Cat, Dog, and Panda Classifier

Tamara Pina, Zoe Powers, Julian Sandjaja

Abstract

In our report, we modify an existing binary Convolutional Neural Network (CNN) to a Categorical Convolutional Neural Network (CNN) capable of classifying images of cats, dogs, and pandas. We began with a pre-trained CNN model that had a high accuracy of 94% in classifying images of cats and dogs. Due to this high accuracy level we decided not to simply adjust the model to reach an only semi higher accuracy level, but instead to expand the models functionality by introducing a third class: pandas. The project involved data cleaning, resizing, augmentation, and splitting the dataset into training, validation, and testing sets. Afterwards, the CNN model was modified with a softmax output layer for three classes, trained on prepared data, and optimized using categorical cross-entropy loss. Afterwards the trained model was evaluated on the test set, and the model weights that corresponded to the best validation performance were retained. Ultimately, adding a third category did result in a decrease in accuracy to 77.3%, however, the model showed consistent results for training, validation, and testing sets, and had the strongest precision levels for pandas at an impressive 92% and weakest results for cats at 66%.

1 Introduction

We live in a world that has an abundance of data. Images make up a large part of this data and can provide highly beneficial insights, however, images have a high degree of visual variability and this can make them significantly more difficult to analyze than other data forms. Furthermore, before machine learning techniques such as deep learning, CNNs, and SVM were created, image analysis was done manually and required various effortful feature extraction techniques. This project explores Convolutional Neural Networks (CNNs) for image analysis. CNNs revolutionized image analysis through utilizing matrix multiplication and linear algebra to automatically discern patterns in images. CNNs were created to automatically learn and extract hierarchical features from images and often achieve human-level accuracy particularly in facial recognition and medical imaging.

Our project demonstrates the flexibility and scalability of CNN models by modifying an existing binary CNN model which classified images of cats and dogs into a three-class image classification model that classifies images of cats, dogs, and pandas. Utilizing a dataset of 3,000 images (1,000 of each animal), the project began with dataset preparation. This included labeling and splitting the data into training, validation, and testing sets, and applying preprocessing. Afterwards, the initial CNN model was modified to include a three-neuron softmax output layer for multi-class classification and trained using categorical cross-entropy loss with training and validation data. Lastly, the trained model was evaluated on the test set and saved for inferences on new images. Through our project we hope to showcase the adaptability and flexibility of CNNs and their ability CNNs to distinguish amongst similar yet different classes.

2 Background

CNNs, developed for classification and computer vision tasks, have three main layer types: Convolutional, Pooling and Fully-connected (FC) layer. The convolutional layer is the first layer (although multiple convolutional layers can exist) and is used for feature extraction. This layer begins by first applying the Convolution function:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

Here I is Input image or feature map. K is a small matrix of weights and S is Output feature map Then the activation function is applied to the convolution functions output. Typical activation functions are:

ReLU:
f(x) = max(0, x)

Softmax (used for multi class output):

$$\sigma(z)_i = \frac{e^{z_i}}{\Sigma_j e^{z_j}}$$

After, the pooling layer applies a pooling function to the convolution layers output. Pooling layers reduce complexity by minimizing the output's dimensionality. The equation for Max Pooling is:

$$P(i,j) = max\{I(m,n)\}, \forall (m,n) \in w$$

Next, the final convolutional and pooling layers output is flattened into a vector. For instance, given a tensor of dimensions h×w×c, the flattened vector has hwc elements. This is then fed into the FC Layer (the final layer) which is used for classification and computes: z=Wx+bz. Where W is Weight matrix, x is Input vector, b is Bias vector. Lastly, the FC layers output passes through an activation function with cost functions and generates a class label for the input image.

3 Dataset

With a course of action in mind, the details of the dataset had clear requirements: include the images of the cat, dog and a third animal as well as containing enough data to properly train the model. For this project, the dataset used for training and evaluating the convolutional neural network was sourced from Kaggle, and consists of 3,000 labeled images, evenly distributed amongst dogs, cats, and the new edition, pandas. This dataset, which is smaller in comparison to the original 25,000 valued dataset observed in the model CNN, moves away from simpler binary classification in exchange for a three class model. This smaller sample size allowed us to minimize computational cost and provide a more time efficient way to expand off of an existing model. While this dataset was on the smaller end of the spectrum, this reduction of the risk of overfitting came at the cost of potential in accuracy. The data found was clean in the sense of containing all valid cat, dog, and panda images but they varied wildly in size which called for the need to resize and standardize all images. Before continuing with the model creation, the training data was resized, had the variation in frame occupance, and other external factors accounted for to reduce overfitting. This allowed for the model to focus on only identifying the differences between the three animals and avoiding disturbances from the surrounding environment or potential obstructions in the images. The same process was not

applied to the test data with the expectation that the model will be trained to identify the three species without alterations.

