Universitat de Lleida

TREBALL FINAL DE MÀSTER

Estudiant: **Alex Ghenghiu**

Titulació: Màster en Enginyeria Informàtica

Títol de Treball Final de Màster: Monitorització de les activitats d'animals de granja mitjançant intel·ligència artificial

Director/a: Ramón Béjar Torres, Carles Mateu Piñol

Presentació

Mes: Juliol

Any: 2019

MONITORING OF THE FARM ANIMAL ACTIVITIES USING ARTIFICIAL INTELLIGENCE

Alex Ghenghiu
Computer Engineering Master UDL
xalexgg94@gmail.com

July 2019

Abstract

An important factor for farms earnings is to detect animals illness on its early stages. The sooner the disease is detected, the less cost it takes to treat it. One way to determine if a **cow** isn't healthy is that the animal won't drink water. So, it would be interesting to have a system to determine which of the **cows** from the farm are not drinking water. Knowing this, we have implemented an object classifier using **Convolution Neural Networks (CNNs)** with **Keras**. The motivation of this project is understanding clearly about deep learning, particularly CNNs, and put in on real life. Therefore, we also tunned the hyper parameter of each models such as learning rate, batch size, and number of epochs. In addition, we also used techniques to optimize networks, acting as activation function, dropout and max pooling. During this process, several models have been generated in order to observe the relationship between number of layers, input data and accuracy.

1 Introduction

To maximize earning on farms, it's important to detect animal illness on its early stages, as detecting it as soon as possible makes it cheaper to treat. A common behaviour when a **cow** isn't healthy is that the animal won't drink water. So, to detect this behaviour, we have devised a system to record **cattle-drinkers** from a overhead perspective. To analyse the video, we have implemented an object classifier using **Convolution Neural Networks (CNNs)** with **Keras**. This classifier is able to distinguish between **cow** and **no-cow** with an accuracy of **82.83%**. So, when a **cow** gets closer to the **cattle-drinker**, we infer that the animal is drinking water, so it's a healthy animal. Finally, we need to build a record of all the **cows** that have been drinking during the day. To do so, we take a look at the **ear tags** in order to identify the animals.

We mention some related work in Section 2. Dataset is remarked in Section 3. Section 4 show what exactly we did, how is our CNN model in detail and how to easily improve it. Content in Section 5 is our results. In ending, we also present some conclusion and future work in Section 6

2 Related Work

Visual recognition is a relatively trivial task for human, but still challenging for automated image recognition systems due to complicated and varied properties of images [20]. Each object of interest can alter an infinite number of different images, generated by variations in position, scale, view, background, or illumination. Challenges become more serious in real-world problems such as wild animal classification from automatic trap cameras, where most captured images are in imperfect quality. Therefore, for the task of image classifying automation, it is important to build models that are capable of being invariant to certain transformations of the inputs, while keeping sensitivity with inter-class objects [11].

Firstly proposed by LeCun et al. [23], CNNs have been showing great practical performance and been widely used in machine learning in the past recent years, especially in the areas of image classification [21], [22], [12], [9], [18], speech recognition [13], and natural language processing [16], [17]. These models have made the state-of-the-art results that even outperformed human in image recognition task [14], due to recent improvements in neural networks, namely deep CNNs, and computing power, especially the successful implementations of parallel computing on graphical processing units (GPUs), and heterogeneous distributed systems for learning deep models in large scale such as TensorFlow [19].

