Accounting for Incomplete Concepts in Concept Bottleneck Models

Irena Gao
Stanford, CA
igao@stanford.edu

Jason Ah Chuen
Stanford, CA
jahchuen@stanford.edu

I. Introduction

Imagine a hospital is using a deep learning model to grade knee osteoarthritis from x-rays. Ideally, an expert radiologist could collaboratively *interact* with the model. If, at test-time, the radiologist disagrees with the model about some prediction, she might want to query *why* the model behaved the way it did — did the model detect joint space narrowing that she missed? Alternatively, she might want to *intervene* on the model — would the model change its prediction if she declared that there was no narrowing?

Recently, Koh et al. (2020) proposed Concept Bottleneck Models [4], a framework for models that support this kind of interaction. In a Concept Bottleneck Model (CBM), a network first learns to predict meaningful high-level concepts (*e.g.* joint space narrowing in knee x-rays) before predicting the final target (*e.g.* osteoarthritis severity). Importantly, our radiologist can intervene on a CBM at test-time by overriding the predicted concepts with her own judgement.

In this project, we extend the CBM paper to account for incomplete sets of concepts while preserving the effectiveness of test-time interventions. Specifically, we increase the width of the concept bottleneck layer to predict both specified concepts $\mathbf{c}$ and unspecified latent concepts $\mathbf{s}$. The final prediction y is based on both $\mathbf{c}$ and $\mathbf{s}$. We also examine the use of regularization to limit the network's reliance on $\mathbf{s}$ and preserve the effectiveness of interventions.

II. Literature Review

Formally, given data points $(x, \mathbf{c}, y)$, where x is the input image, $\mathbf{c}$ is a vector of concept labels, and y is the target, a Concept Bottleneck Model (CBM) predicts $x \to \hat{\mathbf{c}}$, and then $\hat{\mathbf{c}} \to \hat{y}$. Practically, a CBM looks like a deep neural network, with one "bottleneck" layer of length C set aside to predict the C concepts.

The advantage of CBMs is that they support test-time interactions; with a CBM, our radiologist can compare predicted concepts $\hat{\mathbf{c}}$ to her expert grading of concepts $\mathbf{c}_{\text{exp}}$ and, if necessary, intervene by setting $\hat{\mathbf{c}} = \mathbf{c}_{\text{exp}}$. The authors show that these test-time interventions can powerfully improve performance.

The major drawback of CBMs, however, is that they require full specification of concepts $\mathbf{c}$ *a priori*. For many tasks, the set of concepts is incomplete/unknown; we might not know what to look for in a knee x-ray, or we might want the model to discover concepts outside of our specified set. This suggests that it is still useful to maintain a direct channel from input to output $x \to y$, as in conventional deep networks. The challenge is that we must maintain the ability to intervene at test time – a network might learn to ignore concepts $\mathbf{c}$ altogether, rendering interventions on $\mathbf{c}$ useless.

Variable	Phenomenon	Bone Region
xrosfm	osteophytes	femur medial
xrscfm	sclerosis	femur medial
xrjsm	joint space narrowing	medial
xrostm	osteophytes	tibia medial
xrsctm	sclerosis	tibia medial
xrosfl	osteophytes	femur lateral
xrscfl	sclerosis	femur lateral
xrjsl	joint space narrowing	lateral
xrostl	osteophytes	tibia lateral
xrsctl	sclerosis	tibia lateral

TABLE I
All 10 OAI ordinal variables.

III. Dataset

We replicate the knee osteoarthritis grading task used in the original Concept Bottleneck Models paper [4]. The Osteoarthritis Initiative dataset (OAI) contains $n = 36,369$ x-ray images of individual knees from $4,172$ unique patients across multiple visits. Each image is accompanied by an expert evaluation of 10 ordinal variables (Table I) and a target Kellgren-Lawrence grade (KLG), which measures osteoarthritis severity [3]. The task is to predict the KLG grade y from the x-ray image x.

A. Selecting Concepts

In the original CBM paper, the authors provide all 10 variables in Table I as labeled concepts [4]. Using these 10 concepts, CBMs achieve competitive–and even slightly better–accuracy as compared to a baseline network that directly predicts $x \to y$.

