XAI Report – Cats, Dogs and Cars

Team: Yulong MA, Boyuan ZHANG, Huanshan HUANG

1 Introduction

In this lab, we use a small image dataset of cats, dogs, and cars. Each animal or vehicle comes in black and white variants. We put the data into separate folders based on class and color. This helps us easily study color bias. The dataset has 1950 training images, 650 validation images, and 655 test images. We compare a few different models. First is a supervised ResNet-50 baseline. Then we look at zero-shot and linear-probe CLIP models. Finally, we test a Concept Bottleneck Model, or CBM. We also use Explainable AI, or XAI, methods to understand how these models make decisions.

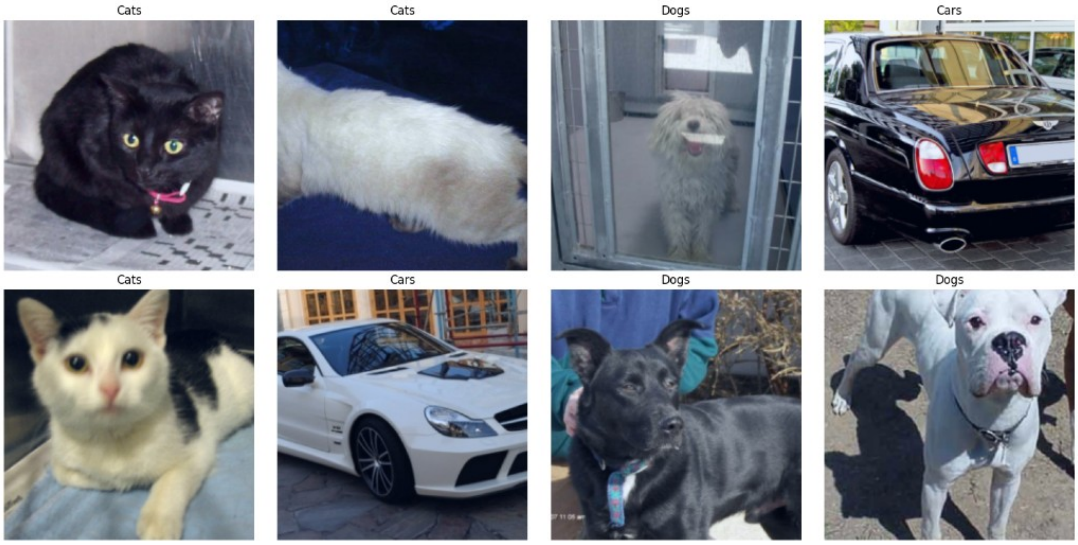

Figure 1: Sample images from the Cats/Dogs/Cars dataset, showing black and white examples of cats, dogs, and cars.

2 Classical explanations with Grad-CAM (Q1–Q2)

First, we train a ResNet-50 classifier on both black and white images across all splits. The best model reaches a test accuracy of 98.32 percent. We then apply Grad-CAM to a convolutional layer in the last block, called `layer4`. To do this, we calculate the gradients of the class score for each image. We average these gradients to get channel weights. Then, we combine the activations using these weights. Finally, we apply a ReLU function, normalize the result, and resize it to create a heatmap on the input image.

We test this on 100 images. We find 4 misclassifications and no weak activation cases based on our quality rules. When the model predicts correctly, the Grad-CAM heatmaps mostly focus on the main object. This makes the explanation easy to understand. However, sometimes the model misclassifies a dog as a cat. In these cases, the heatmap spreads to background areas like fences. This means the model might be using context to guess the class. Grad-CAM gives us a

good basic idea, but it is not perfect. A simple threshold cannot catch all bad explanations, so we still need to check them manually.

