Pets Identification and their characteristic Using Convolutional Neural Networks

Vo Dinh Nghi

Ho Chi Minh City University of Education and Technology

1 Vo Van Ngan, Thu Duc City, Vietnam

19146097@student.hcmute.edu.vn

Abstract

Deep Learning algorithms are designed in such the way that they simulate the function of the human cerebral mantle. These algorithms are representations of deep neural networks aka neural networks with many hidden layers. Convolutional neural networks are deep learning algorithms which will train large datasets with voluminous parameters, in variety of 2D images as input and convolve it with filters to provide the required outputs. In this article, CNN algorithm is used to train a model, which aims to detect 15 pets to teach children from 3-6 years old. The datasets is generated by collecting images of these pets, which is more than 28,867 images, and split them into train and test set to train the model. The accuracy achieved is more than 80% when finish training the model and its performance on real time detection is good.

I. Introduction

Machine learning (ML) has recently gained a lot of traction in research, and it's now used in a number of applications like text mining, spam detection, video recommendation, image categorization, and multimedia idea retrieval. Deep learning (DL) is one of the most widely used machine learning algorithms in these applications. Deep Learning is also known as representation learning (RL). The emergence of new studies in the disciplines of deep and distributed learning is due to both the unpredictability of data availability and the incredible development made in hardware technologies, such as High Performance Computing (HPC).

Artificial Neural Networks (ANNs) are computing systems that are substantially influenced by organic nerve systems (such as the human brain). ANNs are primarily made up of a large number of interconnected computational nodes (also known as neurons) that work together in a dispersed manner to learn from the input and optimize the final output.

Convolutional Neural Networks (CNNs) are similar to classic artificial neural networks (ANNs) in that they are made up of neurons that learn to optimize themselves. Each neuron will still receive an input and conduct an action (such as a scalar product followed by a non-linear function), which is the foundation of innumerable artificial neural networks. The entire network will still express a single perceptual scoring function from the input raw picture vectors to the final output of the class score (the weight).

The last layer will contain the loss functions related with the classes, and all of the standard ANN tips and tactics will still apply.

II. Methods

A. Network Architecture

The network's input is a normalized lung image patch with zero mean and unit variance. A convolutional layer with a kernel size of 5x5 pixels and 16 output channels is the first layer. A max pooling layer with a kernel size of 3x3 is the second layer. The next three layers are fully connected neural layers, each containing 100-50-5 neurons. In comparison to other CNN applications, our architecture just has one convolutional layer. There are no evident big scale or high level elements for the network to learn because the input patches are textures-like images.

A variety of strategies are used to improve and speed up neural network training. The learning parameters associated with these learning techniques are set based on standard values or empirical studies. Each layer of the CNN network is described in detail in the sections that follow.

B. Convolutional Layer

CNN relies heavily on convolutional neuron layers. One or more 2D matrices (or channels) are used as input to the convolutional layer in image classification tasks, and numerous 2D matrices are generated as output. There may be a difference in the

$$A_j = f(\sum_{i=1}^{N} I_i * K_{i,j} + B_j)$$

number of input and output matrices. The procedure for calculating a single output matrix is as follows:

To begin, each input matrix is twisted with the kernel matrix Ki ,j. The sum of all convoluted matrices is then computed, and each member of the resulting matrix is given a bias value Bj. Finally, each member of the previous matrix is subjected to a non-linear activation function f, resulting in one output matrix Aj.

A local feature extractor is a set of kernel matrices that extracts regional features from the input matrices. The learning procedure's goal is to find sets of kernel matrices K that extract good discriminative features that may be utilized to classify images. The kernel matrices and biases can be trained as shared neuron connection weights using the back propagation approach that improves neural network connection weights.

