Concept Bottleneck Models

Pang Wei Koh [*1] **Thao Nguyen** [*1 2] **Yew Siang Tang** [*1]
Stephen Mussmann [1] **Emma Pierson** [1] **Been Kim** [2] **Percy Liang** [1]

Abstract

We seek to learn models that we can interact with using high-level concepts: if the model did not think there was a bone spur in the x-ray, would it still predict severe arthritis? State-of-the-art models today do not typically support the manipulation of concepts like "the existence of bone spurs", as they are trained end-to-end to go directly from raw input (e.g., pixels) to output (e.g., arthritis severity). We revisit the classic idea of first predicting concepts that are provided at training time, and then using these concepts to predict the label. By construction, we can intervene on these *concept bottleneck models* by editing their predicted concept values and propagating these changes to the final prediction. On x-ray grading and bird identification, concept bottleneck models achieve competitive accuracy with standard end-to-end models, while enabling interpretation in terms of high-level clinical concepts ("bone spurs") or bird attributes ("wing color"). These models also allow for richer human-model interaction: accuracy improves significantly if we can correct model mistakes on concepts at test time.

1. Introduction

Suppose that a radiologist is collaborating with a machine learning model to grade the severity of knee osteoarthritis. She might ask why the model made its prediction—did it deem the space between the knee joints too narrow? Or she might seek to intervene on the model—if she told it that the x-ray showed a bone spur, would its prediction change?

State-of-the-art models today do not typically support such queries: they are end-to-end models that go directly from raw input x (e.g., pixels) to target y (e.g., arthritis severity),

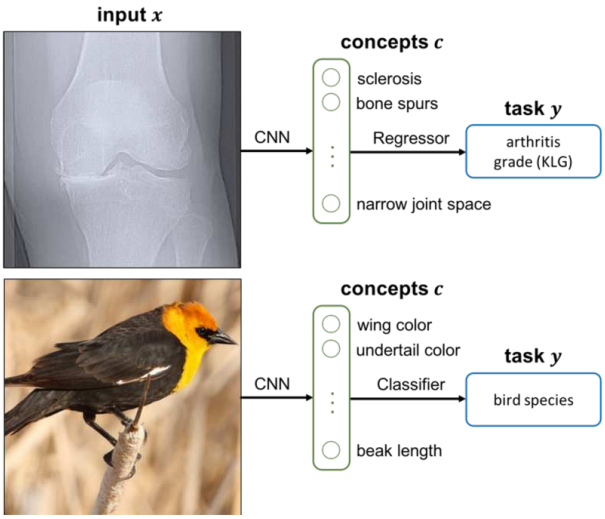

Figure 1. We study concept bottleneck models that first predict an intermediate set of human-specified concepts c, then use c to predict the final output y. We illustrate the two applications we consider: knee x-ray grading and bird identification.

and we cannot easily interact with them using the same high-level concepts that practitioners reason with, like "joint space narrowing" or "bone spurs".

We approach this problem by revisiting the simple idea of first predicting an intermediate set of human-specified concepts c like "joint space narrowing" and "bone spurs", then using c to predict the target y. In this paper, we refer to such models as *concept bottleneck models*. These models are trained on data points (x, c, y), where the input x is annotated with both concepts c and target y. At test time, they take in an input x, predict concepts $\hat{c}$, and then use those concepts to predict the target $\hat{y}$ (Figure 1).

Earlier versions of concept bottleneck models were overtaken in predictive accuracy by end-to-end neural networks (e.g., Kumar et al. (2009) for face recognition and Lampert et al. (2009) for animal identification), leading to a perceived tradeoff between accuracy and interpretability in terms of concepts. Recently, concept bottleneck models have started to re-emerge as targeted tools for solving particular tasks (Fauw et al., 2018; Yi et al., 2018; Bucher et al., 2018; Losch et al., 2019; Chen et al., 2020).

[*]Equal contribution [1]Stanford University [2]Google Research. Correspondence to: Pang Wei Koh <pangwei@cs.stanford.edu>, Been Kim <beenkim@google.com>, Percy Liang <pliang@cs.stanford.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

In this paper, we propose a straightforward method for turning any end-to-end neural network into a concept bottleneck model, given concept annotations at training time: we simply resize one of the layers to match the number of concepts provided, and add an intermediate loss that encourages the neurons in that layer to align component-wise to the provided concepts. We show that concept bottleneck models trained in this manner can achieve task accuracies competitive with or even higher than standard models. We emphasize that concept annotations are not needed at test time; the model predicts the concepts, then uses the predicted concepts to make a final prediction.

Importantly—and unlike standard models—these bottleneck models allow us to intervene on concepts by editing the concept predictions $\hat{c}$ and propagating those changes to the target prediction $\hat{y}$. Interventions enable richer human-model interaction: e.g., if the radiologist realizes that what the model thinks is a bone spur is actually an artifact, she can update the model's prediction by directly changing the corresponding value of $\hat{c}$. When we simulate this injection of human knowledge by partially correcting concept mistakes that the model makes at test time, we find that accuracy improves substantially beyond that of a standard model.

Interventions also make concept bottleneck models interpretable in terms of high-level concepts: by manipulating concepts $\hat{c}$ and observing the model's response, we can obtain counterfactual explanations like "if the model did not think the joint space was too narrow for this patient, then it would not have predicted severe arthritis". In contrast, prior work on explaining end-to-end models in terms of high-level concepts has been restricted to post-hoc interpretation of already-trained models: e.g., predicting concepts from hidden layers (Kim et al., 2018) or measuring the correlation of individual neurons with concepts (Bau et al., 2017).

The validity of interventions on a model depends on the alignment between its predicted concepts $\hat{c}$ and the true concepts c. We can estimate this alignment by measuring the model's concept accuracy on a held-out validation set (Fong & Vedaldi, 2017).[1] A model with perfect concept accuracy across all possible inputs makes predictions $\hat{c}$ that align with the true concepts c. Conversely, if a model has low concept accuracy, then the model's predictions $\hat{c}$ need not match with the true concepts, and we would not expect interventions to lead to meaningful results.

