CECS 456 Machine Learning Project: Animal Identification using Convolutional Neural Networks (CNN)

Edgar Rodriguez-Macias
California State University, Long Beach
edgar.rodriguezmacias@student.csulb.edu

May 15, 2024

GitHub

https://github.com/edgarLB/MLfinal

1 Introduction

1.1 Motivation

As computing power has increased so has the accessibility to machine learning. This is evident with the recent trend of artificial intelligence products emerging such as Dall-E and Google Gemini. In this project I will be creating a CNN model to recognize images of animals in 10 categories.

1.2 CNN

A convolutional neural network (CNN) is a type of deep learning algorithm primarily used for image recognition. It accomplishes this by using thousands to millions of labeled images, data, to train the model.

1.3 Scope

Design and train a Deep Leaning model for dataset.

2 Dataset

2.1 Kaggle

The dataset used is Animals-10 by Corrado Alessio. This set includes about 28k pictures of animals collected from Google Images. The photos are categorized into 10 categories: dog, cat, horse, spider, butterfly, chicken, sheep, cow, squirrel, elephant. Included in the set is a python dictionary with the mapping of the Italian to the English names of the animals. Note: I had to add the translation to spider as it was missing from the dictionary.

To inspect the data to possibly find any biases, I created a pie char that shows if any group of animals is over or under represented (*see Figure 1*).

2.2 Related Work

Kaggle allows users to share their notebooks. There is a user who was able to accomplish 98% accuracy on the same dataset utilizing the “ResNet50” model. Upon some light research I found that it is pre-trained model that can be used for specific tasks. However, I will not be using this model instead; I am creating my own CNN model.

2.3 Preprocessing

Since the images were a variety of different dimensions and aspect ratios, they were preprocessed. I made the images all have a 1:1 aspect ratio by resizing them to be 64x64 pixels. I opted to resizing the images rather than cropping to avoid cropping the subject out of the image if it is not directly in the center (*see Figure 2*).

I chose to reduce the size of the images to 64x64 pixels despite majority of the images were larger than 100x100 pixels to reduce the number of computations. This was important as I was dealing with RGB images.

The data was split into training, validation, and test set with a 70/15/15 split, respectively.

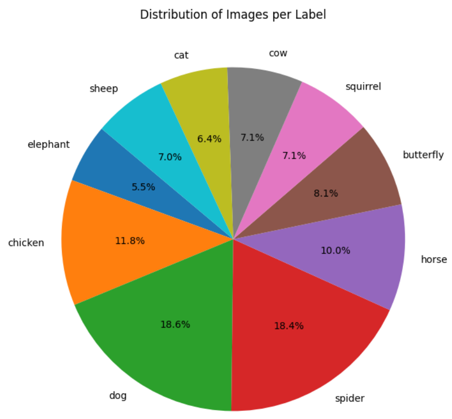

Figure 1: Pie Chart showing distribution of images per label. Dogs and Spiders images are overrepresented.

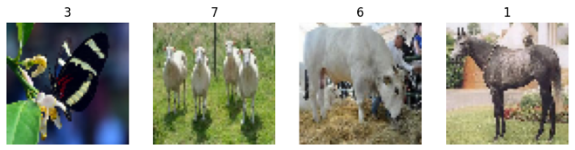

Figure 2: Preprocessed images with numerical label.

3 Methodology

All 3 models were compiled utilizing “adam” optimizer and “sparse_categorical_crossentropy” loss.

3.1 Model 1 (cnn1)

When trying to figure out where to begin I decided to start with the model provided in the demo_for_CNN.ipynb and work to improve the model for my dataset.
This model consists of:

- Input Shape: [64, 64, 3]
- Convolution Layers: 5 total, filters ranging from 64 -256
    - Activation function: ReLU
    - Padding type: same
- 3 pools size 2
- Flatten 3D tensor into 1D tensor
- 2 fully connected layers with a dropout of rate 0.5, after each to help prevent overfitting.
- Output Layer: size 10
    - Activation function: softmax

This model has a total of 3,222,346 trainable parameters.

3.2 Model 2 (cnn2)

Model 1 took long to train and was not accurate enough so, I worked to improve on both of those aspects. To accomplish this, I utilized a different type of convolution layer, SeperableConv2D(), as I read it provided better performance with similar results. I also implemented batch normalization to help improve training speed.
This model consists of:

- Input Shape: [64, 64, 3]
- Separable Convolution 2D Layers: 5 total, filters ranging from 64 -256
    - Activation function: ReLU
    - Padding type: same
- 3 pools size 2
- Batch normalization after layers 2 and 4
- Flatten 3D tensor into 1D tensor
- 2 fully connected layers with a dropout of rate 0.5, after each to help prevent overfitting.
- Output Layer: size 10
    - Activation function: softmax

This model has a total of 2,236,957 trainable parameters. It is less despite not changing the parameters because of the different type of convolution layer.

3.3 Model 3 (cnn3)

Model 2’s training accuracy decreased so, I decided to go back to using Conv2D() again. I also changed the architecture by removing a convolution layer, change batch normalization placement, and the design of the fully connected layers.
This model consists of:

- Input Shape: [64, 64, 3]
- Convolution Layers: 4 total, filters ranging from 32 -256
    - Activation function: ReLU
    - Padding type: same
- 3 pools size 2

- A single batch normalization after the first pooling layer
- Flatten 3D tensor into 1D tensor
- Single fully connected layer with 512 units and a dropout of rate 0.5 after to help prevent overfitting.
- Output Layer: size 10
    - Activation function: softmax

This model has a total of 9,018,378 trainable parameters. This is primarily due to the single fully connected layer with 512 units.

