Project 1: Concept Bottleneck Models

Abstract—**This paper is about exploring the concept bottleneck models in which concepts are used to predict the final output. Unlike the end-to-end models, correcting the concept prediction and propagating this modification to the target prediction are possible at the test time which leads to improving the model's accuracy. Furthermore, the different ways of learning the concept bottleneck models are proposed and the task error of these methods is replicated on the Caltech-UCSD Birds-200-2011 (CUB) dataset for comparison with the standard model. Two of these models are selected to train again from scratch to have concept bottleneck models. The effect of the test-time intervention on the performance of the concept bottleneck is studied.**

I. Introduction

Deep learning models are considered end-to-end models that are trained to predict the output directly from the input and there is no interpretation or intervention. For example, assume that a radiologist is assessing the severity of knee osteoarthritis with the help of a machine-learning model. The logic behind the model's prediction is important for the radiologist. If she adds extra information to the model based on her expertise, the model's prediction would be changed? Is it possible in an end-to-end machine learning model? To address this issue, the concept bottleneck models are proposed. Any end-to-end neural network can converted into this model by providing annotations for concepts in the training phase. It is shown that these bottleneck models achieve higher accuracies compared to standard models as we are allowed to modify the concept prediction to lead the model to predict correctly.

Three different learning methods are suggested to explore the interpretability and performance of concept bottleneck models for specific tasks. Assume y is a target that should be predicted from input x and c is the concept, thus $f(.)$ predicts $c \rightarrow y$ and $g(.)$ predicts $x \rightarrow c$. λ is the task-concept tradeoff hyperparameter. The standard model learns the target directly from the input without considering concepts $\hat{y} = f(g(x))$. The suggested learning methods are as follows:

- The Independent bottleneck: learn $\hat{y} = f(c)$ and $\hat{c} = g(x)$ separately.
- The Sequential bottleneck: first learn $\hat{c} = g(x)$ then use the prediction concept to learn $\hat{y} = f(g(x))$.
- The Joint bottleneck: learn $\hat{y} = f(g(x))$ and $\hat{c} = g(x)$ together in a way that minimizes the weighted sum f and g as $\hat{y}, \hat{c} = f(g(x)) + \lambda g(x)$ for some $\lambda > 0$.

Two different datasets are considered for assessing task accuracy and the high concept accuracy of the bottleneck model, an x-ray grading (OAI) [1] and a bird identification (CUB) [2] dataset. The OAI requires an application for data access, thus the rest of the paper is about replicating the result from the CUB dataset, which is public. The Caltech-UCSD Birds-200-2011 (CUB) dataset contains n = 11, 788 bird photographs that are classified into 200 bird species with k = 112 binary bird attributes representing wing color, beak shape, etc (Figure 1). The neural network architecture used for bird classification is Inception V3 [3]. This model without considering the fully-connected layers is pretrained on ImageNet and then the end-to-end model is fine-tuned [4] on the CUB dataset.

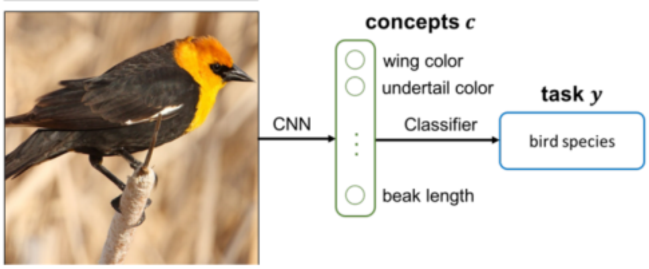


Fig. 1: The concept bottleneck model for bird classification

As the task of the CUB dataset is classification, f and g are turned into a probabilistic prediction to compute the real-valued scores. These models are trained to have a bottleneck layer and the predicted concepts can be read out from the bottleneck layer. This helps to manipulate the predicted concepts and explore the effect of this editing on the final prediction. The drawback of concept bottleneck models is requiring annotated concepts at training time. Also, some CUB concept annotations are noisy since they are provided by a single crowdworker (not a birding expert).

In contrast to the Independent bottleneck model, sequential bottleneck enables the $c \rightarrow y$ model to adjust based on its ability to forecast $x \rightarrow c$. In the joint bottleneck model, the concept model undergoes refinement in order to enhance predictive accuracy.

