PetFace Face Identification Pipeline: Training, Inference, and Accuracy Evaluation Summary

1 System Overview

This module implements various neural network architectures for pet face identification/re-identification. The project compares different loss functions (cross-entropy vs ArcFace) and classification methods (softmax vs cosine similarity) for identifying individual animals across multiple species.

Class Summary

- ArcFaceLayer: Custom layer implementing ArcFace loss for better feature learning
- ReidentModel: Abstract base class for all identification models
- SoftmaxModel: Traditional softmax classification approach
- SimilarityModel: Cosine similarity-based identification
- ArcFaceTrainableModel: Base class for models using ArcFace loss
- SoftmaxArcFaceModel: ArcFace loss with softmax classification
- SimilarityArcFaceModel: ArcFace loss with similarity-based identification

2 Training

We train the model with callbacks for monitoring and early stopping.

Strategy:

1. Save best model weights during training
2. Stop early if validation loss stops improving (i.e. prevent overfitting)
3. Train for up to 25 epochs
4. Save training history for analysis

ReidentModel

Abstract base class for all animal face re-identification models

ReidentModel provides the common functionality needed for training and evaluating different neural network architectures for pet face identification. The project compares multiple approaches:

1. Loss functions: Traditional cross-entropy vs ArcFace angular margin loss
2. Classification methods: Softmax classification vs cosine similarity matching

All models use transfer learning with ResNet50 as the backbone, pre-trained on ImageNet. The functions include:

- _init_: Sets up a place to save model as well as outputs and checkpoints
- number_of_classes: Counts number of individual animals
- checkpoint_file: Path to save and load model weights during training
- history_file: Path to save training history (loss, accuracy over epochs)
- compile: Evaluates model performance on test data. Must be implemented by subclasses
- backbone_model: Creates backbone feature extractor using transfer learning
  - This uses ResNet50, pre-trained on ImageNet, as the foundation. This leverages learned visual features from millions of images then fine-tunes for the specific task of animal face identification.
  - Transfer learning strategy:
    1. Load ResNet50 without final classification layer (include_top=False)
    2. Freeze most layers to preserve learned features
    3. Unfreeze last 30 layers for fine-tuning on animal data
- process_inputs: Optimizes data pipeline for training efficiency
- train: Trains model with callbacks for monitoring and early stopping
  - Strategy:
    1. Save best model weights during training
    2. Stop early if validation loss stops improving (prevent overfitting)
    3. Train for up to 25 epochs
    4. Save training history for analysis
- load_from_checkpoint: Loads previously trained model weights from checkpoint file

- display_training_history: Creates interactive plots showing training progress over time
  - Visualizes key metrics to assess model performance:
    - Accuracy metrics: How well model identifies animals correctly
    - Loss values: How confident model is in predictions
    - Training vs validation: Helps identify overfitting
  - Plots help determine if model is learning effectively and whether or not it generalizes well to unseen data

3 Initial Modeling

ArcFace Layer

We implement a custom ArcFace (Angular Margin Loss) layer. ArcFace is an advanced loss function that improves upon traditional softmax by adding angular margin in the angular space, leading to more discriminative feature embeddings. This is particularly useful for face identification tasks where we need to distinguish between many different individuals.

The layer works by:

1. Normalizing both input embeddings and learned weights to unit vectors
2. Computing angular distance between embeddings and class weights
3. Adding angular margin only to the ground truth class
4. Scaling the final logits for numerical stability

The arguments include:

- num_classes: Number of individual animals/identities
- margin: Angular margin in radians (default 0.05). Larger values create wider gaps between classes
- scale: Scaling factor for logits (default 20). Controls sharpness of distribution

The functions include:

- build: Weights matrix
- call: Implement ArcFace to return scale * logits

SoftmaxModel

Traditional classification model which uses cross-entropy loss and softmax activation

This baseline approach for animal face identification involves:

1. Extracting features using ResNet50 backbone
2. Adding dense layers for classification
3. Using standard cross-entropy loss
4. Making predictions using softmax probabilities

This approach treats each individual animal as a separate class and learns to classify images directly into these classes. The functions are:

- compile: Builds traditional classification model architecture
  - Architecture:
    1. ResNet50 backbone (feature extraction)
    2. Flattens layer (converts 2D features to 1D)
    3. Dense layer (512 units) + BatchNormalization (regularization)
    4. Output layer (number of animal classes)
  - Uses Adam optimizer and cross-entropy loss for multiclass classification
- evaluate: Performs standard evaluation using softmax probabilities
  - Outputs class probabilities for measuring how often correct animal appears in top-k predictions

SimilarityModel

Cosine similarity-based identification model

This approach learns feature embeddings and uses cosine similarity for identification:

1. Extract features using ResNet50 backbone
2. Learn 512-dimensional embeddings
3. At evaluation, compare query embeddings to all known embeddings
4. Rank by cosine similarity to find matches

This system mimics face recognition systems which compare a face to a database of known faces to find the best match. This model is trained exactly like the softmax model, using cross-entropy loss. However, the difference is in evaluation: Instead of using softmax probabilities, we extract embeddings and use cosine similarity. The functions include:

- compile: Builds similarity-based model architecture
- evaluate: Evaluates using cosine similarity instead of softmax probabilities
  - Process:
    1. Extracts embeddings for all training + validation images (reference gallery)
    2. Extracts embeddings for evaluation images (queries)
    3. Computes cosine similarity between query and gallery embeddings
    4. Ranks gallery images by similarity and checks if correct animal is in top-k
  - Simulates real world identification system which compares new image against database of known animals
- add_label_input: Helper function to format data for ArcFace models
  - ArcFace layer needs both images and labels during training to apply angular margin
  - Restructures data into required format with named inputs
  - Args:
    - images: Batch of input images
    - labels: Corresponding ground truth labels
  - Returns: Tuple of (dict with named inputs, labels) suitable for ArcFace training

ArcFaceTrainableModel

Base class for models using ArcFace angular margin loss

ArcFace improves upon traditional cross-entropy by adding angular margin in the embedding space. This creates larger gaps between different classes and leads to more discriminative features for face identification. ArcFace's key difference from traditional models is that it requires both images and labels during training to compute the angular margin. The functions are:

- process_inputs: Restructures inputs to prepare data for ArcFace training
  - ArcFace layer needs labels during forward-pass to apply angular margin, so we restructure the data to provide both images and labels as inputs

- compile: Builds model architecture with ArcFace layer
  - Architecture:
    1. ResNet50 backbone (feature extraction)
    2. Flatten + Dense layer (embedding generation)
    3. ArcFace layer (applies angular margin during training)
  - Model has two inputs (images and labels) because ArcFace needs ground truth labels to apply angular margin

4 Later Modeling

SoftmaxArcFaceModel

ArcFace loss with traditional softmax evaluation

This model trains with ArcFace angular margin loss but evaluates using standard softmax probabilities. This combines the benefits (improved feature learning) of ArcFace training with simple evaluation. The only function is:

- evaluate: Evaluates using trained ArcFace model with softmax probabilities
  - Despite training with ArcFace loss, can still use model to generate class probabilities for standard classification evaluation

SimilarityArcFaceModel

ArcFace loss with cosine similarity evaluation

This model combines ArcFace training (for improved embeddings) with cosine similarity evaluation (for more realistic identification scenarios). This represents the most sophisticated approach in our comparison. The functions are:

- evaluate: Evaluates using ArcFace-trained embeddings with cosine similarity
  - Combines best of both worlds
    1. ArcFace training produces better discriminative embeddings
    2. Cosine similarity evaluation matches real-world identification systems
  - Process is similar to SimilarityModel evaluation but uses embeddings learned with ArcFace angular margin loss
- get_Reidentification_model: Factory function to create appropriate model for experimentation

