Concept Bottleneck Models (CBMs) for XAI

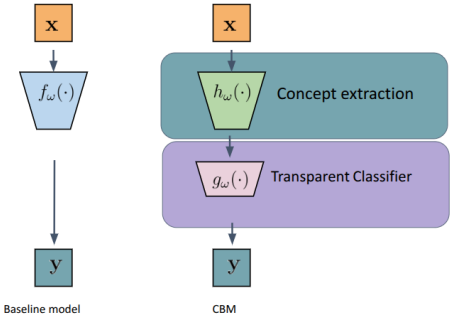
From Foundations to Weakly Supervised VLM-CBMs

Gianni Franchi

MVA course

What is a Concept Bottleneck Model?

- A supervised model that predicts human-interpretable concepts before making final predictions [20]
- Structure:
  - Concept extraction Network: $h_\omega(\cdot)$
  - Classifier to predict the final output class : $g_\omega(\cdot)$
- The final DNN is the composition of these functions: $f_\omega(\cdot) = g_\omega(h_\omega(\cdot))$



Core Components of CBM

A CBM comprises two learnable modules:

1. Concept Extractor

Maps input $x \in \mathcal{X}$ (e.g., an image) into an activation vector of high-level concepts $\mathbf{c} \in [0,1]^k$.

2. Inference Layer

A white-box model (typically a sparse linear layer) that computes predictions y from the extracted concepts $\mathbf{c}$.

$$\hat{y} = g(h(x))$$

Example: The Red Ball

- **Input**: Image of a red ball.
- **Concept Encoding**: The model encodes the image into logits for:
  - (shape = sphere)
  - (color = red)
- **Bottleneck**: This vector serves as interpretable sufficient statistics for the final prediction (e.g., "Sport Equipment").

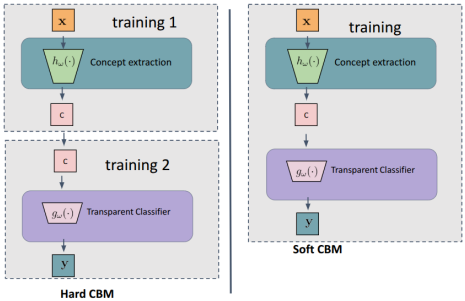
Hard vs. Soft CBMs

Hard CBMs

- Accepts only binary concepts.
- Thresholded/Argmaxed.

Soft CBMs

- Accepts concept probabilities (Soft concepts).
- Joint training.



What is Information Leakage?

Definition

The amount of unintended information used to predict label y with soft concepts $\hat{c}$ that is not present in hard representation c.

- **The Problem**: Information Leakage. Unintentional data signals bypass concepts, making explanations misleading.
- Leakage occurs often with "Soft CBMs"
- Compromises our ability to **intervene** on concepts.
- Hard CBMs are leakage-free.

Measuring Leakage

Leakage is measured as the **conditional mutual information**:

$$I_{\text{Leakage}} = I(y; \hat{c} \mid c) = H(y \mid c) - H(y \mid \hat{c}, c) \tag{1}$$

- $H(y \mid c)$: Uncertainty in target given hard concepts.
- $H(y \mid \hat{c}, c)$: Uncertainty in target given both soft and hard concepts.

EXplainable Neural-Symbolic Learning (X-NeSyL) [2]

- **Concept extraction network**: We propose to use a faster RCNN
- **Classifier network**: We use a MLP
- Explanation is intrinsic to the model (*transparent*), not derived post-hoc

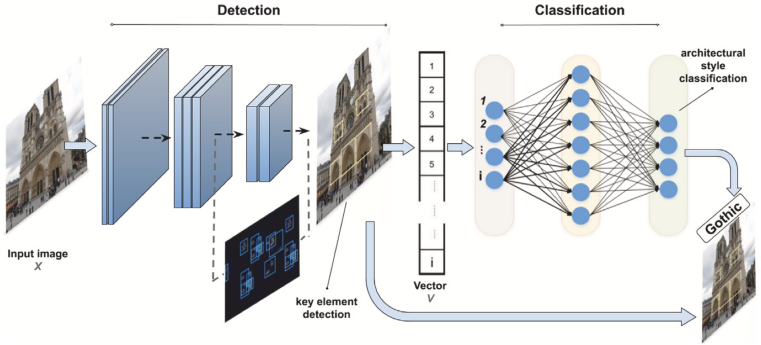


Figure: Representation of X-NeSyL CBM [21].

CBM [1]

- **Concept extraction network**: they use a multi class classifier
- **Classifier network**: They use a ully Connected Layer
- Explanation is intrinsic to the model (*transparent*), not derived post-hoc

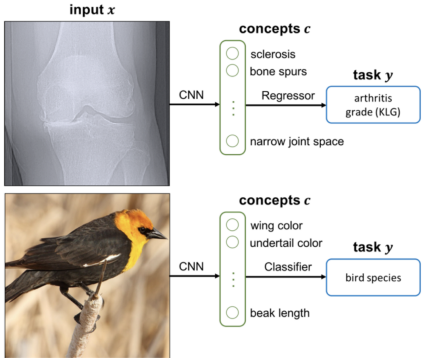
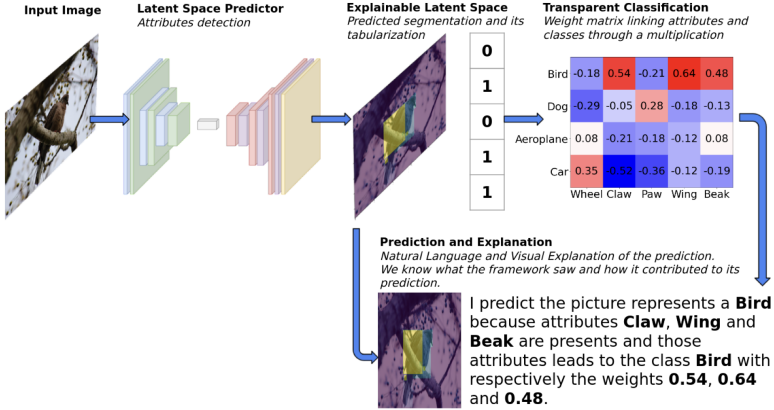


Figure: Representation of X-NeSyL CBM [21].

