Deep Learning for Animal Classification: A Comparative Study of VGG16 and ResNet50 Architectures

CECS 456 - Deep Learning Project Report

Melody Gatan & Matthew Nguyen
California State University, Long Beach
December 5, 2025

GitHub Repository: https://github.com/mel418/CECS456_project

1. Introduction

Image classification remains a fundamental challenge in computer vision with applications spanning autonomous vehicles, medical diagnosis, and wildlife monitoring. This project investigates two prominent convolutional neural network (CNN) architectures—VGG16 and ResNet50—on the Animals10 dataset, a 10-class image classification task.

VGG16 (Simonyan & Zisserman, 2014) represents a classical approach using uniform 3×3 convolutions and deep sequential layers. ResNet50 (He et al., 2016) revolutionized deep learning by introducing residual connections that enable training of significantly deeper networks while avoiding degradation problems. By training both architectures from scratch on identical data splits, we provide direct performance comparisons to guide model selection for animal classification tasks.

2. Dataset and Related Work

2.1 Animals10 Dataset

The Animals10 dataset contains 26,179 RGB images across 10 animal categories: dog (cane), horse (cavallo), elephant (elefante), butterfly (farfalla), chicken (gallina), cat (gatto), cow (mucca), sheep (pecora), spider (ragno), and squirrel (scoiattolo). The dataset presents challenges including variable image quality, diverse backgrounds, different poses/orientations, and inter-class similarities.

Data was split using random sampling with seed 42: 70% training (18,325 images), 15% validation (3,926 images), and 15% test (3,928 images). All images were resized to 128×128 pixels to balance computational efficiency with feature preservation.

2.2 Related Work

VGG's uniform architecture demonstrated that network depth is critical for performance, achieving top ILSVRC-2014 results. ResNet introduced skip connections to address vanishing gradient problems, winning ILSVRC-2015 and becoming foundational for modern architectures. While most animal classification work uses transfer learning from ImageNet, our study trains both models from scratch to evaluate their fundamental learning capabilities.

3. Methodology

3.1 VGG16 Architecture (Melody's Implementation)

Our VGG16 implementation contains 13 convolutional layers organized into 5 blocks with channel progression [64, 128, 256, 512, 512]. Each layer uses 3×3 filters (stride 1, padding 1) followed by batch normalization and ReLU activation. Max pooling (2×2, stride 2) follows each block. The classifier contains three fully connected layers (512×4×4 → 4096 → 4096 → 10) with 50% dropout. The model has 65,103,946 trainable parameters, with the majority in fully connected layers. Weights were initialized using Kaiming initialization.

3.2 ResNet50 Architecture (Matthew’s Implementation)

ResNet50 comprises 50 layers utilizing bottleneck blocks with skip connections. The architecture consists of three stages: initial processing (7×7 convolution, batch norm, max pooling), feature extraction (four residual layers with [3, 4, 6, 3] bottleneck blocks using channel depths [64, 128, 256, 512]), and classification (global average pooling, fully connected layer). Each bottleneck block uses 1×1→3×3→1×1 convolutions with a 4× expansion factor. Skip connections enable identity mapping, facilitating gradient flow. The model contains 23,528,522 parameters—2.8× fewer than VGG16.

4. Experimental Setup

4.1 Hardware and Software

Hardware: NVIDIA A100-SXM4-80GB GPU via Google Colab

Software: Python 3.12, PyTorch 2.9.0+cu126

Training Configuration:

VGG16: 15 epochs, batch size 64, learning rate 0.01, SGD optimizer (momentum 0.9, weight decay 1e-4), CosineAnnealingLR scheduler, dropout 0.5

ResNet50: 15 epochs, batch size 64, learning rate 0.01, SGD optimizer (momentum 0.9, weight decay 1e-4), CosineAnnealingLR scheduler, image size 128×128

Data augmentation included random horizontal flips, 15° rotations, and color jitter. All images were normalized using ImageNet statistics. Data loaders used 4 parallel workers with pinned memory for efficient GPU transfer.

5. Measurement

Model performance was evaluated using multiple metrics to provide comprehensive assessment:

Accuracy: Overall classification accuracy on the test set, calculated as the percentage of correctly classified samples.

Precision, Recall, and F1-Score: Per-class and macro-averaged metrics to assess model performance across imbalanced classes. Precision measures the proportion of true positives among predicted positives, recall measures the proportion of actual positives correctly identified, and F1-score provides their harmonic mean.

Confusion Matrix: Visual representation of classification performance showing true vs. predicted labels, useful for identifying systematically confused class pairs.

Training Time: Total wall-clock time required for 15 epochs of training, measured to assess computational efficiency.