4 Experiments

4.1 Setup

Models evaluated:

- cnn1, cnn2 and, cnn3

Test Dataset:

- Subset of 15% of Animals-10 dataset shuffled to help reduce bias

Tools:

- python libraries tensorflow, sklearn, and numpy

4.2 Measurements

The performance of each model was evaluated on the test data set. This will give the accuracy and loss of each model.

I also generated confusion matrices and classification reports for each model for further performance insight and see areas of confusion between classes.

To measure the model's accuracy considering precision and recall I also evaluated each model's F1 score.

5 Results

5.1 Model 1 (cnn1)

Accuracy	Loss	F1 Score
51.87%	1.9697	0.1231

[cnn1 matrices continue next column]

Confusion Matrix:

```
[[143  58  25  51 114  35  40  37 169  29]
 [ 81  38  14  29  51  21  21  19  92  23]
 [ 49  11  11  22  33   8  17   7  53   5]
 [ 68  21  16  24  45  13  20  18  88  10]
 [ 97  28  21  32  60  22  32  22 122  27]
 [ 47  20   5  18  39  15  17  12  60   7]
 [ 58  19  16  20  47  16  15  16  60  12]
 [ 63  33  18  23  51  12  19  13  59  12]
 [145  50  37  70 127  22  44  41 176  32]
 [ 53  25   6  12  50   9  21  12  79  14]]
```

Classification Report:

```
              precision    recall  f1-score   support

           0       0.18      0.20      0.19       701
           1       0.13      0.10      0.11       389
           2       0.07      0.05      0.06       216
           3       0.08      0.07      0.08       323
           4       0.10      0.13      0.11       463
           5       0.09      0.06      0.07       240
           6       0.06      0.05      0.06       279
           7       0.07      0.04      0.05       303
           8       0.18      0.24      0.21       744
           9       0.08      0.05      0.06       281

    accuracy                           0.13      3939
   macro avg       0.10      0.10      0.10      3939
weighted avg       0.12      0.13      0.12      3939
```

5.2 Model 2 (cnn2)

Accuracy	Loss	F1 Score
60.07%	1.1857	0.1268

Confusion Matrix:

```
[[155  87  28  53  63  17  22  68 169  39]
 [100  61  10  30  30   8  10  29  93  18]
 [ 64  30   7  17  17   5   3  10  52  11]
 [ 78  53  15  27  29   8  12  21  63  17]
 [ 99  57  11  33  37  11  20  41 130  24]
 [ 59  32   7  24  10   9   4  22  66   7]
 [ 60  48  11  21  18   4  11  28  61  17]
 [ 65  45   6  23  21   7  16  25  71  24]
 [183 110  24  57  45  15  24  55 196  35]
 [ 60  43  10  20  22  11  10  28  67  10]]
```

Classification Report:

```
              precision    recall  f1-score   support

           0       0.17      0.22      0.19       701
           1       0.11      0.16      0.13       389
           2       0.05      0.03      0.04       216
           3       0.09      0.08      0.09       323
           4       0.13      0.08      0.10       463
           5       0.09      0.04      0.05       240
           6       0.08      0.04      0.05       279
           7       0.08      0.08      0.08       303
           8       0.20      0.26      0.23       744
           9       0.05      0.04      0.04       281

    accuracy                           0.14      3939
   macro avg       0.11      0.10      0.10      3939
weighted avg       0.12      0.14      0.13      3939
```

5.3 Model 3 (cnn3)

Accuracy	Loss	F1 Score
64.58%	1.4784	0.1226

Confusion Matrix:

```
[[143  80  25  47  82  32  49  56 160  27]
 [ 84  49  18  23  42  12  24  41  77  19]
 [ 38  32   4  17  31  11  14  19  39  11]
 [ 61  46  31  19  42   8  22  24  53  17]
 [ 97  59  18  27  51  20  47  34  91  19]
 [ 44  26  11  11  32  11  17  26  52  10]
 [ 61  32  18  18  34  17  17  17  58   7]
 [ 64  29  14  27  27  10  17  27  71  17]
 [154  78  39  51  82  24  54  66 154  42]
 [ 48  29  20  18  32   9  26  23  56  20]]
```

Classification Report:

```
              precision    recall  f1-score   support

           0       0.18      0.20      0.19       701
           1       0.11      0.13      0.12       389
           2       0.02      0.02      0.02       216
           3       0.07      0.06      0.07       323
           4       0.11      0.11      0.11       463
           5       0.07      0.05      0.06       240
           6       0.06      0.06      0.06       279
           7       0.08      0.09      0.08       303
           8       0.19      0.21      0.20       744
           9       0.11      0.07      0.09       281

    accuracy                           0.13      3939
   macro avg       0.10      0.10      0.10      3939
weighted avg       0.12      0.13      0.12      3939
```

6 Analysis

When comparing all the models we can see that in terms of accuracy Model 3 performs the best. Model 2 shows better balance of precision and recall as evident of the F1 score. However, the F1 scores of all the models are quite similar to one other showing there can be room for improvement.

When looking at the confusion matrix we can see that the models struggled less in categories 0 and 8 compared the others. When looking at out data we can see 0 is dogs and 8 is spiders both, the largest categories. This shows an imbalance in the models.

Model 1 scored the worst in all areas of evaluation while Model 2 has the best loss score. This shows that the inclusion of batch normalization positively influenced the model.

7 Conclusion

To conclude based on the results of my testing, I would not feel comfortable deploying any of the 3 models designed and trained to the public if this was a business I was running. While model 3 seemed promising due to the training accuracy of 84% testing proved training accuracy is not that concrete if your model is not generalized enough. Future work to improve the model could implement techniques like data augmentation. This could help improve the accuracy of the model's predictions especially for underrepresented classes.

This project has demonstrated the challenges in designing and training your own CNN model. I could not have imagined working on this project without the open sourced Kaggle data and python libraries.