Greybox CBM [3]

- **Concept extraction network**: We propose to use a semantic segmentation
- **Classifier network**: We use a Fully Connected Layer
- Explanation is intrinsic to the model (*transparent*), not derived post-hoc



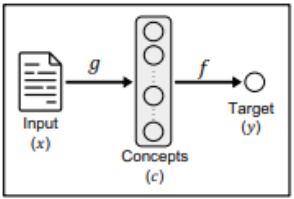
Why is CBM Good for XAI?

- **Interpretability**: Concepts used in prediction are understandable by humans
- **Intervention**: Users can manually modify predicted concepts and see how it changes output
- **Debugging**: Errors can be traced to concept prediction vs decision boundaries
- **Transparency**: The model logic is decomposed into concept reasoning + classification

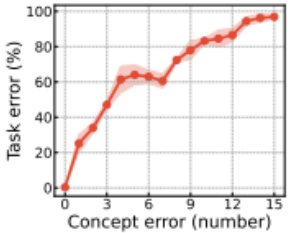
What is Concept Intervention? [4]

In traditional black-box models, if a prediction is wrong, the user is helpless. In **Concept Bottleneck Models (CBMs)**, we can intervene.

- **Definition**: The act of manually correcting or "rectifying" predicted concept values before they reach the final classifier.
- **Human-in-the-loop**: An expert can look at the intermediate concepts, fix errors, and see the target prediction update in real-time.



(a) Diagram of CBMs



(b) Task vs. Concept errors

Formal Notation of Intervention [4]

Let us define the components of the process:

- $x \in \mathbb{R}^d$: The input data (dimensionality d).
- $c \in \{0, 1\}^k$: The ground-truth binary concepts (cardinality k).
- $y \in Y$: Target response (categorical distribution for classification).

The Standard Inference Flow:

1. **Concept Prediction**: $\hat{c} = g(x)$ where g is the concept predictor.
2. **Target Prediction**: $\hat{y} = f(\hat{c})$ where f is the target predictor.

Rectifying the Bottleneck [4]

Suppose we identify a set of concepts $S \subseteq \{1, \ldots, k\}$ that are incorrectly predicted. We intervene by replacing the predicted values $\hat{c}_S$ with the true values c_S.

The Updated Concept Vector $\tilde{c}$

$$\tilde{c} = \{\hat{c} \setminus S, c_S\}$$

The final prediction is then made using this rectified vector:

$$\hat{y}_{intervened} = f(\tilde{c})$$

This allows the model to reach the correct decision even if the original concept extraction was noisy.

Which Concepts Should We Fix? [4]

In real-world applications, intervention is **costly**.

- It requires human experts (doctors, engineers, scientists).
- The cost increases with $|S|$ (the number of intervening concepts).

The Objective: Find a selection score s_i for each concept i. We intervene on concepts in decreasing order of s_i to maximize the increase in task performance with the minimum number of interventions.

Random (RAND) and Uncertainty (UCP)[4]

1. **Random Selection (RAND)**
   - Selects concepts uniformly at random: $s_i \sim U[0, 1]$.
   - **Purpose**: Serves as a baseline to prove that intelligent selection matters.
2. **Uncertainty of Concept Prediction (UCP)**
   - We prioritize concepts where the model is "unsure."
   - For binary concepts, score is based on Entropy $H(\hat{c}_i)$:

$$s_i = \frac{1}{|\hat{c}_i - 0.5|}$$

   - **Intuition**: Uncertain concepts are likely to be wrong and mislead the final classifier.

Loss on Concept Prediction (LCP)[4]

If we knew the ground truth during testing, we would fix the most "wrong" concepts.

LCP Score

$$s_i = |\hat{c}_i - c_i|$$

- **Advantage**: Directly targets the errors in the bottleneck.
- **The Catch**: This score is **unavailable in practice** at test time because the ground truth c_i is exactly what the human is supposed to provide.
- **Role**: Used in research as an "Oracle" or upper bound for performance.

Contribution on Target Prediction (CCTP) [4]

Not all concepts are equally important for the final classification. **CCTP** uses gradients to find which concept "moves the needle" the most for the output classes.

CCTP Formula

$$s_i = \sum_{j=1}^{M} \left| \hat{c}_i \frac{\partial f_j}{\partial \hat{c}_i} \right|$$

where M is the number of classes and f_j is the output for class j.

- This is inspired by **Grad-CAM** and other saliency methods.
- It focuses on concepts the model is actually relying on.

Which Strategy Wins?[4]

- **UCP** is great for fixing noisy extractors but ignores the downstream task importance.
- **CCTP** is efficient for task performance but might ignore concepts that the model currently thinks are "unimportant" but are actually critical.

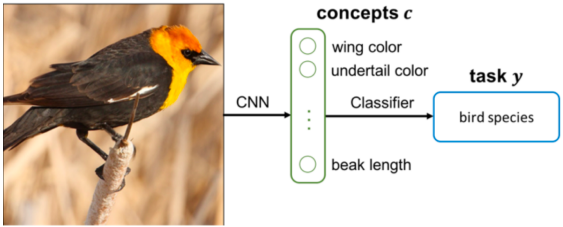
Intervention effectiveness is usually measured by plotting **Task Accuracy vs. Number of Intervened Concepts**.

Summary: Why Intervene?[4]

1. **Correction**: Fixes model errors without retraining.
2. **Trust**: Humans can verify the reasoning process.
3. **Safety**: In critical fields (e.g., medical diagnosis), a human can override a concept (like "Is there a tumor?") to ensure the final recommendation is safe.
4. **Efficiency**: Through criteria like **CCTP**, we minimize the workload for human experts.

Two Main Families of CBM

1. **Fully Supervised CBM**:
   - Requires labels for both the task Y and the concepts C for every sample.
   - High accuracy but extremely expensive to annotate.
2. **Weakly Supervised CBM**:
   - Uses external knowledge or Vision-Language Models (VLMs) to define or supervise concepts without dense manual labels.



