Deep Learning guinea pig image classification using Nvidia DIGITS and GoogLeNet

Lukasz Zmudzinski

University of Warmia and Mazury in Olsztyn, Poland
lukasz@zmudzinski.me

Abstract. In this paper guinea pig classification using deep learning imaging methods was performed on the Nvidia DIGITS 6. Models capable of distinguishing skinny, abyssinian and crested fur types were created in the process. To increase the classification accuracy empty images (with only the background) were added to the data set. Upon evaluation, the created model recognized the animals correctly from images taken in various household backgrounds.

Keywords: deep learning · animal recognition · robotics

1 Introduction

Robotic systems are nowadays increasingly appearing in various industries. This trend is also represented in various animal care facilities like farms, daries, shelters [1,2] and more. New robotic systems are created, that fill the public space as well as connect to the personal home environment. With the growing need of automating work more software and hardware platforms are employed to increase the ease of life.

In this work, deep learning techniques were utilized to create a classification model of guinea pigs in different home environments (living room, office, corridor, etc.), to explore the possibilities of bringing such systems into the world of household animals. Creating such a model was important, to understand how machine learning algorithms would adapt to live creatures, while keeping a high accuracy of the prediction and short inference times needed in robotics.

To address these problems GoogLeNet was used. The pre-trained model connects various techniques known from Deep Learning like convolutions, pooling, adding softmax and more [3], to distinguish all the objects that are present in the image. It implements so called *Inception modules*, that range from 245 filters to 1024 in top inception modules. The consequence of this is the possibility to remove fully connected layers on top completly [4].

Gathered results, will perform as a base for future projects of animal social and care systems. Example appliances could include:

- automatic feeding and cleaning systems,
- automatic pet door management,

– illness and status detectors,
– or mobile, home robots allowing the owners to check on their pets using mobile software.

1.1 Related Works

Related works about animal classification were published in regard of wild animal monitoring [5]. The authors used convolutional neural networks to create a model from the Serengeti National Park camera-trap database snapshot containing 179683 images. Then they tested the set using popular topologies like AlexNet, VGGNet, GoogLeNet and finally ResNets.

Similarly convolutional neural networks were used to recognize 20 species common in North America [6] over a 14346 image training data set. The imagery data from motion triggered cameras was automatically segmented using the graph-cut algorithm.

Another approach was taken for classifying different animals for automated species recognition [7]. The authors enforced ScSPM (Sparse coding Spatial Pyramid Matching) [8] to extract and classify animal species over a 7 thousand image data set. After that multi-class pre-trained SVMs were applied to classify global features of animal species.

All mentioned authors follow a similar pattern when using machine learning for image classification, but none of them concern household animals. All presented projects face different problems from those, that could be encountered in a safe, indoor environment. Hence, different data acquisition techniques had to be used.

1.2 Nvidia DIGITS

DIGITS (Nvidia Deep Learning GPU Training System) was used in this project. It is an open-source project for training deep neural networks (DNNs). The software simplifies common deep learning tasks such as managing data, designing and training neural networks and monitoring performance in real time. [9, 10].

The solution comes with pre-trained models (but it allows usage of self created ones) for example:

– GoogLeNet (Inception),
– AlexNet,
– UNET,
– and more.

In this paper GoogLeNet was used as the model of choice.

2 Background / Formulation

Deep Learning (DL) is a machine learning technique growing in popularity over the past few years. It is connected to the fact that it becomes more useful than before thanks to the amount of available training data and advances in computer hardware/software [11]. It allows to create models that perform the following tasks:

- computer vision,
- speech recognition,
- natural language processing,
- recommendation systems,
- and more.

To understand how DL works, we should first define what learning actually is. A simple definition was provided by Mitchell [12] as follows:

Definition 1. *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at task in T, as measured by P, improves with experience E.*

This can apply to different kind of tasks, performance measures and experiences. In this paper we will focus on the task of object classification using computer vision.

Classification of objects is based on describing to what category a given input belongs. This can be used in robotics for tasks like delivering foods and drinks to clients by the Willow Garage PR2 robot [13]. The general rule is to create an algorithm that produces the funcion: $f : \mathbb{R}^n \to \{1, ..., k\}$, where the category is assigned when $y = f(x)$ for input x.

This paper focused on supervised learning, which means that all data set items were associated with a label (each guinea pig image was added to a specific category). In practice, it means that the algorithm knew how to classify certain objects with similary properties from the start.

