Project | Deep Learning: Image Classification with CNN

Collaborators:

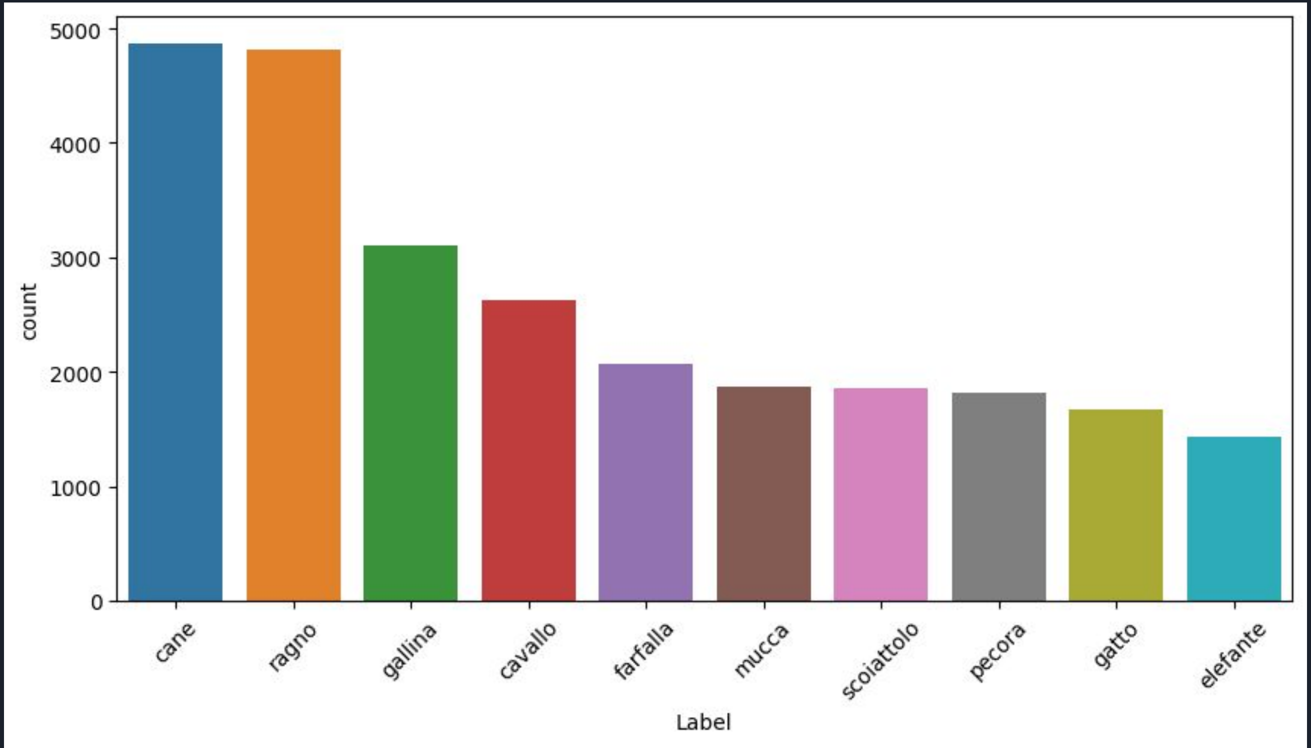
Ginosca Alejandro
Natanael Santiago

Project Overview

- Building a Convolutional Neural Network (CNN)
- Classify images

- Animals-10 Dataset
- ~ 28K images
- Different sizes
- 10 classes:

Dataset preparation



Dataset preparation

- Data exploration
  - Smallest dimensions (60, 57),
  - Largest dimensions (6720, 4480)
  - Approximate average dimension (326, 326)
- Preprocessing
  - Resizing (128 x 128) and normalizing
  - Training set: 80% (20,947)
  - Validation Set: 20% (5,232)

