Animal Image Classification

Isaiah Martinez
CSUN
Computer Science Department
isaiah.martinez.891@my.csun.edu

Joycelyn Tuazon
CSUN
Computer Science Department
joycelyn.tuazon.251@my.csun.edu

12/02/2023

Contents

1 Introduction **2**
1.1 Background . . . 2
1.2 Project Description . . . 2
1.3 Dataset Overview and Preparation . . . 2
1.4 Outline . . . 2

2 Related Works **3**

3 Methods **4**
3.1 SVM . . . 4
3.2 ResNet50 . . . 4
3.3 CNN . . . 4
3.4 VGG16 . . . 4

4 Evaluation **6**

5 Conclusion **7**

6 Future Work **7**

1 Introduction

1.1 Background

In recent years, AI has been an growing topic. Many advancements have been made in several areas within AI including Computer Vision, Natural Language Processing, and other Machine Learning (ML) and Deep Learning (DL) projects [1]. With the release of LLM's such as ChatGPT and Dall-E, the public can now use these tools. However, the general public views AI as a negative advancement in technology. The main belief is that it reduces privacy, but can assist in other areas of life such as finding accurate information online [2].

1.2 Project Description

Our project was designed to show AI can be helpful tool. Additionally, we are beginners in the realm of ML, so this project serves as a step towards real world problem solving. The problem we sought to solve was Image Classification. We employed several ML algorithms to perform Animal Classification for images. Our goal was to learn how to do Image Classification using different methodologies to accurately predict whether an image contained a: cat, dog, or panda.

1.3 Dataset Overview and Preparation

The dataset used contained 3000 RGB images separated into 3 folders, 1 folder per animal [3]. These images ranged varying sizes, ranging less than 100 pixels by 100 pixels to larger than 1000 pixels by 1000 pixels. Each folder contained 1000 images of the depicted animal. From the given images, we selected images that were deemed clean and usable for our models. After cleaning, we had 2,952 images. Our criteria for removing images was as follows:

1. Multiple interested animal species in the same image. For example, a cat and a dog within the same image.
2. Too small of an image size: less than 100 pixels by 100 pixels.
3. Subject of the image was too difficult to see. This may include obscurities, filters, low resolution, etc.
4. The image did not contain the subject animal.

After cleaning the images, we further prepared the dataset to be entered into a readable format for the various models we selected. These included:

- Resizing images to 500 pixels by 500 pixels
- Resiging images to 224 pixels 224 pixels
- Converting images to grayscale
- Converting images to arrays
- Creating new images by augmenting the given images
  - Augment effects included: rotation, zoom, horizontal flipping, vertical/horizonal shifting, shear transforming, and pixel filling.

With these changes in mind, the amount of data becomes: 2,952 images and $(Length * Width * ColorMode) + 1$ Features. Thus for images in RGB sized 500 pixels by 500 pixels, we would have 750,001 features. The reasons for having different image sizes, color modes, and data augmentation was to increase the performance and reduce the runtime to construct the models.

1.4 Outline

The outline of this paper is as follows: First we will discuss the research that contributed to our understanding and gave insights into beginning the project. Second, we will discuss the methods and techniques employed to achieve our goal of making an image classifier. Afterwards, we will evaluate and compare the performance of our models. Lastly, we will discuss some future works that can be done to explore further into this topic.

2 Related Works

[Research and description of algs]

Related work: Explain if there is a relevant paper that worked on the same problem

3 Methods

The SVM, ResNet50, CNN, and VGG16 classifiers were implemented for our project. We also experimented with CLIP to learn about how the model approached image classification.