2.1 Machine learning and neural networks

Articial neural networks are a subgroup of algirthms that are used for machine learning. The main idea behind them is creating artificial neurons, wchich are implemented as a non-linear function over a linear combination of input features. Each neuron generally consist of one to multiple inputs with wages, activation function, bias and one output.

In neural network algorithms we can tweak several parameters, that will greately impact the final output: number of epochs, learning rate, solver type and many more.

- **Epoch** - one complete pass of the data in the data set. The amount of epochs should be determined through tests. Small number of epochs often leads to bad predictions, while a big number leads to overfitting.

- **Learning rate** - describes the rate at which the network abandons old beliefs, for new ones to take their place. The value must be correct, values that are too big or too small may lead to bad predictions.
- **Solver type** - contains information about how the weights are updated for the network. This project used Stochastic Gradient Descent (SGD) [14] and Adam.

When working with machine learning algorithms, we can encounter two problems, that are the result of our actions - underfitting and overfitting.

Overfitting takes place, when the model is training the data too well. That happens when noise, details or random fluctuations are taken into consideration, which negatively impacts the performance of the model on any new data. In such case, the parameters should be adjusted to constrain the amount of detail that the model learns.

Underfitting on the other hand is usually the result of big learning rates. The model becomes too general, which in the end gives bad results for object classification. To provide a solution to the problem, usually adjusting parameters or using different ML algorithms should be used.

2.2 Optimizers used in the project

Stochastic Gradient Descent (SGD) is one of the most popular algorithms in DL. It is an extension to the first-order optimization algorithm: Gradient Descent. In SGD, the gradient is an expectation, which may be estimated even using a small set of samples. SGD has proven to work very well with deep learning models. While it doesn't guarantee finding the local minimum, it usually finds a very low value of the cost funcion quickly.

The estimate from the example x minibatch m' can be written as follows: $g = \frac{1}{m'} \nabla_\theta \sum_{i=1}^{m'} L\left(x^i, y^i, \theta\right)$, where the loss is $L\left(x, y, \theta\right) = -\log p\left(y|x;\theta\right)$ for each example.

Adam is an optimization algorithm that is used for iteratively updating the network weights based on the training data. The algorithm combines two extensions of the SGD: Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation(RMSProp). Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance).

2.3 Convolutional Networks

Convolutional Neural Networks (CNNs) are used for data, that has a known grid-like topology. The name comes from the fact, that it employs an operation called **convolution**, which is a kind of linear operation (instead of general matrix multiplication).

Convolutions are operations on two functions of a real value argument. The convolution can be represented as $s(t) = (x * w)(t)$, where x is a single input, w

is a valid probability density function and t is time. It leverages three important ideas that can improve machine learning systems: sparse interactions, parameter sharing and equivariant representations [11]. The output of the function is often called the feature map.

Each layer of CNNs consists of three distinguishable stages: producing sets of linear activations, detector stage and pooling. Pooling is a method that instead of giving the output of the neural net, provides a summary statistic of all nearby outputs. This is especially helpful, if images of variable size are given as an input. Moreover it helps with feature extraction (as it is known, neural networks loose data over time, with each layer) from convolutional layers.

2.4 GoogLeNet

GoogLeNet was introduced at ILSVRC 2014 competition, where it took first place with a result of 6.67% error rate (which is close to human level performance). The architecture consisted of 22 layers (27 with pooling) of the Deep CNN reducing the number of parameters to 4 million (60 million compared from AlexNet).

The main innovation between normal CNNs and GoogLeNet was the implementation of *Inception modules*. The modules ran several small convolutions in order to reduce the number of parameters. Moreover batch normalization, RMSprop and image distortions were used. Data from the previous layer was run over four 1x1 convolutions, one 3x3 convolution and one 5x5 convolution, with a 3x3 pooling added simultaneously. You can see the network presented on the graph on Fig. 1.

The Inception module 3x3 and 5x5 condolutions ratio increases as higher layers are achieved. This is due to the fact, that stacking Inception modules on top of each other produces an effect where as features of higher abstraction are captured by higher layers, their spatial concentration is expected to decrease [15].

The highest pro of the network is high inference speeds. GoogLeNet was designed to be computational efficient, so that it could be run on devices with limited computational power or low-memory footprint, making it a good choice for robotics and its applications.

2.5 Deep Learning downsides

While Deep Learning is performing well for many cases of image classification, it still has disadvantages, that should be considered when picking the right method. The most known cons of the method are:

– small data sets can produce bad results,
– long calculation time is a big factor,
– debugging is extremly hard,
– picking good neural net parameters takes practice,
– it's hard to gather the logic behind results.

