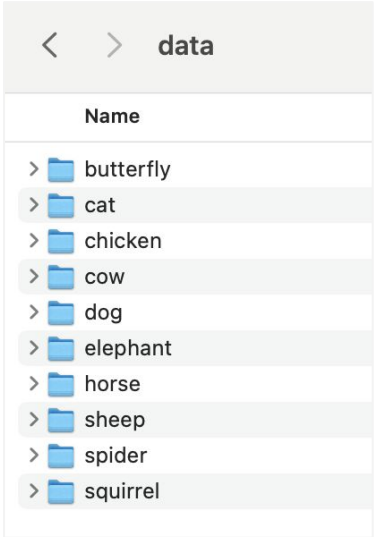
Deep Learning: Image Classification with CNN

Collaborators:

- **Paola Rivera**
- **Javier Dastas**

Data Processing

```
dataset = '/kaggle/input/animals10'
```



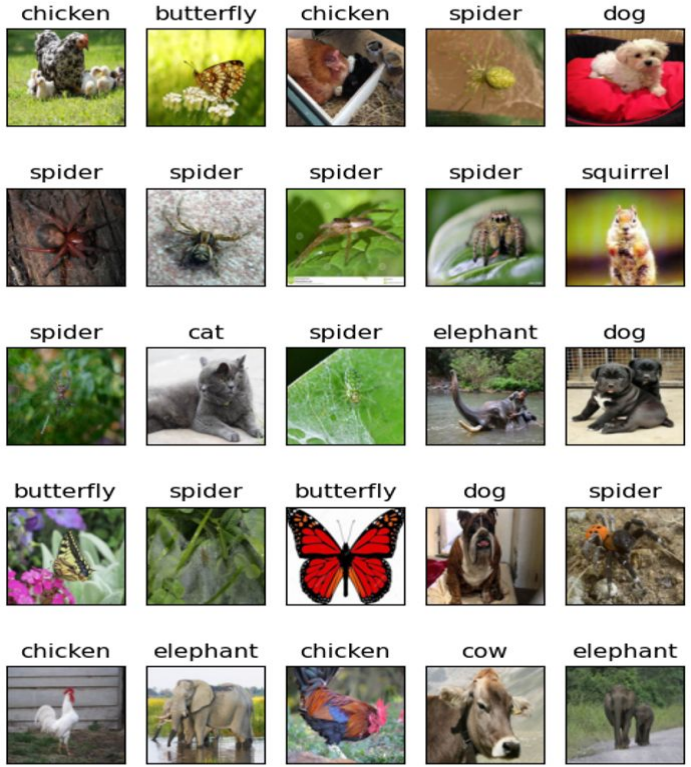

```
label
cane                4863
ragno               4821
gallina             3098
cavallo             2623
farfalla            2112
mucca               1866
scoiattolo          1862
pecora              1820
gatto               1668
elefante            1446
.ipynb_checkpoints     1
Name: count, dtype: int64

Loaded 26180 images with shape (128, 128, 3)
```

```
Total number of images: 26180
DataSet Shapes: torch.Size([128, 128, 3])

Training set size: 22253
Validation set size: 3927

Class names: ['cane', 'cavallo', 'elefante', 'farfalla', 'gallina', 'gatto', 'mucca', 'pecora', 'ragno', 'scoiattolo']

Classes in training set: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
Classes in validation set: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

Data Processing

```
# Visualize some images
batch_images, batch_labels = next(train_generator)
plt.figure(figsize=(5, 7))
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(batch_images[i])
    plt.xlabel(list(train_generator.class_indices.keys())[np.argmax(batch_labels[i])])
plt.show()
```

```
label
cane                 4863
ragno                4821
gallina              3098
cavallo              2623
farfalla             2112
mucca                1866
scoiattolo           1862
pecora               1820
gatto                1668
elefante             1446
.ipynb_checkpoints      1
Name: count, dtype: int64