Parameter Count: Total number of trainable parameters, indicating model complexity and memory requirements.

Overfitting Analysis: Gap between training and validation accuracy to assess generalization capability.

5.1 Evaluation Process

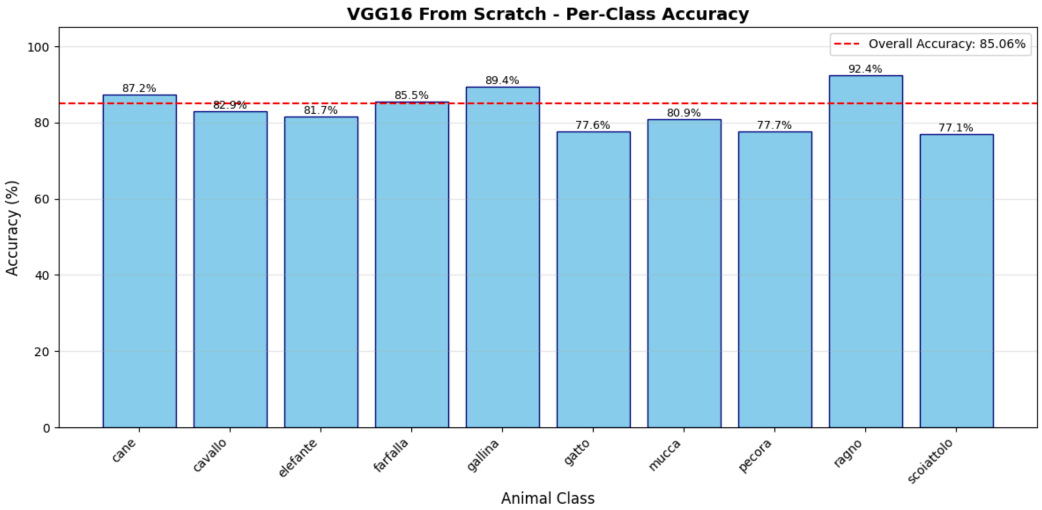
During training, we monitored training loss and accuracy after each epoch and computed validation metrics to track generalization. For final testing, we evaluated both models on the identical held-out test set using the same random seed for data splitting, ensuring fair comparison through consistent preprocessing and evaluation protocols.

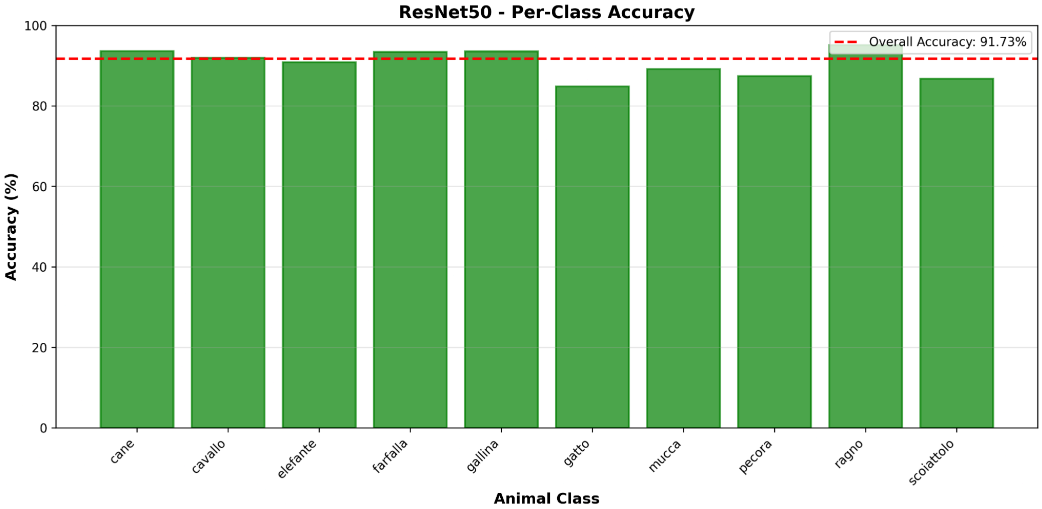
6. Results Analysis, Intuitions, and Comparison

6.1 Overall Performance

Metric	VGG16	ResNet50
Test Accuracy	85.06%	91.73%
Macro Precision	0.837	0.915
Macro Recall	0.832	0.906
Macro F1-Score	0.834	0.911
Parameters	65.1M	23.5M
Training Time	59.7 min	64.8 min