With recent advantages in Deep Learning some of the factors are less limiting. The learning duration for example, is being decreased using parallellization.

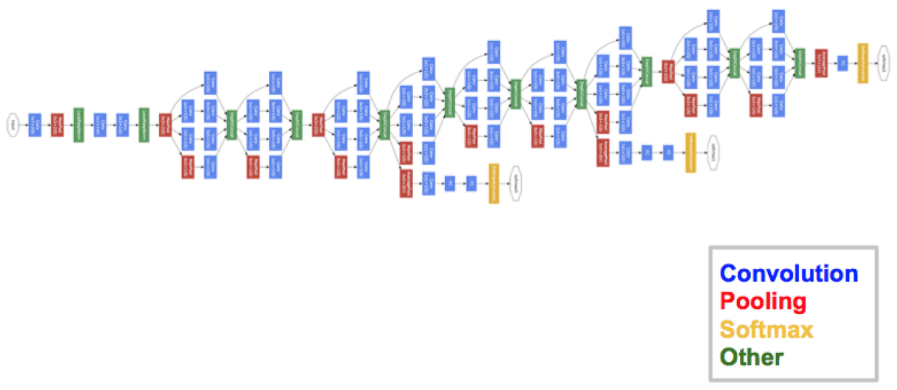


Fig. 1. GoogLeNet 22 Convolutional Neural Network architecture featuring inception layers.

3 Data Acquisition

The data was collected by recording a square video of each guinea pig over a period of 30 seconds in different environments and then extracting frames as an image. Each frame was then lowered in resolution to 256x256 px in order to fit GoogLeNet requirements. Example images taken from the data set can be seen in Fig. 2.

The entire data set consisted of 1098 images. From the initial data set - 25% (274) images were excluded for model validation and 10% (110) were excluded to calculate the test data loss and accuracy. Moreover 32 photos in different environments (animal cage, sleepingroom, guestroom, balcony and bathroom) were taken after the training to test the model behaviour. The visual representation of the training data set can be seen on Fig. 3.

To ensure good accuracy of the model, images for each guinea pig had to cover the whole anatomy of the animal. To achieve that, the following camera positions were covered:

- facing the mouth,
- both sides front and back facing,
- rear view of the animal,
- top view with different distances to the guinea pig.

Moreover empty images (without guinea pigs) were added with the same room backgrounds where previous photos were taken, to increase classification accuracy in distinguishing wanted objects. This set contained 191 images mixed with the guinea pig data set.

Guinea pigs phenotype highly depends on their breed. Therefore three different subjects of diverese ages were used in the experiment:

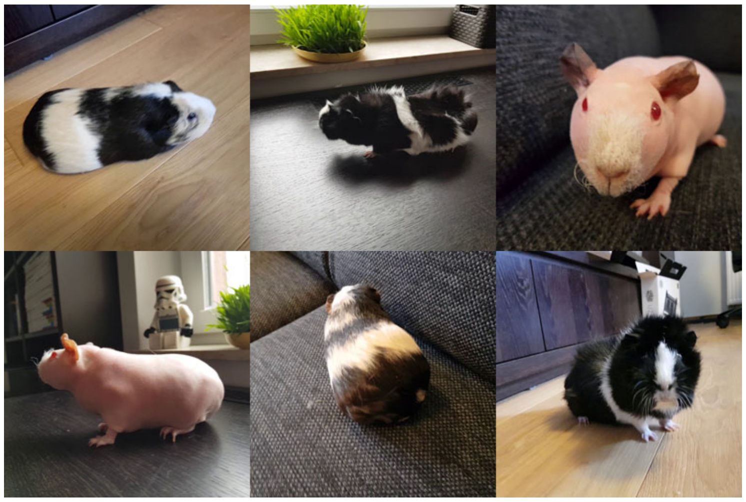


Fig. 2. Example images for crested, abyssinian and skinny guinea pigs from the provided data set.

- Fifi (4 years, abyssinian),
- Rey (2 years, crested),
- Asajj (2 years, skinny).

Using the data, four labels were produced and assigned to the following neural network classes, which provided a base for further classification:

- **None** - when there is no guinea pig in the image,
- **Abyssinian**, **Crested** and **Skinny** - fur type.