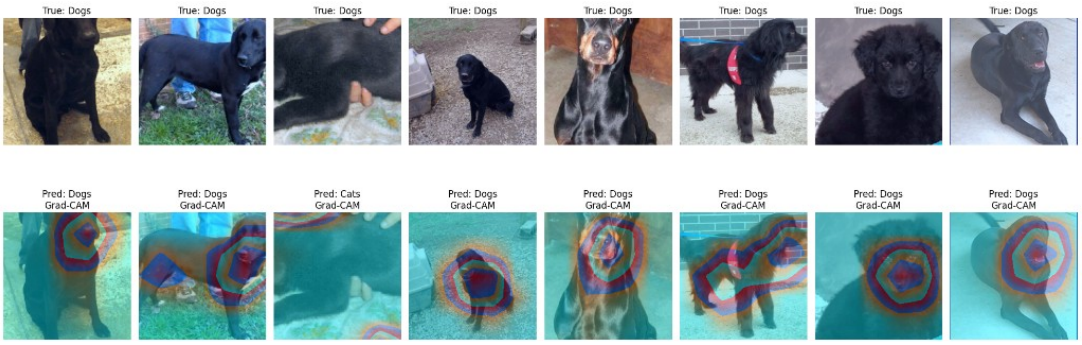

Figure 2: Grad-CAM explanations for correctly classified dog images. The heatmaps focus on the main object, often around the head or body of the dog.

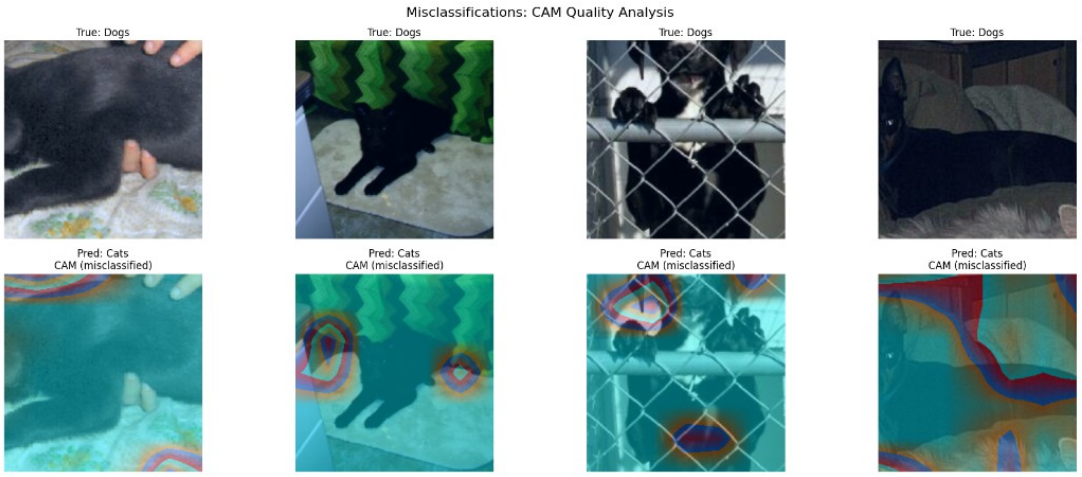

Figure 3: Grad-CAM explanations for misclassified dogs predicted as cats. Attention spreads over both the animals and background structures, making the explanation less clear.

3 CLIP zero-shot and linear probing (Q3–Q5)

Next, we test CLIP in a zero-shot setting using the ViT-B/32 version. We look at one specific white dog image located at `Dogs/White/9192.jpg`. We compare its image embedding to three text prompts. These prompts are "a car", "a dog", and "a cat". The model gives logits of 22.7, 26.8, and 28.0 for these prompts. This turns into probabilities of about 0.4 percent, 22.2 percent, and 77.4 percent. CLIP predicts "a cat" with the highest chance, even though the picture shows a dog. This shows that zero-shot CLIP can still make mistakes on tricky images.