The rest of this paper is organized as follows. To validate the result obtained from the concept bottleneck model on the CUB dataset, the model is trained at least under one of the mentioned learning methods, and then the task error of Independent, Sequential, and Joint concept bottleneck models are compared with each other and with the standard model in section II. In section III, some examples of test-time intervention are provided to show the effect of editing the concept predictions in the target predictions. Conclusions are discussed in Section IV.

II. Validation

As aforementioned, there are three different kinds of methods to train the end-to-end model to have a concept bottleneck model. For this purpose, the CUB dataset and the processed CUB dataset are used to train the pretrained Inception V3 model for the bird classification task. The training process which is reported in [5] is done with a batch size of 64, and SGD with momentum of 0.9 as the optimizer. To assess the validity of this training process, it is tried to replicate this process to train the Joint model. This replication is done by changing the batch size to 32 due to a GPU (graphics processing unit) memory issue. This model is successfully trained and automatically stopped in 690 epochs with a training accuracy of 92.5468 and training loss of 0.2408 as shown in Figure 2.

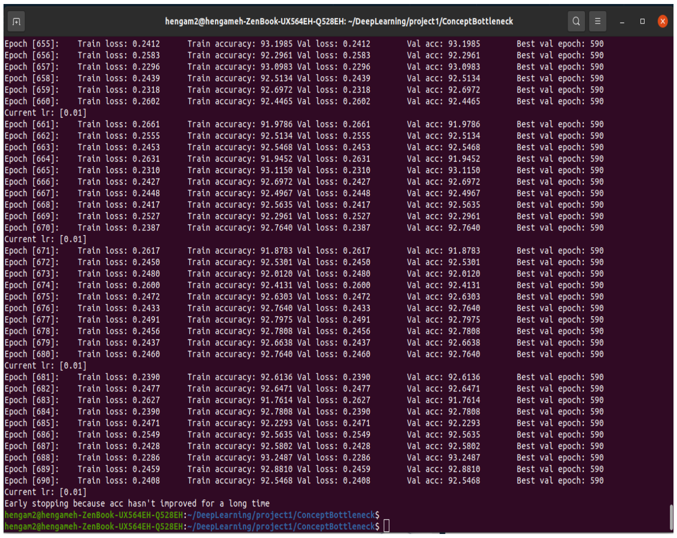
The training process of the Independent model, the part that predicts concepts from the input is also replicated. The training process lasts more than 24 hours and it is stopped in epoch 270 as reaches the training accuracy of 98.3087 and the training loss of 0.3173 as shown in Figure 3.

In addition, the performance of the obtained models (Independent, Sequential, and Joint models) is studied and compared with the standard model. In a way that each of these models is applied to test data to compute the error of concept prediction and target prediction. These models are obtained by training the end-to-end model under different training methods for bird classification. The predicted class labels obtained from each of these models are compared with ground truths to compute the error of concept prediction and target prediction as shown in Table I.

TABLE I: Task errors with ±2SD over concept bottleneck and compared with the standard end-to-end model

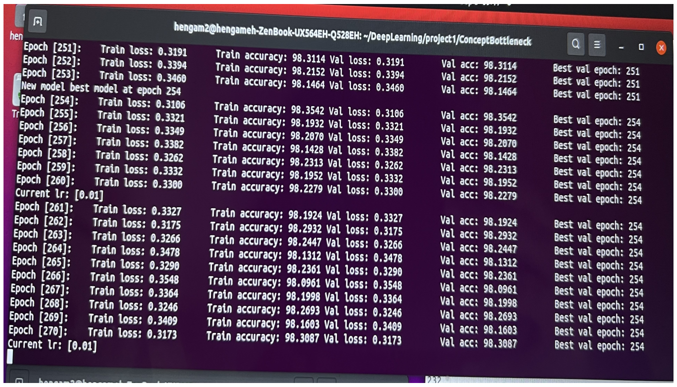
Model	Error of y	Error of c	Accuracy
Independent	0.2400 ± 0.0063	0.0328 ± 0.0009	0.97
Sequential	0.2425 ± 0.0028	0.0328 ± 0.0009	0.97
Joint(0.01)	0.1987 ± 0.0021	0.0297 ± 0.0002	0.97
Standard	0.1746 ± 0.0042	0.5001 ± 0.0029	0.50