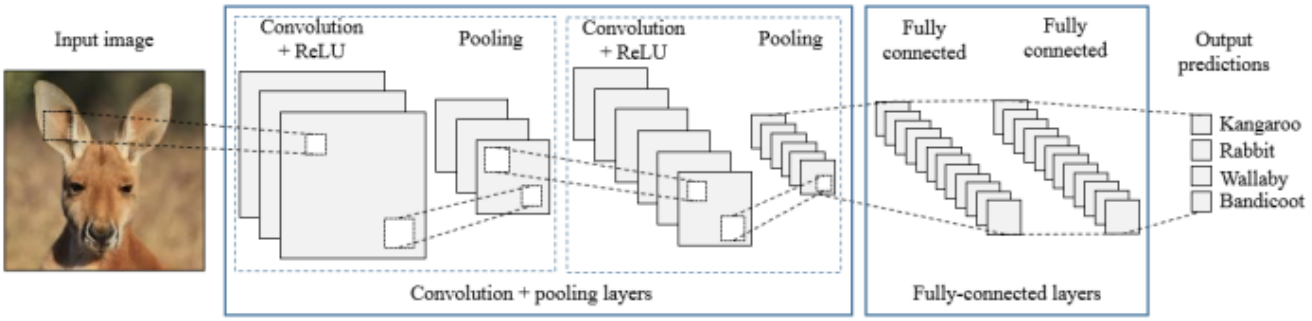

Figure 1: Illustration of a typical convolutional neural network architecture setup

CNNs are basically neural network-based learning models specifically designed to take advance the spatial structure of input images, which are usually in 3-dimensional volume: width, height, and depth (the number of color channels). As illustrated in Figure 1, a CNN is essentially a sequence of layers which can be divided into groups each comprising of convolutional layer plus non-linear activation function, usually the Rectifier Linear Unit (ReLU) [9], and pooling layer, mostly max pooling; ended by several fully-connected layers where the last one is the output layer with predictions. In the standard neural networks, each neuron is fully connected to all neurons in the previous layer and the neurons in each layer are completely independent. When applied to high dimensional data such as natural images, the total number of parameters can reach millions, leading to serious overfitting problem and impractical to be trained. In CNNs, by contrast, each neuron is connected only to a small region of the preceding layer, forming local connectivity. The convolution layer computes the outputs of its neurons connected to local regions in the previous layer, the spatial extent of this connection is specified by a filter size. In addition, another important property of CNNs, namely parameter sharing, dramatically reduces the number of parameters and so does computing complexity. Thus, compared to regular neural networks with similar size of layers, CNNs have much fewer connections and parameters, making them easier to train while their performance is slightly degraded [9]. These three main characteristics – spatial structure, local connectivity and parameter sharing – allow CNNs converting input image into layers of abstraction; the lower layers present detail features of images such as edges, curves and corners, while the higher layers exhibit more abstract features of object.

Apart from using more powerful models and better techniques for preventing overfitting, the performance of data-driven machine learning approaches depends strictly on the size and quality of collected training datasets. Real-life objects exhibit considerable variability, requiring much larger training sets to learn recognizing them [9].

3 Dataset

The model has been trained using two datasets:

- Kaggle dataset [4]
- WSID-100 dataset [1]

For this model, the universe it is made up of two things: **cows** and **no-cows**. So, for the training part, two folders have been created. One with **cow** images and the other one with **no-cow** images. The criteria of the **no-cows** image selection is to cover all the things that could be present on a farm that are not a cow (people, cars, tractors, other animals...).

Following the suggestion of the Deep Learning with Keras book [10], to improve model's accuracy, the technique of **data augmentation** can be used. So, given an image, it is resized to 50x50 as training the convolutional neural network requires to have images of same size. For data augmentation, instead of just saving the resized image, for each image another three images are generated by flipping the original one (horizontally, vertically and both at the same time) See figure 2.

For the testing part, Google images with **Download All Images**[2] plugin has been used.

Figure 2: Output of resize and **data augmentation** technique

The final training dataset is composed by **72.717** images, **16.993 cows** and **55.724 no-cows**

4 Method

4.1 Overview

Convolution Neural Networks (CNNs) are used to recognize **cows**. In CNNs approach, the input image is convolved through a filter collection in the convolution layers to produce a feature map. Each feature map is then combined to fully connected network, and the input image is recognized as belonging to a particular class-based the output of softmax algorithm. There are two main reasons CNNs were chosen for this approach:

- CNNs is the most popular network model among the several deep-learning model available. Understanding CNNs helps to develop researching deep-learning career in the future.
- They have proven that they are the go-to method for any type of prediction problem involving image data as an input.

During this research process, several models were generated. All of them result of trying new network layer structure and tunning hyper parameters (learning rate, batch size and number of epochs) in order to improve the final accuracy.