Figure 1: Irregular data with variation in size and unnecessary background

From that point, the images were organized into categories to allow for the train-validation-test split to call from every animal in equal proportions.

4 Model

The architecture used for this project includes a multi-class classification model to utilize classification for the three animals used. he model consists of six layers: one input layer, one output layer, and four hidden layers. The input layer processes a 128x128 pixel RGB numerical representation of the scaled image where each pixel is represented as a numerical value scaled between 0 and 1.The input is then passed through a convolutional layer with 32 filters, each represented by a 3x3 matrix, and uses the ReLU activation function to capture basic features in the image. The activation outputs are then normalized with a mean of 0 and a standard deviation of 1 to prevent vanishing or exploding gradient. To reduce model complexity and overfitting, the model then grabs the most important feature from a 2x2 window and randomly drops 20% of the neurons.

The next three hidden layers apply the same activation using a 3x3 kernel and ReLU activation, with 64, 128, and 256 filters respectively. Like the input layer, each of these layers includes batch normalization, max-pooling, and dropout to maintain stability and prevent overfitting.

The final hidden layer, the fully connected layers, projects the feature map into a vector. The layer then adds a dense layer with 512 neurons and a ReLU activation function. This dense layer is followed by batch normalization and dropout (20%) to enhance generalization. The output layer contains three neurons to represent the three animals being classified and a softmax activation function. The final output would result to the probability of the given image being a cat, dog, or panda. The maximum probability of the three would be chosen and the image will be classified corresponding to the maximum probability.

5 Methodology

With the architecture set up, the model must be trained. Since the model architecture is

a multi-class classification model, the loss function used is a categorical cross entropy function as it calculates the difference between the predicted and actual class probabilities. The Adam optimization is applied to adjust the model's weights dynamically and minimize the loss function efficiently. A learning rate modifier was included to reduce the learning rate by half if the validation accuracy did not improve for two consecutive epochs, ensuring the model could continue making gradual improvements. The minimum learning rate was set to 0.00001, where the training will terminate once the learning rate becomes less than that value.

With this, the model can finally be trained. The training process incorporated mini-batch stochastic gradient descent, where the data was divided into small batches to improve computational efficiency. Batches of validation data is used to evaluate the model performance within each epoch to detect signs of overfitting and adjust the learning rate accordingly. The model completed 18 epochs before terminating when the learning rate became less than its minimum threshold.

6 Results

The cat-dog-panda CNN model achieved an accuracy of approximately 77.33% on the training dataset with a loss of roughly 0.47. We observed a similar convergence in every run of the model, typically plateauing in accuracy around the 17th epoch.

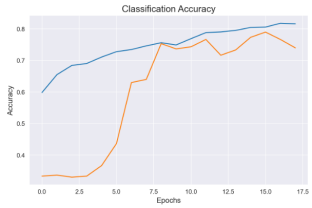

Figure 2: Classification accuracy throughout each epoch where blue line represents training, orange represents validation

The model performed exceptionally well on the panda classification, with a precision of 92%, compared with the lacking cat classification with a precision of 66%, with the dog classification resulting in 81% precision falling between the two.

Table 1: Classification Report

	Precision	Recall	f1-score
Cat	0.66	0.89	0.76
Dog	0.81	0.51	0.63
Panda	0.92	0.94	0.93

With this understanding of the precision of the model, we tested both our testing and validation sets. With an unsurprisingly similar result, the accuracy of the model in the validation data was 77.33%, with a loss of .49, and the accuracy in the test data set was found to be 77.99%, with a loss of.48. Despite the consistency in the results of the model when given all three training sets, the misclassifications observed suggest the model may struggle to distinguish cats specifically within the dataset. These results indicate that the effectiveness of binary classification does not immediately extend to a more complex goal.