As shown in Table I, the error of the Independent and Sequential concept bottleneck models in predicting the output is worse than the standard model while the error of the Joint concept bottleneck model is close to the Standard model. However, the accuracy of concept bottleneck models is better than the standard model. The prediction of concepts in concept bottleneck models is much better than the Standard model. The accuracy of the Independent and Sequential concept bottleneck models is better than the Joint concept bottleneck model and all of the concept bottleneck models have better accuracy compared to the Standard model. The values of this table are not exactly the same as those in Table 1 in the paper [5] but they follow the same pattern.



```
hengam2@hengameh-ZenBook-UX564EH-Q528EH: ~/DeepLearning/project1/ConceptBottleneck
Epoch [655]:    Train loss: 0.2412      Train accuracy: 93.1985 Val loss: 0.2412        Val acc: 93.1985        Best val epoch: 590
Epoch [656]:    Train loss: 0.2583      Train accuracy: 92.2961 Val loss: 0.2583        Val acc: 92.2961        Best val epoch: 590
Epoch [657]:    Train loss: 0.2296      Train accuracy: 93.0983 Val loss: 0.2296        Val acc: 93.0983        Best val epoch: 590
Epoch [658]:    Train loss: 0.2439      Train accuracy: 92.5134 Val loss: 0.2439        Val acc: 92.5134        Best val epoch: 590
Epoch [659]:    Train loss: 0.2318      Train accuracy: 92.6972 Val loss: 0.2318        Val acc: 92.6972        Best val epoch: 590
Epoch [660]:    Train loss: 0.2602      Train accuracy: 92.4465 Val loss: 0.2602        Val acc: 92.4465        Best val epoch: 590
Current lr: [0.01]
Epoch [661]:    Train loss: 0.2661      Train accuracy: 91.9786 Val loss: 0.2661        Val acc: 91.9786        Best val epoch: 590
Epoch [662]:    Train loss: 0.2555      Train accuracy: 92.5134 Val loss: 0.2555        Val acc: 92.5134        Best val epoch: 590
Epoch [663]:    Train loss: 0.2453      Train accuracy: 92.5468 Val loss: 0.2453        Val acc: 92.5468        Best val epoch: 590
Epoch [664]:    Train loss: 0.2631      Train accuracy: 91.9452 Val loss: 0.2631        Val acc: 91.9452        Best val epoch: 590
Epoch [665]:    Train loss: 0.2310      Train accuracy: 93.1150 Val loss: 0.2310        Val acc: 93.1150        Best val epoch: 590
Epoch [666]:    Train loss: 0.2427      Train accuracy: 92.6972 Val loss: 0.2427        Val acc: 92.6972        Best val epoch: 590
Epoch [667]:    Train loss: 0.2448      Train accuracy: 92.4967 Val loss: 0.2448        Val acc: 92.4967        Best val epoch: 590
Epoch [668]:    Train loss: 0.2417      Train accuracy: 92.5635 Val loss: 0.2417        Val acc: 92.5635        Best val epoch: 590
Epoch [669]:    Train loss: 0.2527      Train accuracy: 92.2961 Val loss: 0.2527        Val acc: 92.2961        Best val epoch: 590
Epoch [670]:    Train loss: 0.2387      Train accuracy: 92.7640 Val loss: 0.2387        Val acc: 92.7640        Best val epoch: 590
Current lr: [0.01]
Epoch [671]:    Train loss: 0.2617      Train accuracy: 91.8783 Val loss: 0.2617        Val acc: 91.8783        Best val epoch: 590
Epoch [672]:    Train loss: 0.2450      Train accuracy: 92.5301 Val loss: 0.2450        Val acc: 92.5301        Best val epoch: 590
Epoch [673]:    Train loss: 0.2480      Train accuracy: 92.0120 Val loss: 0.2480        Val acc: 92.0120        Best val epoch: 590
Epoch [674]:    Train loss: 0.2600      Train accuracy: 92.4131 Val loss: 0.2600        Val acc: 92.4131        Best val epoch: 590
Epoch [675]:    Train loss: 0.2472      Train accuracy: 92.6303 Val loss: 0.2472        Val acc: 92.6303        Best val epoch: 590
Epoch [676]:    Train loss: 0.2433      Train accuracy: 92.7640 Val loss: 0.2433        Val acc: 92.7640        Best val epoch: 590
Epoch [677]:    Train loss: 0.2491      Train accuracy: 92.7975 Val loss: 0.2491        Val acc: 92.7975        Best val epoch: 590
Epoch [678]:    Train loss: 0.2456      Train accuracy: 92.7808 Val loss: 0.2456        Val acc: 92.7808        Best val epoch: 590
Epoch [679]:    Train loss: 0.2437      Train accuracy: 92.6638 Val loss: 0.2437        Val acc: 92.6638        Best val epoch: 590
Epoch [680]:    Train loss: 0.2460      Train accuracy: 92.7640 Val loss: 0.2460        Val acc: 92.7640        Best val epoch: 590
Current lr: [0.01]
Epoch [681]:    Train loss: 0.2390      Train accuracy: 92.6136 Val loss: 0.2390        Val acc: 92.6136        Best val epoch: 590
Epoch [682]:    Train loss: 0.2477      Train accuracy: 92.6471 Val loss: 0.2477        Val acc: 92.6471        Best val epoch: 590
Epoch [683]:    Train loss: 0.2627      Train accuracy: 91.7614 Val loss: 0.2627        Val acc: 91.7614        Best val epoch: 590
Epoch [684]:    Train loss: 0.2390      Train accuracy: 92.7808 Val loss: 0.2390        Val acc: 92.7808        Best val epoch: 590
Epoch [685]:    Train loss: 0.2471      Train accuracy: 92.2293 Val loss: 0.2471        Val acc: 92.2293        Best val epoch: 590
Epoch [686]:    Train loss: 0.2549      Train accuracy: 92.5635 Val loss: 0.2549        Val acc: 92.5635        Best val epoch: 590
Epoch [687]:    Train loss: 0.2428      Train accuracy: 92.5802 Val loss: 0.2428        Val acc: 92.5802        Best val epoch: 590
Epoch [688]:    Train loss: 0.2286      Train accuracy: 93.2487 Val loss: 0.2286        Val acc: 93.2487        Best val epoch: 590
Epoch [689]:    Train loss: 0.2459      Train accuracy: 92.8810 Val loss: 0.2459        Val acc: 92.8810        Best val epoch: 590
Epoch [690]:    Train loss: 0.2408      Train accuracy: 92.5468 Val loss: 0.2408        Val acc: 92.5468        Best val epoch: 590
Current lr: [0.01]
Early stopping because acc hasn't improved for a long time
hengam2@hengameh-ZenBook-UX564EH-Q528EH:~/DeepLearning/project1/ConceptBottleneck$
hengam2@hengameh-ZenBook-UX564EH-Q528EH:~/DeepLearning/project1/ConceptBottleneck$
```

