CONTENT

1. Introduction and Project Objective

The main goal of this project is to develop an image classification model capable of correctly identifying 10 different animal species, using the public Animals-10 dataset available on Kaggle. This dataset contains thousands of images distributed among the following categories: dog, cat, horse, elephant, butterfly, chicken, cow, sheep, spider, and squirrel.

The purpose is to build a robust model that not only achieves high prediction accuracy, but also generalizes well to unseen images, avoiding overfitting.

To accomplish this, several convolutional neural network (CNN) approaches are explored—starting from a custom architecture built from scratch and evolving towards transfer learning with pretrained models. Additionally, techniques such as data augmentation, regularization, and fine-tuning are applied to further enhance model performance.

This project combines both experimental design (comparing different models and configurations) and practical optimization, with the ultimate goal of developing an efficient image classifier that can be applied in real-world contexts, such as educational tools or automated animal recognition systems.

2. Dataset Exploration (EDA)

The Animals-10 dataset used in this project contains approximately 28,000 images distributed across 10 different animal classes: *dog, cat, horse, elephant, butterfly, chicken, cow, sheep, spider,* and *squirrel*. The images vary significantly in terms of background, lighting conditions, perspective, and quality — providing a realistic and challenging setting for image classification tasks.

To better understand the dataset, an Exploratory Data Analysis (EDA) was conducted, which included:

- Total number of images: ~28,000
- Image resolution: Varies across samples (rescaled to 224×224 pixels for training)
- Color diversity: Most images are in color, but some are in grayscale (B/W), introducing additional variability
- Dataset split: 80% training, 20% validation

A bar chart was generated to visualize the distribution of images per class, revealing that some classes (such as *dog* and *cat*) have more examples than others (e.g., *spider* or *squirrel*). This indicates a class imbalance, which can negatively affect model performance if not handled properly.

To mitigate this imbalance, data augmentation techniques (rotation, flipping, zooming, brightness adjustment, etc.) were applied, ensuring that the model learns from a more diverse and balanced set of examples.

3. Data Preparation

3.1. Image Preprocessing

All images were rescaled to 224×224 pixels to ensure a consistent input size for the CNN architectures. Pixel values were normalized to the range [0,1] by dividing by 255, allowing the model to train more efficiently and converge faster.

3.2. Data Augmentation

To prevent overfitting and increase data diversity, data augmentation was applied using the ImageDataGenerator class from Keras.The transformations used were moderate to preserve image clarity while simulating real-world variability.This augmentation introduces spatial and color variations, helping the model become more robust to lighting changes, orientations, and slight deformations in the images.

3.3. Train–Validation Split

The dataset was split into 80% training and 20% validation using the validation_split parameter.

3.4. Label Mapping

The dataset originally contained class names in Italian, which were translated into English for readability and clarity during analysis and visualization.

4. Tested Models

4.1. Model 1

Custom CNN Model

- Input: 180×180×3
- Conv Blocks:
  3 blocks (32 → 64 → 1180 filters, 3×3 kernels, ReLU activation, MaxPooling)
- Fully Connected:
  Flatten → Dense(256, ReLU) → Dropout(0.5) → Dense(10, Softmax)
- Parameters: ~8.5M
- Optimizer: Adam (lr=0.0005)
- Final Accuracy: ~60

4.1. Model 2

Custom CNN Model

- Input: 224×224×3
- Conv Blocks:
  3 blocks (32 → 64 → 1180 filters, 3×3 kernels, ReLU activation, MaxPooling)
- Fully Connected:
  Flatten → Dense(256, ReLU) → Dropout(0.5) → Dense(10, Softmax)
- Parameters: ~8.5M
- Optimizer: Adam (lr=0.0002)
- Final Accuracy: ~80%

4.1. Model 3

Custom CNN Model

- Input: 224×224×3
- Conv Blocks:
  4 blocks (32 → 64 → 128→128, 3×3 kernels, ReLU activation, MaxPooling)
- Fully Connected:
  Flatten → Dense(512 neuronas, ReLU) → Dropout(0.5) → Dense(10 neuronas, Softmax)
- Parameters: ~9.6M
- Optimizer: Adam (lr=0.001)
- Final Accuracy: ~ 75%
- Final Accuracy with (Trasnfer Learning,EfficientNetB0):~97%

5. Fine-Tuning and Improvements

This stage involved fine-tuning the pre-trained EfficientNet model to optimize it for animal classification.