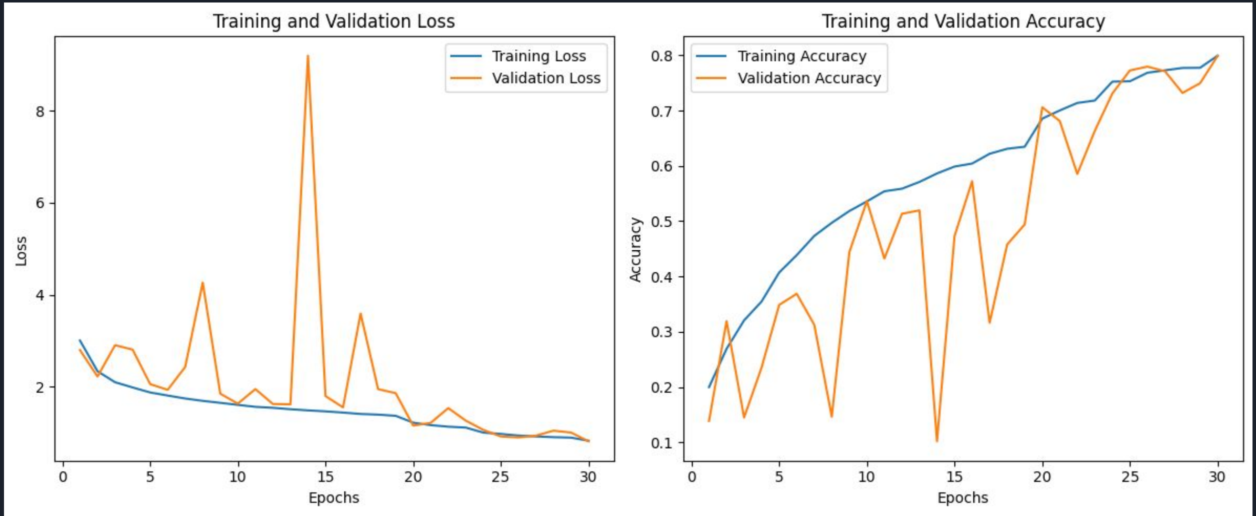
Images after pre-processing

CNN

- Architecture:
  - Convolutional and MaxPooling layers were stacked to extract features from the input images.
  - Fully connected dense layers were used for classification.
  - A final softmax layer outputs predictions for the 10 classes.
- Key Design Choices:
  - Simplicity in design for computational efficiency.
  - Moderate depth to avoid overfitting.

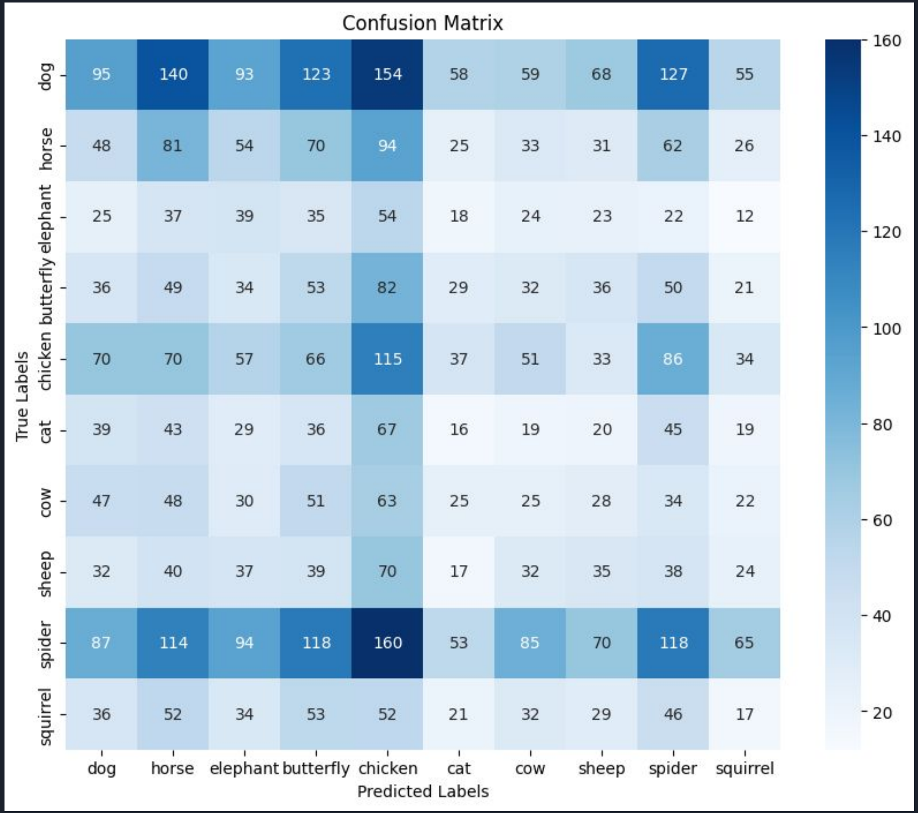
Learning rate	Batch Size	Epochs	Callbacks
0.001	32	30	EarlyStopping ReduceLROnPlateau