Contributions. We systematically study variants of concept bottleneck models and contrast them with standard end-to-end models in different settings, with a focus on the previously-unexplored ability of concept bottleneck models to support concept interventions. Our goal is to characterize concept bottleneck models more fully: Is there a tradeoff between task accuracy and concept interpretability? Do interventions at test time help model accuracy, and is concept accuracy a good indicator of the ability to effectively intervene? Do different ways of training bottleneck models lead to significantly different outcomes in intervention?

We evaluate concept bottleneck models on the two applications in Figure 1: the osteoarthritis grading task (Nevitt et al., 2006) and a fine-grained bird species identification task (Wah et al., 2011). On these, we show that bottleneck models are comparable to standard end-to-end models while also attaining high concept accuracies. In contrast, the concepts cannot be predicted with high accuracy from linear combinations of neurons in a standard black-box model, making it difficult to do post-hoc interpretation in terms of concepts like in Kim et al. (2018). We demonstrate that we can substantially improve model accuracy by intervening on these bottleneck models at test time to correct model mistakes on concepts, and we show that different methods of training bottleneck models lead to different trade-offs between task accuracy with and without interventions. Finally, we show that bottleneck models guided to learn the right concepts can also be more robust to covariate shifts.

2. Related work

Concept bottleneck models. Models that bottleneck on human-specified concepts—where the model first predicts the concepts, then uses only those predicted concepts to make a final prediction—have been previously used for specific applications (Kumar et al., 2009; Lampert et al., 2009). Early versions did not use end-to-end neural networks, which soon overtook them in predictive accuracy. Consequently, bottleneck models have historically been more popular for few-shot learning settings, where shared concepts might allow generalization to unseen contexts, rather than the standard supervised setting we consider here.

More recently, deep neural networks with concept bottlenecks have re-emerged as targeted tools for solving particular tasks, e.g., Fauw et al. (2018) for retinal disease diagnosis, Yi et al. (2018) for visual question-answering, and Bucher et al. (2018) for content-based image retrieval. Losch et al. (2019) and Chen et al. (2020) also explore learning concept-based models via auxiliary datasets.

Feature engineering. Constructing a concept bottleneck model is similar to traditional feature engineering (Lewis, 1992; Zheng & Casari, 2018; Nixon & Aguado, 2019) in that both require specifying intermediate concepts/features. However, they differ in an important way: in the former, we learn mappings from raw input to high-level concepts, whereas in the latter, we construct low-level features that can be computed from the raw input by handwritten functions.

[1] With the usual caveats of measuring accuracy: in practice, the validation set might be skewed such that models that learn spurious correlations can still achieve high concept accuracy.

Concepts as auxiliary losses or features. Non-bottleneck models that use human-specified concepts commonly use them in auxiliary objectives in a multi-task setup, or as auxiliary features; examples include using object parts (Huang et al., 2016; Zhou et al., 2018), parse trees (Zelenko et al., 2003; Bunescu & Mooney, 2005), or natural language explanations (Murty et al., 2020). However, these models do not support intervention on concepts. For instance, consider a multi-task model $c \leftarrow x \rightarrow y$, with the concepts c used in an auxiliary loss; simply intervening on $\hat{c}$ at test time will not affect the model's prediction of y. Interventions do affect models that use c as auxiliary features by first predicting $x \rightarrow c$ and then predicting $(x, c) \rightarrow y$ (e.g., Sutton & McCallum (2005)), but we cannot intervene in isolation on a single concept because of the side channel from $x \rightarrow y$.

Causal models. We emphasize that we study interventions on the value of a predicted concept within the model, not on that concept in reality. In other words, we are interested in how changing the model's predicted concept values $\hat{c}$ would affect its final prediction $\hat{y}$, and not in whether intervening on the true concept value c in reality would actually affect the true label y. This is similar to the notion of causality explored by other concept-based interpretability methods, e.g., in Goyal et al. (2019) and O'Shaughnessy et al. (2020).

More broadly, many others have studied learning models of actual causal relationships in the world (Pearl, 2000). While concept bottleneck models can represent causal relationships between $x \rightarrow c \rightarrow y$ if the set of concepts c is chosen appropriately, they have the advantage of being flexible and do not require c to cause y. For example, imagine that arthritis grade (y) is highly correlated with swelling (c). In this case, c does not cause y (hypothetically, if one could directly induce swelling in the patient, it would not affect whether they had osteoarthritis). However, concept bottleneck models can still exploit the fact that c is highly predictive for y, and intervening on the model by replacing the predicted concept value $\hat{c}$ with the true value c can still improve accuracy, even if c does not cause y.

Post-hoc concept analysis. Many methods have been developed to interpret models post-hoc, including recent work on using human-specified concepts to generate explanations (Bau et al., 2017; Kim et al., 2018; Zhou et al., 2018; Ghorbani et al., 2019). These techniques rely on models automatically learning those concepts despite not having explicit knowledge of them, and can be particularly useful when paired with models that attempt to learn more interpretable representations (Bengio et al., 2013; Chen et al., 2016; Higgins et al., 2017; Melis & Jaakkola, 2018). However, post-hoc methods can fail when the models do not learn these concepts, and also do not admit straightforward interventions on concepts. In this work, we instead directly guide models to learn these concepts at training time.

3. Setup

Consider predicting a target $y \in \mathbb{R}$ from input $x \in \mathbb{R}^d$; for simplicity, we present regression first and discuss classification later. We observe training points $\{(x^{(i)}, y^{(i)}, c^{(i)})\}_{i=1}^n$, where $c \in \mathbb{R}^k$ is a vector of k concepts. We consider bottleneck models of the form $f(g(x))$, where $g : \mathbb{R}^d \rightarrow \mathbb{R}^k$ maps an input x into the concept space ("bone spurs", etc.), and $f : \mathbb{R}^k \rightarrow \mathbb{R}$ maps concepts into a final prediction ("arthritis severity"). We call these *concept bottleneck models* because their prediction $\hat{y} = f(g(x))$ relies on the input x entirely through the bottleneck $\hat{c} = g(x)$, which we train to align component-wise to the concepts c. We define *task accuracy* as how accurately $f(g(x))$ predicts y, and *concept accuracy* as how accurately $g(x)$ predicts c (averaged over each concept). We will refer to $g(\cdot)$ as predicting $x \rightarrow c$, and to $f(\cdot)$ as predicting $c \rightarrow y$.