- Selective Unfreezing: The model's lower layers (0-99) were kept frozen to preserve general knowledge. The top layers (from 100 onward) were unfrozen (trainable = True) to allow subtle adaptation to the specific dataset features.
- Low Learning Rate: The model was recompiled with a very small learning rate . This prevents drastically overwriting the powerful pre-trained weights, ensuring a stable, beneficial refinement. Learning rate: 0.001

6. Results Evaluation

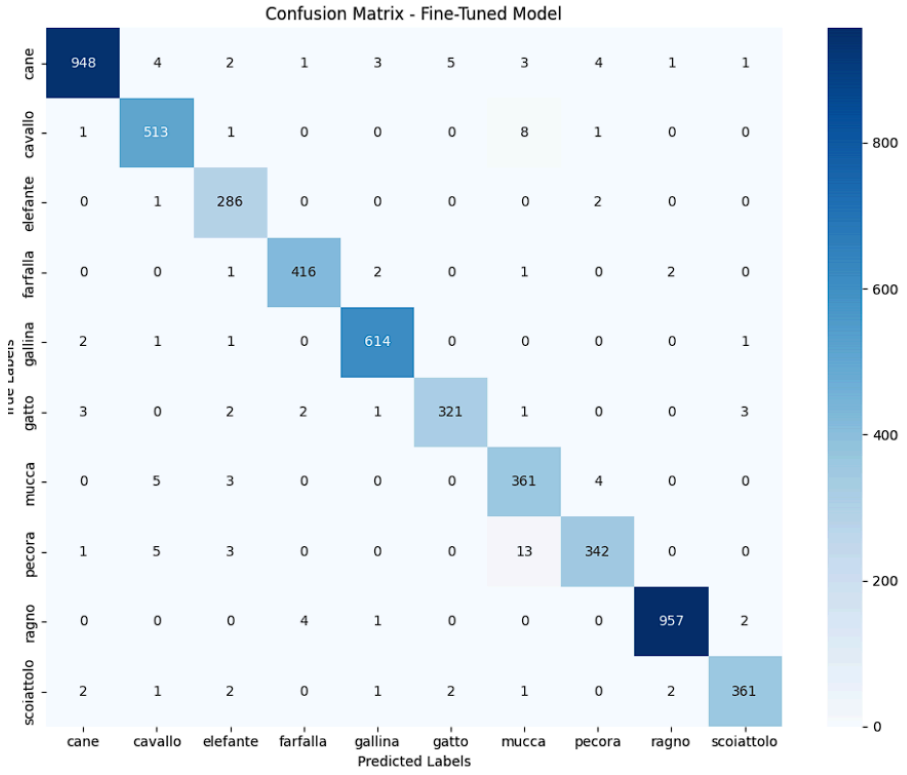
The final performance of the EfficientNet (Fine-Tuned) model is exceptionally strong, validating the success of the Transfer Learning and fine-tuning approach.

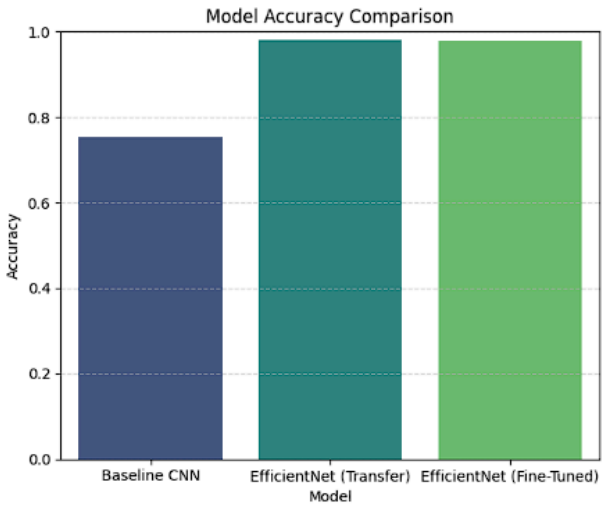
The model achieved high performance across all key validation metrics:

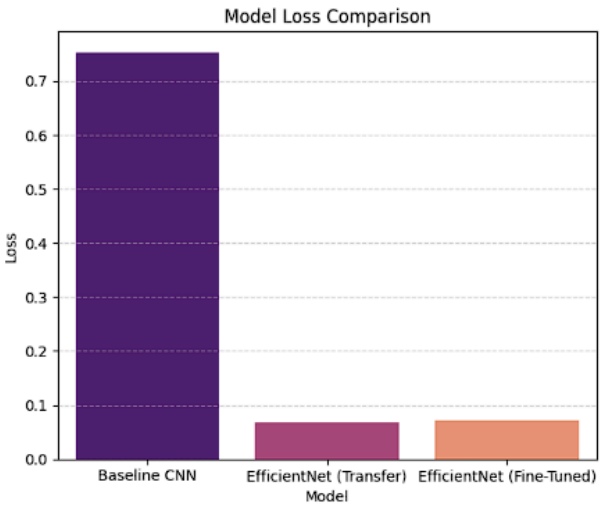
- Validation Accuracy: 0.9784 (97.84%)
- Validation Loss: 0.0722
- Macro F1-score: 0.97

This indicates that the model is highly effective at correctly classifying the animal images, while maintaining a very low error rate.

Model	Accuracy	Loss	Macro F1-score
Baseline CNN	7.525	7.535	734
EfficientNet (Transfer)	9.799	683	970
EfficientNet (Fine-Tuned)	9.784	722	970







7. Model Comparison

Model Name	Input Size	Conv Blocks	Parameters (Approx.)	Optimizer (LR)	Final Accuracy
Model 1 (Custom)	180×180×3	3	8.5M	Adam (0.0005)	~60%
Model 2 (Custom)	224×224×3	3	8.5M	Adam (0.0002)	~80%
Model 3 (Custom)	224×224×3	4	9.6M	Adam (0.001)	~75%
EfficientNet (Transfer)	224×224×3	Base	~4.0M	Adam (Varies)	~97%

8. Conclusions

The project successfully demonstrated that Transfer Learning using the pre-trained EfficientNet model is the optimal approach for this animal classification task, achieving a final accuracy of 97.84%. While performance is already near-perfect, the following avenues represent key areas for future work and potential marginal gains.

1. Advanced Fine-Tuning and Optimization

Given the current model's high performance, future efforts should focus on highly granular optimization to ensure stability, generalization, and efficient deployment.

- Hyperparameter Search for Fine-Tuning Depth: The current fine-tuning fixed the unfreeze point at layer 100. Future work should systematically explore the optimal number of layers to unfreeze (e.g., layers 80, 120, 150) to find the absolute best balance between retaining generalized features and specializing to the animal dataset.
- Exploring Larger EfficientNet Variants: Though EfficientNetB0 performed excellently, investigate scaling up to EfficientNetB2 or B3. While these models are more complex, they

may capture finer visual nuances (especially in the few classes with slightly lower F1-scores, like 'mucca' and 'pecora'), potentially leading to a marginal but critical increase in reliability.

- Learning Rate Scheduling: Implement more advanced learning rate decay strategies (e.g., cosine decay or learning rate reduction on plateau) during the fine-tuning stage. This ensures training steps are highly precise and less likely to cause sudden weight corruption.

2. Improving Generalization with Advanced Data Augmentation

Data augmentation is key to creating a more robust model that generalizes well to unseen or real-world conditions

- Implement Advanced Augmentations: Move beyond basic flips and rotations to use powerful techniques like CutMix or Mixup. These methods blend parts of different images or interpolate between them, significantly increasing data diversity and improving generalization.
- Investigate Domain-Specific Augmentation: Explore methods that simulate real-world challenges in animal photography, such as random blurring, aggressive color jitter, or partial occlusions (simulating animals being partially hidden by objects).

3. Model Robustness and Interpretability

- Cross-Validation: Implement k fold cross-validation on the final model to obtain a more statistically reliable and less dataset-split-dependent measure of performance.
- Model Interpretability : Employ techniques like Grad-CAM or SHAP to visualize which parts of the image the EfficientNet model is focusing on for its classification decision. This would provide valuable insights, especially for the slightly weaker performing classes, and build trust in the model's predictions.
- Adversarial Robustness: Test the model's resilience against small, imperceptible perturbations (adversarial attacks) that can cause misclassification. This is a critical step for deployment in any sensitive application.

GITHUB LINK

Project-I-Deep-Learning-Image-Classification-with-CNN/Animals_CNN.ipynb at main · malvarezgarcia1213-png/Project-I-Deep-Learning-Image-Classification-with-CNN