Dissertation Submitted for the partial fulfillment of the **M.Sc. (Integrated) Artificial Intelligence & Machine Learning** degree to the Department of AI&ML and Data Science

CC-314 & CC-315 Mini Project-I

Semester- 6

Wild vs. Domestic: Image Classification of Animals using Deep Learning

submitted to

By

Meet Solanki and Jenil Poria

Under the guidance of

Anjali Patel

M.Sc. (Integrated) Artificial Intelligence & Machine Learning

Department of AIML & Data Science

School of Emerging Science and Technology

Gujarat University

April 2024
DECLARATION

I hereby declare that the study entitled "Wild vs. Domestic: Image Classification of Animals using Deep Learning" submitted to the Gujarat University, Navarangpura, Ahmedabad (Gujarat) in partial fulfilment of M.Sc. (Int) Artificial Intelligence & Machine Learning degree is the result of investigation done by myself. The material that has been obtained (and used) from other sources has been duly acknowledged in this study. It has not been previously submitted either in part or whole to this or any other university or institution for the award of any degree or diploma.

Place: Ahmedabad

Date:

Name of Student

Meet Solanki

April 2024

DECLARATION

I hereby declare that the study entitled "Wild vs. Domestic: Image Classification of Animals using Deep Learning" submitted to the Gujarat University, Navarangpura, Ahmedabad (Gujarat) in partial fulfilment of M.Sc. (Int) Artificial Intelligence & Machine Learning degree is the result of investigation done by myself. The material that has been obtained (and used) from other sources has been duly acknowledged in this study. It has not been previously submitted either in part or whole to this or any other university or institution for the award of any degree or diploma.

Place: Ahmedabad

Date:

Name of Student

Jenil Poria

Index

Abstract

In this research, using cutting-edge deep learning techniques, this study provides a thorough analysis of the classification of animals into two distinct categories: domestic and wild. The dataset used in this work includes 5000 images from the images.cv websites, representing 10 species that are frequently classified as animals. We conducted extensive trials to properly categorize animals into their respective groups, using the robust custom model as a basis for our work. Our findings show how well our method works to differentiate between Wild animals such as lions, tigers, gorillas, leopards, and bears and domestic animals including dogs, cats, sheep, rabbits, and cows. Our model performed very well in categorizing animal species with excellent accuracy and Adaptability by utilizing deep learning techniques. With significant implications for wildlife monitoring and pet identification applications, this research adds insightful knowledge to automated animal classification systems. The results of this research open the door to the creation of advanced computer vision systems that accurately identify and categorize a wide range of animal species in a variety of environmental situations.

Keywords

Animal Classification, Deep Learning, Convolutional Neural Networks (CNN), Domestic Animals and Wild Animals, Image Processing, Binary Classification, Neural Network Models,

Introduction

Animal classification into domestic and wild categories is an important task that has significant uses in agriculture, ecology, and wildlife protection, among other domains. Conventional techniques for classifying animals sometimes rely on specialist knowledge and manual observation, which may be laborious, subjective, and prone to mistakes. However, the introduction of deep learning methods, especially Convolutional Neural Networks (CNNs), has made it possible to automate and enhance the precision of animal classification tasks using visual data.

In this study, we utilize deep learning to address the challenge of distinguishing between wild vs domestic animals, with particularly focus on CNN techniques. This study attempts to show that CNNs can distinguish between domestic and wild animals with sufficient accuracy based solely on visual features seen in images by extracting edges or pattern in the same kind of images.

It is crucial to distinguish between domestic and wild animals for several reasons. Natural environments are home to wild animals, who are crucial to preserving biodiversity and ecological equilibrium. On the other hand, domestic animals are those that people have carefully cultivated and tamed for uses like food production, agriculture, and companionship. Understanding and correctly categorizing these discrete animal groups is crucial for efficient administration, preservation, and welfare endeavors.

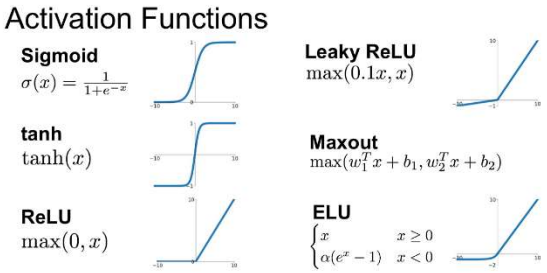
To achieve goals, we gathered a set of 5000 images, 500 of which were taken for each category of animal. To ensure diversity and representation within each category, we specifically chose five species from the categories of domestic and wild animals, respectively. We created and trained a CNN-based deep learning model to handle the classification problem by utilizing this dataset.

Many benefits come from using CNNs for classification of images task. CNNs are highly effective for tasks requiring a lot of visual data since they are made to automatically learn structures from raw picture data. Our CNN model can be trained on a large dataset of photos of both domestic and wild animals, and it can learn to discriminate between these categories with high accuracy by identifying microscopic visual indications and patterns.

The aim of this model is to make a valuable contribution to the expanding literature about deep learning and animal ecology through this study. We hope to offer knowledge and resources that can support practitioners, conservationists, and researchers in their work to monitor, manage, and safeguard species in both natural and tamed areas by showcasing CNNs' ability in classifying wild vs domestic animals.

Basic Terminology

1) Activation Function:

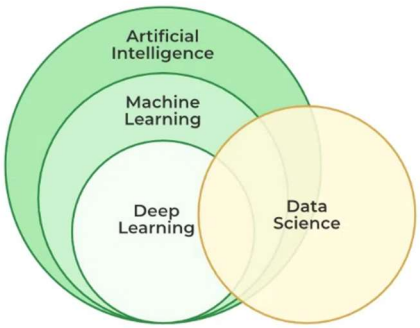


Neural networks [5] are given vitality by activation functions, which introduce non-linearity to reveal complex patterns in the data. The Rectified Linear Unit (ReLU) is a representation of power and simplicity among them, whereas the Sigmoid gently curves paths into gradients of possibilities. These functions are masters of complexity; they turn the network environment into a space for sophisticated comprehension and allow for the easy investigation of large amounts of real-world data.

2) Batch Normalization:

Batch normalization standardizes activations across mini batches of data, bringing order to the unstructured neural network architecture. It guarantees convergence very effectively by preserving consistency and speeding up training. It protects against internal covariate shift in addition to speed and stability, maintaining the network's dynamics in the face of shifting data distributions. Batch normalization acts as a watchful conductor, coordinating neural network training such that activation noise blends in with the learning tune.

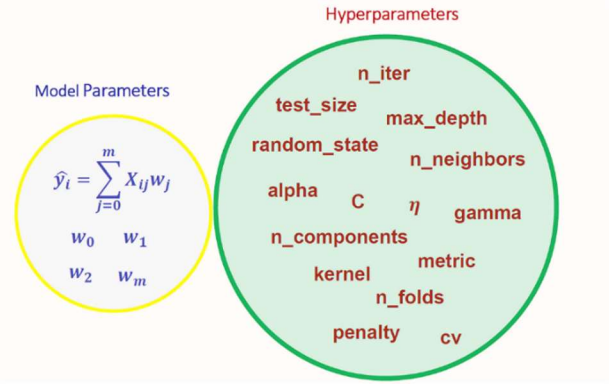
3) Deep Learning:



A branch [6] of machine learning that extracts and learns hierarchical representations from

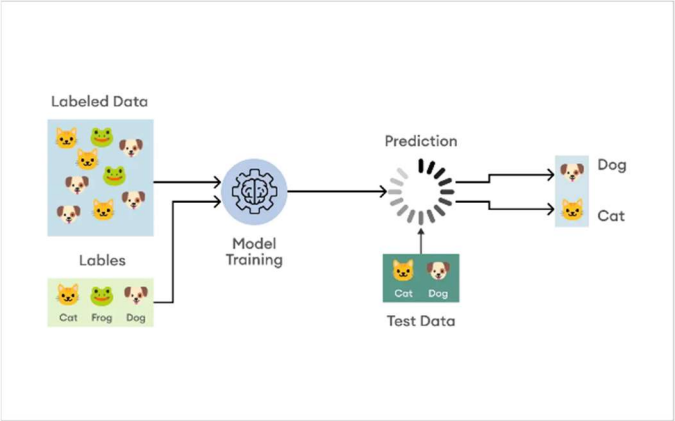
data using multi-layered artificial neural networks. Because deep learning algorithms can automatically learn features from raw data, they are very useful for image classification jobs. One example of this is CNNs.

