Wild Animal Identification and Comparison

**Hugging Face Demo:** <https://huggingface.co/spaces/cse559-final-project/animal-similarity>

We focus on implementing animal detection and classification using pre-trained models from HuggingFace. This approach allows us to apply both object detection and image classification methods learned in class. We may also further explore species similarity comparison using feature embedding.

1. **Object Detection**

• **Model**: facebook/detr-resnet-50 (DETR with ResNet-50 backbone)

<https://huggingface.co/facebook/detr-resnet-50>

• **Purpose**: Detect animals in the image and extract bounding boxes

The DETR model is an encoder-decoder transformer with a convolutional backbone. Two heads are added on top of the decoder outputs in order to perform object detection: a linear layer for the class labels and a MLP (multi-layer perceptron) for the bounding boxes. The model uses so-called object queries to detect objects in an image. Each object query looks for a particular object in the image. For COCO, the number of object queries is set to 100.

1. **Image Pair Generation**

• **Dataset:** Animals-10[**https://www.kaggle.com/datasets/alessiocorrado99/animals10**](https://www.kaggle.com/datasets/alessiocorrado99/animals10)

The dataset contains about 28K medium-quality animal images belonging to 10 categories: dog, cat, horse, spyder, butterfly, chicken, sheep, cow, squirrel, and elephant.

All the images have been collected from "google images" and have been checked by humans. There is some erroneous data to simulate real conditions (eg. images taken by users of your app). The main directory is divided into folders, one for each category. The image count for each category varies from 2K to 5 K units.

• **Approach**:

1. Generated same-species (positive) and different-species (negative) pairs (The four sets of data are 95 pairs, 335, 635, and 935 respectively).
2. Used itertools. combinations to construct sample pairs
3. Data Structure: Stored as a animal\_pairs.csv file with fields: ["image1", "image2", "class1", "class2", "same\_species"]
4. **Image Classification Models (Labelling)**

• **Model**: google/vit-base-patch16-224 <https://huggingface.co/google/vit-base-patch16-224>

To classify the species of each detected animal, we employed the Vision Transformer (ViT) model provided by Hugging Face's Transformers library. Specifically, we used the pre-trained model google/vit-base-patch16-224 along with its associated processor AutoImageProcessor.

ViT is a transformer-based architecture that treats an image as a sequence of fixed-size patches, enabling it to model global context via self-attention. This allows ViT to achieve strong performance on image classification tasks compared to traditional convolutional neural networks (CNNs).

1. **Similarity Comparison**

• **Method**: Cosine similarity of embeddings

To determine whether two animals in a given image pair belong to the same species, we performed similarity comparisons using cosine similarity on the embeddings generated by different models. Each image in the pair was first processed through a feature extractor, and then their embeddings were compared using cosine similarity.

The models and methods used for comparison include:

* **ViT**: Extracted embeddings using ViTModel and AutoModelForImageClassification. Offers strong global feature modeling but was less effective in our binary similarity task, likely due to lack of fine-tuning.
* **ResNet**: Classic CNN-based feature extractor known for strong local texture representation. Produced relatively low recall in our setup, suggesting it may not generalize well without task-specific tuning.
* **MobileNetV2 (Fine-tuned)**: A lightweight CNN architecture that we fine-tuned on the Animals-10 dataset.

To fine-tune MobileNetV2 on the Animals-10 dataset, we followed these steps:

1. **Data Preparation**: Loaded the dataset using PyTorch’s ImageFolder and applied standard image transformations (resize, normalization, and tensor conversion).
2. **Dataset Split**: Divided the dataset into training and validation sets (80/20 split).
3. **Model Adjustment**: Loaded the pre-trained MobileNetV2 model and replaced its final classifier layer with a new linear layer matching the number of animal classes.
4. **Training Setup**: Used cross-entropy loss and Adam optimizer with a learning rate of 2e-4. Trained for 3 epochs using batch size 32.
5. **Performance**: Achieved 97.35% training accuracy and 95.87% validation accuracy after 3 epochs.
6. **Model Saving**: The trained model was saved as mobilenetv2\_animals10\_finetuned.pth for later use in embedding-based similarity comparison.