Fig. 2: The details of Joint model training process



```
hengam2@hengameh-ZenBook-UX564EH-Q528EH: ~/DeepLearning/project1/ConceptBottleneck
Epoch [251]:    Train loss: 0.3191      Train accuracy: 98.3114 Val loss: 0.3191        Val acc: 98.3114        Best val epoch: 251
Epoch [252]:    Train loss: 0.3394      Train accuracy: 98.2152 Val loss: 0.3394        Val acc: 98.2152        Best val epoch: 251
Epoch [253]:    Train loss: 0.3460      Train accuracy: 98.1464 Val loss: 0.3460        Val acc: 98.1464        Best val epoch: 251
New model best model at epoch 254
Epoch [254]:    Train loss: 0.3106      Train accuracy: 98.3542 Val loss: 0.3106        Val acc: 98.3542        Best val epoch: 254
Epoch [255]:    Train loss: 0.3321      Train accuracy: 98.1932 Val loss: 0.3321        Val acc: 98.1932        Best val epoch: 254
Epoch [256]:    Train loss: 0.3349      Train accuracy: 98.2070 Val loss: 0.3349        Val acc: 98.2070        Best val epoch: 254
Epoch [257]:    Train loss: 0.3382      Train accuracy: 98.1428 Val loss: 0.3382        Val acc: 98.1428        Best val epoch: 254
Epoch [258]:    Train loss: 0.3262      Train accuracy: 98.2313 Val loss: 0.3262        Val acc: 98.2313        Best val epoch: 254
Epoch [259]:    Train loss: 0.3332      Train accuracy: 98.1952 Val loss: 0.3332        Val acc: 98.1952        Best val epoch: 254
Epoch [260]:    Train loss: 0.3300      Train accuracy: 98.2279 Val loss: 0.3300        Val acc: 98.2279        Best val epoch: 254
Current lr: [0.01]
Epoch [261]:    Train loss: 0.3327      Train accuracy: 98.1924 Val loss: 0.3327        Val acc: 98.1924        Best val epoch: 254
Epoch [262]:    Train loss: 0.3175      Train accuracy: 98.2932 Val loss: 0.3175        Val acc: 98.2932        Best val epoch: 254
Epoch [263]:    Train loss: 0.3266      Train accuracy: 98.2447 Val loss: 0.3266        Val acc: 98.2447        Best val epoch: 254
Epoch [264]:    Train loss: 0.3478      Train accuracy: 98.1312 Val loss: 0.3478        Val acc: 98.1312        Best val epoch: 254
Epoch [265]:    Train loss: 0.3290      Train accuracy: 98.2361 Val loss: 0.3290        Val acc: 98.2361        Best val epoch: 254
Epoch [266]:    Train loss: 0.3548      Train accuracy: 98.0961 Val loss: 0.3548        Val acc: 98.0961        Best val epoch: 254
Epoch [267]:    Train loss: 0.3364      Train accuracy: 98.1998 Val loss: 0.3364        Val acc: 98.1998        Best val epoch: 254
Epoch [268]:    Train loss: 0.3246      Train accuracy: 98.2693 Val loss: 0.3246        Val acc: 98.2693        Best val epoch: 254
Epoch [269]:    Train loss: 0.3409      Train accuracy: 98.1603 Val loss: 0.3409        Val acc: 98.1603        Best val epoch: 254
Epoch [270]:    Train loss: 0.3173      Train accuracy: 98.3087 Val loss: 0.3173        Val acc: 98.3087        Best val epoch: 254
Current lr: [0.01]
```