We run these three prompts on the whole test set. We get a very high zero-shot accuracy of 99.54 percent. This is great because we did not even fine-tune the model. It shows that CLIP can easily separate these images in its vision-language space. The few errors probably come from weird or blurry pictures. Because this baseline is so good, we do not need to worry much about improving accuracy. Instead, we can focus on understanding how the model makes its choices.

After that, we train a linear probe on top of frozen CLIP embeddings. We turn each image into a vector with 512 dimensions. Then, we train one simple layer to map this vector to the three classes. We only update this single layer during training. After 10 epochs, the training accuracy is about 99.49 percent. The validation accuracy is 99.08 percent, and the test accuracy

reaches 99.39 percent. This proves that CLIP features are great for this task. The linear probe does not beat the zero-shot accuracy. However, it is slightly better than the ResNet-50 score. We mostly use this probe to check our features and set up the CBM.

4 CBM and concept bottlenecks (Q6–Q8)

We want to understand the decisions better, so we build a CBM using CLIP. We start with the 512-dimensional CLIP image vector. We compare it to some text concepts using cosine similarity. This gives us a new concept vector, which acts as our bottleneck. A final layer uses this bottleneck to predict the classes. We try two different sets of concepts. First, we use random concepts like black, white, wood, bottle, and cable. This gives a low test accuracy of 68.40 percent. These words do not help find cats or dogs, so the model loses useful details. Second, we use task-related concepts like "a cat", "a dog", and "a car". This boosts the test accuracy to 99.39 percent. This matches the linear probe and is very close to the zero-shot score. The bottleneck now fits the classes perfectly. The model just picks the class that has the highest similarity score.

Next, we test how dataset bias affects the model. We train a new CBM using only white images for the training and validation steps. We keep the model structure the same. Then, we test it on a normal set containing both black and white images. The accuracy drops to 56.79 percent on this test set. This is much worse than the 99 percent we got with fair training data. It shows that the model took a shortcut and learned to rely on color. It did not learn real features about the animals or cars. A biased dataset can easily trick the model.

We then look at the CBM concept space for the class-aligned setup. We check the 247 test images for dogs. The average similarity score for "a dog" is around 0.24. The score for "a cat" is about 0.20, and for "a car" it is about 0.18. We see a clear pattern when we look at all 655 test images. Cars score highest on the "a car" concept with an average of 0.241. Cats score highest on "a cat" with 0.249. Dogs score highest on "a dog" with 0.243. Boxplots show that each class wins in its own category. However, the other scores do not drop to zero. For example, cats still get a dog score of about 0.20. This makes sense because cats and dogs share some visual features. The bottleneck is easy to read but still captures real-world fuzziness.

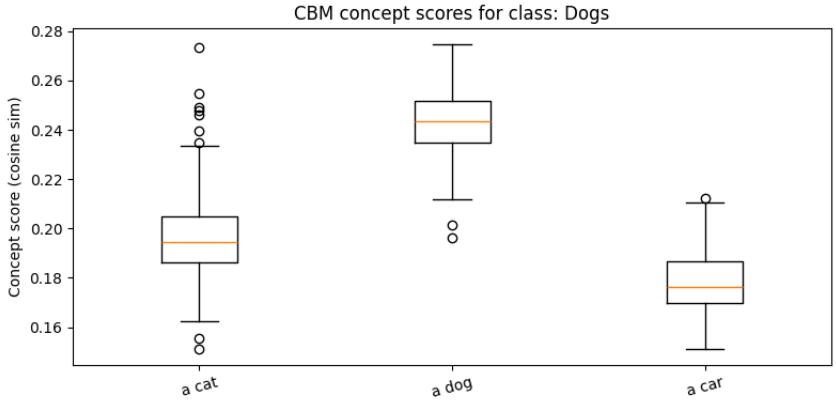

Figure 4: Concept scores in the CBM bottleneck for Dogs test images. The distribution for “a dog” is clearly higher than for “a cat” and “a car”.