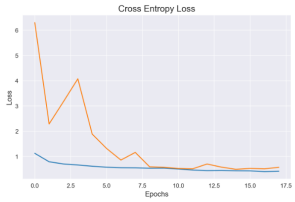

Figure 3: Cross entropy loss throughout each epoch where blue line represents training, orange represents validation

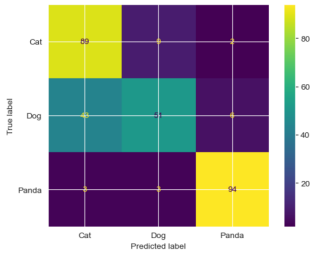

Figure 4: Confusion matrix heat map

7 Conclusion

In this report we were given the opportunity to either build a Convolutional Neural Network from scratch or improve upon an existing CNN with what we have learned throughout the course. Given the high precision of the preexisting model 94%, we opted to challenge ourselves by expanding its capabilities rather than attempting to make minimal adjustments. Our decision to add a third category resulted in a significant decrease in accuracy, and while this result was expected due to the decrease in the size of the data set and purposefully minimal changes in the core of the model creation, the remarkable 16% difference invited opportunity for future experimentation. The newly adapted model proved to raise questions about the trade-off between efficiency and adaptability of a model. Understanding the balance between maintaining computational efficiency and saving space for flexibility is crucial in a further exploration of the applications and limitations of CNN models.

GitHub Repository

`https://github.com/TamiPina/Project156.git`

References

[1] Sachin, *Cats-vs-Dogs*, 12 Mar 2020 *Dogs and Cats*

[2] Sachin, *Cats vs dogs: Image classification using CNN*, Sept, 2023

[3] eward96, *Dog breed image classification*, 31 Mar 2021

[4] "7 Applications of Convolutional Neural Networks - FWS." *7 Applications of Convolutional Neural Networks - FWS*, www.flatworldsolutions.com/data-science/articles/7-applications-of-convolutional-neural-networks.php. Accessed 23 Oct. 2024.

[5] Soham Patel, "Animal Connections Dataset (Cats, Dogs, and Pandas)" *Kaggle*, https://www.kaggle.com/datasets/sohampatel26/animal-detection-dataset-cats-dogs-and-pandas. Accessed 28 Nov. 2024.

[6] Christopher M. Bishop and Nasser M. Nasrabadi, *Pattern Recognition and Machine Learning*, Volume 4, Springer, 2006.

[7] Viso.AI, "Image Classification," *Viso AI*, https://viso.ai/computer-vision/image-classification/. Accessed 6 Dec. 2024.

[8] Farina, "Breaking Down the Mathematics Behind CNN Models: A Comprehensive Guide," *Medium*, https://medium.com/@beingfarina/breaking-down-the-mathematics-behind-cnn-models-a-comprehensive-guide-1853aa6b011e. Accessed 6 Dec. 2024.

[9] University of South Dakota, "Thesis/Dissertation Repository," *USD Red Library*, https://red.library.usd.edu/cgi/viewcontent.cgi?article=1117context=diss-thesis. Accessed 6 Dec. 2024.

[10] Aniruddha Roy, "Mathematical Foundations of CNNs - Part 1: A Deep Dive into Convolutional Layers," *Medium*, https://medium.com/@aniruddharoy535/mathematical-foundations-of-cnns-part-1-a-deep-dive-into-convolutional-layers-b340667fc614. Accessed 6 Dec. 2024.

[11] GeeksforGeeks, "Convolutional Neural Network (CNN) in Machine Learning," *GeeksforGeeks*, https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/. Accessed 6 Dec. 2024.

[12] DataCamp, "Introduction to Convolutional Neural Networks (CNNs)," *DataCamp*, https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns. Accessed 6 Dec. 2024.

[13] Guy Monahan, "Basic Overviews on Convolutional Neural Networks," *Medium*, https://guymonahan.medium.com/basic-overviews-on-convolutional-neural-networks-f205180116be. Accessed 6 Dec. 2024.

[14] IBM, "Convolutional Neural Networks," *IBM Topics*, https://www.ibm.com/topics/convolutional-neural-networks. Accessed 6 Dec. 2024.

[15] Plain Concepts, "Convolutional Neural Network Guide," *Plain Concepts*, https://www.plainconcepts.com/convolutional-neural-network-guide/. Accessed 6 Dec. 2024.

[16] Analytics Vidhya, "Mathematics Behind Convolutional Neural Network," *Analytics Vidhya*, https://www.analyticsvidhya.com/blog/2020/02/mathematics-behind-convolutional-neural-network/. Accessed 6 Dec. 2024.