Loaded 26180 images with shape (128, 128, 3)
```



Model Architecture

Input Layer: Accepts images of size 128x128x3 (RGB images).

Convolutional Blocks:

- **3 blocks of convolutional layers** extract spatial features.
- Each block has **Conv2D** layers with **ReLU** activation and **BatchNormalization** to stabilize learning.
- **MaxPooling** reduces spatial dimensions, focusing on key features.

Global Average Pooling: Reduces feature maps to a single vector, minimizing overfitting.
Fully Connected Layers:

- **Dense layer** (128 neurons) learns complex patterns.
- **Dropout layers** (0.1 and 0.5 rates) prevent overfitting.
- **Final Dense layer** (10 neurons with softmax) predicts probabilities for 10 classes.

Regularization: **BatchNormalization** and **Dropout** improve training stability and generalization.

Model Training

- **Steps Per Epoch:**
  - Calculated based on the number of batches (12) in the training and validation datasets to ensure all data is processed in each epoch.
- **Model Compilation:**
  - Optimizer: Adam (adaptive learning rate).
  - Loss Function: Sparse categorical crossentropy for multi-class classification.
  - Metrics: Accuracy to evaluate model performance.
- **Early Stopping:**
  - Monitors validation loss to prevent overfitting.
  - Stops training if validation loss doesn’t improve for 5 consecutive epochs.
  - Restores the best model weights from the training process.
- **Training:**
  - Runs for 12 epochs, with the training and validation datasets passed via data loaders.
  - History logs metrics like loss and accuracy for both training and validation sets after each epoch.
- **Results:**
  - Training accuracy improves steadily, reaching 78.6%, while validation accuracy peaks at 75.1%.
  - Validation loss decreases initially but fluctuates, indicating potential overfitting.

Model Evaluation

Validation Performance:

- Loss and accuracy are evaluated on the validation dataset, yielding:
  - Validation Loss: *value from evaluation output*.
  - Validation Accuracy: *value from evaluation output*.

Predictions and Ground Truth:

- Predictions are generated using the validation dataset.
- True Labels: Extracted directly from the validation loader.
- Validated that the lengths of true and predicted labels match.

Classification Metrics:

- Classification Report: Summarizes precision, recall, F1-score, and support for each class using class labels.
- Confusion Matrix: Highlights misclassifications and provides a class-wise error breakdown.

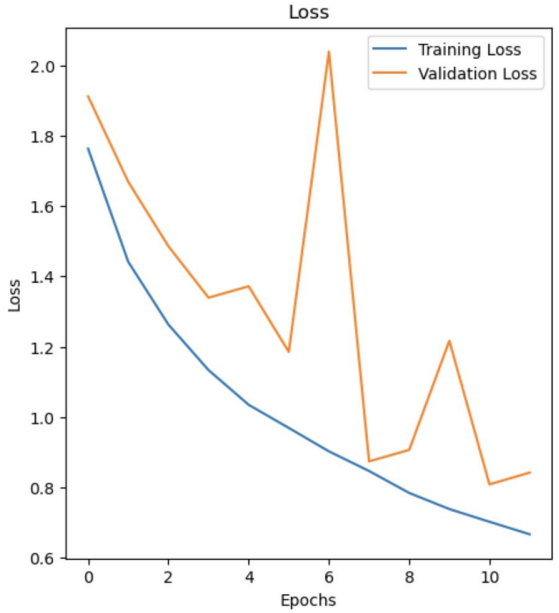
Training History Visualization:

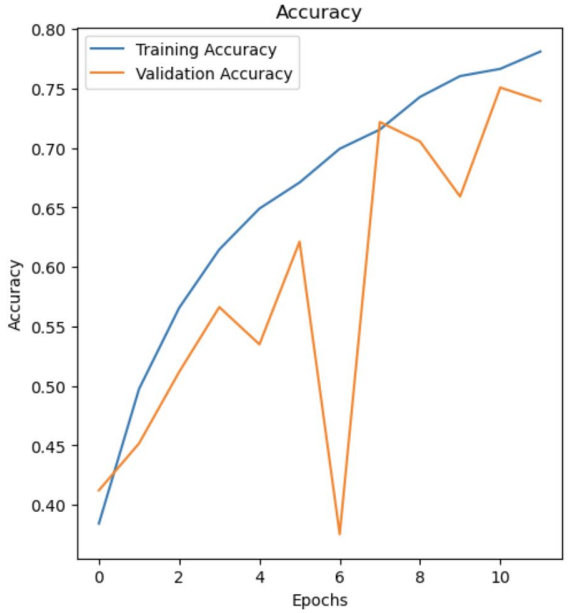
- Loss Plot: Tracks training and validation loss across epochs to identify overfitting or underfitting.
- Accuracy Plot: Shows training and validation accuracy trends, highlighting performance improvements.

Model Evaluation