Weakly Supervised CBMs or Unsupervised CBMs
VLM CBMs

Vision-Language Models (VLM) for Supervision

Collecting manual annotations is hard. A solution: **Querying VLMs**.

- **Method**: Use models like CLIP to check similarity between an image and a concept description (e.g., "a photo of a red wing").
- **Automation**: LLMs (like GPT) can generate the "Vocabulary" of concepts for specific classes.

LABO

The LaBo Foundation [5]

- CBMs decompose classification into two steps:
  1. Map input image x to N_C human-interpretable concepts.
  2. Combine these concepts linearly to predict the final class.
- **Traditional Limitation**: Requires domain experts to manually annotate thousands of images with attributes (e.g., "yellow beak," "spotted wing").

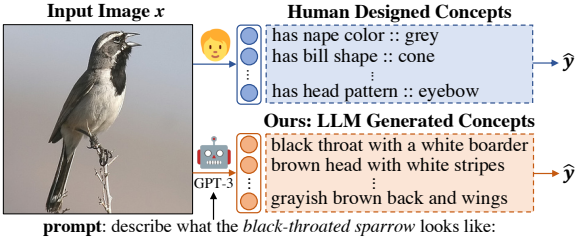


Figure: LaBo removes the need for manual human annotations by prompting LLMs.

Formalizing the Bottleneck [5]

Let $\mathcal{D} = \{(i, y)\}$ be our dataset. They use image encoder $\mathcal{I}$ and text encoder $\mathcal{T}$ (CLIP):

- Image features: $\boldsymbol{x} = \mathcal{I}(i) \in \mathbb{R}^d$
- Bottleneck concepts: $C = \{c_1, ..., c_{N_C}\}$
- Concept embeddings: $\boldsymbol{E}_C \in \mathbb{R}^{N_C \times d}$, where each row is $\mathcal{T}(c)$.

The goal is to solve:

$$\min_{f,C} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\mathcal{L}(f(g(\boldsymbol{x}, \boldsymbol{E}_C)), y)] - \mathcal{F}(C, \mathcal{D})$$

Where $\mathcal{F}$ is the quality of the bottleneck.

Step 1: Generating Candidates [5]

- **Prompting**: GPT-3 is asked: *"Describe what the [CLASS NAME] looks like."*
- **Processing**: 500 sentences per class are generated.
- **Refinement**: A T5 model splits long sentences into shorter, atomic visual concepts.
- **Cleaning**: Class names are automatically stripped to ensure the model doesn't "cheat" by just looking for the name.

Step 2: Choosing the Best Concepts [5]

They cannot use 500 concepts per class (too noisy). They select a subset $C_y \subseteq S_y$ using a **monotone submodular function**:

$$\mathcal{F}(C_y) = \underbrace{\alpha \cdot \sum_{c \in C_y} D(c)}_{\text{Discriminability}} + \underbrace{\beta \cdot \sum_{c_1 \in S_y} \max_{c_2 \in C_y} \phi(c_1, c_2)}_{\text{Coverage}}$$

- **Discriminability**: Concepts that align with class y but not others.
- **Coverage**: Minimizes redundancy; ensures concepts cover different visual appearances.

Measuring Concept Discriminability [5]

The score $D(c)$ uses the mean alignment between class images and concepts:

$$Sim(y, c) = \frac{1}{|\mathcal{X}_y|} \sum_{\boldsymbol{x} \in \mathcal{X}_y} \boldsymbol{x} \cdot \mathcal{T}(c)^\top$$

They normalize this to get a conditional likelihood $\overline{Sim}(y|c)$ and compute the **negative entropy**:

$$D(c) = \sum_{y' \in Y} \overline{Sim}(y'|c) \cdot \log(\overline{Sim}(y'|c))$$

Maximizing this ensures the concept is strongly associated with only a few specific classes.

Step 3: Predicting Concept Scores [5]

The concept predictor g is fixed (not learned) using CLIP's pre-trained zero-shot alignment.

The Bottleneck Activation

For an input image $\boldsymbol{x}$, the concept scores are computed via dot product:

$$g(\boldsymbol{x}, \boldsymbol{E}_C) = \boldsymbol{x} \cdot \boldsymbol{E}_C^\top$$

This produces a vector in $\mathbb{R}^{N_C}$ where each element represents the similarity of the image to a specific textual concept.

Step 4: Learning the Class-Concept Association [5]

The final predictor f is a linear layer with weights $\boldsymbol{W} \in \mathbb{R}^{N \times N_C}$:

$$\hat{y} = \text{argmax}(g(\boldsymbol{x}, \boldsymbol{E}_C) \cdot \sigma(\boldsymbol{W})^\top)$$

- $\sigma(\cdot)$ is the **softmax activation** applied along the concepts axis.
- This ensures the final score is a convex combination of concept scores.
- This weight matrix $\boldsymbol{W}$ is easily interpretable (e.g., how much "green body" contributes to "tree frog").

Training in Few-Shot Scenarios [5]

In low-resource settings, training $\boldsymbol{W}$ from scratch is hard. **LaBo Solution**: Initialize $\boldsymbol{W}$ using a language model prior.

- If concept c was generated by GPT-3 specifically for class y:

$$\boldsymbol{W}_{y,c} = 1$$

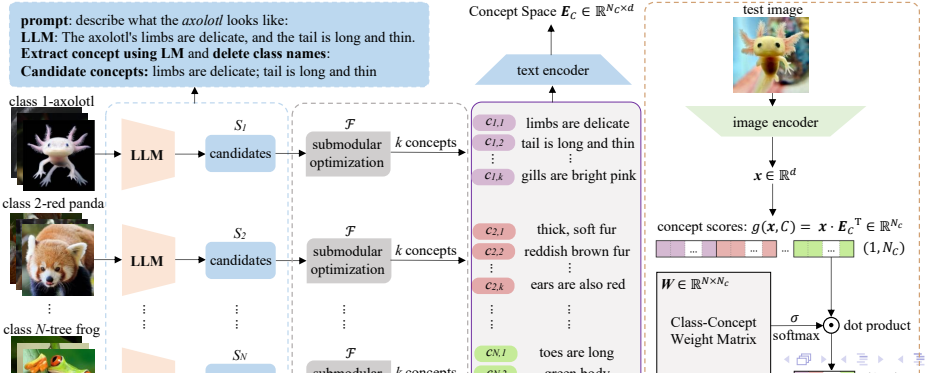
- Otherwise, $\boldsymbol{W}_{y,c} = 0$.