4) Hyperparameter:



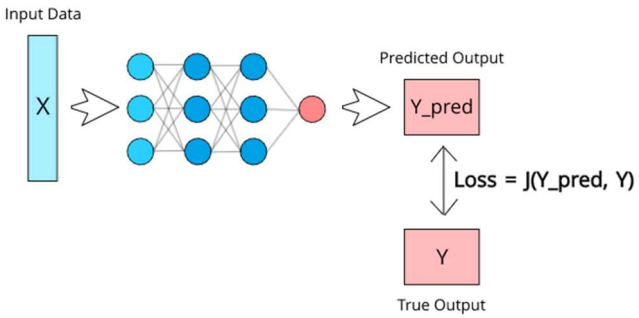
Hyperparameters [7] are important machine learning parameters that direct a model's investigation. They influence the model's development, data intake, and overfitting prevention, respectively, and comprise learning rate, batch size, and dropout rate. These elements have a major role in a model's performance, therefore choosing and refining them carefully is essential to its success. Hyperparameters shape the possibilities of the model like unseen hands.

5) Image Classification:



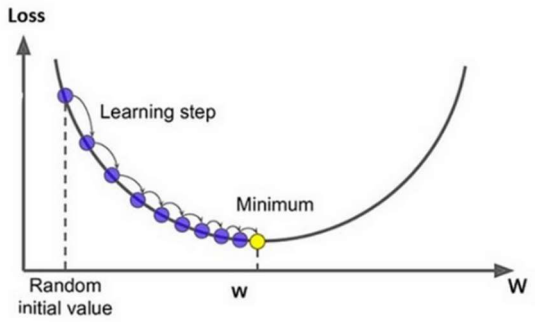
The process [8] of classifying images according to their visual characteristics into pre-established groupings or categories. Within the framework of your study, image classification involves developing a model for distinguishing between domestic and wild animals using the visual clues found in the images.

6) Loss Function:



With its [9] ability to measure the difference between predicted outputs and actual labels, the loss function serves as a sort of compass during an enormous sea of training data. It penalizes inaccurate forecasts, guiding the model toward significant patterns and adjusting its course with every iteration. This mathematical arbitrator, the cornerstone of optimization, makes sure that the model hones its abilities with purpose and accuracy, even while it is hidden from view.

7) Optimizer:



To reduce [10] loss during deep learning training, the optimizer directs parameter modifications. SGD, RMSprop, and Adam are reliable allies, and they each have special tactics. Adam moves across difficult terrain with purpose, whereas SGD combines velocity and flexible learning rates for accuracy. RMSprop protects against gradient problems to provide stability. The model advances toward deep learning expertise thanks to these optimizers.

8) Overfitting:

In machine learning [11], overfitting occurs when a model learns training data excessively well, like a child remembering a textbook by memory but not understanding its contents. Because of its excessive focus on the complex nature of the training set, the model fails to make sense of new data, which results in poor generalization. It happens when a model has too much complexity or is not regularized, leading to an efficiency that is deceptive and breaks down when real-world data becomes unpredictable.

Literature Review

Chen et al. (2019) tackles the manual interpretation challenge of camera trap data in ecological studies, where processing photos and videos is time-consuming and costly, impeding wildlife monitoring initiatives. Their solution involves leveraging deep learning techniques to automate species identification and wildlife detection, thereby enhancing the efficiency and accuracy of ecological studies. Through a two-step process, they train deep learning models using camera trap photos: first, employing a binary classification model to detect animal presence in images, then utilizing a multi-class classification model to identify species. By gathering data from camera trap photos, commonly used in ecological studies, they train their deep learning models for species identification and wildlife detection, ultimately enhancing efficiency and informing conservation efforts.

El Abbadi et al. (2020) aimed to tackle the challenge of manually classifying animals, motivated by the need for effective techniques crucial for ecological surveys, wildlife monitoring, and large-scale image classification. Presented at the 2020 International Conference on Computer Science and Software Engineering (CSASE), their project focused on automating classification using deep Convolutional Neural Networks (CNNs) on vertebrate creature photographs. Training their CNN model with a diverse collection of vertebrate animal photos, including birds and mammals, they achieved an impressive 97.5% accuracy rate through extensive experimentation and model adjustments. Despite their success, a limitation of their approach lies in the dependence on diverse and high-quality training data, which may not always be readily available or fully representative. Furthermore, the model's performance may diminish when faced with novel species or environmental conditions, necessitating ongoing refinement and adaptation.

Komarasamy et al. (2021) seek to optimize animal classification efficiency, accuracy, and scalability through deep learning techniques, employing Convolutional Neural Networks (CNNs) known for their adeptness in image classification. By automating categorization, they reduce time and resource demands, facilitating rapid processing of extensive image datasets. Their CNN-based models are trained on diverse datasets sourced from databases, wildlife surveys, and online repositories, ensuring robust performance across species and environments. However, potential challenges include the necessity for high-quality training data and adaptation to novel species or environmental contexts.

Zeng (2021) identifies the challenge of CNN-based animal classification, particularly in discerning species with high visual similarity, often overlooked by conventional models favoring easily distinguishable species like dogs versus cats. By leveraging CNN algorithms, Zeng adeptly tackles this issue, exploiting their capability to discern minute differences between visually similar species through deep learning techniques. Their systematic approach focuses on species with significant visual resemblance, employing CNN methods and tailored training tactics to effectively distinguish between closely related species. Key to this endeavor is the collection of image datasets containing visually similar animal species, likely sourced from wildlife databases or field surveys, essential for training the CNN models. Despite advancements, limitations persist, as the model's performance may still be constrained when confronted with complex real-world scenarios not fully represented in the training data.

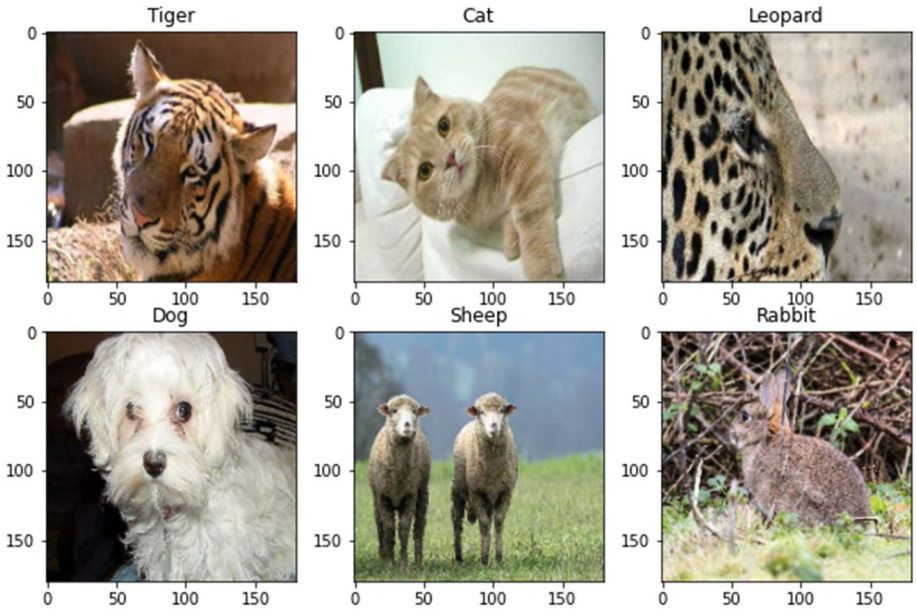
Methodology

This section outlines the approach used to do deep learning-based image classification of domestic and wild animals. Our approach includes gathering data (how we gathered the data), preprocessing (Manipulating the images which helps the model to identify the animals from the images), model selection (Choosing suitable model which appropriate for image data), training process or model building (initialization of model architecture with different layers which helps to accurately classify the two distinct categories), testing and validation or Model evaluation (evaluating the model using different performance matrix to identify how our model performing in specific data), creating research. We break down all these sections in deep discussion to provide a brief information regarding our approach used to classification model with utilize deep learning techniques or methods.

1) Data Collection

The Image data were gathered from secondary sources, mostly from images.cv website where we hit the keyword of a particular animal that we need, and it provides a zip file of several images of animal. This step repeats as per the number of animals that we required. The deep learning model has been trained with 5000 images. For the model, we considered ten animals, five for domestic and another five for wild. For each animal we gathered 500 images. For domestic animals we consider dogs, cats, sheep, rabbit, and cow. For wild animals we consider tiger, lion, leopard, gorilla, and bear. It may provide irrelevant or redundant images which downgrade the performance of the deep learning model, so that we must preprocess all the images.