Finally, following the suggestion of the Deep Learning with Keras book [10], the network configuration that better results had is the following:

conv + conv + maxpool + dropout + conv + conv + maxpool + dropout

This network has four convolution layers and two fully connected (FC) layers (512 and 2 hidden units). In the first convolutional layer, we had 32 filters with kernel size is 3x3, padding is 'same' and value of input shape is (50,50,3) since input image is 50 x 50 and RGB channel (three color channel). The second convolutional is the same as the first one. The third and forth are like the first two but using 64 filters instead. Each pair of convolutional layers, also comes along with max-pooling layer and dropout. Pooling setup is 2x2 with a stride of 2 to reduce the size of the receptive field and avoid overfitting. In dropout layer, a fraction of 0.25 is used.

After the convolutional layers, two fully connected (FC) layers of 512 and 2 hidden units are added. The first one is using **Rectified Linear Unit (ReLU)** as activation function, the second one, **Softmax**.

Also in all the layers, **Rectified Linear Unit (ReLU)** is used as the activation function to model non-linearity. **ReLU** is simply and make high performance.

4.2 Cow or no-cow?

As explained in section 3, the training dataset is composed by **72.717** images, **16.993 cows** and **55.724 no-cows**. All the images resized to 50x50 as for training the convolutional neural network it is required to have images of same size.

Once the images are ready, it's time to train the model. The training process, consists on assigning labels to images to tell the model this one is a cow, this other one is not a cow. In this case, as the number of images is pretty high, it is recommended to use the **GPU** version of Keras [5] as for this kind of work with images the **GPU** is better than **CPU**.

Finally, the final propose of the model. Object recognition on real-time. As explained in section 1, the idea behind this project is to detect illness on **cows**. And to detect illness we can focus on just finding which are the cows that are drinking water, so, the ones that are not, are the ones with some health problem. So, the real input to the system will be a video record of the **cattle-drinkers**.

To analyse this video records, a multi-threading environment is used. While one thread is playing the video and splitting it at a ratio of 1 frame per second, the model, running on a concurrent thread, is receiving this frames and performing the object recognition.

When a cow enters the frame (so is close to a **cattle-drinker**) we infer that the cow is drinking so is a healthy cow. The next step is to know which was this cow.

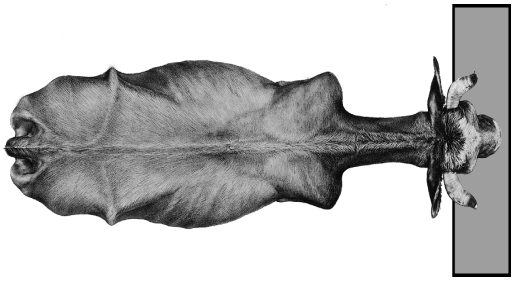

Figure 3: Expected real case of model input

4.3 Which cow?

Now, the final step is to identify the cow. To do so, the only tool right now is the ear tag (See figure 4).

Figure 4: Example of ear tag

So, given an image we need to extract the text from the **ear tag** in order to build, every day, a list of all the cows that have drunk. This technique is known as **OCR (Optical Character Recognition)**. To do so, two tools have been used:

- Tesseract [8]
- Google Cloud Vision [3]

To improve the efficiency, instead of passing the full image as in figure 4, only the **ear tag** should be passed to the **OCR**. This can be achieved by using **OpenCv** [6].

With the release of **OpenCv 3.4.2** and **OpenCv 4**, we can now use a deep learning-based text detector called **EAST**, which is based on [15].

The algorithm is called **EAST** because it's an: Efficient and Accurate Scene Text detection pipeline.

The **EAST** pipeline is capable of predicting words and lines of text at arbitrary orientations on 720p images, and furthermore, can run at 13 FPS, according to the authors.

Perhaps most importantly, since the deep learning model is end-to-end, it is possible to sidestep computationally expensive sub-algorithms that other text detectors typically apply, including candidate aggregation and word partitioning.

To build and train such a deep learning model, the **EAST** method utilizes novel, carefully designed loss functions.

So, given the figure 4, we can easily extract the **ear tags** using **OpenCv**.

Figure 5: Example of text detection with **OpenCv**

Now, given this image of only the number of the **ear tag**, we can pass it to the **OCR** and it will have more accuracy extracting the text that it would have with the original complete image.

