Multi-Label Animal Classification Using the PASCAL VOC Dataset

Loyd Flores
Department of Computer Science
CUNY Queens College
New York, NY 11355

Raphael Attiaala
Department of Computer Science
CUNY Queens College
New York, NY 11355

Abstract

In this paper, we will discuss & analyze our attempt of using CNN architecture for multi-label classification of various animal types on the PASCAL VOC 2012 competition dataset. Building on top of the preexisting neural net found in PyTorch.

1 Literature Review

1.1 Comparative Analysis of Traditional Machine Learning and Deep Learning Algorithms for Image Classification

The referenced article provided key inspiration for our abstract by demonstrating the effectiveness of Convolutional Neural Network(CNN) architecture in addressing multi-label classification challenges within diverse datasets.At present, many pattern recognition systems, such as handwritten character recognition system, face recognition system and speech recognition system, have used CNN, and achieved good results.Wang (2021) Highlighting the CNN's architecture which was inspired by humans cerebral cortex providing strong feature extraction unlike other neural networks thanks to the usage of convolution kernels to extract features in our training images. Furthermore the paper highlights the usage of how CNN's utilize parameter sharing to assist in reducing training time Wang (2021)Through the way of parameter sharing, the parameters in the network are greatly reduced and the network training time is reduced. This approach informed our methodology for classifying animal types using our dataset, emphasizing the importance of building on robust, efficient & preexisting foundations while tailoring solutions to specific classification tasks.

1.2 Animal Recognition System Based on Convolutional Neural Network

The paper outlines the overall performance of CNN's for animal classification compared against other known image recognition methods such as Principal Component Analysis, Linear Discriminant Analysis, Local Binary Patterns Histograms & Support Vector Machines.While using an overall less robust dataset size with a similar class size to our training data, the study demonstrates the power of CNN models under severe conditions comparative to the previously mentioned alternative options showing a significant out performance to the alternative options.This further highlight's the effectiveness of CNN's for the classification of animals.

2 Introduction

The PASCAL VOC (Visual Object Classes) dataset is a widely recognized benchmark for tasks such as object detection, segmentation, and classification. It supports research across a diverse range of object categories and is extensively used to evaluate the performance of computer vision models. This dataset is a critical resource for those working in these areas.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Table 1: Overall Accuracry of Animal Identification

	Bear	Wolf	Fox	Deer	Hog
PCA(%)	82	79	78	76	82
LDA(%)	81	77	78	81	83
LBPH(%)	85	87	83	84	82
SVM(%)	87	86	85	83	81
CNN(%)	97	95	95	93	91

The VOC dataset features two primary challenges: VOC2007 and VOC2012. It contains annotations for 20 object categories, including everyday objects like bicycles, cars, and animals, as well as more niche items such as sofas, boats, and dining tables. These annotations include bounding boxes and class labels for detection and classification tasks, as well as segmentation masks for segmentation tasks.

In this study, we focus on the VOC2012 dataset, specifically addressing the classification of six animal categories: bird, cat, cow, dog, horse, and sheep. This targeted approach highlights the utility of the dataset for advancing model performance on specific object classes.

3 Methodology

3.1 Introduction to Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is a type of artificial neural network specifically designed to process and analyze visual data which makes CNN's a popular model in computer vision applications. CNNs are particularly effective in tasks like image classification because they leverage spatial hierarchies in images to extract and learn features from images at different levels of detail, mimicking how humans might recognize objects such as edges, textures, and shapes. There are three levels of spatial heirarchies. Low-level Features are at the initial layers, CNNs detect basic patterns such as edges, lines, color gradients. In Low-level vision the model learns features that do not represent the entire object but are building blocks for more complex structures. Mid-level is deeper into the network as the basic patterns are learned and are combined into a more recognizable structures, such as corners, shapes, or textures for example the model will know some aspects of the shape of an animals ears and nose. In the final layer the CNN learns High-level features which are more abstract and definitive features such as the entire head of objects such as an animal's head. **A CNN comprises the following primary components:**

- **Convolutional Layers**: Extract spatial features using filters (kernels).
- **Pooling Layers**: Feature maps to reduce computational cost and retain essential features
- **Fully Connected Layers**: Map high-level features to the output space, enabling predictions
- **Activation Functions**: Introduce non-linearity, allowing the network to learn complex patterns.