We carefully selected a wide range of image samples during the data gathering procedure to fully capture the variety of both domestic and wild animal species. Here, we provide an overview of our dataset, which consists of six randomly chosen photos that highlight the wide range of animal species we are looking at. These samples perform as illustrative instances, offering valuable



perspectives on the scope and complexity of our dataset, which is the foundation of our classification system.

2) Data preprocessing

One of the most important parts of our process is preparing the raw image data once we have gathered all our image dataset to make sure that it is consistent and ready for further analysis. Standardization is essential for smooth model training because of the inherent variations in shape and format among the gathered photos. To do this, every image goes through a thorough reshaping procedure, which produces consistent dimensions throughout the collection. To be more precise, every image is downsized to a constant 224 by 224-pixel shape, which guarantees uniformity and makes processing easier.

Moreover, as we are trying to classify complex visual characteristics, we choose to keep the color information in the images. As a result, every image is kept in its original RGB (Red, Green, and Blue) format, which consists of three channels that together capture the whole range of colors and textures that are present in the photographs. Following this defined preprocessing procedure helps us to make sure that our dataset is properly prepped for further model training, which lays the groundwork for reliable and accurate classification results.

All files, whether they images, videos, or other types of data, are eventually encoded in binary format when it comes to computer storage. In the same way, the model is unable to immediately recognize or detect complex patterns or features from the raw image information while processing

images for deep learning tasks. Typically, an image's pixels have values between 0 and 255, which indicate how brilliant or intense a color is.

To standardize our preprocessing, we perform an important step before analyzing our images. This entails converting each image's pixel values into a defined range to guarantee consistency and compatibility throughout the collection. We rescale the image data to a normalized range between 0 and 1 by dividing each pixel value by 255. In addition to promoting consistent data representation, this normalizing procedure helps to improve the model's stability and convergence during training.

40	**53**	**35**	**2**
12	63	59	24
66	45	11	87
36	36	12	45

After Standardization

0.016	**0.207**	**0.137**	**0.008**
0.047	0.246	0.230	0.094
0.258	0.176	0.043	0.340
0.141	0.141	0.047	0.176

Then, the standardized images are transformed into 2D NumPy arrays, which are in a format that makes them easier to handle and manipulate within the computer system. This conversion makes it possible for the image data to be seamlessly integrated into the model pipeline, opening the door for in-depth analysis and pattern identification. Following this preprocessing approach to the letter allows our model to successfully extract important insights and patterns from the picture data, which improves the classification framework's resilience and accuracy.

0.016	**0.207**	**0.137**	**0.008**
0.047	0.246	0.230	0.094
0.258	0.176	0.043	0.340
0.141	0.141	0.047	0.176

Standardized Image Into 2D Numpy Array

[[0.016,0.207,0.137,0.008],
[0.047,0.246,0.230,0.094],
[0.258,0.176,0.043,0.340],
[0.141,0.141,0.047,0.176]]

3) Model Selection

The selection of a suitable model architecture is critical to achieving our goal, which is accurate and reliable image classification. Although neural networks are a good choice for training models, their fundamental constraints in handling complicated visual input sometimes result in less-than-ideal outcomes. Traditional neural networks sometimes struggle with the complexities of image data, in contrast to Convolutional Neural Networks (CNNs), which are specially designed for image analysis tasks. Neural Network ‘s tendency to handle the entire image as a single object, which results in a performance decrease, is the main cause of this.

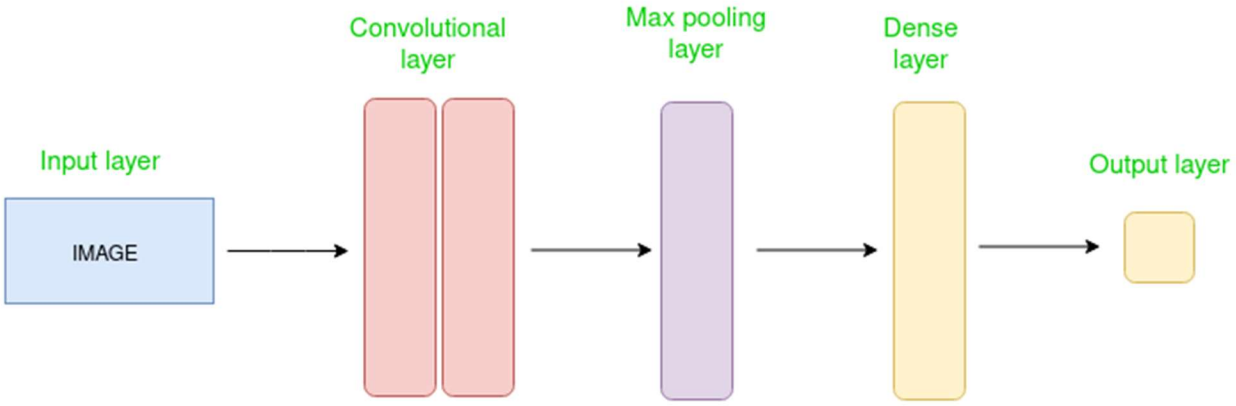
Since animal images are complex and subtle, CNNs were the best option for our classification task. Convolutional and pooling layers, among other specialized layers, are used by CNNs to effectively extract significant patterns and features from images. This allows them to identify minute details like edges and textures. CNNs are designed particularly to evaluate image data, unlike Recurrent Neural Networks (RNNs) and Artificial Neural Networks (ANNs), which might not have this ability by default.

To further improve accuracy and consistency, we decided not to use transfer learning techniques or pretrained models. Rather, we gave top priority to creating a custom CNN architecture, which enables direct learning from the dataset and customized adaptation to the distinct features of our image data. With this customized method, we can optimize the model's performance via convolutional filter configurations, activation functions, and kernel sizes to match the characteristics of the image input.

By following this systematic approach, we make sure that our model is well-suited to the difficulties presented by classifying animal images, which improves its performance and generalization abilities. Through the utilization of CNNs' specific abilities and customization to the complexities of our dataset, we establish the foundation for attaining accurate and dependable image classification outcomes, highlighting the indispensable nature of CNNs in the field of image analysis.

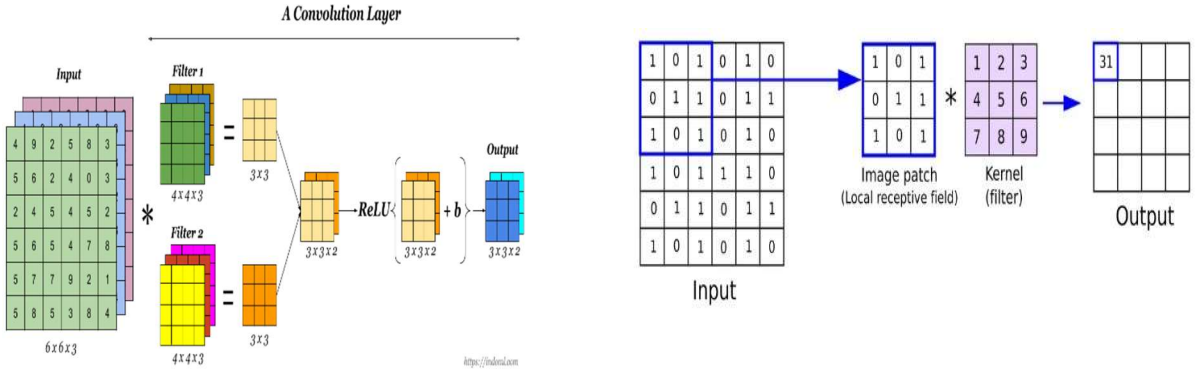
4) Model Building

Convolutional Neural Network (CNN) architectures that are specifically suited to the features of the dataset must be constructed and modified to provide an efficient image classification model. The procedures required in constructing and improving the CNN model for the purpose of classifying wild and domestic animals are described in this section. The first step of building a model was to develop a unique CNN architecture that could be used for image classification. Multiple convolutional layer's structure the design, which follows by fully connected layers for feature aggregation and classification and pooling layers for spatial down sampling. Iterative testing was used to establish the number of convolutional layers, filter sizes, and network depth to achieve a balance between model complexity and performance. Typical CNN architecture is given below.



