0.01	-0.58
has_shape::upland-ground-like	False	False	0.00	-0.23
has_breast_color::black	True	False	0.35	0.16
has_tail_shape::squared_tail	True	False	0.02	-0.20
has_shape::owl-like	False	False	0.00	-0.02
has_back_color::buff	False	False	0.14	0.35
has_leg_color::orange	False	False	0.02	-0.24
has_breast_color::green	False	False	0.00	-0.03
has_nape_color::blue	False	False	0.00	-0.15
has_under_tail_color::black	False	False	0.28	-0.02
has_throat_color::iridescent	False	False	0.00	-0.08
has_primary_color::black	False	False	0.32	-0.20
has_primary_color::iridescent	False	False	0.02	0.09
has_throat_color::purple	False	False	0.00	-0.03

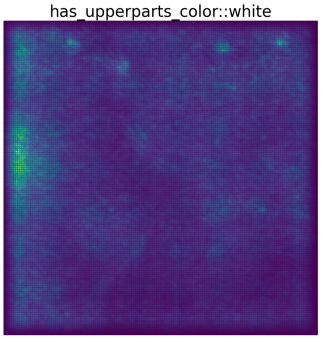
Figure A.17

Sequential model trained on No-Pretaining model and NoN-Majority-voted concepts

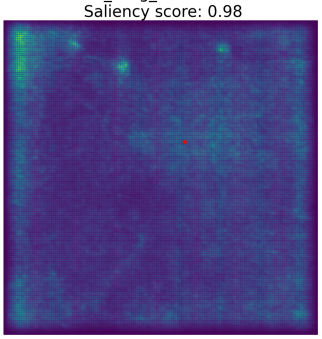
Image validation index 1780
Predicted class: 064.Ring_billed_Gull: 0.96
true class: 064.Ring_billed_Gull: 0.96