This "biases" the model to trust the GPT-3 logic until enough image data is available to refine the weights.

LaBo: System Overview [5]

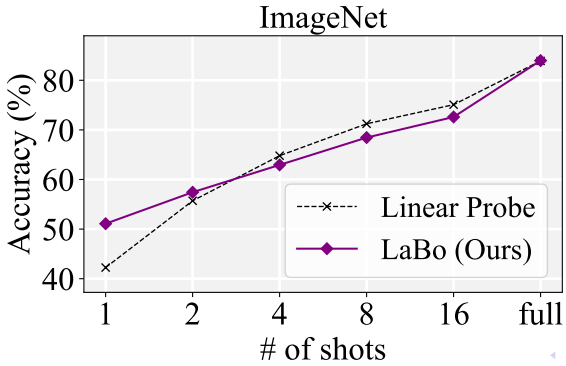
LaBo automates CBM construction in four main stages:

1. **Generation**: GPT-3 generates thousands of candidate concepts for each class.
2. **Selection**: Submodular optimization picks the most discriminative and diverse concepts.
3. **Alignment**: CLIP scores the presence of these textual concepts in images.
4. **Linear Prediction**: A linear layer maps concept scores to class targets.



Key Findings [5]

- **Few-Shot Performance**: LaBo outperforms linear probes by **11.7%** in 1-shot classification.
- **Scaling**: As data increases, performance remains competitive with black-box CLIP linear probes.
- **Generalization**: Evaluated on 11 diverse tasks, from satellite imagery to skin tumors.



Human Evaluation of Concepts [5]

Human annotators compared GPT-3 concepts vs. Wikipedia/WordNet:

- **Factuality**: GPT-3 concepts were found to be more accurate.
- **Groundability**: GPT-3 generated attributes that were more "visual" and easier for CLIP to recognize.
- **Complexity**: Using sentences (e.g., "smooth bumpy skin") captures richer info than single words.

Summary of LaBO [5]

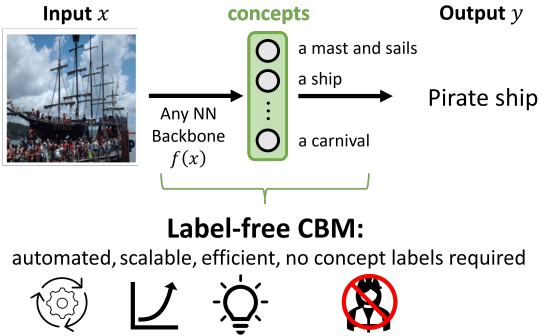
- **Interpretability-by-design** does NOT mean low performance.
- **Automation**: LaBo removes the bottleneck of manual human annotation.
- **Trust**: By tracing decisions through GPT-3 concepts, users can understand why a model makes a specific prediction.
- **Future**: Potential for more complex reasoning layers beyond linear combinations.

Label-free Concept Bottleneck Models

The Label-free CBM Framework [6]

Core Idea: Transform any backbone into a CBM without labeled concept data while preserving accuracy.

- **Scalable:** First CBM to scale to ImageNet.
- **Efficient:** Creation takes only a few hours.
- **Automated:** Minimal human effort required.



The 4-Step Pipeline [6]

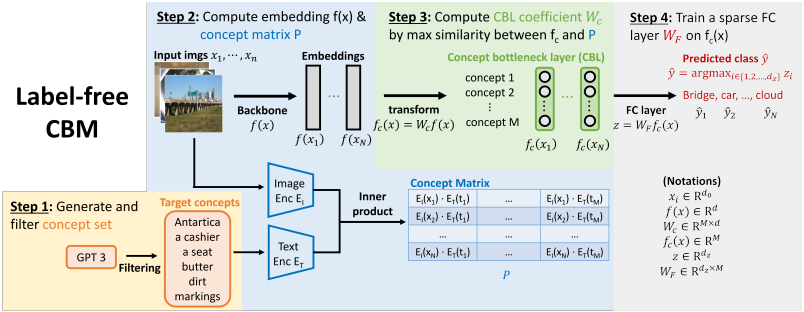


Figure: Overview of the Label-free CBM pipeline.

1. **Step 1:** Concept set creation and filtering.
2. **Step 2:** Compute backbone and concept embeddings.
3. **Step 3:** Learn projection weights (W_c) for the CBL.
4. **Step 4:** Learn sparse final layer weights (W_F).

Step 1A: Initial Concept Set Creation [6]

- Goal: Avoid manual selection by domain experts.
- Method: Generate concepts via **GPT-3**.
- **Prompts used:**
  - "List the most important features for recognizing a {class}."
  - "List the things most commonly seen around a {class}."
  - "Give superclasses for the word {class}."
- Result: A large, noisy set of candidate concepts.

Step 1B: Concept Set Filtering [6]

1. **Concept Length:** Remove concepts > 30 characters.
2. **Class Similarity:** Remove concepts too similar to output class names (Cosine similarity using CLIP/Sentence-BERT).
3. **Redundancy:** Remove synonyms or duplicates (Similarity > 0.9).

Step 1B: Concept Set Filtering (cont.) [6]

4. **Presence in Data:** Remove concepts that do not activate CLIP highly on training images.
5. **Interpretability:** Remove neurons that cannot be projected accurately into the CBL (determined after Step 3).

Steps 2 & 3: Learning the Projection [6]

- Utilize **CLIP-Dissect** to map backbone feature space to interpretable concepts.
- $P_{i,j}$ = Matrix of CLIP similarities between image x_i and concept t_j.
- Define $f_c(x) = W_c f(x)$.
- Goal: Maximize similarity between neuron activation q_k and the target concept pattern.