C. Pooling Layer

In CNN, the pooling layer is critical for reducing feature dimension. Pooling methods should be used to merge the neighbouring elements in the convolution output matrices in order to reduce the number of output neurons in the convolutional layer. Max-pooling and average-pooling are two common pooling methods. In this paper, a maxpooling layer with a 2x2 kernel size selects the highest value from the input matrix's four neighboring members to generate one element in the output matrix. The gradient signal must only be transmitted back to the neurons that contribute to the pooling output during the error back propagation procedure.

D. ReLU Activation

The neuron activation function in ANN is non-linear transfer functions. The sigmoid function f(x) = 1/(1 + e x) and the hyperbolic tangent function f(x) = tanh are two examples of regularly used activation functions (x). Both sigmoid and hyperbolic tangents are saturating non-linear functions, meaning that as the input grows, the output gradient approaches zero.

According to several recent studies, using a non-saturating non-linear function such the rectified linear function f(x) = max(0, x) (ReLU) in CNN applications enhances both learning speed and classification performance . The ReLU activation function is used in the convolutional layer of our

CNN model. The ReLU activation function enhances classification performance by 2.5 percent and the network converges considerably faster than the sigmoid activation function, according to testing data.

E. Dropout function

During training, the drop-out method is used to increase performance by randomly deactivating neurons in each layer. At the start of each training iteration, a drop-out map with the same size as the neurons in each layer is randomly initialized to mark the on or off state of the relevant neuron.

During the training iteration, the neurons with off states are deleted from the network by inhibiting the neuron's activation signal forward propagation and error signal backward propagation. It's the same as switching between different models for each learning iteration, resulting in a large number of different models being trained at once. All neurons are turned on during testing, however the activation signal is attenuated to the probability of average turn on rate

$$\phi : \mathcal{S} \mapsto \mathcal{T}$$

during training.

During training, we picked a drop-out rate of 0.5, and during testing, the neuron activation output was multiplied by 0.5.

F. Data Augmentation

We propose investigating and comparing alternative data augmentation approaches for image and video categorization. It is well understood that the more data a machine learning algorithm has, the more effective it may be. Even when the data is of poor quality, algorithms can outperform the original data set if the model can extract relevant information from it.

DA refers to any method that artificially inflates the original training set with label preserving transformations and can be represented as the mapping:

Where, S is the original training set and T is the augmented set of S. The artificially inflated training set is thus represented as:

Where, S ′ contains the original training set and the

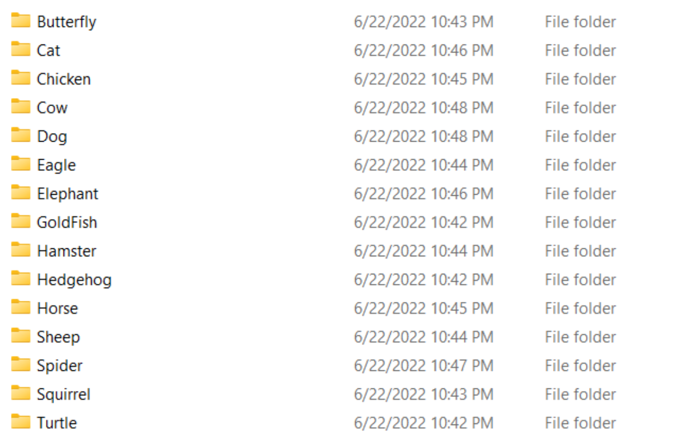
$$\mathcal{S}' = \mathcal{S} \cup \mathcal{T}$$

respective transformations defined by φ. Note the term label preserving transformations refers to the fact that if image x is an element of class y then φ(x) is also an element of class y.

III. Experimental Result

A. Datasets

I create my own datasets to use for training CNN Deep Learning model. They are mostly collected on Google and Kaggle because it’s hard to find good image about pets in my area. I saved all images of each species into one folder and it will be called in my Python program using ‘os’ library.