The parameters for the convolutional layers are optimal for

3.1.1 Convolutional Layers: Feature Extraction

The convolutional layers progressively extract hierarchal features. The first layer detects simple patterns like edges, corners, pixel intensities. The second layer builds upon these patterns.

Python Implementation:

```
# self.conv1 = nn.Conv2d(Input Channels, Output Channels, Kernel Size, Padding)
self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1)
self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
```

Convolutional Layer parameters:

- **Input Channels** are set to **3**, representing the Red Green Blue channels of the input image because the dataset consists of colored images.
- **Output Channels** differ in the two layers of the CNN. The first layer is set to **32** feature maps. This number balances the model's capacity to learn diverse features while maintaining computational efficiency. The second output channel is set to **64** allowing the model to capture more complex and abstract features.
- **Kernel Size** size of **3x3** was selected as this is the standard choice in convolutional neural networks, as it effectively captures local patterns such as edges and textures while keeping computational costs low.
- **Padding** of **1** ensures that the spatial dimensions of the output match the input, facilitating consistent feature extractions across multiple layers. This approach avoids the reduction of image dimensions, which could lead to the loss of important spatial information because low level computer vision operations operate on neighboring pixels a padding or frame of 1 to prevent unwanted results.

3.1.2 Pooling Layers: Dimensionality Reduction

Pooling layers reduce the spatial dimensions of feature maps while retaining critical information. Max pooling, a common pooling method, selects the maximum value from each region in the feature map, preserving prominent features.

Python Implementation:

```
# self.pool = nn.MaxPool2d(Kernel Size, Stride)
self.pool = nn.MaxPool2d(2, 2)
```

- **Kernel Size 2x2** kernel reduces the spatial dimensions by half, effectively summarizing features while reducing computational cost.
- **Stride** of **2** ensures non-overlapping pooling, further simplifying the feature maps and maintaining efficiency.

3.1.3 Fully Connected Layers: Classification

Fully connected layers process high-level features and produce the final output by computing weighted sums of all inputs. These layers enable the network to learn complex combinations of features for accurate predictions. The FC layers map the spatial features extracted by convolutional layers to the output classes, ensuring meaningful predictions for multi-label classification tasks.

Python Implementation:

```
self.fc1 = nn.Linear(64 * 56 * 56, 128)
self.fc2 = nn.Linear(128, 6 #6 Animal Classes to predict)
```

Two layers provide a good balance between model complexity while preventing overfitting, allowing the network to learn nonlinear relationships in data.

- **fc1**
  - **Input Size (64 * 56 * 56):** This corresponds to the flattened output from the final convolutional and pooling layers, where 64 is the number of feature maps, and 56 x 56 represents the spatial dimensions.
  - **Output Size (128):** A smaller dimensional space is chosen to reduce the feature representation while maintaining the model's capacity to learn meaningful patterns.
- **fc2**

- **Input Size (128):** Connects the 128 learned features from fc1.
- **Output Size (6):** Matches the number of target classes in the dataset, enabling the model to produce a class score for each category.

3.1.4 Activation Functions: Non-Linearity

Activation functions introduce non-linearity, enabling the model to learn complex patterns. Without them, the network would reduce to a linear model, incapable of solving non-linear problems.

Python Implementation:

```
self.relu = nn.ReLU()
self.sigmoid = nn.Sigmoid()  # For multi-label classification
```

- **ReLU** (Rectified Linear Unit): Used in the hidden layers to prevent vanishing gradients and speed up convergence.
- **Sigmoid**: Applied in the output layer for multi-label classification, normalizing predictions to the [0, 1] range.