5 Post-hoc methods with CBM: LIME and SHAP (Q9–Q10)

Now we use LIME and SHAP to explain our CBM predictions. First, we try LIME on images. We change small image patches and run them through the whole CBM pipeline using a black

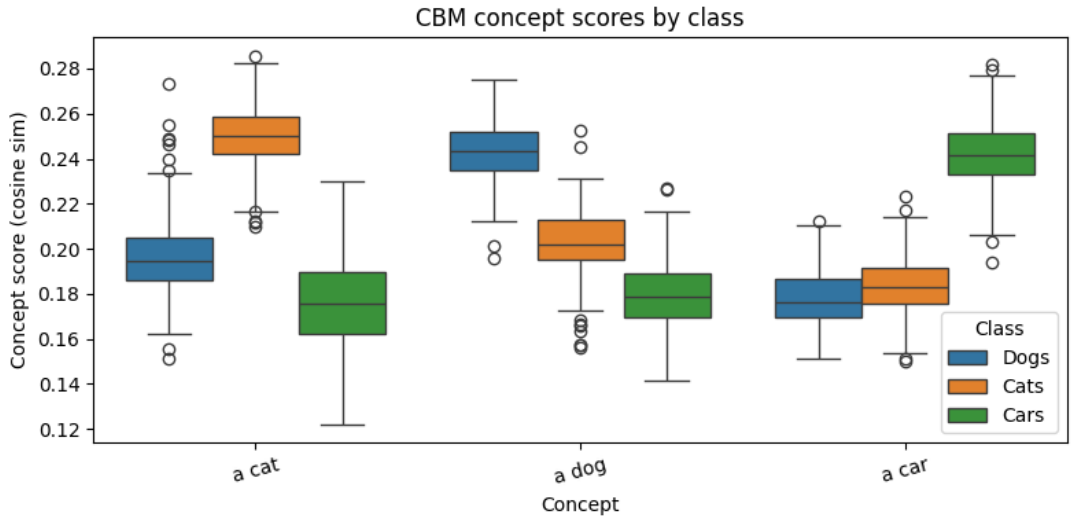

Figure 5: CBM concept scores by class. Each class tends to score highest on its own concept (diagonal pattern), while non-matching concepts remain non-zero and overlap, reflecting natural ambiguity.

dog image. The result highlights the dog and a bit of the background. This means the model looks at the animal but also uses some background texture. Next, we use LIME on the concept space. We check the three concept scores for the same image and explain the final linear layer. The results show that a dog score greater than 0.25 strongly supports the dog class. On the other hand, high car or cat scores hurt the dog prediction. This gives us a very clear explanation using words. If we just run LIME on the old ResNet-50 model, we only get highlighted image patches. We miss out on the word-based explanations because ResNet has no concept layer.

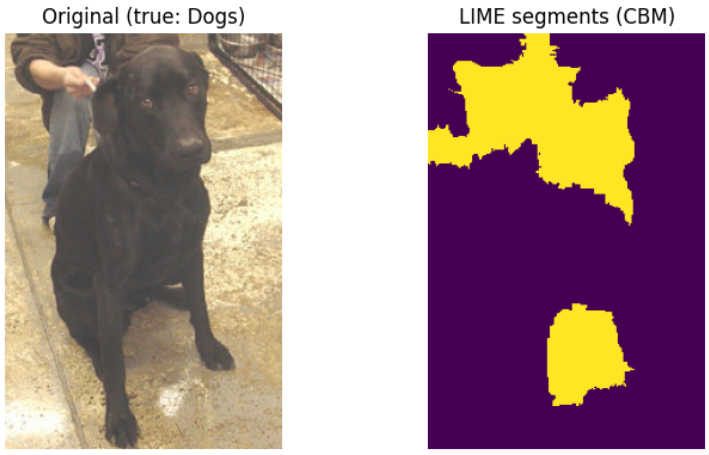

Figure 6: LIME image-space explanation for the CBM on a Dogs example. Superpixels covering the dog and a background patch are marked as important for the prediction.