3.1 SVM

3.2 ResNet50

3.3 CNN

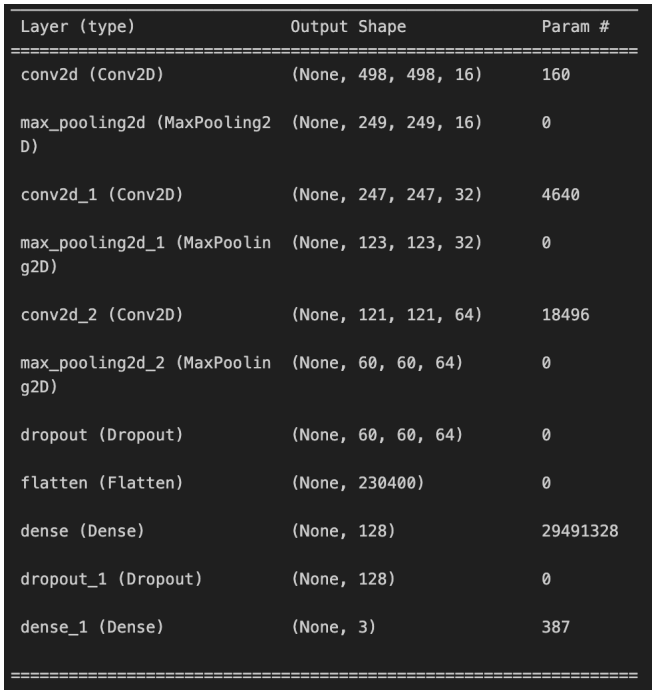
Convolutional Neural Networks (CNN) is commonly used for image classification tasks. Typically, CNN models consist of convolutional layers that apply filters to extract features and patterns, pooling layers to retain the important features and patterns, and fully connected layers to make predictions based on the features learned in the convolutional and pooling layers. Our CNN model implements a sequential structure, where the layers are added linearly. The input layer takes in the parameters of the image, in our case our images were 500x500 pixels with a color channel of 1 (since they are greyscale images). After input, the first convolutional layer applies filters to capture the features in the image data by moving a matrix of learnable weights over the image. In our model, the filters had a size of 3x3 and the number of filters increased in each convolutional layer. At the beginning of training, the weights are random before adapting its value throughout the training process. The extracted features are put into a feature map that the following pooling layer uses to further extract important features. Our model uses max-pooling, which looks for the maximum value extracted by each filter from the previous layer so that the initial feature map is downsampled. This pattern of convolutional layer followed by pooling layer iterates to continue extracting features and downsampling the feature map before a flatten layer reshapes the feature map into a one-dimensional vector that can be inputted into a fully connected dense layer. The convolutional layers and max-pooling layers iterate 3 times in our model before the feature map is flattened. We found that less iterations yielded poorer performance (model was not complex enough) and more iterations resulted in longer training time (model was more complex and computational load was larger). The dense layer is considered a "fully connected" layer because each neuron in a dense layer is connected to every extracted feature and is assigned a weight value. The initial weights are random but are learned throughout the training process. The weight of each of the connections in the dense layer attempt to understand the relationship between the important features extracted from the previous layers as a representation of a prominent feature. The output layer is a dense layer with neurons equal to the number of classes in the dataset; in our case there are 3 classes (cat, dog, and panda). The summary of the structure of our CNN model is shown in Figure 1 below.

Our convoluted layers use the Rectified Linear Unit (ReLu) activation function to introduce non-linearity, this way our model can learn complex relationships between the features of our data. The ReLu activation function sets negative values found in the convolution matrix to 0, allowing the model to focus on the positive values that likely represent relevant features. In between the dense layers, we implemented dropout regularization to prevent overfitting. Dropout works by temporarily setting the weights of a random set of neurons to 0 ("dropping out") per epoch. This prevents overfitting by forcing the model to learn more robustly instead of relying too much on the recognition of certain features and "memorizing" the training data. The output dense layer uses the Softmax activation function to scale the numerical output of the model into probabilities distributed over the possible output classes represented by indexes. The class with the highest probability is considered the predicted class which is mapped to the string value of the class using the index value.

While fitting the model, we implemented the EarlyStopping callback to reduce the likelihood of overfitting by stopping training if the validation accuracy did not improve after 5 consecutive epochs. The EarlyStopping callback can also be useful in preventing underfitting if the number of epochs is set to a higher number than what is expected to be necessary.

3.4 VGG16

VGG16 is a pre-trained CNN model available for use through Tensorflow. VGG16 was pre-trained for image classification on large datasets such as ImageNet where the input image size was 224x224 pixels with 3 color channels (RGB). Its structure consists of 13 convoluted layers and 3 fully connected dense layers (16 layers total). We decided to use a pre-trained VGG16 model instead of building one from scratch to learn how efficiently it would execute our particular image classification task. To utilize the



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 498, 498, 16)	160
max_pooling2d (MaxPooling2D)	(None, 249, 249, 16)	0
conv2d_1 (Conv2D)	(None, 247, 247, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 123, 123, 32)	0
conv2d_2 (Conv2D)	(None, 121, 121, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 60, 60, 64)	0
dropout (Dropout)	(None, 60, 60, 64)	0
flatten (Flatten)	(None, 230400)	0
dense (Dense)	(None, 128)	29491328
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 3)	387

Figure 1: CNN model summary.

pre-trained VGG16 model, we first load the base model without the top layers as these layers are what we need to modify to have the model classify our cat, dog, and panda images. Initially, the custom layers we added included 3 fully connected dense layers; however, we found the training time to be especially long and after experimenting with removing a dense layer to make the model less complex, we found that the training time was shortened with very little change in performance. Ultimately, we chose to keep working with the shortened training time with two dense layers. The first dense layer utilizes ReLu activation as well as the He uniform kernel initializer to initialize the weights of the neurons within a uniform range dependent on the number of input neurons. He uniform combats the possibility of too many dead neurons by increasing the likelihood that neurons will activate and not be stuck in an inactive state throughout the entirety of training. This gives more neurons the chance to contribute to learning. The structure of the pre-trained VGG16 is shown in 2 below. The last output dense layer acts the same as it did in our previous CNN model, utilizing Softmax to generate probabilities distributed over the three output classes mapped to their corresponding string values.

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14714688
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 64)	1605696
dense_1 (Dense)	(None, 3)	195

Figure 2: Pre-trained VGG16 model summary. The base model is loaded and the top dense layers are added manually.

4 Evaluation

[Evaluate and compare the models performance] Evaluation: Compare the methods you used

5 Conclusion

[Summarize] Conclusion: Write a conclusion and future work for your project
Here you summarize the essential aspects and findings of your work and analysis.

6 Future Work

[What improvements could be done]
better alg?

References

[1] Stanford: What are the most important Advances in AI?

[2] PRC: Growing public concern about the role of artificial intelligence in daily life

[3] Kaggle: Animal Image Dataset(DOG, CAT and PANDA)