Figure 6: Example of **OCR** output

This image was a easy one. Both **Tesseract** and **Google Cloud Vision** output is correct. But by testing both **OCRs**, **Google Cloud Vision** has turned out to have more accuracy. Also, **Tesseract** is harder to use as some of the images need a lot of preparation (resize, delete noise, change colors...) before sending them to the **OCR** in order to improve the accuracy.

4.4 How to improve it

The easiest way to train the model for better results is do it *in situ*. Knowing the real data. Train the model with the exact breed of cows that the farm would have. Also train the model with the exact things of the farm that are **no-cows**. For example, training

the model by putting **horses** on the selection of **no-cows** if in that farm would be no **horses**, would be counterproductive as from some perspectives a **horse** may look like a **cow**...
Also an important thing are the **perspectives**. If the model has been trained with only **cow** profile pictures, then if it has to predict an overhead picture of a **cow**, it may struggle.
Same as before for other factors like the illumination.

5 Results

The first accuracy obtained by the model with the network configuration explained in section 4, was **75.37%**. This first version of the model was only trained with a dataset composed by 1000 images (500 cows and 500 no-cows). Here it can be seen the effect of the **data augmentation** technique, as the final accuracy obtained by the model trained with 72.717 images (16.993 cows and 55.724 no-cows) is **82.83%**. This accuracy could be easily improved by deleting the unnecessary counterproductive data from the datasets and training the model as suggested in section 4.4.
When it comes to the **OCR** part, as explained in previous sections, **Google Cloud Vision** has given better results than **Tesseract** in terms of extracting the full clean text from the ear tags.
The full project can be checked in the following repository [7].

6 Conclusion

In this paper, were explored CNNs for object classification. This project, has not only provided a learning of technologies that are increasingly used, but also has forced me to face things I had never experienced before, leading to a great professional growth in the subject.
It's very likely that the model will struggle in the real environment, so it would need a training with real and quality data.

References

[1] Wsid-100 dataset. `http://www.multimediauts.org/dataset/WSID-100.html`.

[2] Download images plugin. `https://chrome.google.com/webstore/detail/download-all-images/ifipmflagepipjokmbdecpmjbibjnakm`.

[3] Google cloud vision. `https://cloud.google.com/vision/?hl=es`.

[4] Kaggle dataset. `https://www.kaggle.com/datasets`.

[5] Keras gpu. `https://anaconda.org/anaconda/keras-gpu`.

[6] Opencv. `https://opencv-python-tutroals.readthedocs.io/en/latest/`.

[7] Github project link. `https://github.com/alexgg94/TFM/`.

[8] Tesseract. `https://pypi.org/project/pytesseract/`.

[9] I. S. A. Krizhevsky and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *in Advances in Neural Information Processing Systems*, 2012.

[10] S. P. Antonio Gulli. *Deep Learning with Keras*. Packt Publishing Ltd, 2017. Implementing deep learning models and neural networks with the power of Python.

[11] C. M. Bishop. Pattern recognition. *Machine Learning, vol. 128, pp. 1–58*, 2006.

[12] Y. J. P. S. S. R. D. A. D. E. V. V. C. Szegedy, W. Liu and A. Rabinovich. Going deeper with convolutions. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[13] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. *in Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

[14] U. M. D. Ciregan and J. Schmidhuber. Multi-column deep neural networks for image classification. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[15] Z. et al. East: An efficient and accurate scene text detector. 2017.

[16] D. G. J. Gehring, M. Auli and Y. Dauphin. A convolutional encoder model for neural machine translation. *arXiv:1611.02344*, 2016.

[17] D. G. D. Y. J. Gehring, M. Auli and Y. N. Dauphin. Convolutional sequence to sequence learning. *ArXiv e-prints*, 2017.

[18] S. R. K. He, X. Zhang and J. Sun. Deep residual learning for image recognition. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[19] P. B. E. B. Z. C. C. C. G. S. C. A. D. J. D. M. D. M. Abadi, A. Agarwal. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2016.

[20] D. D. C. N. Pinto and J. J. DiCarlo. Why is real-world visual object recognition hard? *PLOS Computational Biology, vol. 4, no. 1, p. e27*, 2008.

[21] H. S. O. Russakovsky, J. Deng. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252*, 2015.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[23] J. S. D. D. H. R. E. H. W. H. Y. LeCun, B. Boser and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation, vol. 1, no. 4, pp. 541–551*, 1989.