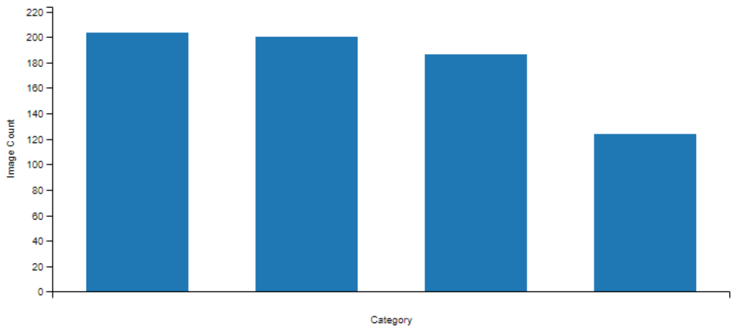


Fig. 3. Data representation in the training data set. Category labels from left: crested, skinny, abyssinian, none.

Finally the data was processed by the neural network, on different settings. Two parameters were changed during testing: learning rate and optimizer used, while the epoch count remained at 15. The settings can be seen below:

1. **First test:**
   - Epoch count: 15,
   - Learning rate: 0.001 with fixed policy,
   - Optimizer: Stochastic Gradient Descent.
2. **Second test:**
   - Epoch count: 15,
   - Learning rate: 0.001 with step down policy,
   - Optimizer: Adam.
3. **Third test:**
   - Epoch count: 15,
   - Learning rate: 0.01 with step down policy,
   - Optimizer: Stochastic Gradient Descent.

4 Results

Final results for classification models were gathered from the neural networks described in the previous section.

Performing the first test gave good results, although overfitting was discovered around epoch 13, with accuracy of 98,95% and loss of 0.04. Later epochs drastically fell in value, producing an accuracy of 65,63% and loss of 1.03. Overfitting appeared because of the low learning rate from the very start, which should have been avoided.

Second test, using Adam as the solver type, provided the best results. The final acurracy and loss after 15 epochs were 99.31% and 0.04 respectfully. A different training rate (0.01) was also tested, but it didn't provide any useful results.

The final test gave good results, but not satisfactory - it produced an accuracy of 87,15% and loss of 0.32. That was not enough to be used for the guinea pig classification system (the predictions would give false-positives).

The final classification model that was selected for further use and testing, was taken from test number two, using the Adam optimizer. Over 15 epochs with a learning rate of 0.001 ran on the GoogLeNet model on Nvidia DIGITS, it has provided the best results. Using more epochs and a different learning rate was tested afterwards, but it led to overfitting of the data. You can see the final result on Fig. 4.

4.1 Manual image test results

After the model was created, a series of tests were performed to check, if the inference is correct. The first set of tests consisted of images from the previously gathered data set. The prediction was always right for provided images, with the value between 70%-95%. You can see an example result on Fig. 5.

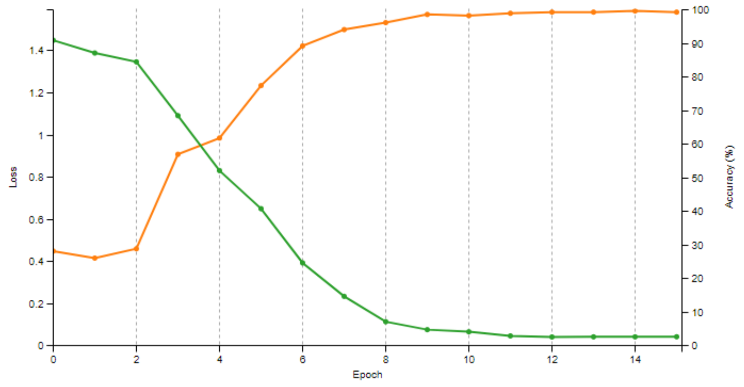
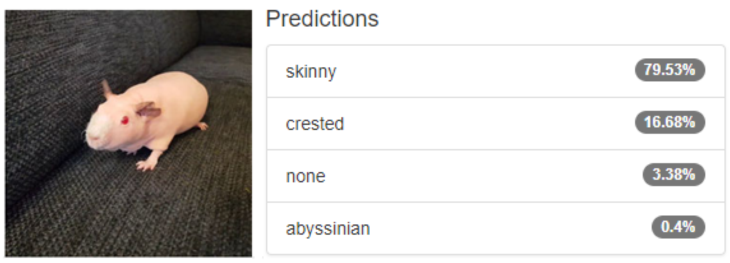


Fig. 4. Graph showing the final accuracy (orange) and loss (green) of the second test running Adam optimizer.