In our work, we systematically study different ways of learning concept bottleneck models. Let $L_{C_j} : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}_+$ be a loss function that measures the discrepancy between the predicted and true j-th concept, and let $L_Y : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}_+$ measure the discrepancy between predicted and true targets. We consider the following ways to learn a concept bottleneck model $(\hat{f}, \hat{g})$:

1. The *independent bottleneck* learns $\hat{f}$ and $\hat{g}$ independently: $\hat{f} = \arg\min_f \sum_i L_Y(f(c^{(i)}); y^{(i)})$, and $\hat{g} = \arg\min_g \sum_{i,j} L_{C_j}(g_j(x^{(i)}); c_j^{(i)})$. While $\hat{f}$ is trained using the true c, at test time it still takes $\hat{g}(x)$ as input.

2. The *sequential bottleneck* first learns $\hat{g}$ in the same way as above. It then uses the concept predictions $\hat{g}(x)$ to learn $\hat{f} = \arg\min_f \sum_i L_Y(f(\hat{g}(x^{(i)})); y^{(i)})$.

3. The *joint bottleneck* minimizes the weighted sum $\hat{f}, \hat{g} = \arg\min_{f,g} \sum_i \left[L_Y(f(g(x^{(i)})); y^{(i)}) + \sum_j \lambda L_{C_j}(g(x^{(i)}); c^{(i)}) \right]$ for some $\lambda > 0$.

As a control, we also study the *standard model*, which ignores concepts and directly minimizes $\hat{f}, \hat{g} = \arg\min_{f,g} \sum_i L_Y(f(g(x^{(i)})); y^{(i)})$.

The hyperparameter λ in the joint bottleneck controls the tradeoff between concept vs. task loss. The standard model is equivalent to taking $\lambda \rightarrow 0$, while the sequential bottleneck can be viewed as taking $\lambda \rightarrow \infty$. Compared to independent bottlenecks, sequential bottlenecks allow the $c \rightarrow y$ part of the model to adapt to how well it can predict $x \rightarrow c$; and joint bottlenecks further allow the model's version of the concepts to be refined to improve predictive performance.

We propose a simple scheme to turn an end-to-end neural network into a concept bottleneck model: simply resize

Table 1. Task errors with ±2SD over random seeds. Overall, independent, sequential, and joint concept bottleneck models are comparable to standard end-to-end models on task error. Removing the bottleneck from the standard model ("no bottleneck") does not significantly affect task error. Multi-task learning on the standard models further improves task error, but does not allow for interventions.

MODEL	y RMSE (OAI)	y ERROR (CUB)
INDEPENDENT	0.435± 0.024	0.240±0.012
SEQUENTIAL	0.418± 0.004	0.243±0.006
JOINT	0.418± 0.004	0.199±0.006
STANDARD	0.441± 0.006	0.175±0.008
NO BOTTLENECK	0.443 ± 0.008	0.173±0.003
MULTITASK	0.425± 0.010	0.162±0.002

one of its layers to have k neurons to match the number of concepts k, then choose one of the training schemes above.

Classification. In classification, f and g compute real-valued scores (e.g., concept logits $\hat{\ell} = \hat{g}(x) \in \mathbb{R}^k$) that we then turn into a probabilistic prediction (e.g., $P(\hat{c}_j = 1) = \sigma(\hat{\ell}_j)$ for logistic regression). This does not change the independent bottleneck, since f is directly trained on the binary-valued c. For the sequential and joint bottlenecks, we connect f to the logits $\hat{\ell}$, i.e., we compute $P(\hat{c}_j = 1) = \sigma(\hat{g}_j(x))$ and $P(\hat{y} = 1) = \sigma(\hat{f}(\hat{g}(x)))$.

4. Benchmarking bottleneck model accuracy

We start by showing that concept bottleneck models achieve both competitive task accuracy and high concept accuracy. While this is necessary for bottleneck models to be viable in practice, their strength is that we can interpret and intervene on them; we explore those aspects in Sections 5 and 6.

4.1. Applications

We consider an x-ray grading and a bird identification task. Their corresponding datasets are annotated with high-level concepts that practitioners (radiologists/birders) use to reason about their decisions. (Dataset details in Appendix A.)

X-ray grading (OAI). We use knee x-rays from the Osteoarthritis Initiative (OAI) (Nevitt et al., 2006), which compiles radiological and clinical data on patients at risk of knee osteoarthritis (Figure 1-Top; $n = 36,369$ data points). Given an x-ray, the task is to predict the Kellgren-Lawrence grade (KLG), a 4-level ordinal variable assessed by radiologists that measures the severity of osteoarthritis, with higher scores denoting more severe disease.[2] As concepts, we use $k = 10$ ordinal variables describing joint space narrowing, bone spurs, calcification, etc.; these clinical concepts are also assessed by radiologists and used directly in the assessment of KLG (Kellgren & Lawrence, 1957).

[2]Due to technicalities in the data collection protocol, we use a modified version of KLG where the first two grades are combined.

Table 2. Average concept errors. Bottleneck models have lower error than linear probes on standard and SENN models.