Step 4: Learning Sparse Predictors [6]

- Final layer W_F maps concepts to output classes.
- **Goal:** Sparsity (more interpretable).
- Solved via **Elastic Net** objective:

Elastic Net Loss

$$\min_{W_F, b_F} \sum_{i=1}^{N} L_{ce}(W_F f_c(x_i) + b_F, y_i) + \lambda R_\alpha(W_F)$$

- $\alpha = 0.99$ (heavy L1 penalty).
- Target: 25 to 35 non-zero weights per class.

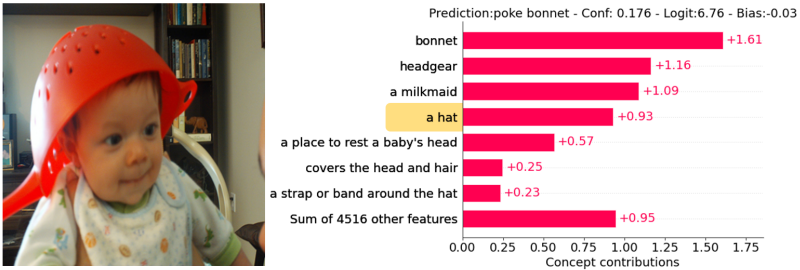
Debugging ImageNet Models [6]

- Analyzing a well-trained model to manually fix errors.
- First example of manually editing a large network to **improve test accuracy**.
- Identifies where model weights conflict with domain knowledge.

Type 4: Incorrect Final Layer Weight [6]

- **Definition:** Concepts are activated correctly, but W_F uses them incorrectly for prediction.
- **Solution:** Manually edit the final layer weights.
- This is the focus of model refinement.

Test-time Intervention [6]



Intervene: activation of “a hat” 2.70 -> 0. New prediction: strainer ✓

Figure: Fixing a prediction by zeroing out an incorrect "hat" activation.

How to Edit Model Weights [6]

1. **Identify Type 4 Error:** Find a visual mismatch in the explanation.
2. **Select Concept:** Choose a highly activated concept to adjust.
3. **Update Weights:**
   - $W_{F[gt,concept]} \uparrow$ (increase for ground truth).
   - $W_{F[pred,concept]} \downarrow$ (decrease for incorrect prediction).

Quantifying the Refinement [6]

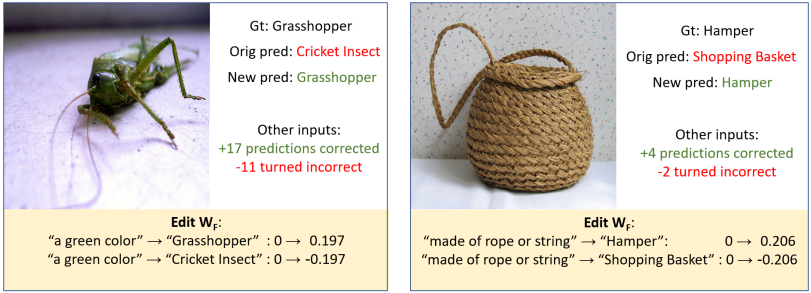


Figure: Manual edits fix errors on unseen inputs.

- Edited only **10 out of 4.5 million** weights.
- Corrected 38 predictions while turning only 17 incorrect.
- Accuracy on the edited subset increased by **4.2%**.

Future Directions [6]

- Automated weight editing based on massive feedback.
- Practitioners can quickly devise fixes for production errors.
- Checking global impacts of local fixes using validation sets.

CLIP QDA

Methodology Overview

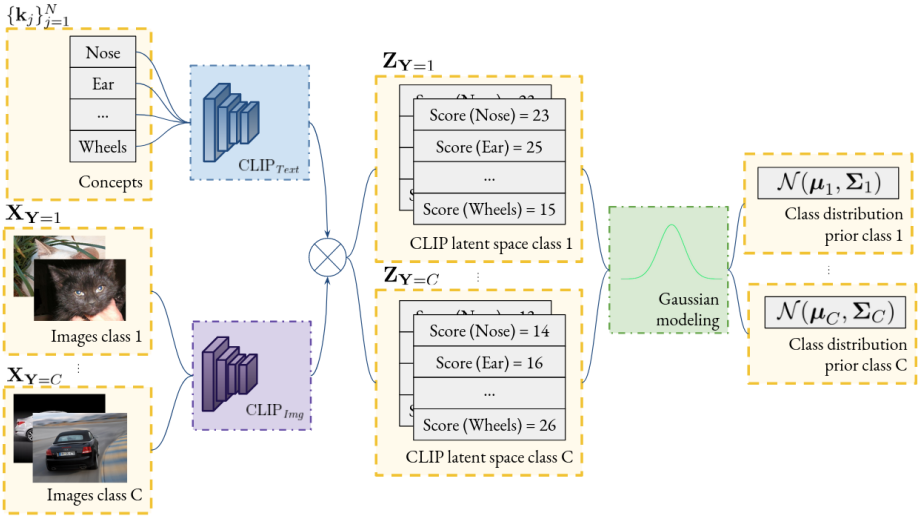


Figure: Overview of our modeling approach.

Contribution Summary

1. **Gaussian Modeling:** Representing CLIP scores as a Mixture of Gaussians.
2. **CLIP-QDA:** A Quadratic Discriminant Analysis classifier for CBMs.
3. **New XAI Tools:** QDA-Data (global) and QDA-Sample (local).
4. **CBM-specific adaptations:** Extending LIME and SHAP to the concept level.
5. **Benchmark:** A new evaluation protocol for CBM explanations.

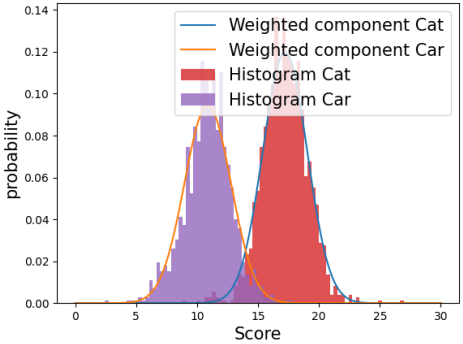
Background: CLIP Notations

- $\boldsymbol{x}$: Image input, y: Class label $\in [\![1, C]\!]$.
- $\{\boldsymbol{k}^j\}_{j=1}^N$: Set of N natural language concepts.
- CLIP Score for concept j: $z^j = g_{\omega^g}(\boldsymbol{x}, \boldsymbol{k}^j)$.
- Latent vector: $\boldsymbol{z} = [z^1, \ldots, z^N]$.
- Conditional distribution for class c: $\boldsymbol{Z}_{Y=c} \sim \mathcal{P}_{\boldsymbol{Z}_{Y=c}}$.