Fig. 3: The detail training process of Independent $x \rightarrow c$ model

The aim of this paper is to show the effect of the intervention on the concept bottleneck model at test time on the final prediction. For this purpose, the paper shows the case study in two images of the CUB dataset which is shown in Figure 3 of the paper [5]. Replicating this part was challenging as the intervention process on concepts at test time is not elaborated in the paper or the GitHub repository. Based on Figure 3 of the paper [5], two images of the dataset are selected, it is tried to select those two same images (Glaucous_Winged_Gull_0112_44731.jpg and Worm_Eating_Warbler_0014_176042.jpg). To find those two images in the dataset, the test pickle file is read out and then tried to find images that have the same class labels (Glaucous_winged_Gull and Worm_eating_Warbler with class labels of 59 and 180) as those images in the paper.

As the paper has not specified which model was used to obtain those results, the performance of all models is surveyed on the new dataset. I figured out that seed1 of the Independent model predicts the class labels of 61 (Herring_Gull) and 98 (Ovenbird) as reported in the paper (Figure 4). Then, the test-time intervention with $n = 28$ is applied to those images, and the class prediction is changed to Glaucous_winged_Gull (59) and Ovenbird (98) (Figure 5). As shown, after concept intervention the model cannot predict the class label of the second image correctly. I checked all other models and different values for n to reach the same values reported in the paper but I did not succeed.

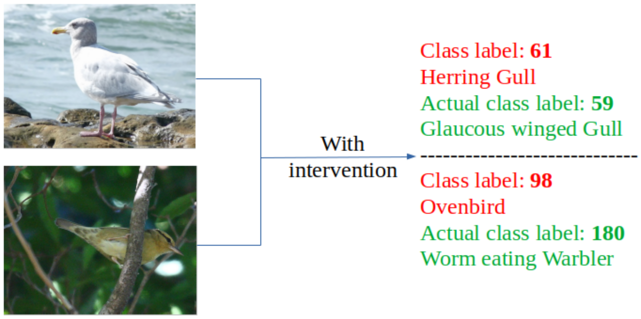


Fig. 4: Class prediction of two examples of images without intervention

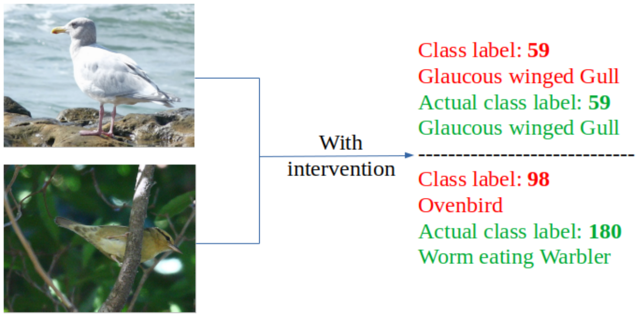


Fig. 5: Class prediction of two examples of images with intervention

TABLE II: Test-time intervention on different concept bottleneck models with $n = 1$