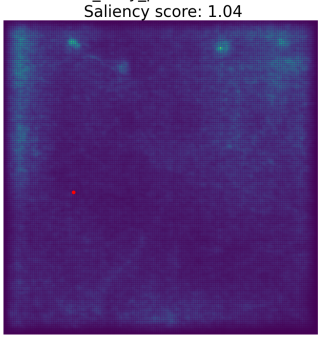
has_upperparts_color::white



has_wing_color::blue
Saliency score: 0.98



has_belly_pattern::solid
Saliency score: 1.04

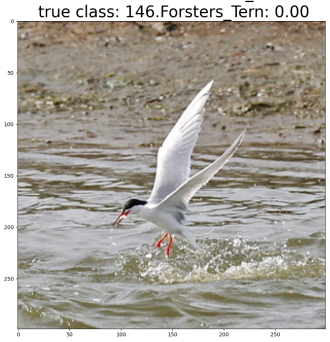


Concept	Orginal Lables	Majority-voted	Prediction	C to Y weight
has_upperparts_color::white	True	True	0.51	-0.09
has_wing_color::blue	False	False	0.00	-1.18
has_belly_pattern::solid	True	True	0.90	-0.09
has_bill_color::yellow	True	False	0.36	-4.95
has_throat_color::orange	False	False	0.00	-0.48
has_crown_color::rufous	False	False	0.00	-0.34
has_upper_tail_color::red	False	False	0.00	0.98
has_breast_color::white	True	True	0.90	1.50
has_nape_color::grey	False	False	0.65	4.72
has_primary_color::brown	False	False	0.00	-0.71
has_eye_color::buff	False	False	0.00	1.93
has_throat_color::grey	False	False	0.27	-3.73
has_throat_color::buff	False	False	0.01	2.43
has_bill_color::grey	False	False	0.13	0.70
has_upperparts_color::iridescent	False	False	0.00	0.14
has_bill_shape::needle	False	False	0.00	-2.75
has_head_pattern::malar	False	False	0.01	1.50
has_underparts_color::white	True	True	0.90	4.24
has_nape_color::red	False	False	0.00	-0.03
has_head_pattern::plain	False	False	0.46	-6.44
has_nape_color::yellow	False	False	0.00	-0.16
has_nape_color::brown	False	False	0.00	-0.33
has_underparts_color::yellow	False	False	0.00	-1.02
has_throat_color::rufous	False	False	0.00	-0.20
has_forehead_color::white	True	True	0.92	2.00
has_back_color::iridescent	False	False	0.00	0.06
has_breast_color::purple	False	False	0.00	0.27
has_nape_color::pink	False	False	0.00	-0.61
has_belly_color::orange	False	False	0.00	-0.98
has_head_pattern::eyering	False	False	0.06	-0.05
has_belly_pattern::striped	False	False	0.02	-2.15
has_forehead_color::brown	False	False	0.00	-0.44
has_primary_color::rufous	False	False	0.00	-0.56
has_bill_color::green	False	False	0.00	0.45
has_nape_color::black	False	False	0.05	-1.46
has_forehead_color::olive	False	False	0.00	-0.30
has_bill_length::about_the_same_as_head	True	True	0.95	1.33
has_underparts_color::blue	False	False	0.00	-0.60
has_breast_color::black	False	False	0.01	-1.29
has_bill_color::olive	False	False	0.00	-0.05
has_under_tail_color::black	True	False	0.45	6.59
has_upperparts_color::yellow	False	False	0.00	-1.22
has_tail_shape::forked_tail	False	False	0.14	0.86
has_underparts_color::black	False	False	0.01	-1.97
has_belly_color::brown	False	False	0.00	-1.06
has_upper_tail_color::olive	False	False	0.00	0.07
has_breast_color::buff	False	False	0.00	2.36
has_belly_color::iridescent	False	False	0.00	-0.24
has_upper_tail_color::green	False	False	0.00	-0.32
has_crown_color::orange	False	False	0.00	-0.20

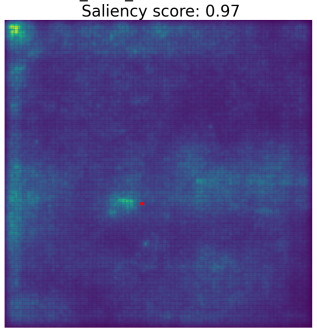
Figure A.18

Independent model trained on NoN-Majority-voted concepts

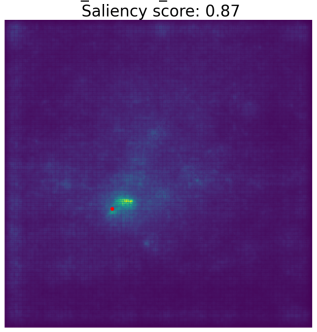
Image validation index 4186
Predicted class: 144.Common_Tern: 1.00
true class: 146.Forsters_Tern: 0.00



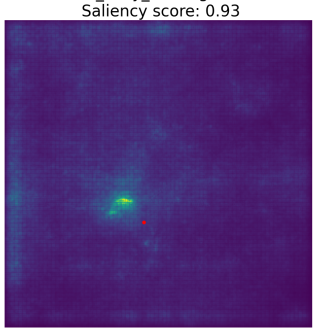
has_back_color::iridescent
Saliency score: 0.97