```
Len true_classes: 3927
Len predicted_classes: 3927
Classification Report:
              precision    recall  f1-score   support

        cane       0.79      0.73      0.76       701
     cavallo       0.75      0.67      0.71       422
    elefante       0.84      0.70      0.77       220
    farfalla       0.68      0.87      0.76       296
     gallina       0.85      0.78      0.82       486
       gatto       0.73      0.58      0.65       226
       mucca       0.72      0.58      0.65       289
      pecora       0.61      0.75      0.67       277
       ragno       0.74      0.94      0.83       720
  scoiattolo       0.80      0.62      0.70       290

    accuracy                           0.75      3927
   macro avg       0.75      0.72      0.73      3927
weighted avg       0.76      0.75      0.75      3927

Confusion Matrix:
[[512  31   4  21  16  21  14  42  29  11]
 [ 29 284  12  10  11   1  31  18  24   2]
 [  9  15 155   3   2   3   4  16  10   3]
 [  2   0   0 257   2   1   0   0  33   1]
 [ 21   3   2  26 379   3   0   8  36   8]
 [ 42   1   1  12   2 132   2   3  27   4]
 [ 16  33   2   4   7   2 169  36  13   7]
 [  4   8   6   2   9   5  13 207  16   7]
 [  6   1   2  29   2   4   0   0 675   1]
 [ 11   2   0  14  14   8   1   9  52 179]]
```