MODEL	c RMSE (OAI)	c ERROR (CUB)
INDEPENDENT	0.529±0.004	0.034±0.002
SEQUENTIAL	0.527±0.004	0.034±0.002
JOINT	0.543±0.014	0.031±0.000
STANDARD [PROBE]	0.680±0.038	0.093±0.004
SENN [PROBE]	0.676±0.026	-

Bird identification (CUB). We use the Caltech-UCSD Birds-200-2011 (CUB) dataset (Wah et al., 2011), which comprises $n = 11,788$ bird photographs (Figure 1-Bot). The task is to classify the correct bird species out of 200 possible options. As concepts, we use $k = 112$ binary bird attributes representing wing color, beak shape, etc. Because the provided concepts are noisy (see Appendix A), we denoise them by majority voting, e.g., if more than 50% of crows have black wings in the data, then we set all crows to have black wings. In other words, we use class-level concepts and assume that all birds of the same species in the training data share the same concept annotations. In contrast, the OAI dataset uses instance-level concepts: examples with the same y can have different concept annotations c.

Models. For each task, we construct concept bottleneck models by adopting model architectures and hyperparameters from previous high-performing approaches; see Appendix B for experimental details. For the joint bottleneck model, we search over the task-concept tradeoff hyperparameter λ and report results for the model that has the highest task accuracy while maintaining high concept accuracy on the validation set ($\lambda = 1$ for OAI and $\lambda = 0.01$ for CUB). We model x-ray grading as a regression problem (minimizing mean squared error) on both the KLG target y and concepts c, following Pierson et al. (2019); we learn g, which goes from $x \to c$, by fine-tuning a pretrained ResNet-18 model (He et al., 2016), and we learn f, which goes from $c \to y$, by training a small 3-layer multi-layer perceptron. We model bird identification as multi-class classification for the species y and binary classification for the concepts c. Following Cui et al. (2018), we learn g by fine-tuning an Inception-v3 network (Szegedy et al., 2016), and learn f by training a single linear layer (i.e., logistic regression).

4.2. Task and concept accuracies

Table 1 shows that concept bottleneck models achieve comparable task accuracy to standard black-box models on both tasks, despite the bottleneck constraint (all numbers reported are on a held-out test set). On OAI, joint and sequential bottlenecks are actually better in root mean square error

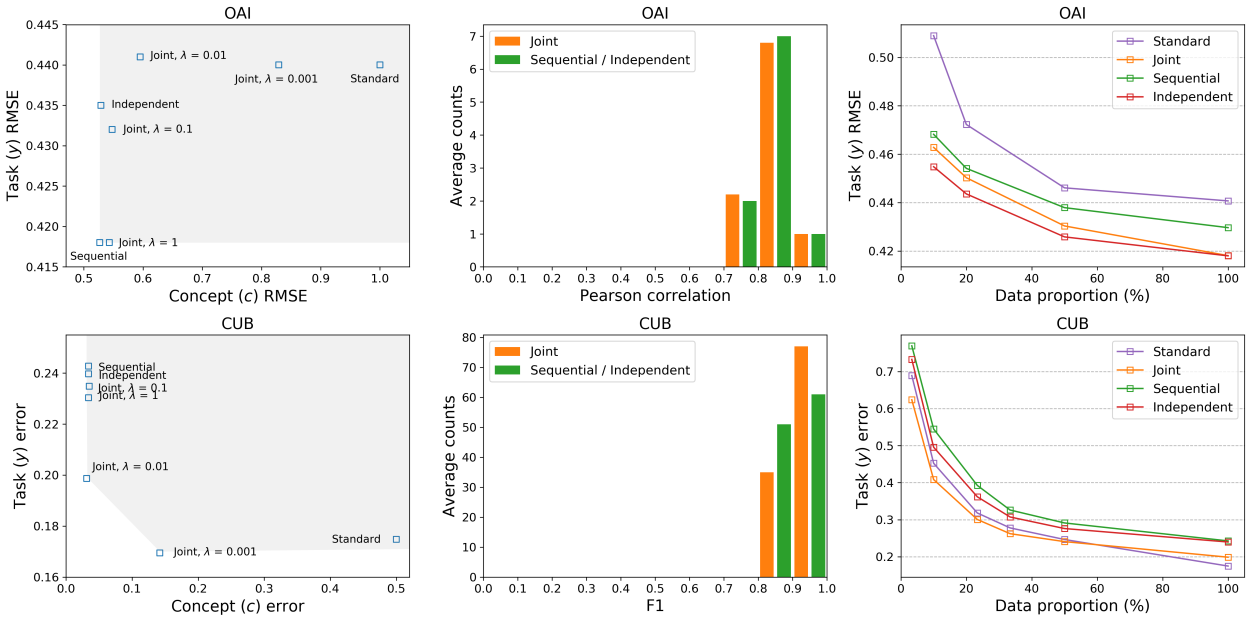

Figure 2. **Left**: The shaded regions show the optimal frontier between task vs. concept error. On OAI, we find little trade-off; models can do well on both task and concept prediction. On CUB, there is some trade-off, with standard models and joint models that prioritize task prediction (i.e., with sufficiently low λ) having lower task error. For standard models, we plot the concept error of the mean predictor (OAI) or random predictor (CUB). **Mid**: Histograms of how accurate individual concepts are, averaged over multiple random seeds. In our tasks, each individual concept can be accurately predicted by bottleneck models. **Right**: Data efficiency curves. Especially on OAI, bottleneck models can achieve the same task accuracy as standard models with many fewer training points.

(RMSE) than the standard model,[3] and on CUB, sequential and independent bottlenecks are worse in 0-1 error than the standard model, though the joint model (which is allowed to modify the concepts to improve task performance) closes most of the gap.

At the same time, the bottleneck models are able to accurately predict each concept well (Figure 2), and they achieve low average error across all concepts (Table 2). As discussed in Section 1, low concept error suggests that the model's concepts are aligned with the true concepts, which in turn suggests that we might intervene effectively on them; we will explore this in Section 6.

Overall, we do not observe a tradeoff between high task accuracy and high concept accuracy: pulling the bottleneck layer towards the concepts c does not substantially affect the model's ability to predict y in our tasks, even when the bottleneck is trained jointly. We illustrate this in Figure 2-Left, which plots the task vs. concept errors of each model.