We simulate having an incomplete set of concepts by only using a subset ($C = 6$) of the original 10 variables. We considered five sets of concepts (Table II). On each, we trained a standard CBM ($x \to \mathbf{c} \to y$) using a mini-dataset of $n = 500$.

We selected the incomplete set *No Tibia* for experimentation, which showed the largest degradation in accuracy ($+0.224$ Y RMSE).

	Y RMSE	C RMSE
No Sclerosis	0.490	0.706
No Osteoporosis	0.588	0.701
No Tibia	0.643	0.741
No Femur	0.614	0.801
Low Intervention Influence	0.588	0.706

TABLE II
Subsets of variables considered as concept sets. Each set consists of 6 variables of the original 10 in Table I. *No Sclerosis* removes four variables measuring sclerosis. *No Osteoporosis* removes four variables measuring osteoporosis. *No Tibia* removes four variables related to the tibia medial/lateral. *No Femur* removes four variables related to the femur medial/lateral. *Low Intervention Influence* removes the top four variables, ordered by influence on Y RMSE when intervened upon.

B. Preprocessing

We use three random splits: train ($n = 21,340$ from $2,456$ patients), validation ($n = 3,709$ from 421 patients), and test ($n = 11,320$ from $1,295$ patients), where no patient overlaps data splits. For preprocessing, each x-ray image is downsampled to 512×512 pixels and normalized by the maximum pixel value, followed by z-scoring.

Each of the 10 considered variables in Table I were z-scored using the training set. Any variables that were recorded as fractions (*e.g.* 1.2) were truncated to integers; this is because the decimal portion represents the temporal visit number and not a meaningful fractional grade.

As in the original CBM paper [4], we also use an adjusted 4-level KLG target (0 to 3) as opposed to the raw 5-level KLG score (0 to 4). This change was due to the data collection process, which makes KLG-0 and KLG-1 patients indistinguishable.

IV. Model

To predict $x \to \mathbf{c}$, we use a ResNet-18 [2] pre-trained on ImageNet [1]. A small 3-layer Multilayer Perceptron is used to predict $\mathbf{c} \to y$. The concept bottleneck layer is a fully-connected layer with C units, to match the number of provided concepts.

In our experiments, we compare two CBM training schemes:

1) *Using Joint Loss*: in this setup, the CBM trains all weights in $x \to \mathbf{c} \to y$ on the joint loss of $\text{Loss}(\mathbf{c}, \hat{\mathbf{c}}) + \text{Loss}(y, \hat{y})$
2) *Using Sequential Loss*: in this setup, the CBM first trains $x \to \mathbf{c}$ using $\text{Loss}(\mathbf{c}, \hat{\mathbf{c}})$. Then, we freeze the bottleneck concept weights and separately train $\mathbf{c} \to y$ on $\text{Loss}(y, \hat{y})$. Note that conv layers preceding the bottleneck layer are unfrozen so that they can learn new information to pass to the latent units.

For all models, we follow [4] and employ learning rate decay with decay factor of 2 every 10 epochs. All models are trained for 30 epochs with early stopping; the model weights are set at the end of training to those after the epoch with the lowest validation loss.

	Y RMSE	C RMSE
Original Paper	0.418 $\pm$0.004	0.543 $\pm$0.014
Our Setup	0.419	0.530

TABLE III
Standard CBM trained with joint loss using a complete set of 10 concepts. The original paper reported the average of 6 seeds, while we only used 1.

V. Baseline

A. Preliminary Experiment: Reproducing the CBM Paper

To test our set-up, we train a standard CBM with joint loss on $C = 10$ concepts and compare to the original paper's results. We matched the original paper's setup of training with initial learning rate $\eta = 0.005$. Table III shows our results compared to the original paper.

B. Standard CBM on Incomplete Concepts

For our baseline, we trained a standard CBM using joint loss for the set of $C = 6$ incomplete concepts *No Tibia*. The initial learning rate was $\eta = 0.0005$. Table IV shows the baseline performances.

VI. Main Approach

A. Direct $x \to y$ Channel

For a CBM to be effective, we must specify good concepts. Because our bottleneck layer has exactly C units, equal to the number of concepts, the CBM is constrained to predict y *only* based on $\mathbf{c}$. If $\mathbf{c}$ captures meaningful predictors of y, the CBM will perform well; if $\mathbf{c}$ is uncorrelated with y, no CBM can perform well. Our choice of concept set directly influences model performance.