We then try SHAP to see how each feature adds up. We start in the embedding space. We trace the gradients back to the 512-dimensional CLIP vector to compute SHAP values. A bar plot shows the top 20 numbers that push the dog score up or down. Sadly, these raw numbers mean nothing to humans. We switch to the concept space and apply SHAP to the three CBM scores. For our dog image, the "a dog" concept score of 0.253 adds about 0.14 to the final prediction. The car score of 0.183 and the cat score of 0.199 push the prediction down by 0.12 and 0.10. This is super easy to read. It simply says the model picks dog because it looks like

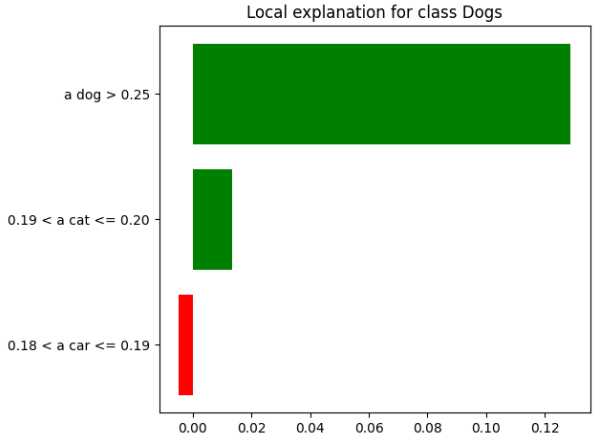

Figure 7: LIME concept-space explanation for the CBM. The rule "a dog > 0.25" strongly supports the Dogs class, while certain intervals of "a car" and "a cat" scores contribute negatively or only weakly.

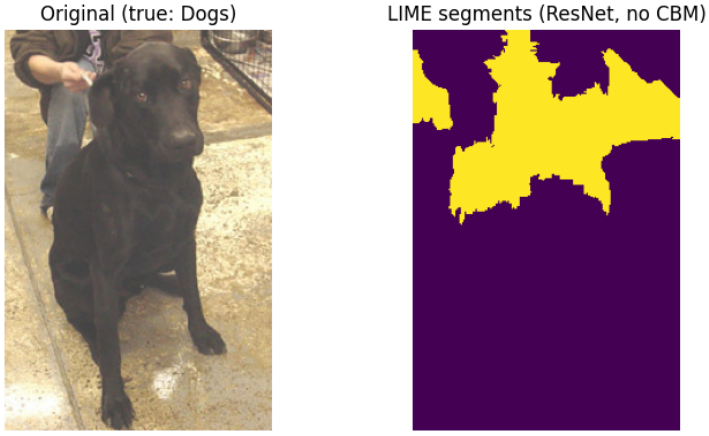

Figure 8: LIME image-space explanation for the ResNet baseline. Important superpixels cover a large part of the background, suggesting a stronger reliance on contextual cues compared to the CBM.

a dog and does not look like a car or cat. We can run SHAP on the linear probe without the CBM, but again, we lose these nice human-friendly concepts.