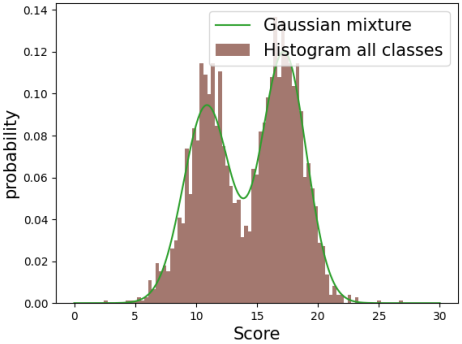
Gaussian Modeling Intuition

- Consider a Toy Example: Cats vs. Cars.
- Concept: "Pointy-eared".
- Histogram shows two modes:
  - Low scores: Images without pointy ears (Cars).
  - High scores: Images with pointy ears (Cats).

Visualizing the Distribution



(a) Class conditioned modeling.



(b) Full mixture modeling.

Figure: CLIP scores for the concept "Pointy-eared".

Formal Gaussian Assumptions

Assume class-conditioned distributions are normal:

$$p(\boldsymbol{Z} = \boldsymbol{z} \mid Y = c) = \mathcal{N}(\boldsymbol{z} \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{2}$$

The total distribution is a Gaussian Mixture Model (GMM):

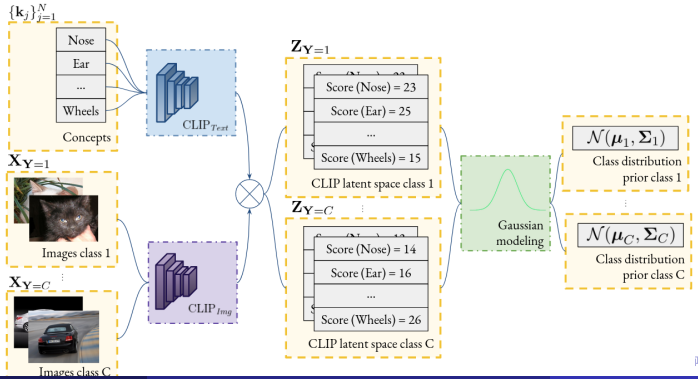
$$p(\boldsymbol{Z} = \boldsymbol{z}) = \sum_{c_i=1}^{C} p_{c_i} \mathcal{N}(\boldsymbol{z} \mid \boldsymbol{\mu}_{c_i}, \boldsymbol{\Sigma}_{c_i}) \tag{3}$$

where $p_c = P(Y = c)$ are the class priors.

CLIP-QDA Architecture

- **Concept extraction network**: We use CLIP
- **Classifier network**: We use a Quadratic Discriminant Analysis

$$p(Y = c \mid \boldsymbol{Z} = \boldsymbol{z}) = \frac{p_c \mathcal{N}(\boldsymbol{z} \mid \mu_c, \Sigma_c)}{\sum_{c_i=1}^{N} p_{c_i} \mathcal{N}(\boldsymbol{z} \mid \mu_{c_i}, \Sigma_{c_i})}. \tag{4}$$



XAI for Concept Bottleneck Models

Two distinct perspectives of explanation:

- **Global (QDA-Data):** Behavior across the entire dataset.
- **Local (QDA-Sample):** Actions on individual samples.

Extended methods: CBM-SHAP and CBM-LIME produce explanations on the **concept level**.

Global Explanations: QDA-Data

- Measures the separation between class distributions c_1 and c_2.
- Uses **Wasserstein-2 distance** for each concept j.
- Large distance implies concept j is a strong differentiator.

Signed Wasserstein-2 Distance for Gaussians

$$\tilde{W}_2 = \text{sign}([\boldsymbol{\mu}_{c_1}]_{(j)} - [\boldsymbol{\mu}_{c_2}]_{(j)}) \left(([\boldsymbol{\mu}_{c_1}]_{(j)} - [\boldsymbol{\mu}_{c_2}]_{(j)})^2 + \Lambda^j_{c_1,c_2}\right)$$

Local Explanations: QDA-Sample

- Based on **Counterfactual Analysis**.
- Question: "What is the minimal change to concept j that flips the class prediction?"
- Perturbation ϵ_s^j is sought where $s \in \{+, -\}$.

Local Metric: Formal Optimization

Finding the minimal perturbation:

$$\min \|\epsilon_s^j\|^2 \quad \text{s.t.} \quad h_{\omega^h}(\boldsymbol{z} + \epsilon_s^j) \neq h_{\omega^h}(\boldsymbol{z}) \tag{5}$$

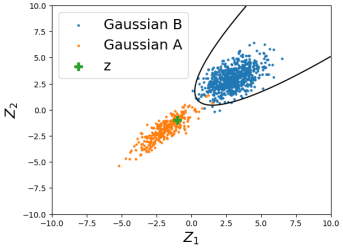
Constraints for Interpretability:

- **Sparsity:** Change only one attribute at a time.
- **Directional Sign:** Evaluate addition $(+)$ or subtraction $(-)$ of concept intensity separately.

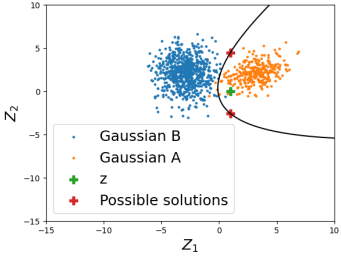
Counterfactual Interpretations

- Simulates an **intervention**: $do(\boldsymbol{Z} = \boldsymbol{z} + \boldsymbol{\epsilon}_s^j)$.
- "Would this cat image be classified as a car if I removed its pointy ears?"
- Closed-form solution exists for QDA parameters $(\boldsymbol{\Sigma}_c, \boldsymbol{\mu}_c, p_c)$.

