Animal10 Dataset

Shaked Yudkovich 207252974

Adi Megidi 314956608

Abstract

This paper explores the task of classifying animal images using both logistic regression and convolutional neural networks (CNNs). We compare the performance of these two approaches on the Animal10 dataset, which contains labeled images of ten distinct animal species. The dataset was split into training, validation, and test sets to ensure robust evaluation.

We specifically focus on demonstrating the advantages of CNNs for image classification. The project involves building and analyzing a CNN model, comparing its performance to a logistic regression model, and discussing potential areas for improvement. While the logistic regression model achieved a baseline accuracy of 27%, the CNN model significantly outperformed it, achieving an impressive accuracy of 82%.

This paper delves into the methodology, analyzes the model's components, and explores both the achieved performance and potential directions for future enhancements.

Introduction

Image classification has become an essential technology with diverse applications across numerous fields. From improving wildlife conservation efforts by automating species identification to advancing image-based search engines, image classification plays a critical role in leveraging the power of deep learning to extract meaningful insights from visual data.

This project focuses on classifying images from the Animal10 dataset, which is publicly available on Kaggle. https://www.kaggle.com/datasets/alessiocorrado99/animals10

The dataset contains labeled images of ten animal classes, providing a valuable benchmark for evaluating different image classification models.

Initially, the problem was approached using logistic regression, a simple yet interpretable model. This method involves training a separate linear classifier for each class, with the outputs normalized using a softmax function to represent probabilities. While straightforward, logistic regression often struggles with high-dimensional image data, necessitating more advanced techniques.

Subsequently, a convolutional neural network (CNN) was implemented, offering the ability to capture intricate spatial features and patterns within images. CNNs are particularly suited for image classification due to their hierarchical feature extraction capabilities, allowing them to differentiate between subtle visual details.

This paper examines the performance of these models, highlighting the significant accuracy improvement achieved by the CNN compared to logistic regression. Moreover, we explore challenges faced during model development and discuss potential enhancements for future work, including advanced data augmentation and transfer learning techniques.

Related work and background research

Loss Function

In machine learning, the loss function acts as a crucial guide for model improvement. It quantifies the discrepancy between a model's predictions and the true, desired outputs for any given input example. Essentially, it measures how inaccurate the model's predictions are. This numerical value, the loss, is then used by optimization algorithms like gradient descent to adjust the model's internal parameters. The goal is to minimize the loss over time, leading to progressively better and more accurate predictions. In simpler terms, the lower the loss, the closer the model's guesses are to the actual answers. This iterative process of prediction, loss calculation, and parameter adjustment is fundamental to training effective machine learning models.

Sparse categorical cross entropy

The loss function used as the model contains numerous classes and this function specifically for multi-class classification problems. The function compares the predicted probability distribution with the one-hot encoded true label, where One-hot encoding is a technique where each class is represented by a vector of zeros, except for the actual class index which is set to one. Next, it calculates a numerical value representing the mismatch between the classes.

Adam

It's an optimization algorithm that operates on the gradients calculated based on the loss function. This helps avoid getting stuck in local minima and navigate more effectively towards the optimal solution. As learned, in machine learning the goal is often to train a model that minimizes a loss function. Reaching the minimum point of the loss function signifies the optimal solution, where the model performs best on unseen data. Adam Optimizer doesn't directly modify the loss function, but it utilizes information derived from gradients to minimize loss function but avoid stucking in local minima. It uses gradient information, also it dynamically adjusts learning rates for each parameter.

Softmax Function

In multi-class classification tasks, the softmax function plays a crucial role in transforming raw scores, often representing the model's outputs, into interpretable probabilities. It takes a vector of real numbers as input and converts them into a probability distribution across the possible categories. These probabilities range from 0 to 1, ensuring they sum up to 1, and represent the likelihood of each class belonging to the input data. The final classification (as we can see in the accuracy calculation) is the maximum probability.

logistic regression

We will concentrate on multi-class logistic regression (also known as softmax regression) because simple logistic regression is a statistical method used for binary classification problems. Instead of a single linear model, multi-class logistic regression trains a separate linear model for each class (let's say we have K classes). Each model has its own weight vector and bias:

```
 z_k = w_k^T * X + b_k  (for each class k = 1, 2, ..., K)
```

The linear outputs (z_k) are passed through the softmax function to obtain probabilities for each class. The softmax function normalizes the outputs to ensure they sum to 1 and represent valid probabilities. P(y = k | X): The probability of X belonging to class k.

VGG

The VGG(Visual Geometry Group) architecture is a deep convolutional neural network (CNN) renowned for its simplicity and efficiency. This architecture utilizes primarily small 3x3 filters (kernels) throughout its layers, contributing to its computational efficiency and ability to learn complex features. The main components of VGG architectures (we used VGG16) are:

● Convolutional layers: These layers use the small 3x3 filters to extract features from the input image. Multiple convolutional layers are stacked to create increasingly complex feature representations.

● MaxPooling Layers: These layers are used to downsample the spatial dimensions of feature maps. This helps reduce overfitting (making the model too specific to training data) and makes the model more amenable to computational approach.