Additional baselines. We ran two further baselines to determine if the bottleneck architecture impacted model performance. First, standard models in the literature generally do not use architectures that have a bottleneck layer with exactly k units, so we trained a variant of the standard model without that bottleneck layer (directly using a ResNet-18 or Inception-v3 model to predict $x \to y$); this performed similarly to the standard bottleneck model ("Standard, no bottleneck" in Table 1). Second, we tested a typical multi-task setup using an auxiliary loss to encourage the activations of the last layer to be predictive of the concepts c, hyperparameter searching across different weightings of this auxiliary loss. These models also performed comparably ("Multitask" in Table 1), but since they do not support concept interventions, we focus on comparing standard vs. concept bottleneck models in the rest of the paper.

Data efficiency. Another way to benchmark different models is by measuring data efficiency, i.e., how many training points they need for a desired level of accuracy. To study this, we subsampled the training and validation data and retrained each model (details in Appendix B.4). Concept bottleneck models are particularly effective on OAI: the sequential bottleneck model with $\approx 25\%$ of the full dataset performs similarly to the standard model. On CUB, the joint bottleneck and standard models are more accurate throughout, with the joint model slightly more accurate in lower data regimes (Figure 2-Right).

[3] To contextualize RMSE, our modified KLG ranges from 0-3, and average Pearson correlations between each predicted and true concept are $\geq$0.87 for all bottleneck models.

5. Benchmarking post-hoc concept analysis

Concept bottleneck models are trained to have a bottleneck layer that aligns component-wise with the human-specified concepts c. For any test input x, we can read out predicted concepts directly from the bottleneck layer, as well as intervene on concepts by manipulating the predicted concepts $\hat{c}$ and inspecting how the final prediction $\hat{y}$ changes. This enables explanations like "if the model did not think the joint space was too narrow for this patient, then it would not have predicted severe arthritis". An alternative approach to interpreting models in terms of concepts is post-hoc analysis: take an existing model trained to directly predict $x \to y$ without any concepts, and use a probe to recover the known concepts from the model's activations. For example, Bau et al. (2017) measure the correlation of individual neurons with concepts, while Kim et al. (2018) use a linear probe to predict concepts with linear combinations of neurons.

A necessary condition for post-hoc interpretation is high concept accuracy. In this section, we therefore evaluate how accurately linear probes can predict concepts post-hoc. We emphasize that this is a necessary but not sufficient condition for accurate post-hoc interpretations, as post-hoc analysis does not enable interventions on concepts: even if we find a linear combination of neurons that predicts a concept well, it is unclear how to modify the model's activations to change what it thinks of that concept alone. Without this ability to intervene, interpretations in terms of concepts are suggestive but fraught: even if we can say that "the model thinks the joint space is narrow", it is hard to test if that actually affects its final prediction. This is an important limitation of post-hoc interpretation, though we will set it aside for this section.

Following Kim et al. (2018), we trained a linear probe to predict each concept from the layers of the standard model (see Appendix B). We found that these linear probes have lower concept accuracy compared to simply reading concepts out from a bottleneck model (Table 2). On OAI, the best-performing linear probe achieved an average concept RMSE of 0.68, vs. 0.53 in the bottleneck models; average Pearson correlation dropped to 0.72 from 0.84. On CUB, the linear probe achieved an average concept error of 0.09 instead of 0.03; average F1 score dropped to 0.77 from 0.92.

We also tested if we could predict concepts post-hoc from models designed to learn an interpretable mapping from $x \to y$. Specifically, we evaluated self-explaining neural networks (SENN) (Melis & Jaakkola, 2018). As with standard models, SENN does not use any pre-specified concepts; it learns an input representation encouraged to be interpretable through diversity and smoothness constraints. However, linear probes on SENN also had lower concept accuracy on OAI (0.68 concept RMSE; see Appendix B).[4]

The comparative difficulty in predicting concepts post-hoc suggests that if we have prior knowledge of what concepts practitioners would use, then it helps to directly train models with these concepts instead of hoping to recover them from a model trained without knowledge of these concepts. See Chen et al. (2020) for a related discussion.

6. Test-time intervention

The ability to intervene on concept bottleneck models enables human users to have richer interactions with them. For example, if a radiologist disagrees with a model's prediction, she would not only be able to inspect the predicted concepts, but also simulate how the model would respond to changes in those predicted concepts. This kind of *test-time intervention* can be particularly useful in high-stakes settings like medicine, or in other settings where it is easier for users to identify the concepts c (e.g., wing color) than the target y (exact species of bird).

We envision that in practice, domain experts interacting with the model could intervene to "fix" potentially incorrect concept values predicted by the model. To study this setting, we use an oracle that can query the true value of any concept for a test input. Figure 3 shows examples of interventions that lead to the model making a correct prediction.

6.1. Intervening on OAI

Recall that concept bottleneck models first predict concept values $\hat{c}$ from the input x, and then use those predicted concept values to predict the target $\hat{y}$. On OAI, we define intervening on the j-th concept as replacing $\hat{c}_j$ with its true value c_j, as provided by the oracle, and then updating the prediction $\hat{y}$ after this replacement. Similarly, we can intervene on multiple concepts by simultaneously replacing all of their corresponding predicted concept values, and then updating the prediction. (Intervening on CUB, which we describe in the next subsection, is similar but slightly more complicated because of its regression setting.)

Concretely, we iteratively select concepts on which to intervene on, using an input-independent ordering over concepts computed from the held-out validation set. This means that we always intervene on the same concept c_{i_1} first, followed by intervening on both c_{i_1} and c_{i_2}, and so on (see Appendix B for more detail).

We found that test-time intervention in this manner significantly improved task accuracy on OAI: e.g., querying for just 2 concepts reduces task RMSE from >0.4 to ≈ 0.3

[4] We were unable to run SENN on CUB because the default implementation was too memory-intensive; CUB has many more classes/concepts than the tasks SENN was originally used for.