This fine-tuning significantly improved the quality of feature embeddings used in similarity calculations, making MobileNetV2 one of the top-performing models in our evaluation.

* **Label Matching**: Involves classifying both images independently and checking if the labels match. While precise, it relies heavily on classification accuracy and lacks semantic embedding comparison.
* **CLIP**: A powerful multi-modal model that produces high-level semantic embeddings. It performed well in capturing similarity beyond surface-level features.

1. **Model Evaluation – Metrics (Based on similarity comparison)**

We evaluated the performance of each model on datasets containing different numbers of image pairs (95, 335, 635, and 935). For each dataset, the following classification metrics were computed to assess the ability of each model to identify whether a pair of animals belongs to the same species: **Accuracy, Precision, Recall, F1 Score.**

A table with numbers and a number on it

AI-generated content may be incorrect.

We observed a clear trend: **MobileNetV2 (after fine-tuning)**and**CLIP**consistently outperformed other models across all pair sets. In particular, MobileNetV2 achieved high accuracy and F1 scores due to its domain-specific tuning, while CLIP provided robust semantic matching thanks to its large-scale pretraining.

• **Conclusion:** MobileNet and CLIP are the strongest contributors so we will use these two models as the main model and label matching as the submodel. ViT and ResNet embeddings alone are too weak (likely under-trained or unsuited for similarity).

1. **Grid Search to Find the Best Weights for Score Fusion**

To determine the optimal fusion strategy for combining similarity scores from different models, we conducted a small-scale grid search over a set of predefined candidate weight combinations. The fusion score is computed as a weighted sum of three components: MobileNetV2 similarity score (fine-tuned on Animals-10), CLIP similarity score, Label match score (1 if predicted labels are identical, else 0)

We selected six reasonable weight combinations based on prior experiments and model importance:

candidate\_weights = [

(0.6, 0.3, 0.1), # MobileNet dominant

(0.5, 0.3, 0.2), # Slightly higher label contribution

(0.4, 0.4, 0.2), # Balanced across all three

(0.4, 0.5, 0.1), # CLIP-favored

(0.3, 0.6, 0.1), # CLIP dominant

(0.6, 0.2, 0.2) # Conservative label usage

]

Output: Best Config: ((0.6, 0.3, 0.1), 0.7) | F1 Score: 0.9473684210526316

1. **Score Fusion – Final Decision**

To determine the fusion weights, we evaluated each model independently using Accuracy, Precision, Recall, and F1 Score. MobileNetV2 demonstrated the highest performance across all datasets, achieving up to 97% F1 Score. CLIP also performed well, especially in semantically similar but visually distinct cases. Label match was moderately helpful but prone to misclassification if the base classifier failed.

Based on this, we assigned weights of 0.6 (MobileNet), 0.3 (CLIP), and 0.1 (Label Match), prioritizing the models with better reliability while still allowing label agreement to contribute. Fusion improves robustness by leveraging the strengths of multiple models. In the same time, the confusion matrix accuracy achieves 94.05%.

• Combines multiple model scores using weighted average:

final\_score =

mobilenet\_weight \* sim\_mobile

+ label\_weight \* label\_sim

+ clip\_weight \* sim\_clip

• Based on the final score, we produce a final decision:

*"🟢 Highly likely same species"* ( >= 0.88)*,*

*"🟡 Probably in the same family but not the same species"* ( >= 0.7)

*"🔴 Likely different species"* (< 0.7)

• **Final CM Accuracy by Dataset Size:**

A table with numbers and letters

AI-generated content may be incorrect.

The CM accuracy remained consistently high (above 90%) across all dataset sizes, indicating the robustness of our comparison-based model. The highest accuracy was observed with 95 pairs (93.63%), while slight fluctuations were noted with larger datasets. This suggests that while the model scales well with data, some variability may arise due to increased diversity and complexity in larger pair sets.