Convolutional neural networks (CNNs) [12] are effective instruments for identifying critical visual characteristics that are essential for differentiating between animal classifications. The input layer is at the forefront of this design, processing and receiving animal images as matrices of pixels to capture vital features like hue and fur texture. The convolutional layer of the CNN processes these images and extracts complex patterns, key features, edges, and structures, such stripes, or spots, that are necessary for differentiating between domestic and wild species. These characteristics are refined by further max pooling layers, which decrease the dimensionality of the input without losing any important information. Then the flattened layer converts the 2D image into 1D vector as further layers required 1D vector format. To identify minute variations throughout animal groups, the dense layer also known as fully connected layers carefully reviews the flattened outputs, opening the door for the network's sophisticated classification choices. The output layer provides final class predictions after this complex analysis, which guides the network's classification decisions. By means of these repeated processes, CNNs gradually convert raw image input into meaningful insights, enabling automatic classification of animals according to their visual attributes.

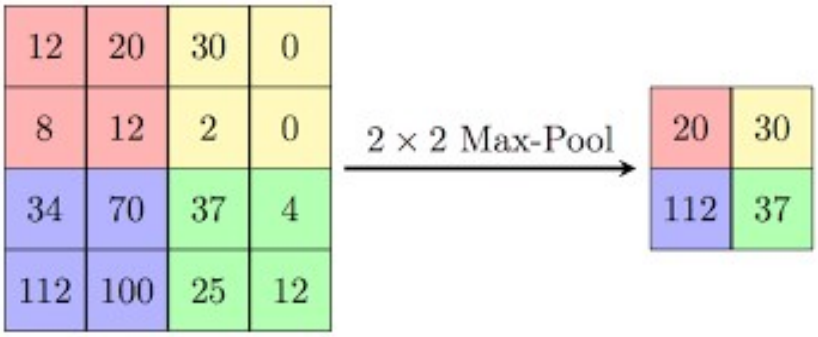
4.1) Convolution Layer



The key element [13], [14] in extracting relevant features from the input images is the convolutional layer. Convolution operations are carried out by this layer between an input image and a collection of kernels, often referred to as filters. Every kernel oversees identifying aspects or patterns in the image, such textures, forms, or edges. These kernels come in a variety of types and orientations, offering a range of filter designs that are specifically suited to capture the many visual traits found in images of animals. Each kernel performs element-wise multiplications between its weights and the associated pixel values in the input image while sliding across it with a predetermined stride, usually set to 1. The activation of that specific filter at a given spatial point in the input image is represented by a single output value that is created by adding these multiplications. The convolutional layer creates a series of feature maps that encode various features of the input image by convolving several kernels across the whole image.

In our case, where we want to classify animals into distinct categories according to their visual features, the convolutional layer serves as a feature extractor, picking out significant patterns and characteristics that correspond to various animal species. For example, vertical or horizontal filters might be useful in identifying contour forms or texture patterns associated with animal classifications. To help subsequent layers in the neural network obtain the discriminative features necessary for precise classification, the convolutional layer generates output feature maps that are rich representations of the input images. Therefore, recognizing the basic principles behind our deep learning model's effectiveness in animal classification tasks requires a comprehension of the specifics of the convolutional layer's activities.

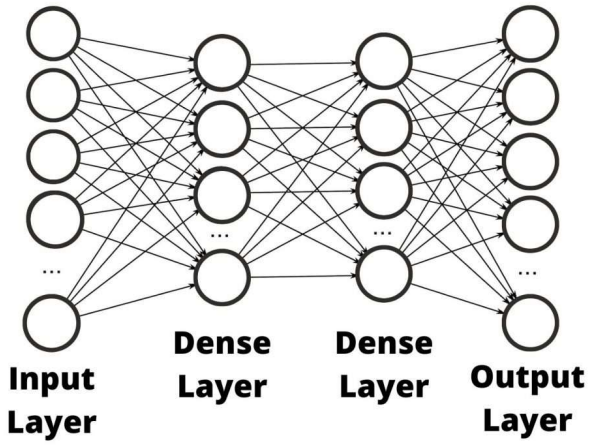
4.2) Max-Pooling Layer



To recognize [17] dominating features and reduce spatial dimensionality at the same time, we intentionally included max-pooling layers between the convolutional layers in our CNN design. By keeping the most noticeable characteristics and removing less important information, max pooling plays a vital part in feature extraction and improves the model's capacity to identify important patterns in animal images. Furthermore, batch normalization techniques were added to improve the training process's stability and effectiveness. By normalizing the activations inside each mini batch, these techniques help to mitigate problems like gradient and speed up convergence during training. Through the strategic combination of batch normalization and max pooling, our CNN architecture is strengthened to efficiently extract discriminative features and accelerate the model training process, which maximizes the model's performance for precise animal classification tasks. We have provided the code that we used to implement max pooling below.

```
MaxPooling2D(pool_size=(2, 2))
```

4.3) Dense Layer



The dense layer [22] is an essential part of our convolutional neural network (CNN) design that helps us extract complex data and make accurate classification decisions. The dense layer, which comes in after the convolutional and pooling layers, is essential for processing the outputs from the earlier layers that have been flattened. Each neuron in the thick layer is tightly connected to every other neuron in the preceding layer by dense interconnections, which allows for feature aggregation and thorough information processing. Using learnable weights and biases, this layer applies a linear transformation to the input to extract high-level characteristics that are necessary for precise animal species classification. Through a thorough examination of the input images' flattened representations, the dense layer of data identifies minute patterns and subtleties that help the network differentiate between various animal types with impressive accuracy. By carefully incorporating the dense layer into our CNN architecture, we can take use of its power to extract complex visual signals and fine-tune classification outcomes, which in turn improves the overall effectiveness and performance of our classification framework.

4.4) Relu or Rectified Linear Unit and Sigmoid Activation Function

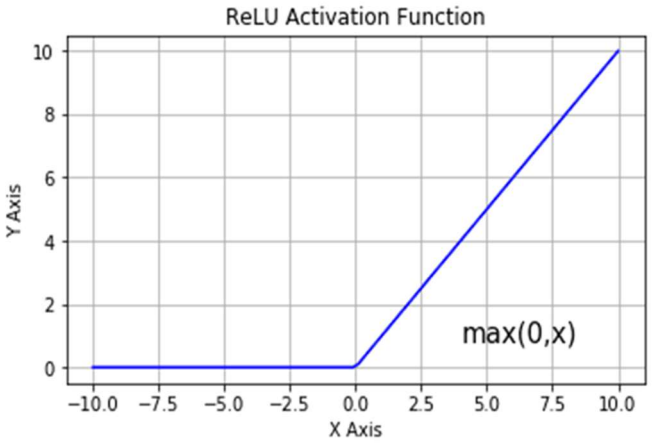
4.4.1) Relu or Rectified Linear Unit Activation Function

Rectified Linear Unit (ReLU) [15] activation functions were used in each convolutional layer of our CNN architecture as well as in hidden layers to provide non-linearity and improve the model's capacity to identify complex patterns in the input data. ReLUs are essential parts of deep learning architectures that allow for the effective processing and representation of the subtle information found in images of animals. ReLUs increase classification accuracy by utilizing non-linearities to improve the model's ability to capture and adjust to various attributes displayed by animal species. This deliberate application of ReLUs demonstrates our dedication to utilizing state-of-the-art techniques to maximize the effectiveness and performance of our classification system, guaranteeing its applicability and relevance in actual situations. We have provided the code that we used for relu activation function in convolutional layer as well as in dense layer or fully connected layer below.