```
Validation Loss: 0.8091505169868469
Validation Accuracy: 0.7509549260139465
```





Transfer Learning (Pre-Trained Models)

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0

https://keras.io/api/applications/

Transfer Learning (Freezing)

Pre-Trained Model:

- Used EfficientNetB0 (pre-trained on ImageNet) as the base model.
- The base model's weights are frozen, retaining its learned features.

Custom Classification Head:

- Added layers on top of the base model to adapt it to the current task:
  - Global Average Pooling: Reduces the spatial dimensions of feature maps.
  - Dense Layer: 128 neurons with ReLU activation for feature learning.
  - Dropout (0.5): Prevents overfitting.
  - Output Layer: 10 neurons with softmax activation for multi-class classification.

Model Compilation:

- Optimizer: Adam with a learning rate of 0.001.
- Loss Function: Sparse categorical crossentropy for classification.
- Metric: Accuracy to evaluate model performance.

Transfer Learning - Data Set Preparation

- **Data Augmentation:**
  - Applied to the training set to improve model generalization and handle overfitting:
    - Normalization: Scales pixel values to [0, 1].
    - Transformations: Includes random rotations, shifts, zoom, shear, and horizontal flips.
  - Validation data is normalized but not augmented to evaluate the model fairly.
- **Data Splitting:**
  - Used ImageDataGenerator to split the dataset:
    - 80% Training: For model learning.
    - 20% Validation: For performance evaluation.
- **Class Imbalance Handling:**
  - Class Weights: Computed using `compute_class_weight` to address imbalanced class distribution.
  - Ensures the model gives equal importance to underrepresented classes during training.
  - Example: Class weights dynamically adjust the loss contribution for each class.

Transfer Learning (Unfreezing)

Fine-Tuning:

- Fine-tuning was performed twice, with improved results in the second iteration.

Unfreezing the Base Model:

- The EfficientNetB0 base model was unfrozen, allowing its pre-trained layers to be updated during training.

Recompilation:

- Lower Learning Rate: Reduced to 0.0001 to avoid large weight updates and maintain the pre-trained knowledge.
- Retained Sparse Categorical Crossentropy as the loss function and Accuracy as the metric.

Training Process:

- Epochs: Fine-tuned for 5 additional epochs.
- Used class weights to address class imbalances in the dataset.
- Training and validation steps were calculated based on the dataset size.

Key Outcome:

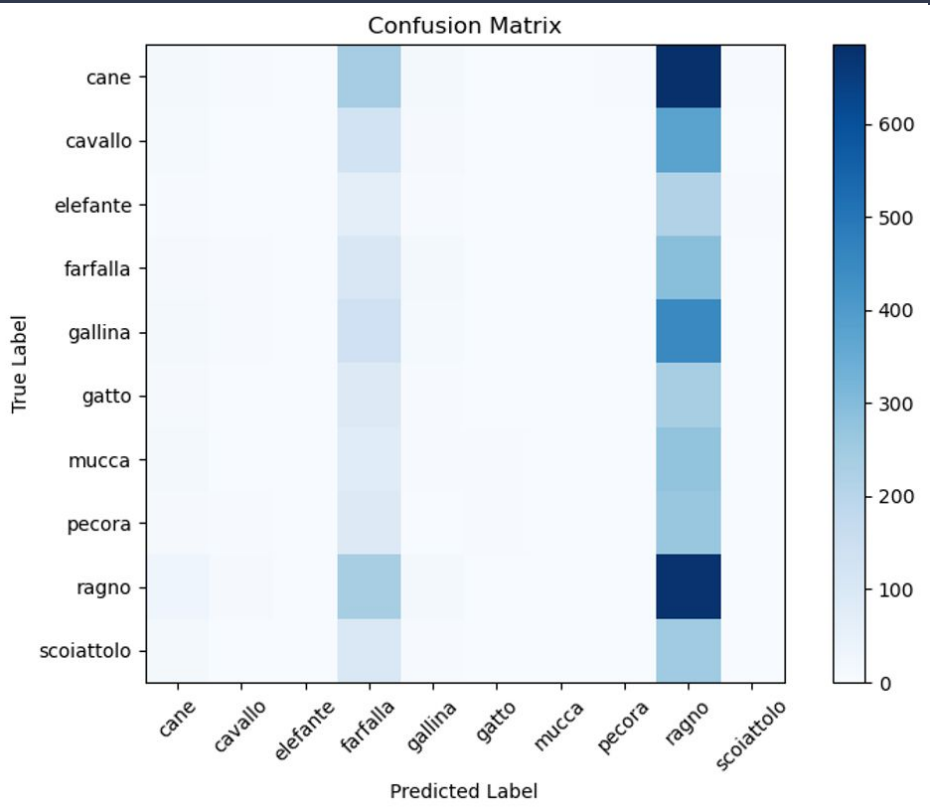
- Fine-tuning enhanced the model’s performance, leveraging both pre-trained features and task-specific updates.

Model Deployment

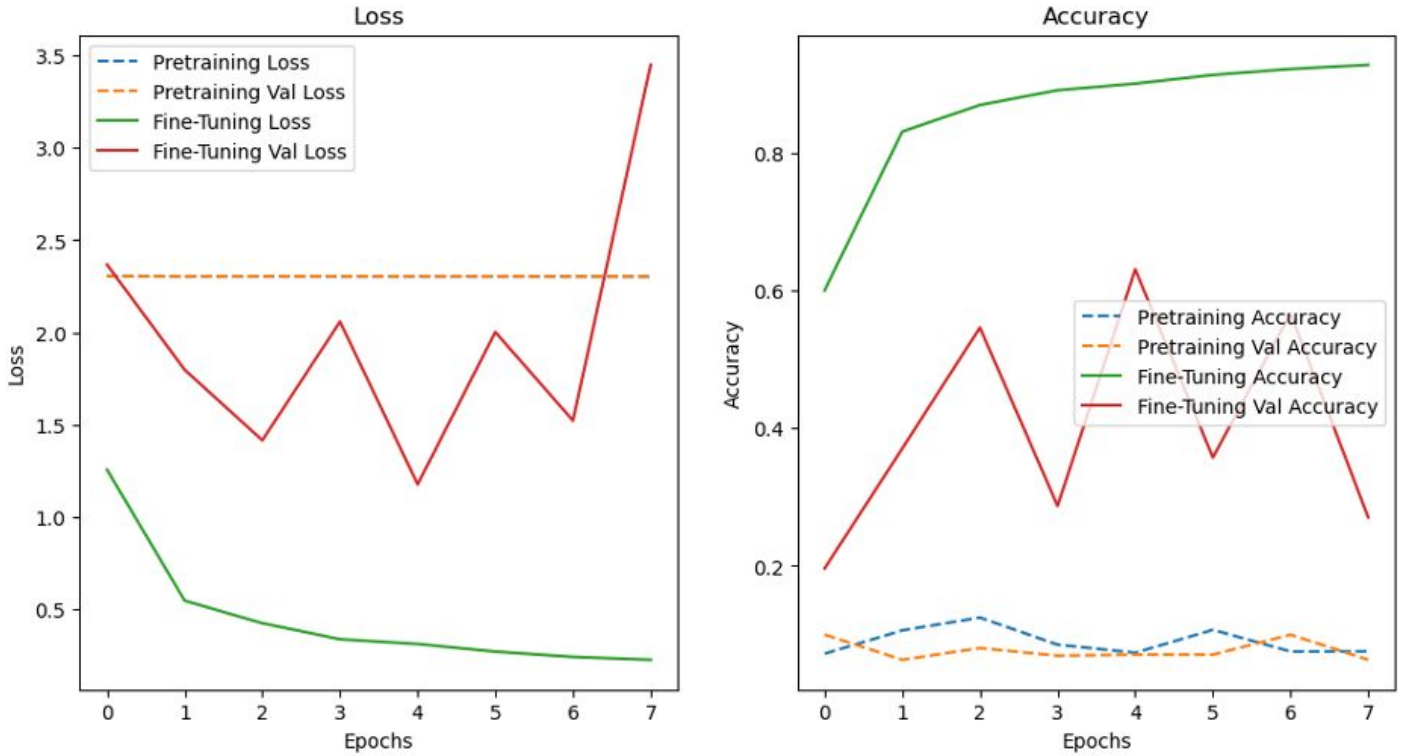
```
164/164 ━━━━━━━━━━━━━━━━━━━━ 12s 71ms/step - accuracy: 0.2661 - loss: 3.4453
Validation Loss: 3.4260590076446533
Validation Accuracy: 0.2696865499019623
164/164 ━━━━━━━━━━━━━━━━━━━━ 13s 74ms/step
Len true_classes: 5232
Len predicted_classes: 5232
Classification Report:
              precision    recall  f1-score   support