	Class prediction		
Number of Concepts Intervened	$n = 1$		
Bottleneck models	Seed1	Seed2	Seed3
Joint	[61, 98]	[61, 177]	[61, 98]
Independent	[61, 98]	[59, 98]	[65, 98]
Sequential	[61, 98]	[61, 98]	[61, 178]

III. Extension

In this section, it is tried to extend some methods proposed in the paper and study the effect of hyperparameters in the performance of concept bottleneck models. To better see the effect of editing the concepts at the test time on the class prediction, different bottleneck models and their seeds are considered as shown in Tables II and III. As you can see in some cases by changing at least one value of concepts the class prediction can be changed. Tables II and III show the class prediction of different models under the intervention of $n = 1$ and $n = 28$ (n is the number that is intervened). The dataset is as same as the previous part and the ground truth class labels are [59, 180]. It seems that the number of concepts that intervened has an effect on the model performance. Based on the graphs in Figure 4 of the paper, increasing the number of concepts intervened leads to decreasing the task error.

Based on the paper, as the CUB dataset is a classification, not a regression, the intervention on the CUB dataset is complicated. Assume $\hat{l} = \hat{g}(x)$, to modify a concept $\hat{c}j$, the true value cj cannot be replaced simply. Instead, we must adjust $\hat{l}j$ so that the probability of $\hat{c}j$ being 1 is close to the true value of cj. We intervene on $\hat{c}j$ by setting $\hat{l}j$ to either the 5th percentile (if $cj = 0$) or the 95th percentile (if $cj = 1$) of $\hat{l}j$ over the training distribution.

In addition, to better see the effect of concepts on class prediction, the whole concept values are changed to 0 and 1 and the results are shown in Table IV. As it is obvious, the second class label is finally predicted correctly when the whole values of concepts intervened are changed to 1 or 0 for the Joint model.

TABLE III: Test-time intervention on different concept bottleneck models with $n = 28$

	Class prediction		
Number of Concepts Intervened	$n = 28$		
Bottleneck models	Seed1	Seed2	Seed3
Joint	[61, 98]	[61, 180]	[61, 98]
Independent	[63, 98]	[59, 98]	[65, 98]
Sequential	[61, 98]	[59, 98]	[59, 36]

TABLE IV: Test-time intervention on different concept bottleneck models

	Class prediction	
Bottleneck models	$concepts = 0$	$concepts = 1$
Joint	[61, 180]	[61, 180]
Independent	[65, 98]	[65, 98]
Sequential	[59, 36]	[59, 36]

IV. Conclusion

We are trying to duplicate the results that are reported in the main paper. At first, we train two models, Joint (from input to output) and Independent (from input to concept). Then, the results of task errors overall seeds of bottleneck models are computed and compared with the Standard model. The effect of test time intervention on two specific images is also studied but replicating the result is not successful. The specific model used to achieve those results has not been mentioned in the paper, after investigating all the models, the model that has similar behavior to those results is selected. The test time intervention is also applied to this model but it leads to predicting one of the class labels correctly instead of both.

In the extension section, some other test time intervention is studied to show the role of concepts in predicting class labels in bottleneck models. The effect of the values of concept and the number of concepts intervened is studied on the performance of models. Concept bottleneck models can compete on task accuracy while supporting intervention and interpretation, however, a drawback of these models is that they require annotated concepts at training time.

References

[1] Nevitt, M., Felson, D. T., and Lester, G. The Osteoarthritis Initiative. Cohort study protocol, 2006.

[2] Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The Caltech-UCSD Birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

[3] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the Inception architecture for com- puter vision. In Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826, 2016.

[4] Cui, Y., Song, Y., Sun, C., Howard, A., and Belongie, S. Large-scale fine-grained categorization and domain-specific transfer learning. In Computer Vision and Pattern Recognition (CVPR), pp. 4109–4118, 2018.

[5] Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B. and Liang, P. Concept bottleneck models. In International conference on machine learning, 2020.