CNN



Metric	Custom Model
Validation Loss	1.0980
Validation Accuracy	72.32%
Validation Precision	78.12%
Validation Recall	67.62%

CNN

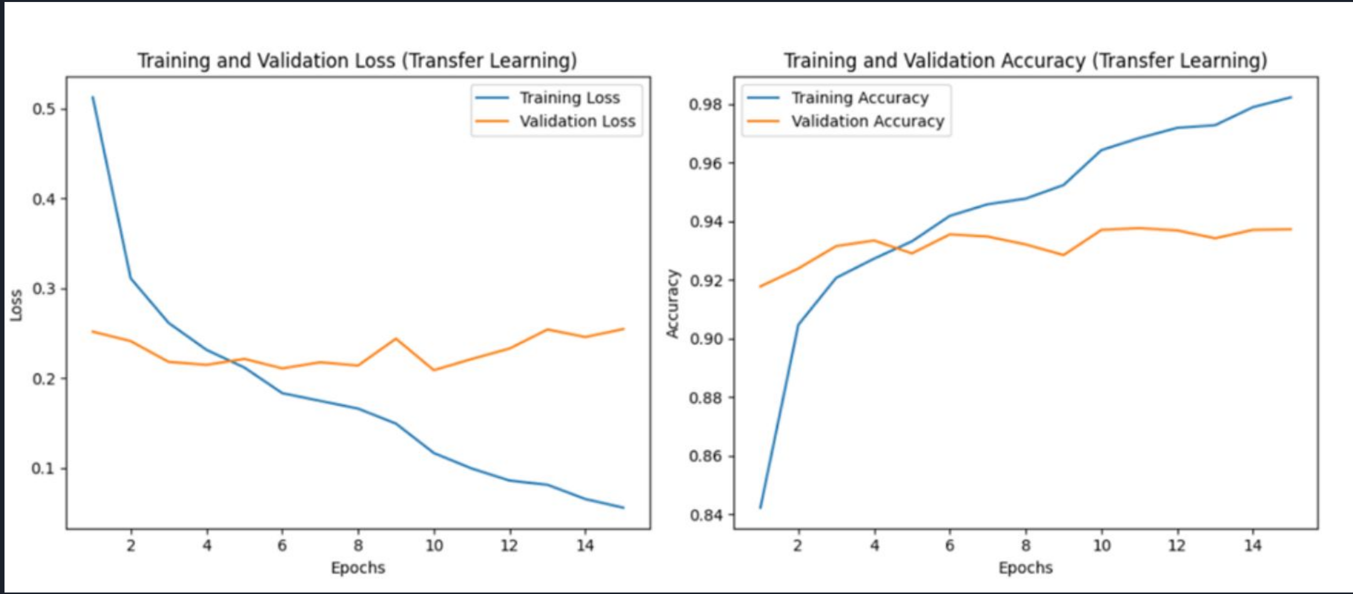

Confusion Matrix
True Labels
Predicted Labels
dog horse elephant butterfly chicken cat cow sheep spider squirrel
dog 95 140 93 123 154 58 59 68 127 55
horse 48 81 54 70 94 25 33 31 62 26
elephant 25 37 39 35 54 18 24 23 22 12
butterfly 36 49 34 53 82 29 32 36 50 21
chicken 70 70 57 66 115 37 51 33 86 34
cat 39 43 29 36 67 16 19 20 45 19
cow 47 48 30 51 63 25 25 28 34 22
sheep 32 40 37 39 70 17 32 35 38 24
spider 87 114 94 118 160 53 85 70 118 65
squirrel 36 52 34 53 52 21 32 29 46 17
160
140
120
100
80
60
40
20

Transfer Learning

- MobileNetV2 pretrained on ImageNet.