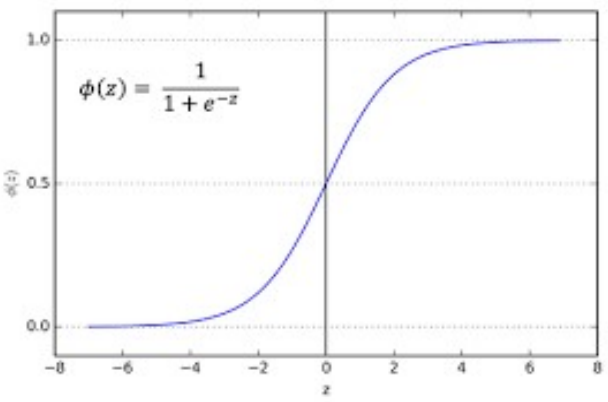
```
Conv2D(32, (3, 3), activation='relu')
Dense(64, activation='relu')
```

Mathematical intuition:

$F(x) = \max(0,x)$



Relu



Sigmoid

4.4.2) Sigmoid Activation Function

Similarly, Sigmoid activation functions [16] were utilized in the output layer to help our CNN design make final classification judgments. In binary classification tasks, sigmoid are essential because they allow the model to produce probabilistic outputs that show the probability that image belongs to a specific class. Our model can distinguish between domestic and wild animals based on visual traits by using Sigmoid, which improves classification accuracy. This deliberate incorporation of Sigmoid activation functions emphasizes our dedication to using customized methods to maximize the functionality and effectiveness of our classification framework, hence enhancing its relevance in real-world scenarios. We have provided the code that we used for sigmoid activation function in output layer below.

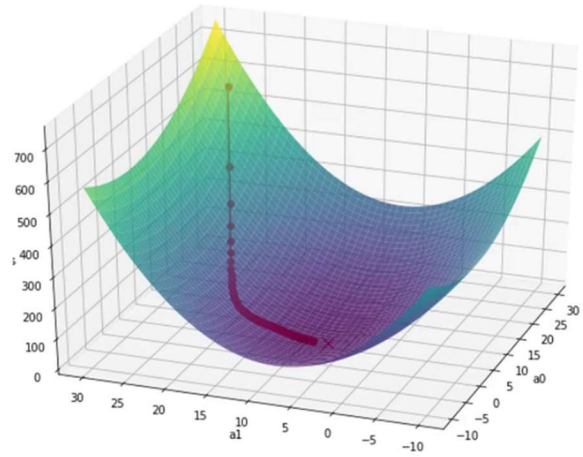
Dense(1, activation='sigmoid')

Mathematical intuition:

As sigmoid is differentiable which helps in backpropagation in weights updating such as **δ**(loss)/δ(weight).

σ(x) = 1/(1+e^-1)

4.5) Adam (Adaptive Moment Estimation)



Adaptive Moment Estimation, or Adam [21], is a crucial optimization approach that is frequently used in the training of deep learning models, such as Convolutional Neural Networks (CNNs). In contrast to conventional optimization methods, Adam combines the ideas of two well-known algorithms—RMSprop and momentum—to effectively update model parameters as the model is being trained. Its adaptive learning rate technique, which allows Adam to dynamically modify the learning rate for every parameter based on estimations of the first and second moments of the gradients, is one of its distinguishing features. Because of this flexibility, Adam is especially good in quickly convergent situations, such as those with noisy or sparsely distributed data. Adam also adds a momentum term, which is like momentum optimization, to enhance the optimization process by fusing a portion of the prior update with the present update. This momentum term is essential for optimizing the trajectory of the search and making it easier to navigate towards minima. Additionally, Adam does bias correction to reduce initialization bias and bias toward zero in the first and second moment estimations. This improves parameter update accuracy, especially in the early phases of training. Adam's adaptive nature allows it to independently modify learning rates for various parameters according to their gradients and past updates. Because of his flexibility, Adam can adjust to varied learning rates for various parameters, resulting in steady and effective optimization all the way through the training process. Thanks to its reputation for stability and efficiency on a wide range of deep learning applications and architectures, Adam has become a popular option for deep learning optimization. It is a great option for training deep learning models, like CNNs, because of its effectiveness, simplicity, and good empirical performance—especially for tasks like image categorization. All things considered, the Adam optimizer has several benefits, such as quick convergence, robustness against noisy gradients, and learning rate auto-adjustment. These characteristics make it a good choice for CNN training on image classification tasks, such as identifying domestic vs wild animals—a job that depends on effective optimization techniques to identify complex visual patterns and features in pictures. We have provided the code that we used to implemented adam as optimizer below.

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

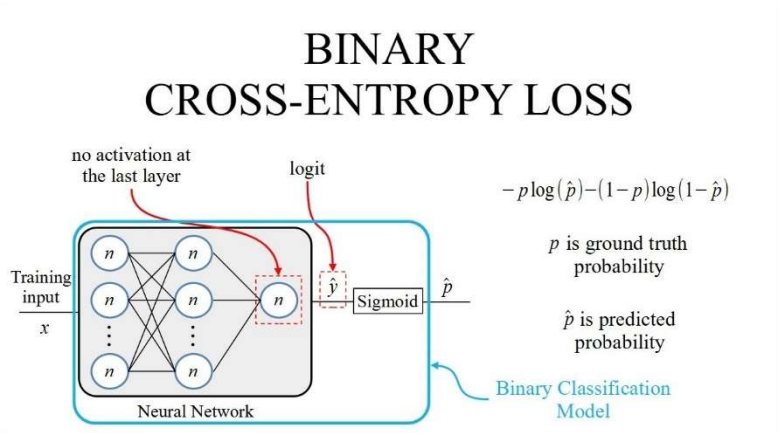
W(t +1) = W(t) – n/sqrt(V(t) + E) * M(t)

Where:
M(t) = Betta1*M(t-1) + (1 – Betta1) * Gradient of W(t)
V(t) = Betta2*V(t-1) + (1 – Betta2) * (Gradient of V(t))^2

After that we do bias correction
M_cap(t) = M(t)/(1-Betta1^t)
V_cap(t) = V(t)/ (1-Betta2^t)

Generally, the value of Betta1 is 0.9 and Betta2 is 0.99 but we can change as per our need.

4.6) Binary Cross Entropy



As we utilized the Adam optimizer and backpropagation approach in the CNN model's training phase, this combination was effective in updating the model parameters and reducing the loss function. To optimize the model parameters efficiently, we specifically chose the binary cross-entropy loss function [18], which is designed for binary classification problems. The difference between the predicted probabilities and the actual class labels is measured using binary cross-entropy, which offers a reliable metric for directing the model toward accurate classification results.

Binary cross-entropy appears as a logical choice in the context of animal classification, as the task usually entails discriminating between two separate categories (e.g., domestic, and Wild). Binary cross-entropy is a loss function that is especially suited for binary classification scenarios, which aligns with the nature of our classification problem better than other loss functions that are created for multi-class classification tasks, like categorical cross-entropy.

We take advantage of binary cross-entropy's built-in capacity to punish departures from the real class labels, which helps to nudge the model in the direction of more confident and accurate

predictions. In addition to improving the model's performance, this focused loss function also simplifies the optimization process, resulting in quicker convergence and increased overall efficiency. Thus, our dedication to improving the model's performance for the unique needs of binary animal classification tasks is highlighted by the purposeful selection of binary cross-entropy. We have provided the code that we used to implemented binary cross entropy as loss function below.

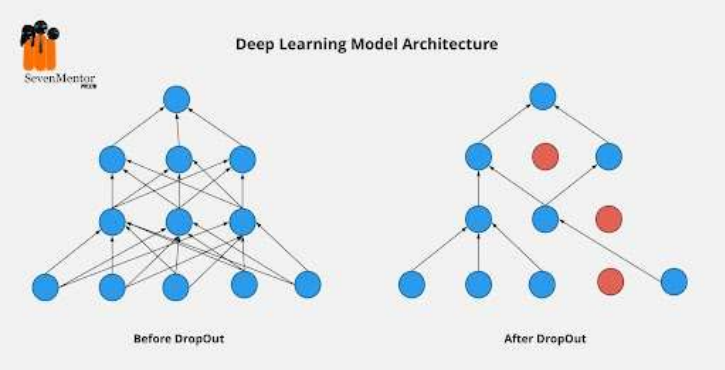
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

Mathematical intuition:

loss function = $-[y\log(\text{ycap}) + (1-y)\log(1-\text{ycap})]$

We utilized strategies such as dropout and early stopping to prevent the chances of overfitting in our model.