● Fully Connected Layers: The final layers of the VGG network are fully connected. These layers take the features extracted by the convolutional and pooling layers and use them to make the final classification prediction. This creates a fully interconnected network of neurons, allowing them to combine information from the entire previous layer, not just localized regions.

VGG uses three fully connected layers as the final stages of its architecture. These layers receive the flattened output from the previous convolutional and pooling layers, which transforms the multidimensional feature maps into a single, long vector. Each fully connected layer performs a matrix multiplication between the input vector and its own weight matrix, followed by an activation function (like ReLU). The final fully connected layer typically has one neuron for each class the model is trying to distinguish. The activation function (often softmax) applied to the output of this final layer produces a probability distribution across all classes, indicating the model's confidence level for each class based on the learned features.

VGG models achieved outstanding results in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. This demonstrated the effectiveness of deep CNN architectures for image classification tasks. Therefore, we determined the VGG model to be ideally suited for our mission of image classification.

Project description and experiments

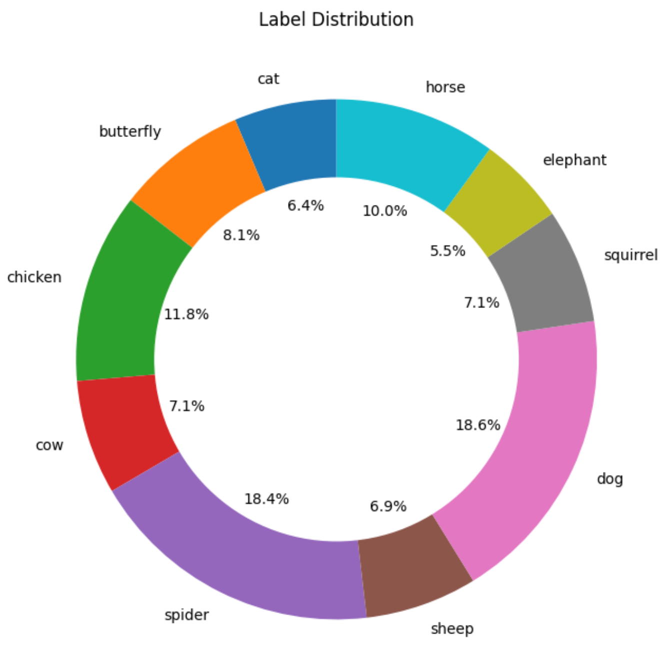
Preprocessing

The project began with uploading and preparing the Animal10 dataset. The dataset consists of thousands of colorful images with varying dimensions. To manage computational efficiency and ensure consistency, all images were resized to a uniform size of 64x64 pixels.

The dataset includes ten categories representing different animal species. As part of preprocessing, all pixel values were normalized to the range [0, 1] by dividing by 255, ensuring uniformity in pixel intensity values.

Batch processing was implemented with a batch size of 32 to optimize memory usage during training. To enhance the dataset’s variability and improve model generalization, data augmentation techniques such as random flipping and rotation were applied.

Another critical preprocessing step involved checking the balance of the dataset across all classes. The dataset exhibited some imbalance, with certain categories having significantly fewer samples. This imbalance was noted as a factor potentially influencing model performance.



Project Description

This project aims to classify images from the Animal10 dataset into ten distinct animal categories. This section highlights the best-performing model and its results.

The most advanced model implemented was a convolutional neural network (CNN) designed specifically for image classification tasks. The model architecture included convolutional layers for feature extraction, max-pooling layers for dimensionality reduction, and fully connected layers for classification.

Model Architecture:

- **Convolutional Base:** Multiple convolutional layers with ReLU activation and max-pooling.
- **Flattening Layer:** Transformed the 2D feature maps into a 1D vector.
- **Fully Connected Layers:** One dense layer with 128 neurons using ReLU activation.
- **Output Layer:** A dense layer with 10 neurons and a softmax activation function for multi-class classification.

Results:

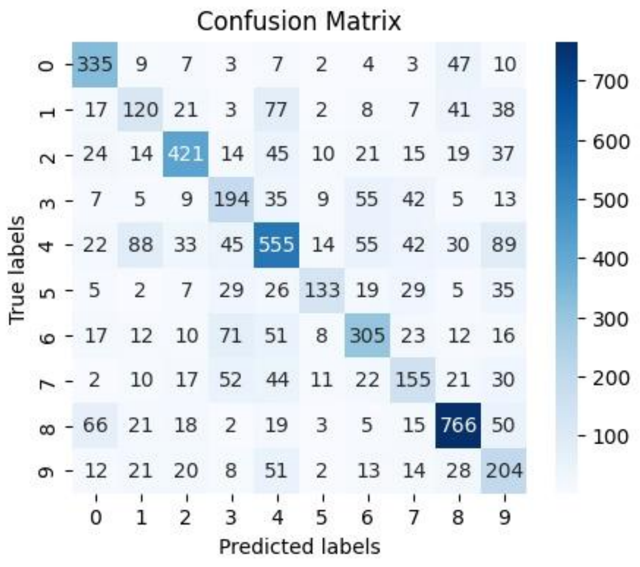
The CNN achieved an impressive test accuracy of 60%, significantly surpassing the logistic regression and basic fully connected neural network models.