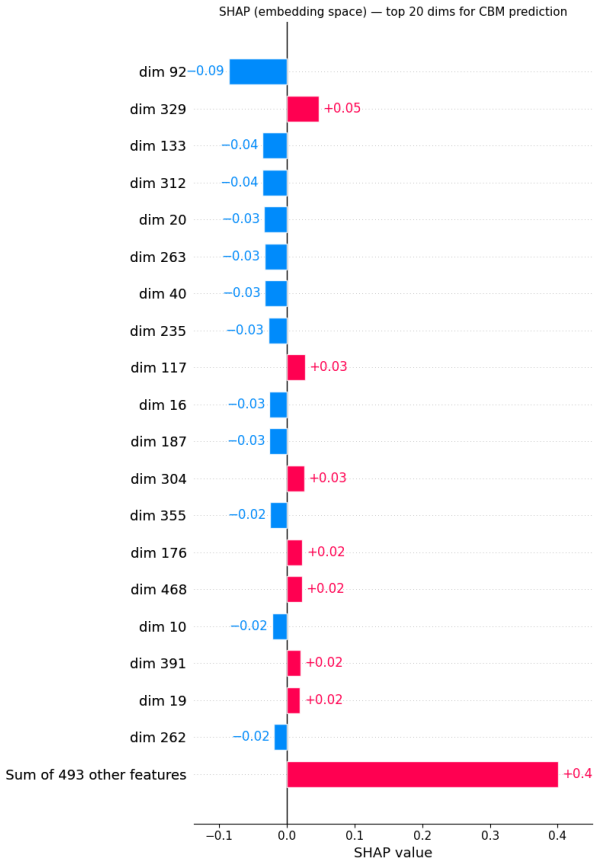

Figure 9: SHAP bar plot in CLIP embedding space for the CBM prediction (top 20 dimensions). The remaining dimensions are aggregated, and individual dimensions are not directly interpretable.

Figure 10: Test image used for the SHAP explanations (predicted Dogs, true Dogs).

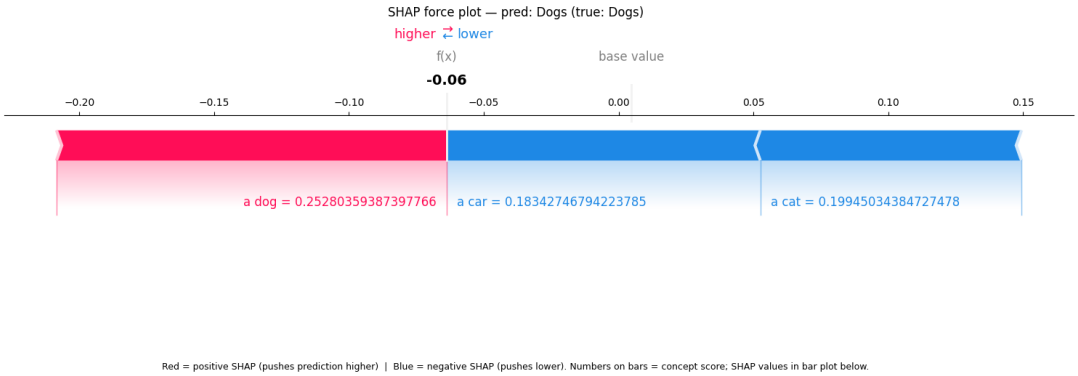

Figure 11: SHAP force plot in concept space for the Dogs prediction. The "a dog" concept pushes the prediction higher, while "a car" and "a cat" push it lower.

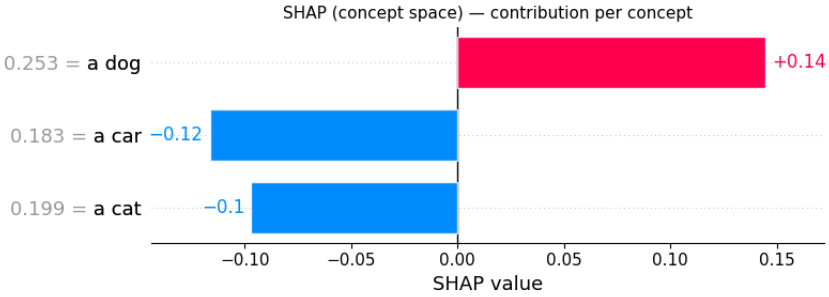

Figure 12: SHAP bar plot in concept space showing per-concept contributions for the Dogs prediction (positive for "a dog", negative for "a car" and "a cat").