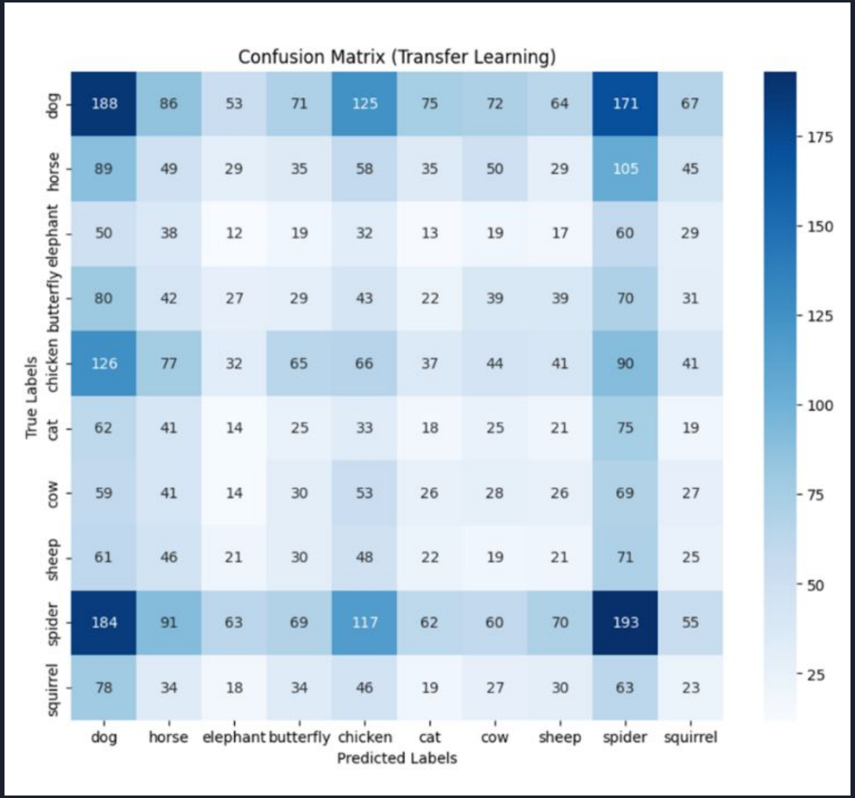
```
# Add custom classification layers
x = base_model.output
x = GlobalAveragePooling2D()(x)  # Global Average Pooling layer
x = Dense(256, activation='relu')(x)  # Fully connected layer
x = Dropout(0.5)(x)  # Dropout for regularization
x = Dense(128, activation='relu')(x)  # Fully connected layer
x = Dropout(0.3)(x)  # Dropout for regularization
output = Dense(10, activation='softmax')(x)  # Output layer for 10 classes
```

Transfer Learning



Validation Loss	0.2086
Validation Accuracy	93.71%
Validation Precision	95.29%
Validation Recall	92.78%