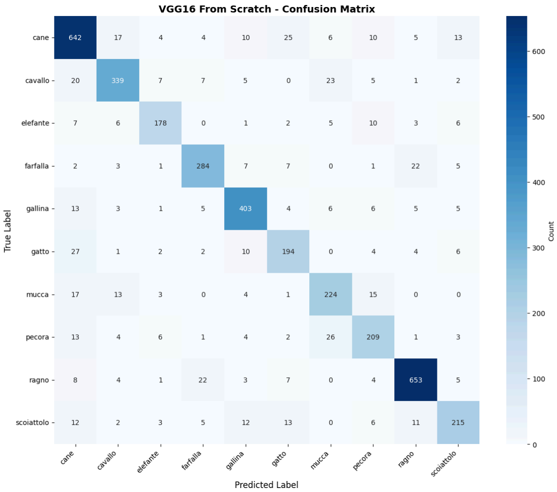
6.2 Per-Class Analysis

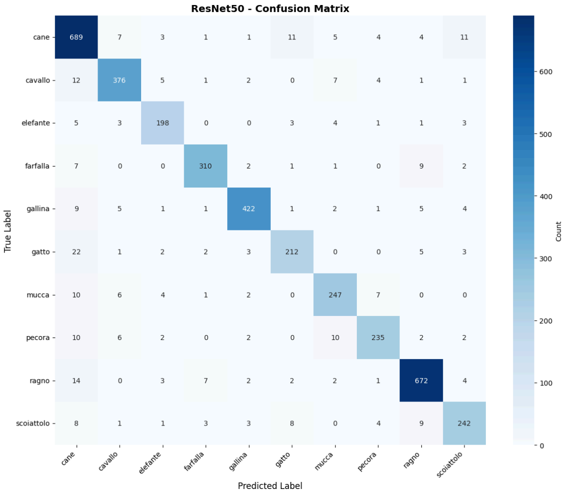




ResNet50 outperformed VGG16 across all 10 animal classes. The largest improvements were observed for challenging classes: cat (84.8% vs 77.6%), squirrel (86.7% vs 77.1%), and sheep (87.4% vs 77.7%). Both models performed best on spider classification (VGG16: 92.4%, ResNet50: 95.0%), likely due to spiders' distinctive morphological features. The most challenging class for both models was cat, where inter-class similarity with dogs contributed to confusion.