5 Model Summary

This project compares four different model configurations:

1. cross_entropy + softmax: Traditional classification (baseline)
2. cross_entropy + cosine_similarity: Standard embeddings with similarity matching
3. ArcFace + softmax: ArcFace training with traditional evaluation
4. ArcFace + cosine_similarity: ArcFace training with similarity evaluation (most advanced)

– Args:
  * animal: Animal species to train on (e.g., 'dog', 'cat', 'rabbit')
  * loss_type: 'cross_entropy' or 'ArcFace'
  * classification_type: 'softmax' or 'cosine_similarity'
  * batch_size: Training batch size

– Returns:
  * Configured model ready for training and evaluation

6 Accuracy Results

Cross-Entropy + Softmax

Overall Accuracy: 29.85 %

Species	Accuracy
guineapig	52.00
chinchilla	45.00
chimp	41.70
rabbit	41.00
degus	32.10
hamster	31.80
hedgehog	27.50
parakeet	23.40
pig	**21.50**
javasparrow	21.30
ferret	21.00
dog	19.50
cat	10.20

This model achieved some accuracy (including the highest accuracy for pigs), but accuracy is surprisingly lowest for those animals–dogs and cats–with abundant photos. This indicates a possibly vanishing gradient problem. With the fewest observations in the dataset, ferrets have the lowest accuracy apart from cats and dogs.

Cross-Entropy + Cosine Similarity

Overall Accuracy: 34.83 %

Species	Accuracy
guineapig	53.90
rabbit	48.20
chinchilla	45.00
chimp	41.70
dog	41.60
parakeet	36.30
cat	36.00
degus	34.10
hamster	33.50
hedgehog	27.50
javasparrow	24.00
ferret	21.00
pig	10.00

This model improves upon the previous one, and we see gains for guinea pigs, rabbits, dogs, parakeets, cats, degus, hamsters and javasparrows. Dogs and cats have large gains, meaning that the cosine similarity had some advantage over softmax in mitigating vanishing and/or exploding gradients. Only pigs suffered a loss in accuracy. This could be because there is low variance among the pig images. Because pigs are farmed in the largest volume of any animal on the list and because they also have the largest litter size, there could be more uniformity among them which is better picked up by softmax. Pigs' have flexible snouts which can change their face shape between images. Their pale skin with few markings makes them difficult to distinguish from each other, and it also makes them look very different when dirty. For these reasons, they are more difficult to recognize.

ArcFace + Softmax

Overall Accuracy: 24.49 %

Species	Accuracy
guineapig	39.80
chinchilla	37.70
rabbit	34.00
chimp	32.70
dog	27.80
degus	23.20
hedgehog	21.90
hamster	21.80
cat	21.40
parakeet	17.70
ferret	15.00
javasparrow	14.10
pig	11.30

This model performed the worst of the four. Only dogs and pigs performed worse in other models.

ArcFace + Cosine Similarity

Overall Accuracy: **39.89 %**

White Paper Accuracy: 53.80 %

Species	Accuracy (%)	White Paper Top Accuracy (any model)
guineapig	58.00	68.66
rabbit	54.00	69.13
chinchilla	53.00	69.86
dog	50.00	77.86
chimp	48.90	43.27
cat	44.00	70.30
parakeet	40.20	62.19
hamster	36.70	54.33
degus	35.80	56.08
hedgehog	31.50	44.38
javasparrow	25.90	42.27
ferret	23.10	46.12
pig	17.50	41.49

This model achieved the highest accuracy for all species except for pigs. We can see that there is some similarity in ranking between our model and the white paper's, with pigs ranking last in both cases. Our model did outperform the white paper's identification of chimps, but there are marked differences between most of the species. Notably, cats and dogs did not have the highest accuracies; although cats had the second highest accuracy in the white paper model, despite having vastly more images than dogs.