However, in many tasks, the set of useful concepts is incomplete/unknown *a priori*. In these cases, we might want the model to discover latent concepts outside of our specified set. This suggests that it is still useful to maintain a direct channel from input to output $x \to y$, as in conventional deep networks.

We integrate this channel by increasing the width of the concept bottleneck layer to predict both specified concepts $\mathbf{c}$ and unspecified *latent* concepts $\mathbf{s}$. The final prediction y is based on both $\mathbf{c}$ and $\mathbf{s}$. As before, only $\mathbf{c}$, not $\mathbf{s}$ directly affects the loss. Figure 1 depicts our model architecture.

When training on joint loss, all 10 units are trained simultaneously. When training on sequential loss, $x \to \mathbf{c}$ has a bottleneck layer of only $C = 6$ units. When training $\mathbf{c} \to y$,

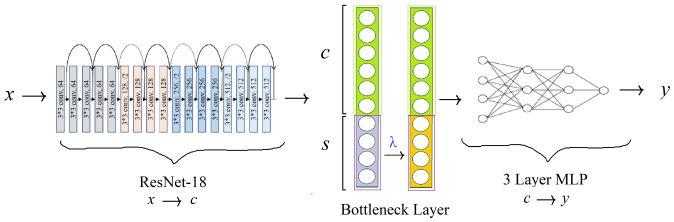

Fig. 1. Model architecture, using ResNet-18 for $x \to c$, a widened bottleneck layer, and a 3-Layer MLP for $c \to y$

Joint Loss			
Concept Set	**Model**	**Y RMSE**	**C RMSE**
No Tibia	$\lambda = 0.005$	0.433	0.555
	$\lambda = 0.05$	0.454	0.570
	$\lambda = 0.5$	0.430	0.533
	Baseline (Joint)	0.497	0.588

TABLE IV
Results for CBMs trained using the joint loss of $\text{Loss}(\mathbf{c}, \hat{\mathbf{c}}) + \text{Loss}(y, \hat{y})$. We evaluate four models varying by regularization strength λ on incomplete set *No Tibia*. Columns represent test Y RMSE and test C RMSE.

Sequential Loss			
Concept Set	**Model**	**Y RMSE**	**C RMSE**
No Tibia	$\lambda = 0.005$	0.425	0.533
	$\lambda = 0.05$	0.426	0.540
	$\lambda = 0.5$	0.432	0.556
	Baseline (Joint)	0.497	0.588

TABLE V
Results for CBMs trained using a sequential loss where $x \to \mathbf{c}$ is trained using $\text{Loss}(\mathbf{c}, \hat{\mathbf{c}})$, the bottleneck concept weights are frozen, and then $\mathbf{c} \to y$ is trained on $\text{Loss}(y, \hat{y})$. Note that the baseline is the same as in Table IV and was trained on a joint loss. We evaluate four models varying by regularization strength λ on incomplete set *No Tibia*. Columns represent test Y RMSE and test C RMSE.

we expand the width of the bottleneck layer and randomly initialize the new 4 weights.

Specifically, we use a 10-unit fully-connected layer for our bottleneck layer. The first $C = 6$ units are used to predict concepts $\mathbf{c}$, while the last $10 - C$ "latent" units $\mathbf{c}$ have no pre-specified semantic meaning. $\mathbf{s}$ is additionally passed through a fully-connected layer and ReLU before being concatenated with $\hat{\mathbf{c}}$.

Formally, let $h^{(c)}$ represent our concept bottleneck layer. Then

$$h^{(c)}[1:C] = \mathbf{c}, \quad h^{(c)}[C+1:10] = \mathbf{s} \tag{1}$$

The input to our 3-layer MLP predicting $\mathbf{c} \to y$ is

$$\text{concat}(\mathbf{c}, f(\mathbf{s})) \in \mathbb{R}^{10} \tag{2}$$

where f is a fully-connected layer followed by ReLU.

B. Regularization

The danger of with introducing the $x \to y$ channel is that it may impair our ability to intervene at test time. In the worst case, a network might learn to ignore concepts $\mathbf{c}$ altogether, predicting y only from $\mathbf{s}$, rendering interventions on $\mathbf{c}$ useless.