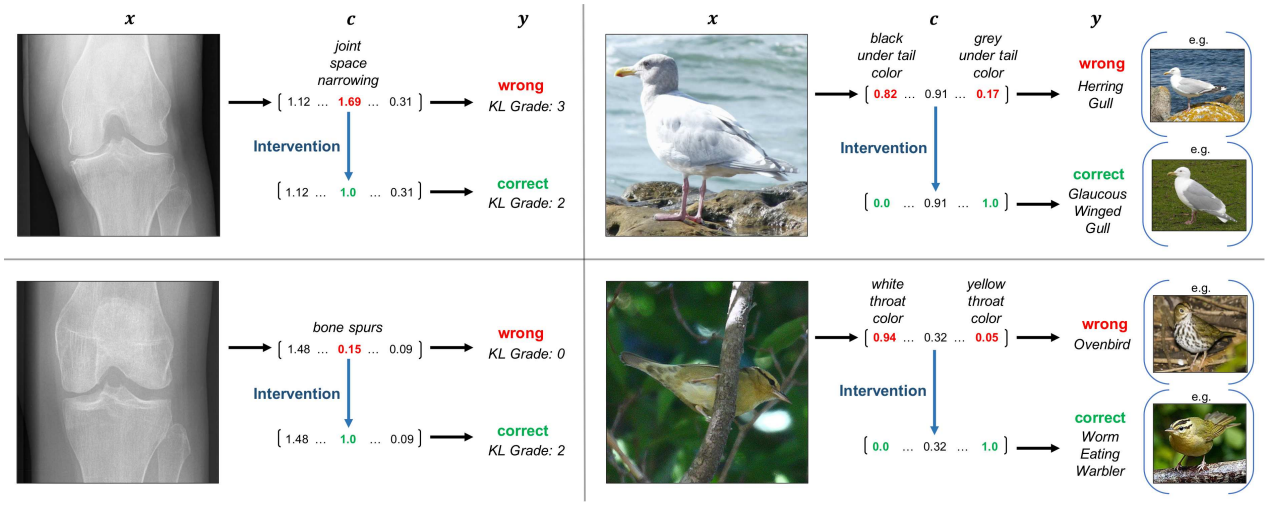

Figure 3. Successful examples of test-time intervention, where intervening on a single concept corrects the model prediction. Here, we show examples from independent bottleneck models. **Right**: For CUB, we intervene on concept groups instead of individual binary concepts. The sample birds on the right illustrate how the intervened concept distinguishes between the original and new predictions.

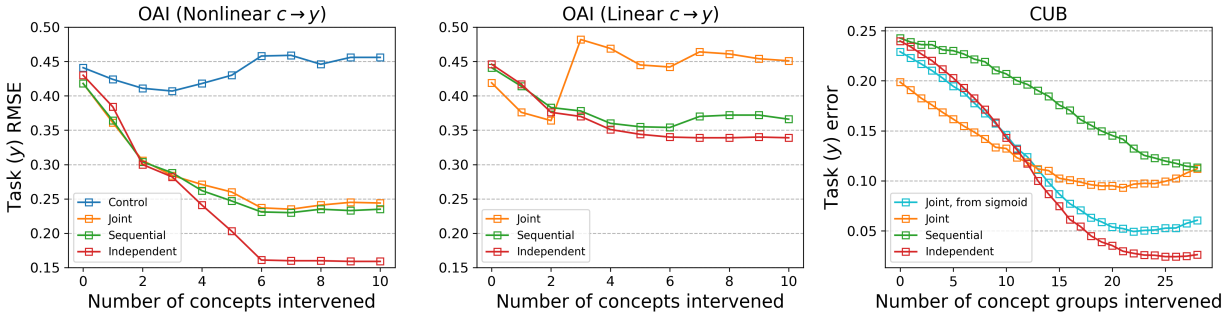

Figure 4. Test-time intervention results. **Left**: Intervention substantially improves task accuracy, except for the control model, which is a joint model that heavily prioritizes label accuracy over concept accuracy. **Mid**: Replacing $c \to y$ with a linear model degrades effectiveness. **Right**: Intervention improves task accuracy except for the joint model. Connecting $c \to y$ to probabilities rescues intervention but degrades normal accuracy.

(Figure 4-Left). These results hint that a single radiologist collaborating with bottleneck models might be able to outperform either the radiologist or model alone, since the concept values used for intervention mostly come from a single radiologist instead of a consensus reading (see Appendix A), and neural networks similar to ours are comparable with individual radiologist performance in terms of agreement with the consensus grade (Tiulpin et al., 2018; Pierson et al., 2019). However, definitively showing this would require more careful human studies.

Furthermore, we found a trade-off between intervenability and task accuracy: the independent bottleneck achieved better test error when all $k = 10$ concepts are replaced than the sequential or joint bottlenecks, but performed slightly worse without any intervention (Figure 4-Left). This behavior is consistent with how these different bottleneck models are trained. Recall that in the independent bottleneck, $c \to y$ is trained using the true c, which is what we replace the predicted concepts $\hat{c}$ with. In contrast, in the sequential and joint models, $c \to y$ is trained using the predicted $\hat{c}$, which in general will have a different distribution from the true c. Without any interventions, we might therefore expect the independent bottleneck to perform worse, as at test time, it receives the distribution over the predicted $\hat{c}$ instead of the distribution of the true c that it was trained with. However, when all concepts are replaced, the reverse is true.

To better understand what influences intervention effectiveness, we ran two ablations. First, we found that intervention can fail in joint models when we prioritize fitting y over c too much (i.e., when λ is too small). Specifically, the joint model with $\lambda = 0.01$ learned a concept representation that was not as well-aligned with the true concepts, and replacing $\hat{c}$ with the true c at test time slightly *increased* test error ("control" model in Figure 4-Left). Second, we changed the

$c \to y$ model from the 3-layer multi-layer perceptron used throughout the paper to a single linear layer. Surprisingly, test-time intervention was less effective here compared to the non-linear counterparts (Figure 4-Mid), even though task and concept accuracies were similar before intervention (concept RMSEs of the sequential and independent models are not even affected by the change in $c \to y$). It is unclear to us why a linear $c \to y$ model should be worse at handling interventions, and this observation warrants further investigation in future work.