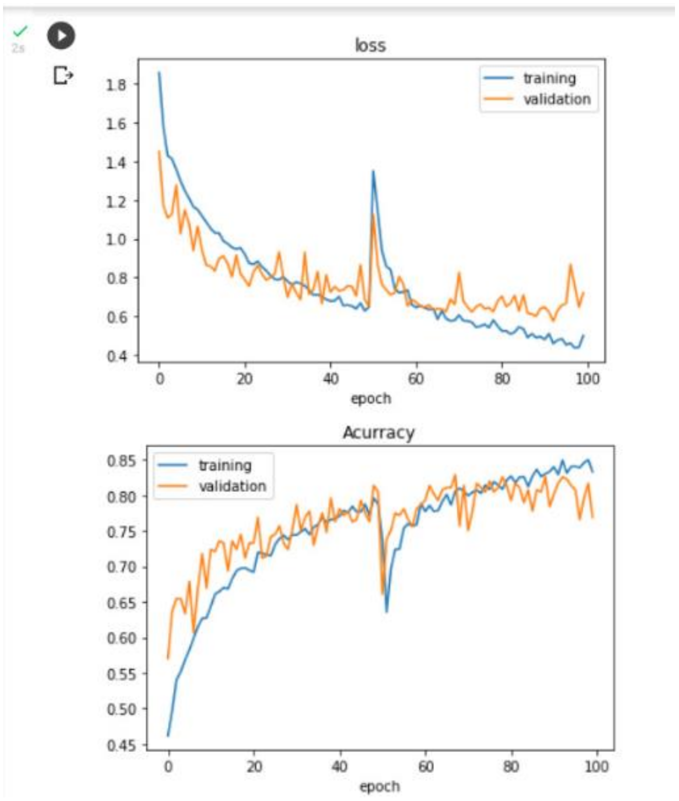


Butterfly	6/22/2022 10:43 PM	File folder
Cat	6/22/2022 10:46 PM	File folder
Chicken	6/22/2022 10:45 PM	File folder
Cow	6/22/2022 10:48 PM	File folder
Dog	6/22/2022 10:48 PM	File folder
Eagle	6/22/2022 10:44 PM	File folder
Elephant	6/22/2022 10:46 PM	File folder
GoldFish	6/22/2022 10:42 PM	File folder
Hamster	6/22/2022 10:44 PM	File folder
Hedgehog	6/22/2022 10:42 PM	File folder
Horse	6/22/2022 10:45 PM	File folder
Sheep	6/22/2022 10:44 PM	File folder
Spider	6/22/2022 10:47 PM	File folder
Squirrel	6/22/2022 10:43 PM	File folder
Turtle	6/22/2022 10:42 PM	File folder

Each folder will include 100 photos and some species can up to 3000 photos and it will be divided into training and test set using train_test_split in sklearn library.

Totally there’s 28,867 used to implement the project. The test size is set at 0.2, meaning that 23,094 images will be used for training, and the rest will be used as test images.

B. Model evaluation



As can be seen, the model is going through overfitting problem. Its performance is good enough to be used for prediction

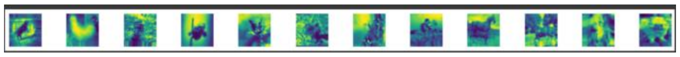
C. Confusion Matrix

A confusion matrix is a technique for summarizing the performance of a classification algorithm.

Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset.

Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.

D. Image Preprocessing



Go through many steps like:

- Light balance for photos

- Scale the image to a value of 0-1

- Take the processed image and re-insert the training and test sets

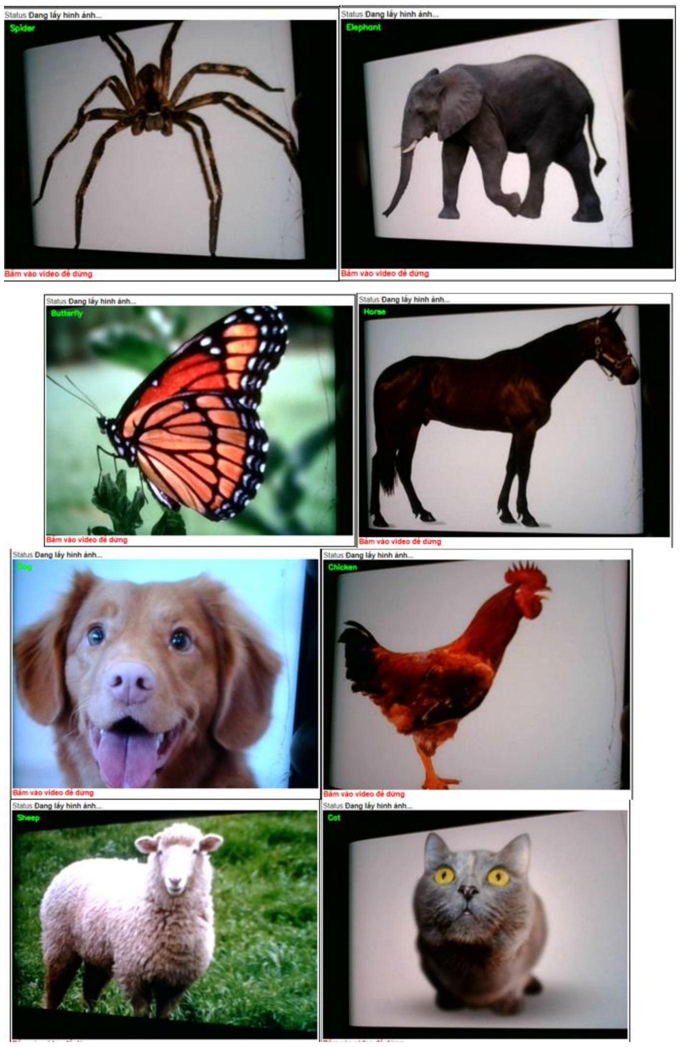
- Reformat to format (32, 32, 1)

- Rotate, shift-L/R, image zoom

- Requesting data generator to create more images

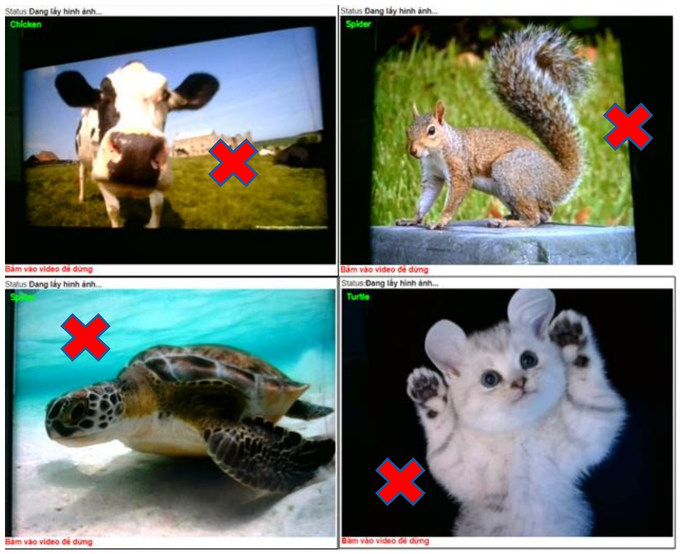
- Convert to one-hot and convert from numberical to array to be trainable

E. Test Result



I conducted a real-time test through the laptop's webcam. As seen above, those images are predicted by the CNN model. The result was better than I expected because all the predictions were correct.

But beside that there some of the predictions are wrong:



- Drawback

After completing the model, I realized that there are still many drawback such as a lot of photos that are mistakenly identified with other animals.