3.2 Extending PyTorch's Neural Network

PyTorch enables modular and reusable designs by allowing custom neural network architectures to extend **torch.nn.Module**. This enables Encapsulation. Encapsulation is the principle of bundling data and methods that operate on the data within a class, restricting direct access to some components to enforce modularity and maintain control. A subclass is a derived class that inherits properties and behaviors from a parent class, allowing for reuse and extension of functionality.

3.2.1 Implementation of the Animal Classifier

The Animal Classifier class encapsulates the architecture and overrides the forward method to define data flow specific to the task of classifying the animals from the PASCAL VOC dataset.

3.3 Hyperparameters

- **Number of Epochs:** The model trains for a fixed number of iterations to balance underfitting and overfitting.
- **Optimizer:** We used the Adam optimizer, which adapts the learning rate and handles sparse gradients efficiently.
- **Loss Function:** Binary Cross Entropy Loss (BCELoss) computes the error for each label independently, making it ideal for multi-label classification.

4 Experimentation

4.1 Description of our dataset

Derived from the PASCAL VOC challenge, originally consists of 11,530 images, annotated with over 25,000 region of interest (ROI) objects and approximately 5,000 segmentations. However, for the purposes of this project, we focused exclusively on classifying Animal objects. This reduced the dataset to 4,241 images, containing annotations relevant to the following six Animal classes:

- Bird
- Cat
- Cow
- Dog
- Horse

- Sheep

The original dataset is divided into four main object categories—Person, Animal, Vehicle, and Indoor—but we excluded non-Animal categories to streamline the scope of our classification task. The decision to narrow the focus to Animal classes allowed us to optimize the model specifically for these six categories, ensuring targeted feature learning.

To further enhance model performance, we combined the original training and validation sets into a unified dataset. This approach maximized the number of samples available for training while maintaining meaningful data distribution for evaluation purposes.

A breakdown of the dataset focusing on the Animal classes is provided in the table below. It highlights the key aspects of the data distribution that may have influenced our model's performance, including potential imbalances or biases in the representation of specific classes.

Table 2: Animal Class Breakdown

Class	Total Images
Bird	765
Cat	1080
Cow	303
Dog	1286
Horse	482
Sheep	325

Some potential concerns include the distribution of images along the 6 classes with two classes being under represented with slightly over 300 images for each. Beyond that we see that our problem set consists of approximately 40% of our entire image set.

4.2 Performance Measurement

4.2.1 Accuracy

Accuracy is a performance metric that measures the proportion of correct predictions made by a model out of the total predictions. It is a straightforward and commonly used metric for evaluating classification models. Accuracy calculates how often the model's predictions match the actual labels. It works by comparing the predicted labels to the true labels and counting the number of correct predictions.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

4.2.2 Precision

Precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive. It indicates the accuracy of positive predictions. Precision evaluates how many of the predicted positives are actually correct. It is useful in scenarios where minimizing false positives is critical.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

4.2.3 Recall

Recall measures the proportion of correctly predicted positive instances out of all actual positive instances. It is also called sensitivity or true positive rate. Recall evaluates how well the model captures all the actual positives. It is crucial in scenarios where minimizing false negatives is important.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

4.2.4 F1-Score

The F1-score is the harmonic mean of precision and recall, providing a balanced metric that considers both false positives and false negatives. F1-score is particularly useful when there is an uneven class distribution or when a balance between precision and recall is needed

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.3 Validation process

4.3.1 Experiment 1: Validation Using the PASCAL VOC Dataset Splits

In the first experiment, we adhered to the standard PASCAL VOC dataset splits. The training dataset consisted of 2,091 images, while the validation dataset comprised 2,108 images. This split allowed us to evaluate the model's performance on unseen data using a traditional holdout validation approach. This model was trained on only 5 epochs and serves as a baseline model.