Altogether, these results suggest that task and concept accuracies alone are insufficient for determining how effective test-time intervention will be on a model. Different inductive biases in different models control how effectively they can handle distribution shifts from $\hat{c} \to y$ (pre-intervention) to $c \to y$ (post-intervention). Even without this distribution shift, as in the case of the linear vs. non-linear independent bottlenecks, the expressivity of $c \to y$ has a large effect on intervention effectiveness. Moreover, it is possible that the average concept accuracy masks differences in individual concept accuracies that influence these results.

6.2. Intervening on CUB

Intervention on CUB is complicated by the fact that it is classification instead of regression. Recall from Section 3 that for sequential and joint bottleneck classifiers, the final predictor f takes in the predicted concept logits $\hat{\ell} = \hat{g}(x)$ instead of the predicted binary concepts $\hat{c}$. To intervene on a concept $\hat{c}_j$, we therefore cannot directly copy over the true c_j. Instead, we need to alter the logits $\hat{\ell}_j$ such that $P(\hat{c}_j = 1) = \sigma(\hat{\ell}_j)$ is close to the true c_j. Concretely, we intervene on $\hat{c}_j$ by setting $\hat{\ell}_j$ to the 5th (if $c_j = 0$) or 95th (if $c_j = 1$) percentile of $\hat{\ell}_j$ over the training distribution.

Another difference is that for CUB, we group related concepts and intervene on them together. This is because many of the concepts encode the same underlying property, e.g., $c_1 = 1$ if the wing is red, $c_2 = 1$ if the wing is black, etc. We assume that the human (oracle) returns the true wing color in a single query, instead of only answering yes/no questions about the wing color; see Figure 4-Right.

An important caveat is that we use denoised class-level concepts in the CUB dataset (Section 4.1). To avoid unrealistic scenarios where a bird part is not visible in the image but we still 'intervene' on it, we only replace a concept value with the true concept value if that concept is actually visible in the image (visibility information is included in the dataset). The results here are nonetheless still optimistic, because they assume that human experts do not make mistakes in identifying concepts and that birds of the same species always share the same concept values.

Test-time intervention substantially improved accuracy on CUB bottleneck models (Figure 4-Right), though it took intervention on several concept groups to see a large gain. For simplicity, we queried concept groups in random order, which means that many queries were probably irrelevant for any given test example.[5]

Table 3. Task and concept error with background shifts. Bottleneck models have substantially lower task error than the standard model.

MODEL	y ERROR	c ERROR
STANDARD	0.627±0.013	-
JOINT	0.482±0.018	0.069±0.002
SEQUENTIAL	0.496±0.009	0.072±0.002
INDEPENDENT	0.482±0.008	0.072±0.002

As with OAI, test-time intervention was more effective on independent bottleneck models than on the sequential and joint models (Figure 4-Right). We hypothesize that this is partially due to the ad-hoc fashion in which we set logits to the 5th or 95th percentiles for the latter models. To study this, we trained a joint bottleneck with the same task-concept tradeoff λ but with the final predictor f connected to the probabilities $P(\hat{c}_j = 1) = \sigma(\hat{\ell}_j)$ instead of the logits $\hat{\ell}_j$. We show the performance of this model under test-time intervention as the "Joint, from sigmoid" curve in Figure 4-Right. It had a higher task error of 0.224 vs. 0.199 with the normal joint model; we suspect that the squashing from the sigmoid makes optimization harder. However, test-time intervention worked better, and it is more straightforward as we can directly edit $\hat{c}$ instead of having to arbitrarily choose a percentile for the logits $\hat{\ell}$. This raises the question of how to effectively intervene in the classification setting while maintaining the computational advantages of avoiding the sigmoid in the $c \to y$ connection.

7. Robustness to background shifts

Finally, we investigate if concept bottleneck models can be more robust than standard models to spurious correlations that hold in the training distribution but not the test distribution. For example, in the bird recognition task, a model might spuriously associate the image background with the label and therefore make more errors if the relationship between the background and label changes. However, if the relationship between the concepts and the label remains invariant, then one might hope that concept bottleneck models might still perform well under this change.

To test this, we constructed the TravelingBirds dataset, a

[5] This differs from OAI, which used a fixed ordering computed on a held-out validation set. For CUB, it is common to retrain the model on the training + validation set after the hyperparameter search, which makes it difficult to subsequently use the held-out validation set. See Appendix B for more details.

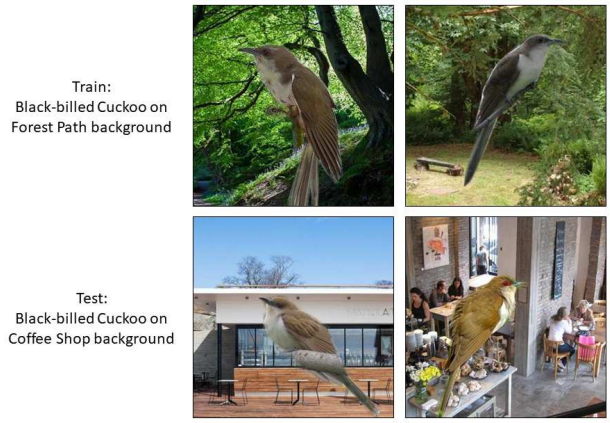

Figure 5. In the TravelingBirds dataset, we change the image backgrounds associated with each class from train to test time (illustrated above for a single class).

variant of the CUB dataset where the target y is spuriously correlated with image background in the training set. Specifically, we cropped each bird out of its original background (using segmentation masks from the original dataset) and onto a new background from the Places dataset (Zhou et al., 2017), with each bird class (species) assigned to a unique and randomly-selected category of places. At test time, we shuffle this mapping, so each class is associated with a different category of places. For example, at training time, all robins might be pictured against the sky, but at test time they might all be on grassy plains (Figure 5).[6]

The results on TravelingBirds are shown in Table 3. Concept bottleneck models do better than standard models: they rely less on background features, since each concept is shared among multiple bird classes and thus appears in training data points that span multiple background types, reducing the correlation between the concept and the background. On the other hand, standard models that go straight from the input x to the label y leverage the spurious correlation between background and label, and consequently fail on the shifted test set.