We counter this using simple L2 regularization on the effect of $\mathbf{s}$. Specifically, we regularize the weights of f that $\mathbf{s}$ is passed through in eq. (2).

We control the regularization strength with hyperparameter λ. To study the effect of regularization, we systematically study $\lambda \in [0.005, 0.05, 0.5]$.

VII. Evaluation Metric

For all experiments, we evaluate model performance using the root mean squared error (RMSE) of y and $\mathbf{c}$ in all cases.

$$\text{RMSE}(y) = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{3}$$

$$\text{RMSE}(c) = \sqrt{\sum_{i=1}^{n} \sum_{k=1}^{C} \frac{(\hat{c}_i^{(k)} - c_i^{(k)})^2}{n}} \tag{4}$$

A. Interventions

We define an intervention on a test point x as correcting some predicted concept $\hat{c}^{(k)}$ by overwriting it with the true value $c^{(k)}$. The model's responsiveness to the intervention is given by comparing its Y RMSE before and after the intervention.

We measure the effect of intervening on m concepts, $m \in [0, 1, \cdots, C-1]$. When $m = 0$, we make no intervention; when $m = C - 1$, we correct the values of $C - 1$ concepts. As in [4], we correct concepts by order of largest intervention influence; the intervention which marginally corrects performance the most is applied first.

VIII. Results and Analysis

Table IV shows model performances using joint training, and Table V shows performances using sequential training. We will analyze the effect of regularization in each case, and then compare the two.

A. Joint Loss

On the whole, our wider CBM models outperform the narrower Baseline (Table IV). Our worst-performing model (CBM with $\lambda = 0.05$) outperforms the baseline model by a margin of 0.043 Y RMSE, roughly a 9% decrease. This verifies our hypothesis that latent units can compensate for an incomplete set of concepts.

We note that, surprisingly, our models outperform the baseline not only in task accuracy (Y RMSE), but also in concept accuracy (C RMSE); our worst-performing model improves C RMSE by a margin of 0.018. We interpret this result in the next section.

The effect of regularization on our joint models is less clear. We hypothesized that stronger regularization should result in worse performance; under this hypothesis, the RMSEs in Table IV should strictly increase. This is not the case: instead of the expected $\text{RMSE}_{\lambda=0.005} < \text{RMSE}_{\lambda=0.05} < \text{RMSE}_{\lambda=0.5}$, we observe that $\text{RMSE}_{\lambda=0.5} < \text{RMSE}_{\lambda=0.005} < \text{RMSE}_{\lambda=0.05}$, a somewhat bizarre result. Additionally, we hypothesized that stronger regularization would lead to more effective interventions, *i.e.* the slope of the $\lambda = 0.5$ model in Figure 2 should be more steeply negative than the $\lambda = 0.005$ model. This also does not appear to be the case; if anything, models improve to interventions equally well.

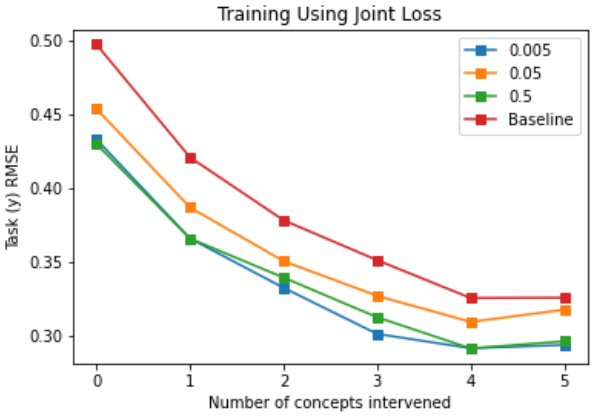

Fig. 2. Intervention effectiveness for CBMs trained using joint loss. Lines are color-coded by model; we evaluate four models varying by regularization strength λ on incomplete set *No Tibia*. Values plot test Y RMSE by the number of concepts intervened upon (m).