4.3.2 Experiment 1: Results

```
Validation Accuracy: 80.23%
Precision: 0.36
Recall: 0.23
F1 Score: 0.28
```

4.3.3 Experiment 2: 5-Fold Cross-Validation

To address the limitations of the first experiment, we implemented k-fold cross-validation with 5 folds. Cross-validation provides a more robust evaluation of model performance by splitting the dataset into multiple folds, ensuring that every data point gets a chance to be in both the training and validation sets. **k-Fold Cross-Validation** is a technique where the dataset is divided into k equal subsets (folds). The model is trained on k-1 folds and validated on the remaining fold. This process is repeated k times, with each fold serving as the validation set once. The final accuracy is averaged across all folds to provide a more reliable estimate of model performance. The number of epochs was also doubled.

Implementation

- The images were ivided the 4,241 images (focused on the Animal classes) into 5 equal folds.
- In each iteration, the model was trained on 4 folds and validated on the remaining fold.
- The process was repeated 5 times, with each fold acting as the validation set once.

4.3.4 Experiment 2: Results

Fold	Validation Accuracy (%)	Precision	Recall	F1 Score
1	80.69	0.6373	0.6065	0.6175
2	81.31	0.6521	0.6227	0.6339
3	80.02	0.6219	0.5962	0.6054
4	80.02	0.6222	0.6012	0.6093
5	80.15	0.6254	0.5990	0.6085

Table 3: Validation metrics for each fold in k-fold cross-validation.

4.3.5 Modeling time decrease when RAM is doubled

The 5-fold cross validation with 10 epochs was run twice on a computer that has a **AMD Ryzen 5 with 6 cores** with **16GB** of RAM which ran for *86 minutes*. Before running it for the second time RAM was doubled to **32GB**. Leading the researchers to discover that doubling the RAM led to a a **38.37%** decrease in running time as retraining the model only had **56 minutes** of run time.

5 Discussion and conclusion

The results from the k-fold cross-validation show consistent validation accuracy, precision, recall, and F1 scores across all five folds. This consistency suggests that the model generalizes well across different subsets of the data. The validation accuracy remains stable around 80%, with minor variance across the folds. Precision, recall, and F1 score also exhibit low variance, indicating the model's reliability.

Compared to the initial results using the default Pascal VOC split and a base model trained for 5 epochs, the k-fold results are notably better in terms of both performance and stability. The base model yielded less reliable results, likely due to the limited training epochs and the non-stratified splitting of the dataset. The k-fold approach leverages the entire dataset more effectively, providing a robust estimate of model performance. The k-fold cross-validation confirms that the model is both reliable and effective in classifying animal images. The stable metrics across folds highlight the model's ability to generalize and its robustness to variations in training data. The use of k-fold validation provided a more comprehensive assessment compared to the single split with Pascal VOC, reinforcing the model's capability to perform well on unseen data. Future work could focus on fine-tuning hyperparameters or experimenting with more complex architectures to push the performance metrics beyond the current plateau.

6 Next Steps: Deployment

To prepare for deployment, the model's weights can be captured and saved for external conversion. The goal is to encapsulate the model within a function that accepts images as input, preprocesses and analyzes them, and outputs the predicted animal in the image. The model is expected to classify the six animals it was trained on accurately but may produce incorrect or unexpected results for animals outside its training set.

7 References

[1] Pin Wang & En Fan & Peng Wang (2021) Comparative analysis of image classification algorithms based on traditional machine learning and deep learning

[2] Tibor Trnovszky & Patrik Kamencay & Richard Orjesek & Miroslav Benco & Peter Sykora (2017) *Animal Recognition System Based on Convulutional Neural Network.*

A Appendix

Loyd Flores: Research/ Paper writing and Implementation of CNN in Python
Raphael Attiaala: Research/ Paper writing