4.7) Dropout



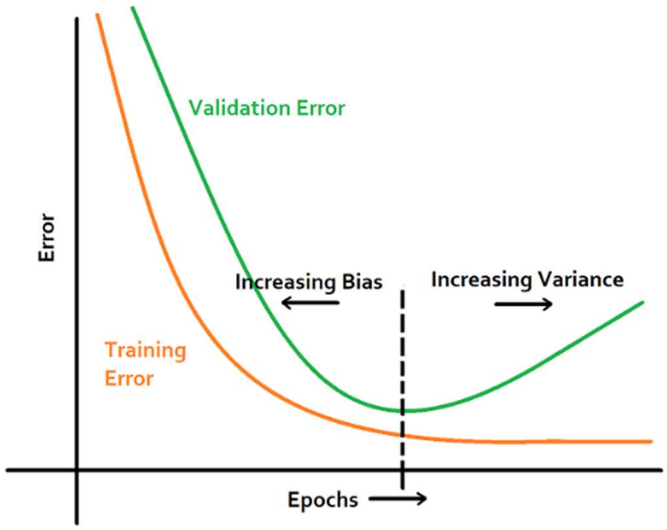
We used dropout regularization technique [19] as a tactical approach in the training phase to reduce overfitting and improve our CNN model's capacity for generalization. By randomly deactivating a percentage of the network's neurons during each training iteration, this regularization strategy helps prevent overfitting. By doing this, dropout ensures that the model does not to depend on characteristics or patterns seen in the training data by preventing neurons from co-adapting.

A trade-off is made between regularization strength and model capacity with a dropout rate of 0.2 to 0.5. This rate was carefully chosen using insights from prior research as well as empirical evidence from real data. Dropout regularization allows us to give the model's resilient feature representations top priority, which helps the model generalize effectively to new data and a variety of environmental situations. Dropout regularization is essential for improving the model's resistance to noise and variability present in real-world datasets when it comes to animal classification. Dropout regularization contributes to the development of a more flexible and adaptive model by encouraging neuronal diversity and avoiding an over-reliance on traits. Thus,

our CNN model continues to work well across a variety of animal species and environmental conditions, leading to increased classification accuracy and dependability. We have provided the code that we used to implemented dropout in hidden layer as well as convolutional layer below.

Dropout(0.1)

4.8) Early Stopping



During the training phase, we used early stopping [20] as an intentional regularization method in addition to the comprehensive model architecture design. In deep learning tasks, early stopping prevents overfitting, a typical problem when the model gets too specialized to the training data and loses its capacity to generalize to new samples. Training was terminated early when no discernible improvement in validation performance was seen to ensure optimal generalization to unknown data. This active method ensures that the model learns strong and transferable characteristics that enable it to successfully identify unknown images, while also preventing the model from forgetting the training set. For our animal classification task, early stopping is essential to improve the efficiency and dependability of the model. We achieve an optimal balance between model complexity and generalization ability by stopping training at the right time, which maximizes the model's capacity to accurately classify a variety of animal species. We reduce the possibility of overfitting and increase the model's ability to handle unknown data by carefully using early stopping, which strengthens the model's applicability for actual uses like conservation and wildlife monitoring. This complex approach demonstrates our dedication to improving our classification framework's robustness, dependability, and accuracy.

To maintain optimal model generalization and avoid overfitting, we integrated the early stopping strategy into our code. This improved the stability and dependability of our classification architecture.

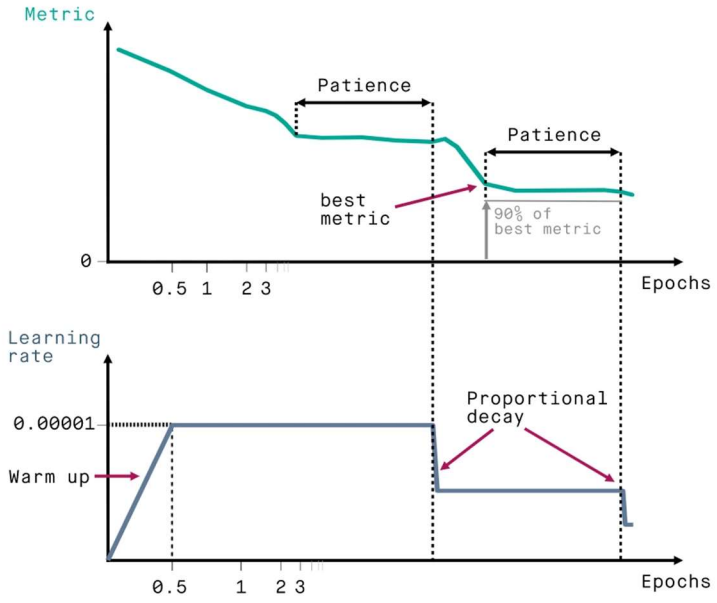
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

Stop training if.
$Ev(t) \geq Ev(t-1) \geq \ldots \geq Ev(t-p+1)$

Where:

$Ev(t)$ is the validation error at epoch t.
p is the patience parameter, indicating the number of epochs to wait before early stopping if the validation error doesn't improve.

4.9) ReduceLROnPlateau



One useful method in the toolbox of neural network training strategies is the ReduceLROnPlateau callback [23]. Its goal is to dynamically modify the training rate in response to visible modifications in a selected statistic, usually validation accuracy or validation loss. Throughout training, this callback maintains focus on the designated statistic and steps in when it notices a performance plateau or stagnation. The learning rate is lowered by a predetermined factor when such a circumstance is identified, suggesting that the model could be failing to converge efficiently or being trapped in a local minimum. The callback helps the model navigate difficult optimization landscapes more successfully by gradually decreasing the learning rate in response to stagnant performance. This may allow the model to avoid suboptimal solutions and converge to a better minimum. More stable training dynamics are encouraged by this adaptive learning rate modification, which may enhance model performance and generalization. Furthermore, the ReduceLROnPlateau callback provides further versatility by allowing for the configuration of parameters like the factor parameter, which sets the amount of the learning rate decrease, and the

patience parameter, which chooses how many epochs to wait before changing the learning rate. In general, the ReduceLROnPlateau callback helps to train neural networks more effectively and efficiently, improving their capacity to recognize intricate patterns and extrapolate to previously unknown material.

We understand that choosing hyperparameters like learning rate, batch size, and dropout rate is crucial to determining how well our CNN model performs. The choice of these hyperparameters is not random; rather, it necessitates an in-depth understanding of their impact on the convergence, generalization, and overall effectiveness of the model. In this study, we used a manual technique for hyperparameter fine-tuning, guided by experience and practical insights, instead of automated search techniques. This choice was motivated by the understanding that although automated search methods are computationally efficient, they can miss domain-specific subtleties and do not completely understand the complexities of our classification problem. We used our domain knowledge to manually choose the model's hyperparameters, adjusting the model's arrangement to the particulars of our dataset and classification problem. We were able to carefully tune the hyperparameters using our customized method, taking into factors like dataset size and complexity. Manual hyperparameter tuning gives us more control and insight into the behavior of the model, allowing us to make well-informed decisions that maximize performance and generalization, even if it may require more work and iteration. By using a systematic strategy, we want to fully realize the potential of our CNN model, making it suitable for practical use and increasing the standard for animal classification research.

We optimized the learning rate which is critical hyperparameter for determining the magnitude of parameter updates during training through repeated experimentation and thorough testing. Upon analyzing many values, it was determined that 0.001 was the best option since it could provide steady convergence and stable training dynamics. Our dedication to enhancing the accuracy and dependability of our CNN model for animal classification is demonstrated by this rigorous selection procedure.

```
Adam(learning_rate=0.001)
```

We carefully selected a value of 32 for the batch size, a crucial parameter that determines how many samples are handled in each training cycle. After giving it some thought, the decision was taken with the goal of finding a careful balance between model convergence and computing efficiency. The training parameters remain consistent, and the available computational resources may be efficiently utilized with a batch size of 32. We ensure robust model convergence and computational tractability in the training process by processing a reasonable amount of data every iteration. This strategy makes the most of our CNN model's training regimen while maintaining the stability and quality of the training process. It also ensures that computing resources are used efficiently. By carefully choosing the batch size, we can maximize the training process and strike the ideal balance between model convergence and computational efficiency, which improves the overall performance and dependability of our classification framework.

```
model.fit(X_train,y_train,batch_size=32,epochs=20,callbacks=[reduce_lr,early_stopping],validation_data=(X_test,y_test))
```