This toy experiment shows that concept bottleneck models can be more robust to spurious correlations when the target y is more correlated with training data artifacts compared to the concepts c. We emphasize that whether bottleneck models are more robust depends on the choice of the set of concepts c and the shifts considered; a priori, we do not expect that an arbitrary set of concepts c will lead to a more robust model.

[6]TravelingBirds is constructed in a similar manner to the Waterbirds dataset introduced in Sagawa et al. (2020). In Waterbirds, which was designed to benchmark methods for handling group shifts, the classes are collapsed into just two classes (waterbirds and landbirds), and only land or water backgrounds are used. In TravelingBirds, which is a more adversarial setting, we retain the multi-class labels, use a larger set of backgrounds, and change the test distribution so that the relationship between the backgrounds and labels are completely altered.

8. Discussion

Concept bottleneck models can compete on task accuracy while supporting intervention and interpretation, allowing practitioners to reason about these models in terms of high-level concepts they are familiar with, and enabling more effective human-model collaboration through test-time intervention. We believe that these models can be promising in settings like medicine, where the high stakes incentivize human experts to collaborate with models at test time, and where the tasks are often normatively defined with respect to a set of standard concepts (e.g., "osteoarthritis is marked by the presence of bone spurs"). A flurry of recent papers have used similar human concepts for post-hoc interpretation of medical and other scientific ML models, e.g., Graziani et al. (2018) for breast cancer histopathology; Clough et al. (2019) for cardiac MRIs; and Sprague et al. (2019) for meteorology (storm prediction). We expect that concept bottleneck models can be applied directly to similar settings.

A drawback of concept bottleneck models is that they require annotated concepts at training time. However, if the set of concepts are good enough, then fewer training examples might be required to achieve a desired accuracy level (as in OAI). This allows model developers to trade off the cost of acquiring more detailed annotations against the cost of acquiring new training examples, which can be helpful when new training examples are expensive to acquire, e.g., in medical settings where adding training points might entail invasive/expensive procedures on patients, but the incremental cost in asking a doctor to add annotations to data points that they already need to look at might be lower.

Below, we discuss several directions for future work.

Learning concepts. In tasks that are not normatively defined, we can learn the right concepts by interactively querying humans. For example, Cheng & Bernstein (2015) asked crowdworkers to generate concepts to differentiate between adaptively-chosen pairs of examples, and used those concepts to train models to recognize the artist of a painting, tell honest from deceptive reviews, and identify popular jokes. Similar methods can also be used to refine existing concepts and make them more discriminative (Duan et al., 2012).

Side channel from $x \to y$. We can also account for having an incomplete set of concepts by adding a direct side channel from $x \to y$ to a bottleneck model. This is equivalent to using the concepts as auxiliary features, and as discussed in Section 2, has the drawback that we cannot cleanly intervene on a single concept, since the $x \to y$ connection might also be implicitly reasoning about that concept. De-

vising approaches to mitigate this issue would allow concept models to have high task accuracy even with an incomplete set of concepts; for example, one might consider carefully regularizing the $x \to y$ connection or using some sort of adversarial loss to prevent it from using existing concepts.

Theoretically analyzing concept bottlenecks. In OAI, we found that concept bottleneck models outperform standard models on task accuracy. Better understanding when and why this happens can inform how we collect concepts or design the architecture of bottleneck models. As an example of what this could entail, we sketch an analysis of a simple well-specified linear regression setting, where we assume that the input $x \in \mathbb{R}^d$ is normally distributed, and that the concepts $c \in \mathbb{R}^k$ and the target $y \in \mathbb{R}$ are noisy linear transformations of x and c respectively. Our analysis suggests that in this setting, concept bottleneck models can be particularly effective when the number of concepts k is much smaller than the input dimension d and when the concepts have relatively low noise compared to the target.

Concretely, we compared an independent bottleneck model (two linear regression problems for $x \to c$ and $c \to y$) to a standard model (a single linear regression problem) by deriving the ratio of their excess mean-squared-errors as the number of training points n goes to infinity:

$$\frac{\text{Excess error for indp bottleneck model}}{\text{Excess error for standard model}} \leq \frac{\frac{k}{d}\sigma_Y^2 + \sigma_C^2}{\sigma_Y^2 + \sigma_C^2},$$

where σ_C^2 and σ_Y^2 are the variances of the noise in the concepts c and target y, respectively. See Appendix C for a formal statement and proof. The asymptotic relative excess error is small when $\frac{k}{d}$ is small and $\sigma_Y^2 \gg \sigma_C^2$, i.e., when the number of concepts is much smaller than the input dimension and the concepts are less noisy than the target.

Intervention effectiveness. Our exploration of the design space of concept bottleneck models showed that the training method (independent, sequential, joint) and choice of architecture influence not just the task and concept accuracies, but also how effective interventions are. This poses several open questions, for example: What factors drive the effectiveness of test-time interventions? Could adaptive strategies that query for the concepts that maximize expected information gain on a particular test example make interventions more effective? Finally, how might we have models learn from interventions to avoid making similar mistakes in the future?

Reproducibility

The code for replicating our experiments is available on GitHub at https://github.com/yewsiang/ConceptBottleneck. An executable version of the CUB experiments in this paper is on CodaLab at https://worksheets.codalab.org/worksheets/0x362911581fcd4e048ddfd84f47203fd2. The TravelingBirds dataset can also be downloaded at that link. While we are unable to release the OAI dataset publicly, an application to access the data can be made at https://nda.nih.gov/oai/.