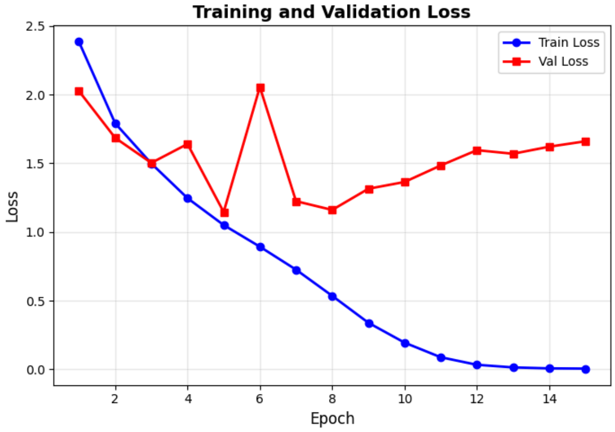
6.3 Confusion Patterns

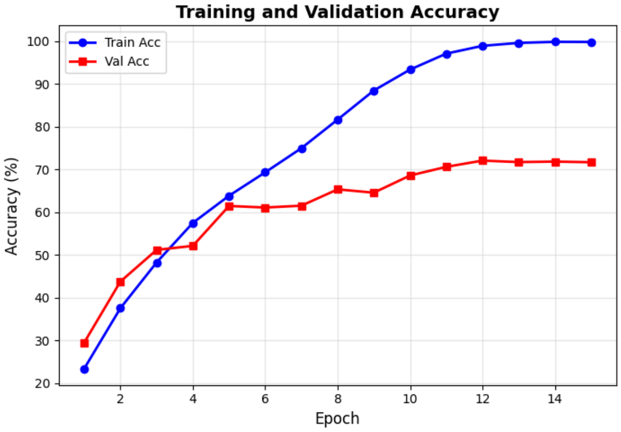


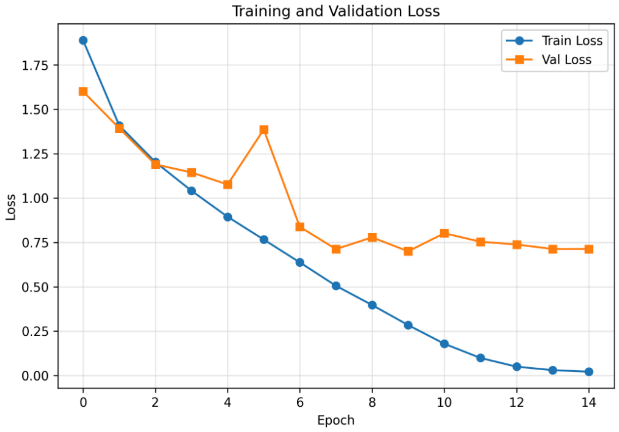


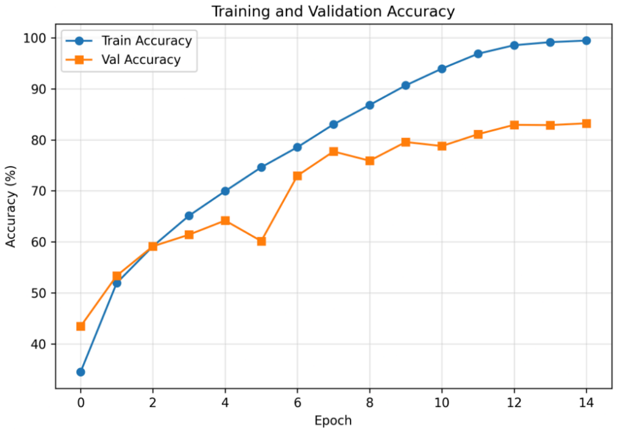
Analysis of confusion matrices revealed similar misclassification patterns. Both models most frequently confused cats with dogs (VGG16: 25 times, ResNet50: 22 times) and horses with dogs (VGG16: 20 times, ResNet50: 12 times). These confusions are understandable given visual similarities in fur texture, body structure, and common poses. ResNet50's reduced confusion rates suggest better feature discrimination.

6.4 Training Dynamics and Overfitting









Both models exhibited significant overfitting, with train-validation accuracy gaps exceeding 27% (VGG16: 28.1%, ResNet50: 28.0%). Training accuracy reached 99%+ while validation accuracy plateaued around 70%. This pattern indicates that both architectures have sufficient capacity to memorize the training set but struggle to generalize. The similar overfitting levels suggest that architectural differences had minimal impact on generalization when training from scratch on this dataset size.

6.5 Computational Efficiency

VGG16 completed training slightly faster (59.7 vs 64.8 minutes) despite having 2.8× more parameters. This is attributed to VGG16's simpler forward pass with uniform convolutions, while ResNet50's skip connections and bottleneck blocks require additional computational overhead. However, ResNet50's superior accuracy with fewer parameters makes it more memory-efficient and better suited for deployment scenarios.

6.6 Intuitions

ResNet50's superior performance can be attributed to several factors. First, residual connections enable more effective gradient flow, allowing the network to learn more discriminative features. Second, the bottleneck design (1×1→3×3→1×1) provides computational efficiency while maintaining representational power. Third, deeper architecture (50 vs 16 layers) captures more hierarchical features crucial for distinguishing similar animals.

VGG16's limitations stem from its depth constraint without skip connections. Gradients must backpropagate through 16 sequential layers, potentially leading to vanishing gradients. Additionally, large fully connected layers (4096→4096→10) contribute most parameters but may overfit easily, especially when training from scratch on limited data.

7. Conclusion

This study demonstrates ResNet50's clear superiority over VGG16 for animal classification. ResNet50 achieved 91.73% test accuracy with 64% fewer parameters, validating the effectiveness of residual learning for training deep networks from scratch. Both models suffered substantial overfitting (>27% train-val gap), indicating that 18,325 training images are insufficient for learning generalizable features from random initialization.

For practitioners, ResNet50 offers superior accuracy-efficiency trade-offs. The reduced parameter count (23.5M) makes it more suitable for resource-constrained deployment while delivering better performance. However, the persistent overfitting suggests that production-quality models require either transfer learning or substantially larger datasets. Future work should investigate transfer learning from ImageNet, stronger regularization techniques, and ensemble methods.

8. Individual Contributions

Melody Gatan (VGG16): Implemented VGG16 architecture with batch normalization, designed and executed training pipeline, generated all VGG16 visualizations, performed error analysis, wrote VGG16 methodology and results sections, set up Google Colab environment, contributed to comparative analysis.

Matthew Nguyen (ResNet50): Designed and implemented ResNet50 with bottleneck blocks, built custom BottleneckBlock class with skip connections, configured training pipeline, trained model achieving 91.73% test accuracy, generated ResNet50 visualizations, analyzed training dynamics, wrote ResNet50 methodology section, contributed to comparative analysis.

References

[1] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770-778).

[3] Animals10 Dataset. Kaggle. https://www.kaggle.com/datasets/alessiocorrado99/animals10

[4] Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32 (pp. 8024-8035).