We present in detail the Convolutional Neural Network (CNN) architecture that forms the foundation of our system for image classification. Several convolutional layers are carefully arranged in this carefully constructed architecture to extract complex information from input images. Furthermore, pooling layers are used for spatial down sampling, which makes it easier to extract important information at different sizes. The fully linked layers that handle feature aggregation and final classification serve to complement these. Each layer is painstakingly built to extract the rich and intricate information contained in animal photos, enabling the model to identify minute details and patterns that are essential for accurate classification with unmatched Adaptability and accuracy.

```
model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),activation="relu",input_shape=(224,224,3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2))

model.add(Conv2D(64,kernel_size=(3,3) ,activation="relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2))

model.add(Conv2D(128,kernel_size=(3,3), activation="relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2))

model.add(Conv2D(256,kernel_size=(3,3), activation="relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2))


model.add(Flatten())

model.add(Dense(64,activation="relu"))
model.add(BatchNormalization())
model.add(Dropout(0.1))
model.add(Dense(1,activation="sigmoid"))
```

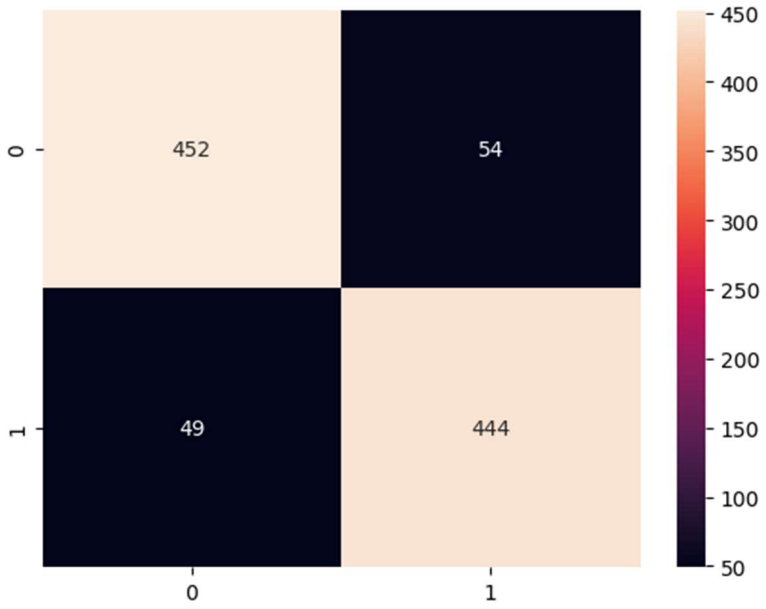
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896

 batch_normalization (BatchN  (None, 222, 222, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496

 batch_normalization_1 (Batc  (None, 109, 109, 64)     256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 52, 52, 128)       73856

 batch_normalization_2 (Batc  (None, 52, 52, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 24, 24, 256)       295168

 batch_normalization_3 (Batc  (None, 24, 24, 256)      1024
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 12, 12, 256)      0
 2D)

 flatten (Flatten)           (None, 36864)             0

 dense (Dense)               (None, 64)                2359360

 batch_normalization_4 (Batc  (None, 64)               256
 hNormalization)