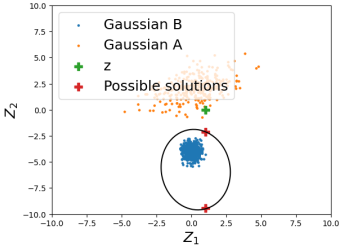
Visualization of Counterfactuals



(a) No counterfactual possible.



(b) Two counterfactuals ($\pm$).



(c) Only one possible ($-$).

Scaled Counterfactuals

- Raw CLIP scores are unit-less and hard to interpret.
- **Normalization:** Divide by the standard deviation of the predicted class distribution.

$$\epsilon^{j}_{s,*,scaled} = \frac{\epsilon^{j}_{s,*}}{\sqrt{[\Sigma_c]_{(j,j)}}} \tag{6}$$

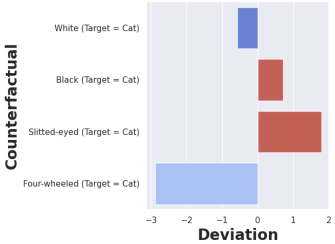
Meaning: "The addition/subtraction of X standard deviations of concept j flips the label."

Post-hoc Extensions: CBM-LIME and CBM-SHAP

- Conventional LIME/SHAP explain images at the pixel level.
- **Novelty:** We adapt them to explain class predictions at the **concept level**.
- Allows users to see the relative contribution of each human-understandable concept to the final classification decision.

CLIP-QDA: Principle

- Learn classification in concept space using Quadratic Discriminant Analysis (QDA)
- Enables a closed-form, mathematical formulation of counterfactual explanations

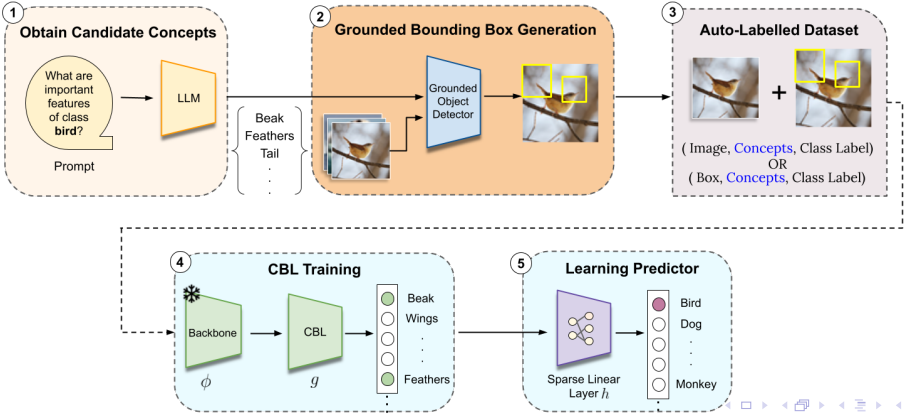


(b) **CLIP-QDA explanation.** Only the top 5 concepts are displayed. Note that the classifier misclassified the image as "Car". The first row must be read as follows: removing a little of the concept "White" to the vector $\boldsymbol{z}$

VLG CBM

Overall Pipeline [8]

- The pipeline consists of three main stages:
  1. Automated generation of an auxiliary grounded dataset.
  2. Training the Concept Bottleneck Layer (CBL).
  3. Mapping concepts to classes via a sparse layer.



Stage 1: Auxiliary Dataset Generation [8]

- Let $D = \{(x_i, y_i)\}$ be the original dataset.
- **Language Supervision**: Generate candidate concepts S_c for each class c using an LLM (following LF-CBM).
- **Vision Supervision**: Use **Grounding-DINO** to ground these concepts spatially in the images.

Spatial Grounding and Filtering [8]

For each image x_i, Grounding-DINO provides bounding boxes:

$$B_i = \{(b_j, t_j, s_j)\}_{j=1}^{K_i}$$

- b_j: Bounding box coordinates.
- t_j: Confidence score.
- s_j: Concept label.

Confidence Filtering

They filter boxes where $t_j < T$ (Confidence Threshold) to ensure only **visually recognizable** concepts remain.

Visual Concept Consolidation [8]

- They filter out concepts that never appear in any bounding box with high confidence.
- The final concept set $\tilde{S}$ is:

$$\tilde{S} = \{s \in S \mid \exists(\cdot, \cdot, s) \in \cup_{i=1}^{|D|} \tilde{B}_i\}$$

- The one-hot encoded concept label vector $o_i \in \{0,1\}^{|\tilde{S}|}$ is defined as:

$$(o_i)_j = 1 \text{ if } s_j \text{ appears in } \tilde{B}_i, \text{ else } 0$$

Stage 2: CBL Training [8]

- **Backbone**: $\phi : \mathcal{X} \to \mathbb{R}^d$ (pretrained or from scratch).
- **Concept Predictor**: g maps embeddings to concept logits.
- **Loss Function**: Binary Cross Entropy (BCE) for multi-label prediction.

$$\min_g \mathcal{L}_{CBL} = \frac{1}{|D'|} \sum_{i=1}^{|D'|} BCE[g \circ \phi(x_i), o_i]$$

Stage 2: Improving Diversity [8]

- They leverage the location information from Grounding-DINO for data augmentation.
- **Strategy**: Randomly crop images to selected bounding boxes.
- **Target**: Modify the one-hot vector o_i to prioritize the concept within that specific crop.
- **Benefit**: Forces the model to focus on the actual visual attribute rather than global image context.

Stage 3: The Sparse Final Layer [8]

- Final layer $h : \mathbb{R}^d \to \mathbb{R}^C$ maps concept logits to class labels.
- **Interpretable Mapping**: They use a sparse linear layer with weight matrix W_F.
- The CBL remains **frozen** during this stage to prevent concept degradation.

Optimization for Sparsity [8]

We minimize the Cross-Entropy (CE) loss with Elastic-Net regularization:

$$\min_h \mathcal{L}_{SL} = \frac{1}{|D|} \sum_{(x,y)\in D} CE[h \circ g \circ \phi(x), y] + \lambda R_\alpha$$

Elastic-Net Term

$$R_\alpha = (1-\alpha)\frac{1}{2}\|W_F\|_2^2 + \alpha\|W_F\|_1$$

Number of Effective Concepts (NEC) [8]

- To control leakage and ensure fair comparison, they introduce **NEC**.
- This metric facilitates fair comparison between CBMs with different bottleneck sizes.
- **ANEC-5**: Accuracy at NEC=5.
- **ANEC-avg**: Average accuracy across different NEC values.