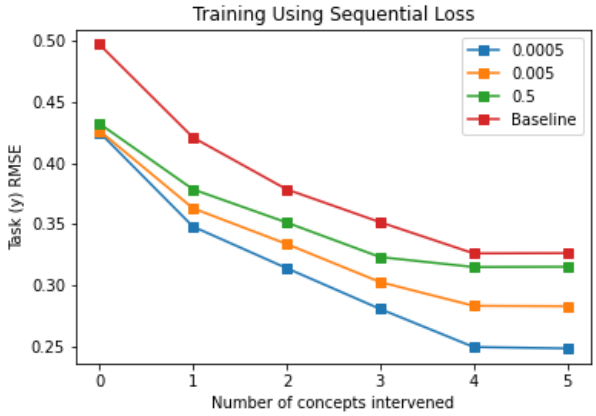

Fig. 3. Intervention effectiveness for CBMs trained sequential joint loss. Lines are color-coded by model; we evaluate four models varying by regularization strength λ on incomplete set *No Tibia*. Note that the baseline is the same as in Table IV and was trained on a joint loss. Values plot test Y RMSE by the number of concepts intervened upon (m).

We believe that our inconsistent results are due to the large model space of training all 10 bottleneck units simultaneously. The model behaves inconsistently as it swings between optimizing $\mathbf{c}$ and $\mathbf{s}$. We hypothesized that freezing the concept units via training on sequential loss might give more consistent results.

B. Sequential Loss

As expected, training on sequential loss improved result consistency. As before, our wider CBM models outperform the narrower Baseline in both Y RMSE and C RMSE (Table V). This second result is surprising; although we froze the concept unit weights in the bottleneck, it appears that learning latent units $\mathbf{s}$ guided by $Loss(y, \hat{y})$ pushes earlier convolutional filters to learn better representations that also improve $Loss(\mathbf{c}, \hat{\mathbf{c}})$. We believe this also explains the boost in C RMSE observed using joint loss.

Our sequential models behave under regularization as we hypothesized — stronger regularization worsens performance, and the Y RMSEs in Table V strictly increase. Whereas we hypothesized that stronger regularization would lead to more effective interventions, however, Figure 3 again suggests that all of our models notably improve with interventions; the weakest regularization level ($\lambda = 0.005$) actually shows the largest improvement with interventions. We note that the baseline model also responds comparatively better to interventions; whereas the baseline error starts much higher ($+0.065$ Y RMSE) than the $\lambda = 0.5$ model initially, by $m = 5$ interventions the two perform similarly.

Our results suggest that, contrary to our hypothesis, there is no tradeoff between initial accuracy and intervention effectiveness in wider models. Our lowest level of regularization ($\lambda = 0.005$) performs significantly better than our highly regularized model ($\lambda = 0.5$) both initially and after interventions. We note that the small width of $\mathbf{s}$ (4) is itself a form of regularization, and we would expect a tradeoff to emerge as the width of $\mathbf{s}$ increases.

IX. Future Work

Our results suggest that incorporating latent units to account for incomplete sets of concepts is an effective extension of Concept Bottleneck Models. We remain unsure, however, of how this result generalizes to (a) different widths of $\mathbf{s}$, as outlined in the previous section, and to (b) different datasets. Initial work by [4] suggests that other datasets may show drastically different results, such as the Caltech-UCSD Birds-200-2011 (CUB) dataset [5].

In general, our work confirms that Concept Bottleneck Models are a valuable approach to maintaining model *interactiveness* through test-time interventions. Small boosts in accuracy via wider bottleneck layers do not detract from this asset.

X. Code and Acknowledgements

Our experiment code can be found at https://github.com/jason-alouda/ConceptBottleneck/.
Our repo is forked from the original experiment's at https://github.com/yewsiang/ConceptBottleneck/.

Huge thanks to Pang Wei for authoring the original paper, securing dataset and compute access, and advising us through this project.

References

[1] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (2009), Ieee, pp. 248–255.

[2] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[3] Kellgren, J., and Lawrence, J. Radiological assessment of osteo-arthrosis. *Annals of the rheumatic diseases 16*, 4 (1957), 494.

[4] KOH, P. W., NGUYEN, T., TANG, Y. S., MUSSMANN, S., PIERSON, E., KIM, B., AND LIANG, P. Concept bottleneck models. *arXiv preprint arXiv:2007.04612* (2020).
[5] WAH, C., BRANSON, S., WELINDER, P., PERONA, P., AND BELONGIE, S. The caltech-ucsd birds-200-2011 dataset.