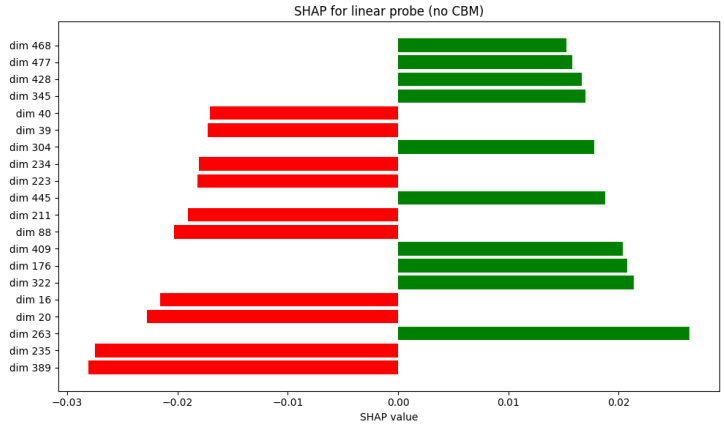

Figure 13: SHAP values for the linear probe without a CBM. Explanations are in embedding space only, so they are harder to map to human concepts.

6 Counterfactuals in concept space (Q11)

Finally, we create counterfactual explanations using the CBM concepts. We take a test image predicted as a dog. We look at its concept scores for cat, dog, and car. We change one score at a time and keep the others the same. We sort the concepts by their importance. Then, we slowly tweak the scores to see how much change is needed to flip the prediction. In our test, we need to drop the dog score by 0.1. Alternatively, we can raise the car or cat score by 0.1.

We plot these changes on a heatmap. It shows that slightly lowering the dog score changes the prediction to a cat. Slightly raising the car score changes the prediction to a car. Raising the cat score changes it to a cat. All these tweaks are small, around 0.1. This means the model is pretty close to changing its mind on this image. These counterfactuals are great because they tell us exactly what to change, which way to go, and by how much.

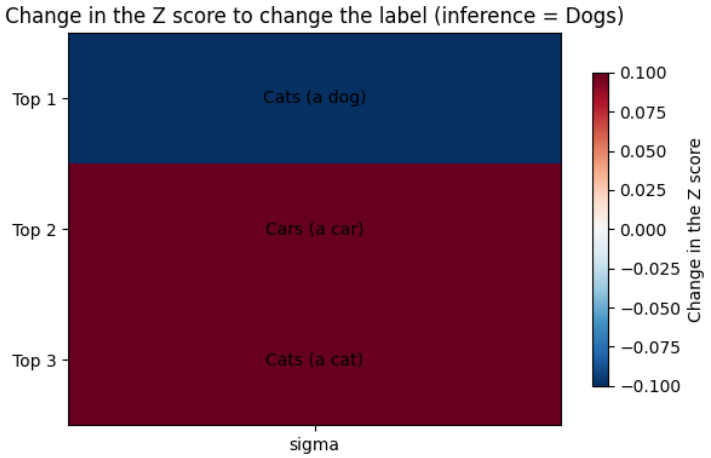

Figure 14: Counterfactual explanation in concept space. Each row shows the minimal change in one concept score (delta in z) needed to flip the Dogs prediction to Cats or Cars for this example.

7 Conclusion

In conclusion, the CLIP and CBM models beat the standard ResNet-50 on our dataset. Zero-shot CLIP gets an accuracy of 99.54 percent. The linear probe and the aligned CBM both hit 99.39 percent. The ResNet-50 baseline trails at 98.32 percent. The CBM shines when we use task-related concepts like cat, dog, and car. It keeps high accuracy while being easy to understand. If we use random concepts, the accuracy crashes to 68.40 percent. This shows that bad concepts ruin the model. LIME and SHAP work very well with the CBM. They can explain choices using both pictures and words, which is much better than just looking at raw numbers. We also learned that bias is dangerous. Training on only white images dropped our accuracy to 56.79 percent. Also, tiny changes around 0.1 can completely flip a prediction. These issues show why we really need good testing and XAI tools to keep models reliable.