Accuracy Comparison[8]

VLG-CBM demonstrates superior performance across 5 benchmarks:

- **ANEC-5 Improvement**: +4.27% to +51.09% over state-of-the-art.
- **ANEC-avg Improvement**: +0.45% to +29.78%.
- Achieves high sparsity (0.2% non-zero weights) on Places365.

Concept Attribution [8]

- VLG-CBM provides more **faithful** concept attributions.
- Non-visual concepts suggested by LLMs are effectively removed by the grounding detector.
- Localized concepts match human visual intuition better than global image-level scores.

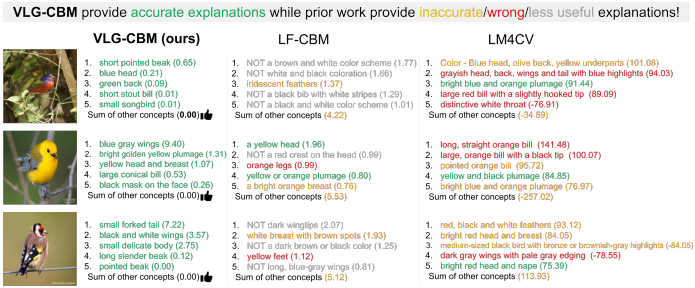


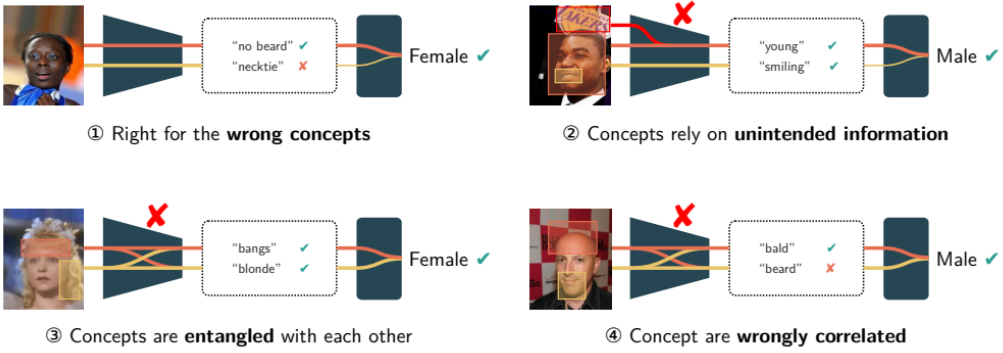
Figure: Comparison of concept localization.

Risks with VLM CBM

Issues with VLM-CBMs [9]

VLMs are not a "silver bullet." They bring risks:

- **Hallucinations**: VLMs might claim a concept exists when it doesn't.
- **Shortcut Learning**: Models might use background pixels to "predict" a concept.
- **Erroneous Agreements**: CLIP often relies on cosine similarity which can be misleading [46].



① Right for the **wrong concepts**

② Concepts rely on **unintended information**

③ Concepts are **entangled** with each other

④ Concept are **wrongly correlated**

Metric 1: Accurate Concept Predictions [9]

How do we measure if $h(x)$ is doing a good job?

Area Under the ROC Curve (AUC)

$$\text{AUC}(C) = \frac{1}{K} \sum_{k=1}^{K} \text{AUC}(c_k)$$

- Robust to unbalanced classes.
- High AUC is necessary but **not sufficient** for interpretability.

Metric 2: Concept Leakage [9]

Leakage occurs when concepts carry information semantically incongruent with the concept label.

- *Example*: A model predicts the concept "young" by looking at a basketball team logo in the background.
- This compromises "Intervenability" (changing the concept "young" won't behave as expected).

Metric 3: Disentanglement [9]

Concepts should be independent.

- **Definition**: Manipulating the input to alter one concept should not change others.
- **DCI Metric**: Measures the degree to which a concept c_i is responsible for only one semantic attribute.
- Unwanted correlations (e.g., "Wheels" and "Road") often lead to entanglement.

Bibliography:

1 Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., & Liang, P. (2020, November). Concept bottleneck models. In International conference on machine learning (pp. 5338-5348). PMLR.

2 Díaz-Rodríguez, Natalia, et al. "EXplainable Neural-Symbolic Learning (X-NeSyL) methodology to fuse deep learning representations with expert knowledge graphs: the MonuMAI cultural heritage use case." Information Fusion 79 (2022): 58-83.

3 Bennetot, A., Franchi, G., Del Ser, J., Chatila, R., & Diaz-Rodriguez, N. (2022). Greybox XAI: A Neural-Symbolic learning framework to produce interpretable predictions for image classification. Knowledge-Based Systems, 258, 109947.

4 Shin, Sungbin, et al. "A closer look at the intervention procedure of concept bottleneck models." International Conference on Machine Learning. PMLR, 2023.

5 Yang, Yue, et al. "Language in a bottle: Language model guided concept bottlenecks for interpretable image classification." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2023.

Bibliography:

6 Oikarinen, Tuomas, et al. "Label-free concept bottleneck models." arXiv preprint arXiv:2304.06129 (2023).

7 Kazmierczak, R., Berthier, E., Frehse, G., & Franchi, G. (2023). CLIP-QDA: An explainable concept bottleneck model. arXiv preprint arXiv:2312.00110.

8 Srivastava, Divyansh, Ge Yan, and Lily Weng. "Vlg-cbm: Training concept bottleneck models with vision-language guidance." Advances in Neural Information Processing Systems 37 (2024): 79057-79094.

9 Debole, Nicola, et al. "If Concept Bottlenecks are the Question, are Foundation Models the Answer?." arXiv preprint arXiv:2504.19774 (2025).

10 Kazmierczak, Rémi, et al. "Benchmarking xai explanations with human-aligned evaluations." arXiv preprint arXiv:2411.02470 (2024).