        cane       0.12      0.01      0.03       972
     cavallo       0.05      0.00      0.00       524
    elefante       0.00      0.00      0.00       289
    farfalla       0.08      0.25      0.12       422
     gallina       0.11      0.01      0.03       619
       gatto       0.00      0.00      0.00       333
       mucca       0.00      0.00      0.00       373
      pecora       0.00      0.00      0.00       364
       ragno       0.18      0.70      0.29       964
  scoiattolo       0.00      0.00      0.00       372

    accuracy                           0.15      5232
   macro avg       0.05      0.10      0.05      5232
weighted avg       0.08      0.15      0.07      5232
```

```
Confusion Matrix:
[[ 14   3   2 239  16   2   2   5 685   4]
 [  9   1   2 126   8   2   0   0 375   1]
 [  4   0   0  65   4   0   0   0 213   3]
 [  7   3   0 104  16   0   0   1 291   0]
 [ 15   3   1 139   9   1   2   1 447   1]
 [  8   2   0  86   4   0   0   0 233   0]
 [ 11   1   0  78   5   3   0   1 273   1]
 [  7   3   0  87   1   3   0   0 263   0]
 [ 27   6   2 235  15   1   1   1 675   1]
 [ 14   0   0 101   4   2   0   0 251   0]]
```

Model Deployment



Model Deployment



Model Deployment