And Validation accuracy = ~80% is too low, so the model is not optimal

- Cause of the Drawback

Some species have similar colors and patterns

Poor image quality, color is close to color

Parameters such as loop, CNN layer

- Ways to Overcome

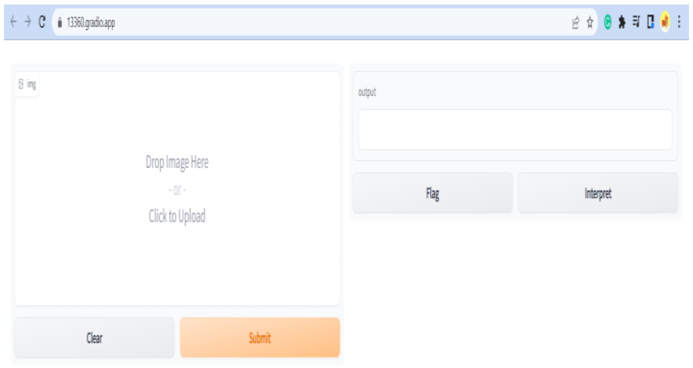
Choose clear, large-sized animal photos

Increase the amount of image data of each species

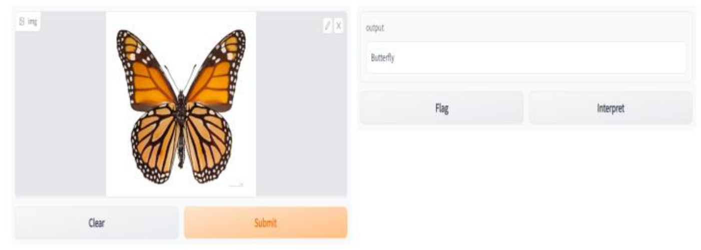
Changing parameters and rerun more times

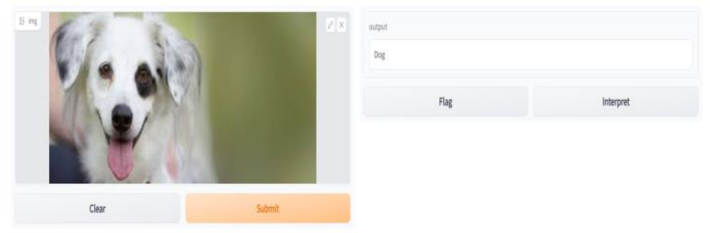
- Web App

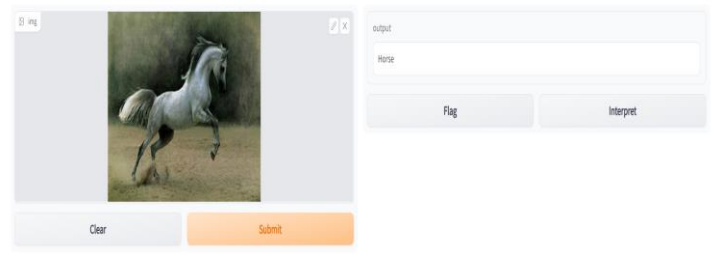
Gradio is a Python toolkit for fast creating web-based machine learning demos, data science dashboards, and other types of web apps. These web apps may be launched from everywhere you use Python (jupyter notebooks, colab notebooks, Python terminal, and so on) and quickly shared with anyone using Gradio's auto-generated share links.



After launching the interface successfully, I did some test on it and it show a wonderful result.







IV. Conclusion

Through the project, I know how to train a CNN deep learning model and how to improve its performance using data augmentation technique. I also had chance to know more about how we can apply an AI model in real life and how to conduct them online through the Internet. In this project, there're also limitation. The output model performance is not as expected and I wasn't able to complete all the requirement (like show their Characteristics). This is the point I will need to improve in the future. Anyway, this project is very helpful and it will continue to help me in the future.

Reference

[1] Luke Taylor, Geoff Nitschke: Improving Deep Learning using Generic Data Augmentation. 1708.06020

[2] Andrew G. Howard: Some Improvements on Deep Convolutional Neural Network Based Image Classification. arXiv:1312.5402