has_forehead_color::white
Saliency score: 0.87



has_belly_color::green
Saliency score: 0.93

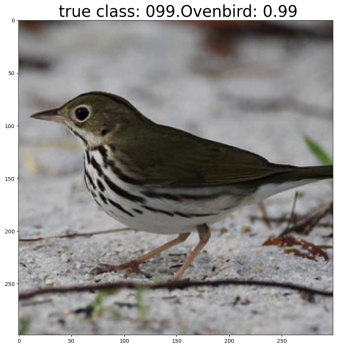


Concept	Orginal Lables	Majority-voted	Prediction	C to Y weight
has_back_color::iridescent	False	False	0.00	0.00
has_forehead_color::white	False	False	0.08	-3.00
has_belly_color::green	False	False	0.00	0.00
has_primary_color::white	True	True	0.98	4.00
has_belly_color::brown	False	False	0.00	-2.00
has_nape_color::blue	False	False	0.00	0.00
has_eye_color::green	False	False	0.00	-1.00
has_primary_color::brown	False	False	0.00	-7.00
has_bill_shape::hooked	False	False	0.00	-1.00
has_leg_color::pink	False	False	0.00	-2.00
has_primary_color::buff	False	False	0.00	3.00
has_upper_tail_color::brown	False	False	0.00	-5.00
has_shape::duck-like	False	False	0.02	-5.00
has_belly_color::pink	False	False	0.00	-1.00
has_under_tail_color::grey	False	False	0.05	3.00
has_shape::upland-ground-like	False	False	0.01	-4.00
has_underparts_color::rufous	False	False	0.00	-1.00
has_underparts_color::purple	False	False	0.00	0.00
has_wing_shape::rounded-wings	False	False	0.04	0.00
has_breast_color::rufous	False	False	0.00	-1.00
has_leg_color::white	False	False	0.06	-1.00
has_crown_color::orange	False	False	0.00	-1.00
has_wing_shape::long-wings	True	False	0.36	-1.00
has_nape_color::yellow	False	False	0.00	-1.00
has_tail_pattern::multi-colored	False	False	0.06	5.00
has_throat_color::pink	False	False	0.00	0.00
has_throat_color::yellow	False	False	0.00	-1.00
has_belly_pattern::spotted	False	False	0.00	-1.00
has_eye_color::buff	False	False	0.00	0.00
has_wing_color::pink	False	False	0.00	-1.00
has_nape_color::purple	False	False	0.00	-1.00
has_crown_color::white	False	False	0.02	3.00
has_belly_pattern::solid	True	True	0.96	-3.00
has_size::large_(16_-_32_in)	True	False	0.04	-7.00
has_wing_shape::tapered-wings	False	False	0.26	10.00
has_bill_shape::hooked_seabird	False	False	0.01	-1.00
has_bill_length::longer_than_head	True	False	0.33	0.00
has_shape::long-legged-like	False	False	0.07	-4.00
has_breast_color::orange	False	False	0.00	-2.00
has_wing_color::white	True	True	0.93	-3.00
has_under_tail_color::buff	False	False	0.00	-6.00
has_throat_color::orange	False	False	0.00	-2.00
has_forehead_color::iridescent	False	False	0.00	-1.00
has_underparts_color::orange	False	False	0.00	-4.00
has_upper_tail_color::iridescent	False	False	0.00	0.00
has_eye_color::yellow	False	False	0.00	-5.00
has_upperparts_color::olive	False	False	0.00	-2.00
has_head_pattern::unique_pattern	False	False	0.00	-5.00
has_under_tail_color::green	False	False	0.00	-1.00
has_upper_tail_color::blue	False	False	0.00	-1.00

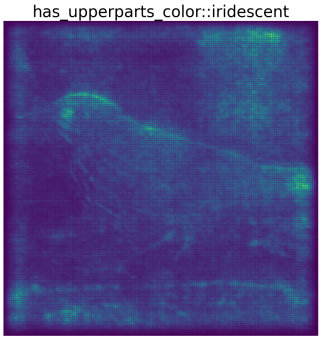
Figure A.19

Sequential model trained on No-Pretaining model and NoN-Majority-voted data

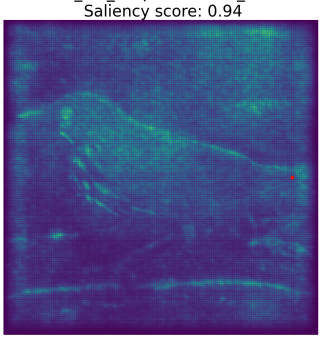
Image validation index 2811
Predicted class: 099.Ovenbird: 0.99
true class: 099.Ovenbird: 0.99