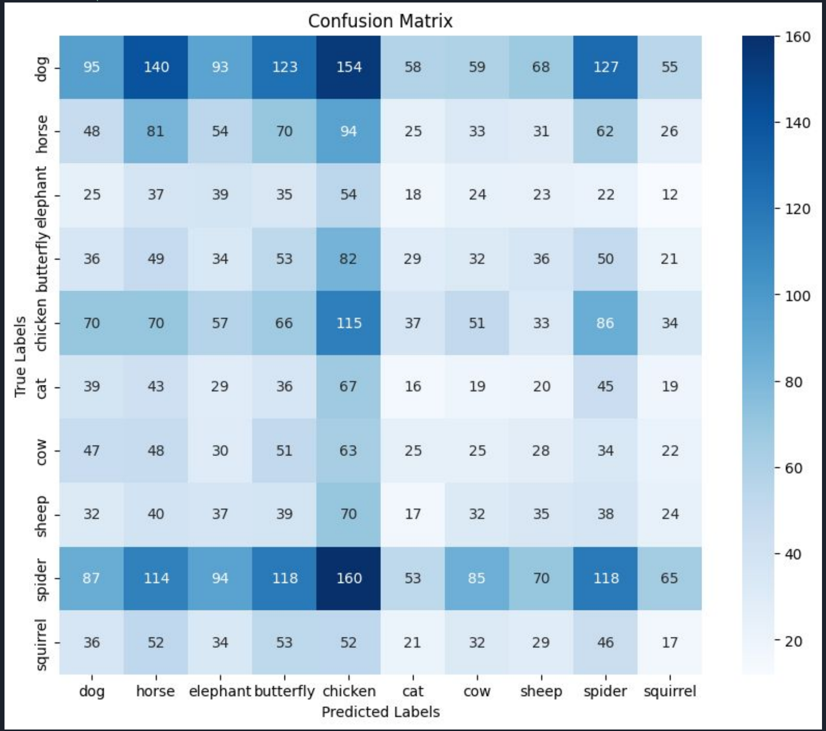
Transfer Learning

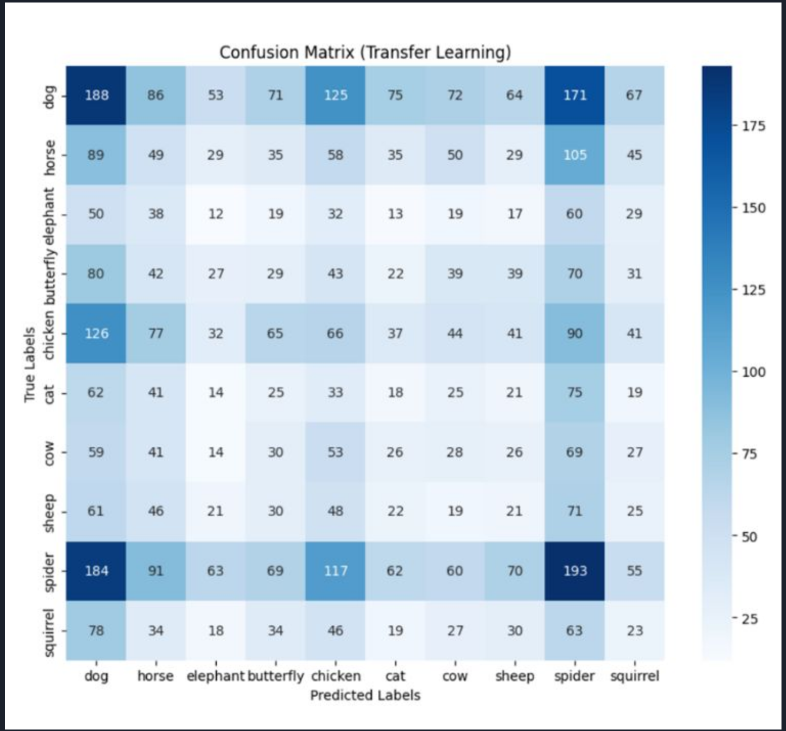


Comparison

Metric	Custom Model	Transfer Learning Model
Validation Loss	1.0980	0.2086
Validation Accuracy	72.32%	93.71%
Validation Precision	78.12%	95.29%
Validation Recall	67.62%	92.78%

Comparison (CNN vs Transfer Learning)





Insights and Learnings

- Impact of Transfer Learning:
    - Leveraging pretrained models significantly improved performance compared to building a model from scratch.
- Importance of Preprocessing:
    - Data augmentation and normalization played a crucial role in achieving generalization.
- Challenges with Specific Classes:
    - Categories like "dog" and "butterfly" require more refined feature extraction due to visual similarities.
- Role of Callbacks:
    - EarlyStopping and ReduceLROnPlateau contributed to efficient training and prevented overfitting.

Next Steps

1. Fine-Tuning:
    - Unfreeze additional layers in MobileNetV2 for task-specific learning.
2. Dataset Enhancements:
    - Collect more diverse examples for underperforming classes.
3. Exploring Alternate Architectures:
    - Experiment with other lightweight architectures like EfficientNet or ResNet.
4. Deploying the Model:
    - Integrate the model into an application for real-time image classification.

THANK YOU!

IRON HACK