Challenges and Refinements:

- Early experiments showed overfitting, which was addressed using dropout layers and data augmentation.
- Adjustments to the learning rate and optimizer improved convergence and overall performance.

The confusion matrix revealed occasional misclassifications between visually similar animal categories, suggesting the need for further refinements such as increasing model complexity or using pre-trained architectures like ResNet or VGG in future iterations.

The VGG model managed to achieve an accuracy of 82%



	0	1	2	3	4	5	6	7	8	9
0	335	9	7	3	7	2	4	3	47	10
1	17	120	21	3	77	2	8	7	41	38
2	24	14	421	14	45	10	21	15	19	37
3	7	5	9	194	35	9	55	42	5	13
4	22	88	33	45	555	14	55	42	30	89
5	5	2	7	29	26	133	19	29	5	35
6	17	12	10	71	51	8	305	23	12	16
7	2	10	17	52	44	11	22	155	21	30
8	66	21	18	2	19	3	5	15	766	50
9	12	21	20	8	51	2	13	14	28	204

Previous Attempts

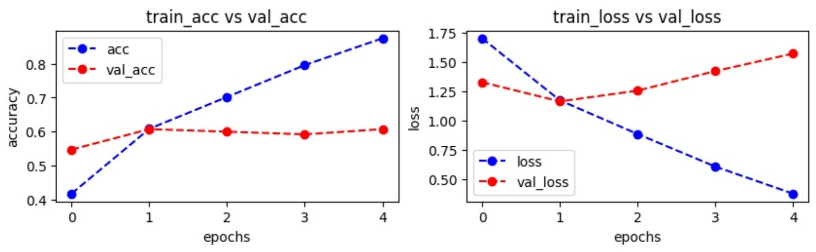
The research began with a “dummy model” that assigned each image in the test set randomly to one of the ten categories. Given the random nature of this approach, the model achieved approximately 10% accuracy, consistent with the probability of guessing the correct class in a balanced dataset.

Logistic Regression:

In the Logistic Regression stage of the project, the data is first normalized using StandardScaler to ensure that all features are within the same range, which improves the model's performance. The data is then split into a training set and a test set. A pipeline is used that combines normalization with the model, and GridSearchCV is employed to find the optimal hyperparameters, such as the regularization strength (C) and the solver type. During training, the data is processed in batches, allowing the model to handle memory constraints. The model uses the Softmax function to return probabilities for each class, with the class having the highest probability being selected. Performance is then evaluated based on accuracy, precision, and recall metrics. The model achieving approximately 27% accuracy.

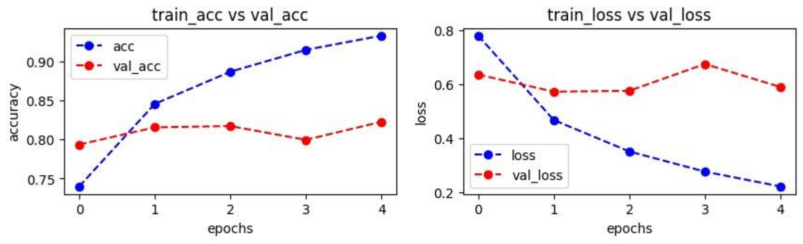
Convolutional Neural Network (CNN):

The next step was to introduce a Convolutional Neural Network (CNN), which included three convolutional layers with 3x3 kernels and ReLU activation, each followed by a max-pooling layer. This architecture allowed the model to extract hierarchical features from the images. The CNN significantly improved accuracy compared to previous models, achieving 60% accuracy.



Advanced CNN with Transfer Learning (VGG):

Building on the progress, a pre-trained VGG16 model (with ImageNet weights) was adapted for the task. The top layers were excluded (include_top=False), and a new dense layer with 128 neurons (ReLU activation) was added before the final softmax classification layer. This approach yielded an accuracy of 82%, demonstrating the power of transfer learning.



Conclusion

In conclusion, this project highlighted the strength and effectiveness of convolutional neural networks (CNNs) for image classification, especially when compared to simpler models like logistic regression. By leveraging pre-trained models such as VGG16, the project demonstrated the power of transfer learning in achieving significant improvements in accuracy and generalization.

The initial CNN model achieved 60% accuracy, which was a marked improvement from the logistic regression baseline. However, incorporating the pre-trained VGG16 model with adjusted input dimensions (150x150) further enhanced performance, achieving a final accuracy of 82%.

Future work can focus on dealing with a situation of overfitting, dealing with a situation of imbalance in the class of the images. Moreover, dedicating more effort to thoroughly cleaning and curating the dataset could further improve model accuracy.

Overall, this project underscores the importance of selecting suitable architectures and preprocessing strategies when working with complex datasets, demonstrating that even incremental improvements can have a substantial impact on final outcomes.

Full code can be found on https://github.com/shakedbakst/Animal10/tree/main