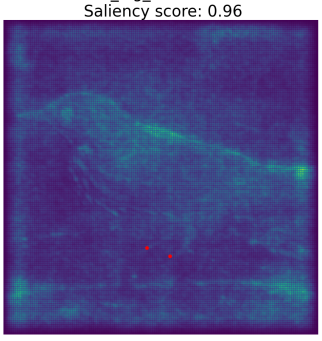
has_upperparts_color::iridescent



has_tail_shape::notched_tail
Saliency score: 0.94



has_leg_color::blue
Saliency score: 0.96



Concept	Orginal Lables	Majority-voted	Prediction	C to Y weight
has_upperparts_color::iridescent	False	False	0.00	-0.32
has_tail_shape::notched_tail	True	True	0.41	3.14
has_leg_color::blue	False	False	0.00	0.04
has_throat_color::red	False	False	0.00	-0.33
has_under_tail_color::red	False	False	0.00	0.20
has_leg_color::yellow	False	False	0.00	0.11
has_forehead_color::grey	False	False	0.02	-3.67
has_bill_color::brown	False	False	0.48	2.67
has_bill_color::purple	False	False	0.00	0.15
has_primary_color::olive	True	False	0.00	-0.12
has_underparts_color::brown	False	False	0.80	0.50
has_forehead_color::pink	False	False	0.00	0.08
has_crown_color::buff	False	False	0.00	4.10
has_size::medium_(9_-_16_in)	False	False	0.00	-4.47
has_underparts_color::purple	False	False	0.00	0.24
has_bill_color::orange	False	False	0.01	1.87
has_breast_color::orange	False	False	0.00	0.64
has_belly_color::red	False	False	0.00	-0.51
has_upper_tail_color::iridescent	False	False	0.00	-0.09
has_underparts_color::iridescent	False	False	0.00	-0.09
has_belly_color::blue	False	False	0.00	-0.01
has_back_color::white	False	False	0.00	-0.69
has_upperparts_color::pink	False	False	0.00	0.05
has_leg_color::brown	False	False	0.10	1.59
has_leg_color::orange	False	False	0.27	-0.16
has_belly_color::pink	False	False	0.00	0.34
has_tail_shape::rounded_tail	False	False	0.02	-0.33
has_underparts_color::black	True	False	0.43	3.06
has_back_color::pink	False	False	0.00	0.04
has_wing_color::purple	False	False	0.00	0.01
has_upper_tail_color::green	False	False	0.00	2.82
has_shape::gull-like	False	False	0.00	-0.37
has_eye_color::white	False	False	0.00	-2.32
has_head_pattern::crested	False	False	0.00	-0.26
has_breast_color::blue	False	False	0.00	-0.04
has_back_color::orange	False	False	0.00	0.50
has_upperparts_color::orange	False	False	0.00	1.23
has_head_pattern::striped	False	False	0.02	3.79
has_wing_color::buff	False	False	0.00	-2.79
has_throat_color::iridescent	False	False	0.00	-0.14
has_crown_color::iridescent	False	False	0.00	-0.45
has_leg_color::olive	False	False	0.00	-0.29
has_back_pattern::solid	True	True	1.00	0.59
has_back_color::olive	False	False	0.00	0.86
has_bill_length::shorter_than_head	True	True	0.85	1.44
has_bill_shape::hooked	False	False	0.00	-0.28
has_back_color::purple	False	False	0.00	-0.02
has_wing_color::green	False	False	0.00	1.51
has_underparts_color::green	False	False	0.00	0.32
has_wing_shape::broad-wings	False	False	0.00	-0.94

Figure A.20

Technical
University of
Denmark

Richard Petersens Plads Building 324
2800 Kgs. Lyngby
Tlf. 4525 1700

www.compute.dtu.dk/