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 2,750,017
Trainable params: 2,748,929
Non-trainable params: 1,088
_________________________________________________________________
```

5) Model Evaluation

To determine how well the CNN model performs in terms of classifying both domestic and wild animals from input visual data, performance evaluation is crucial. We provide the methods and metrics in this part for assessing the trained model's performance. The model's performance was assessed once training was finished using a different test dataset from the training dataset. To ensure that the model was evaluated on hypothetical samples that were typical of actual situations, the test dataset was a subset of the original picture data. Using the test dataset, a confusion matrix was created to show how well the model classified animals into various groups. The confusion matrix sheds light on any misclassifications or confusion across classes, as well as how effectively the model classifies instances of each class. We were able to assess the model's generalization abilities to new data and practical situations by testing it on a different test dataset. Important information on the model's adaptability, functionality, and dependability in real-world applications has been gathered from this testing procedure.

To illustrate how well the CNN model classified animals into various groups, a confusion matrix was created. The confusion matrix allows for a thorough examination of classification mistakes and model performance across classes by providing a tabular representation of true positive, true negative, false positive, and false negative predictions. Using a confusion matrix made it easier to analyze the model's classification performance in detail for binary animal categories. This analysis tool gave specific insights into how well the model classified animals into the binary classes. The model's ability to discriminate between domestic and wild species was thoroughly explained by carefully examining true positive, true negative, false positive, and false negative predictions. Such detailed analysis adds to a better understanding of the model's classification skills by providing insightful information about the model's advantages and possible areas for improvement.

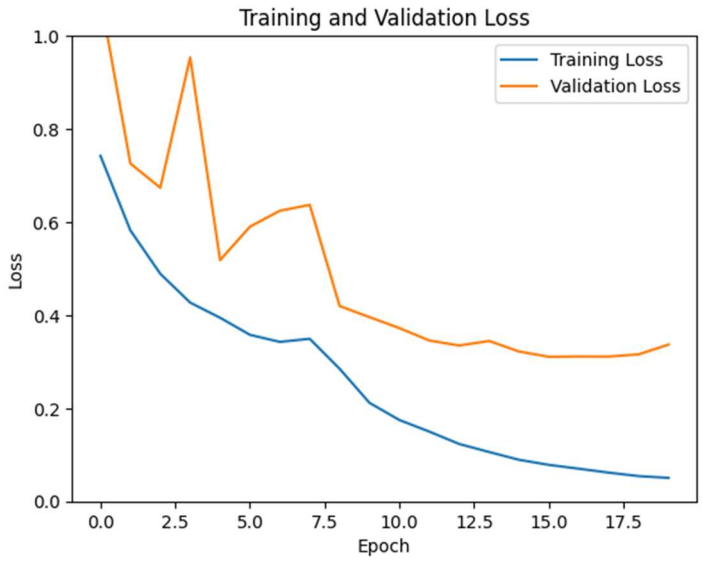
5.1) Confusion Matrix



We were able to identify several important measures that provided insight into the model's classification performance by examining the confusion matrix. When a true positive result of 452 is present, it indicates that the model accurately classified domestic species as such, indicating high classification accuracy. In the same way, a true negative value of 444 indicates cases in which the model correctly identified wild animals as such, demonstrating its strong ability to distinguish non-domestic categories. With a score of 49, false positives, on the other hand, draw attention to cases in which wild species were mistakenly categorized as domestic—a type one error that might have serious consequences. On the other hand, false negatives, which have a value of 54, represent situations in which domestic species were mistakenly identified as wild—a type two error that has its own difficulties. Specifically, type one mistakes are particularly significant in our case since misclassifying a wild animal as domestic might have life-threatening consequences, highlighting how crucial proper classification is in practical applications.

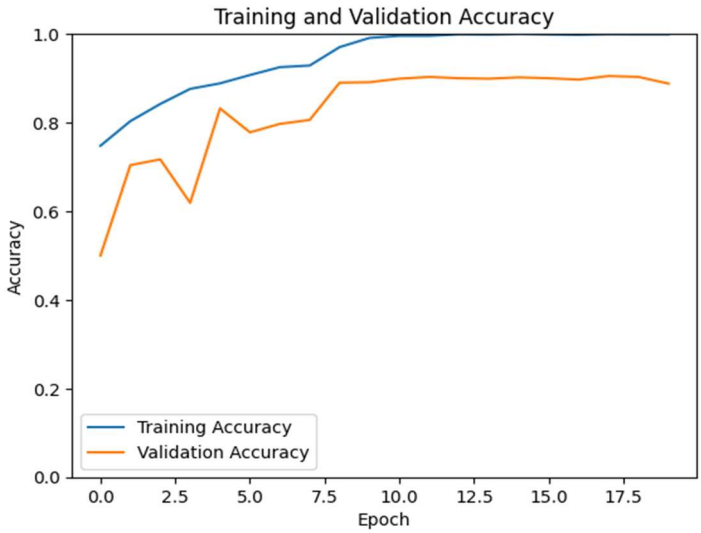
5.2) Loss Graph

Important identifiers of the model's performance on never-before-seen data were the accuracy measures and the loss computations. These validation measures confirmed the model's effectiveness outside of the training dataset by offering insightful information about how well it generalizes to new and untested situations. Interestingly, the validation accuracy showed a steady trend that mirrored the training accuracy, highlighting the model's strong generalization abilities. This consistent agreement between training and validation accuracy not only confirms the model's robustness but also highlights its ability to correctly classify animal classes in a variety of datasets, providing assurance about its usefulness in real-world applications.



5.3) Accuracy Graph

In addition to the accuracy assessments, the loss measures were equally important for assessing the model's performance on unobserved data sets. These loss indicators gave rise to a thorough evaluation of the model's predictive skills, revealing how well it can reduce mistakes and disparities in its forecasts. Specifically, the validation loss functioned as an essential reference point to assess the model's flexibility and capacity to generalize. The model's stability and dependability in categorizing animal categories across a variety of datasets is further supported by the observed consistency between the training and validation losses. This balanced match between training and validation loss enhances the model's trustworthiness and highlights how well it can identify complex patterns and characteristics that are crucial for precise classification tasks.



Result & Discussion

The results obtained from utilizing the CNN model to classify animals into separate domestic and wild categories are systematically revealed in the results and discussion section. After these data are presented, a thorough investigation uncovers the wider relevance of the study's findings by researching into the implications and hidden insights derived from the observed results.

The following important insights show that the CNN model performed remarkably well in dividing animals into domestic and wild categories: The model showed a high rate of accuracy of 89%, successfully categorizing a sizable percentage of cases into the appropriate classes domestic and wild. During training, the model's loss function showed a tendency toward decrease throughout the epochs, indicating efficient learning and convergence. Still after utilizing overfitting preventing methods such as dropout, batch normalization, early stopping, our model has several overfittings in performance.

The findings highlight the effectiveness of deep learning methods in automated animal classification tasks, especially CNN models. The CNN model's excellent accuracy and resilience show that it has practical uses in animal monitoring, pet identification, and environmental protection. The effective division of animals into domestic and wild groups provides the groundwork for the creation of computer vision systems that can reliably and accurately recognize a wide range of animal species.

Future research directions might include different deep learning architectures, creative augmentation techniques, and improvements to the current dataset to improve model performance and mitigate any drawbacks. This planned approach aims to improve and extend the model's existing capabilities. Furthermore, the integration of real-time monitoring systems that is planned, followed by their deployment in real-world environments, show promise as application trajectories that will allow the created system to be practically implemented in real-world scenarios.

Accurately classifying animals into separate domestic and Wild classifications provides strong foundation for the advancement of computer vision systems. It is intended for these systems to be able to recognize and classify a wide variety of animal species accurately effectively. With the development of complex algorithms made possible by such a thorough categorization framework, improved species detection and classification in a variety of environmental settings will become possible.

By providing a summary of the predictions made by our CNN model, we are hoping for some understanding of the significant achievements as well as the inherent difficulties that we had while attempting to classify. It proceeds through the analysis of four different images, demonstrating the algorithm's ability to correctly identify animal species in three cases, contrasted with one mistake that occurs in testing phase. Even with the extra cautious use of early stopping and dropout strategies, this misclassification serves as an actual reminder of our model's optimization process' iterative nature. It is a moving call to action that emphasizes how crucial it is to keep improving to strengthen and further enhance the model's classification abilities. Here is a glimpse of how the model performs on unseen images.

Uploaded Image.

Proceed to Classification

Prediction: Domestic (Probability: 0.99)

Uploaded Image.

Proceed to Classification

Prediction: Wild (Probability: 0.81)

Uploaded Image.

Proceed to Classification

Prediction: Wild (Probability: 1.00)

Uploaded Image.

Proceed to Classification

Prediction: Domestic (Probability: 0.77)

Conclusion

In conclusion, our research provides insightful information about the potential use of deep learning techniques, particularly Convolutional Neural Networks (CNNs), for the classification of animals. By careful analysis, we have outlined the details of our approach, including the architectural subtleties and training protocols used to create a reliable CNN model. We have successfully distinguished between domestic and wild animal species using this model, which is a strong tool that we have developed through a thorough study of our novel methodology.

Our in-depth analysis of the model's performance highlighted how well it classified animals from visual data. By utilizing validation measures like accuracy and loss in conjunction with visualizations like confusion matrices, we were able to obtain critical insights into the model's classification performance as well as regions that need more improvement. This comprehensive study acted as a compass for future optimization efforts in addition to validating the model's efficacy.

Although our research showed remarkable precision in classifying animal species in certain cases, it also revealed underlying difficulties and areas that may be improved. The discovery of misclassifications highlights the constant search for optimization and serves as an irritating reminder of the iterative nature of model refining. By actively using early stopping and dropout techniques, we strengthened the model's robustness and generalization abilities. These steps not only reduce overfitting but also provide the model with more flexibility to suit different datasets and real-world situations, demonstrating our dedication to improving the performance of our classification system.

Following studies might focus on investigating different deep learning architectures, creative augmentation techniques, and improvements to the current dataset. The goal of this active research is to improve animal classification systems to previously unheard-of levels of accuracy and dependability. Furthermore, the incorporation of automated animal classification has significant potential across many fields, such as wildlife surveillance, pet recognition, and environmental conservation endeavors. By combining this capability with real-time monitoring systems, it is possible to further enhance it and eventually use automatic categorization frameworks in real-world scenarios that are authentic. These kinds of projects not only represent a paradigm change in biodiversity study, but they also portend revolutionary consequences for global environmental management and conservation initiatives.

To summarize, our study is a noteworthy advancement in the dynamic field of computer vision, demonstrating the revolutionary potential of deep learning techniques in automating intricate tasks like classifying animals. Through disentangling the complex relationship between wildlife research and artificial intelligence, we open fresh possibilities for creative solutions that advance both international conservation efforts and our comprehension of the natural world. This work brings in a new age of technical innovation that promises to fundamentally change how we interact with and conserve the rich and diverse ecosystems of our world, acting as an inspiration for creative discovery.

In conclusion, with an incredible false-positive rate of 89%, our model has shown promise in correctly differentiating between two unique groups; yet there is room for improvement. This noteworthy accomplishment highlights the enormous potential that deep learning approaches have in the field of animal classification, opening doors for useful applications in fields like agriculture, conservation, and wildlife observation. As we commemorate this achievement, we must also recognize that improvement is still a continuous process. Realizing how important it is to keep improving, we turn our attention to enhancing the model's functionality and performance in practical situations. To improve the model's accuracy and effectiveness, a concentrated effort must be made to increase its dataset and enhance repeatedly.

Bibliography

1. Chen, R., Little, R., Mihaylova, L., Delahay, R., & Cox, R. (2019). Wildlife surveillance using deep learning methods. *Ecology and evolution*, *9*(17), 9453-9466.
2. El Abbadi, N. K., & Alsaadi, E. M. T. A. (2020, April). An automated vertebrate animals' classification using deep convolution neural networks. In 2020 International Conference on Computer Science and Software Engineering (CSASE) (pp. 72-77). IEEE.
3. Komarasamy, G., Manish, M., Dheemanth, V., Dhar, D., & Bhattacharjee, M. (2021). Automation of Animal Classification Using Deep Learning. In *Integrated Emerging Methods of Artificial Intelligence & Cloud Computing* (pp. 419-427). Cham: Springer International Publishing.
4. Zeng, P. (2021, December). Research on similar animal classification based on CNN algorithm. In *Journal of Physics: Conference Series* (Vol. 2132, No. 1, p. 012001). IOP Publishing.
5. https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092
6. https://www.geeksforgeeks.org/introduction-deep-learning/
7. https://towardsdatascience.com/model-parameters-and-hyperparameters-in-machine-learning-what-is-the-difference-702d30970f6
8. https://www.superannotate.com/blog/image-classification-basics
9. https://medium.com/deep-learning-demystified/loss-functions-explained-3098e8ff2b27
10. https://medium.com/codersarts/understanding-optimizers-in-deep-learning-exploring-different-types-88bcc44ff67e
11. https://www.superannotate.com/blog/overfitting-and-underfitting-in-machine-learning
12. https://www.geeksforgeeks.org/introduction-convolution-neural-network/
13. https://anhreynolds.com/blogs/cnn.html
14. https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d
15. https://www.nomidl.com/deep-learning/what-is-relu-and-sigmoid-activation-function/
16. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
17. https://paperswithcode.com/method/max-pooling
18. https://pub.aimind.so/binary-crossentropy-with-keras-61a227ed5bcd
19. https://www.sevenmentor.com/dropout-layers
20. https://medium.com/@rahuljain13101999/why-early-stopping-works-as-regularization-b9f0a6c2772
21. https://www.linkedin.com/pulse/getting-know-adam-optimization-comprehensive-guide-kiran-kumar/
22. https://pysource.com/2022/10/07/flatten-and-dense-layers-computer-vision-with-keras-p-6/
23. https://wiki.cloudfactory.com/docs/mp-wiki/scheduler/reducelronplateau