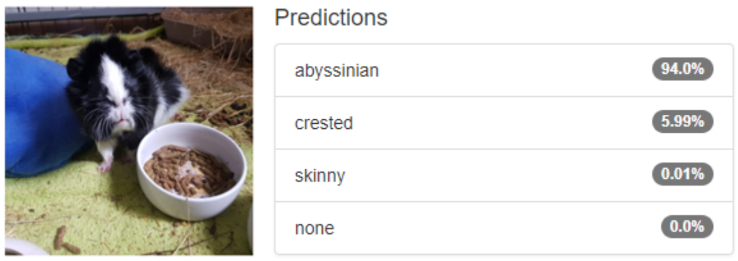
Moreover, as soon as acquiring enough information from predictions was done, tests on new images were performed (different environments, same guinea pigs) giving satisfactory results - guinea pigs were classified correctly on each provided picture containing the animal. Example result one such case can be seen on Fig. 6.

One failed prediction was encountered, when a picture of a background with a cat was used. The cat was badly classified as an abyssinian guinea pig. This happened due to the fact, that the data set didn't contain images that were in any case similar to the cat picture used. More cases like that can be produced with the provided model, as it was trained on specific data in the first place.


Predictions

skinny	79.53%
crested	16.68%
none	3.38%
abyssinian	0.4%

Fig. 5. Example correct prediction results (79.53%) using previously captured images in selected environments.


Predictions

abyssinian	94.0%
crested	5.99%
skinny	0.01%
none	0.0%

Fig. 6. Example correct prediction results (94.0%) using post-model captured images in animal cage environment.

5 Conclusion / Future work

Modern robotics highly depend on sensor readings of the surrounding environment. They often use camera input as one of the parameters to perceive the world. Due to that, imaging methods for decision-making were introduced.

In this paper Deep Learning was implemented for guinea pig classification in order to explore the possibilities of introducing household animal care using robotics and automation, while keeping them safe.

The provided GoogLeNet model from Nvidia DIGITS has proven successful in identifying the guinea pigs in different environments, even when taking images that weren't originally added to the data set. Some errors were observed (false-positives) when no guinea pigs were present in the tested image. The model will behave poorly when other animals are present in the pictures, since the classification was based purely on guinea pigs.

Increasing the accuracy of the model, can greately improve the robot-animal interactions, allowing to tailor behaviours to specific beings. This could be achieved by using modifying the learning rate, using more images, creating more labels or finally using a different optimizer or pre-built model. There are many possible variables to take into consideration, when building a model for a defined task.

The guinea pig classification model after building with GoogLeNet, was able to provide results almost instantly. This is important, if used for robotics, because while waiting for an action, the environment can change quite drasticly. Moreover, such model are built to be deployed on an autonomous platform (Raspberry, Jetson TX2 or any other), so the memory usage will be limited, increasing the inference calculation time.

The created model proves that guinea pig fur recognition for robotic systems is possible. The project gave good results - the created model recognized the animals correctly from images taken in various household backgrounds. The prediction was acquired fast making the inference time low. This is especially important for robotic systems that deal with live animals, because the reaction times need to be rapid.

Future work might include robotic systems that monitor the state of specific animals, adjust food distribution depending on image readings or alert when the guinea pig suffers from any kind of illness.

Moreover, different types of models can be employed to see, which one fits the needs the most. The project shouldn't be limited to GoogLeNet.

References

1. J. J. Roldán, J. del Cerro, D. Garzón-Ramos, P. Garcia-Aunon, M. Garzón, J. de León, and A. Barrientos, "Robots in agriculture: State of art and practical experiences," *Service Robots Antonio Neves, IntechOpen*, 2018. DOI: 10.5772/intechopen.69874. Available from: https://www.intechopen.com/books/service-robots/robots-in-agriculture-state-of-art-and-practical-experiences.
2. M. BD, S. Adil, and S. Ranvir, "Robotics: An emerging technology in dairy and food industry: Review," *International Journal of Chemical Studies*, 2018.
3. J. Long, E. Shelhammer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *arXiv: 1411.4038*, (v2) 2015.
4. F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2017.
5. A. Gomez, A. Salazar, and F. Vargas, "Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks," 2016.
6. G. Chen, T. X. Han, Z. He, R. Kays, and T. Forrester, "Deep convolutional neural network based species recognition for wild animal monitoring," *International Conference on Image Processing (ICIP) IEEE pp. 858-862*, 2014.
7. X. Yu, J. Wang, R. Kays, P. A. Jansen, T. Wang, and T. Huang, "Automated identification of animal species in camera trap images," *EURASIP Journal of Image and Video Processing*, 2013.
8. J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
9. "Nvidia DIGITS." https://developer.nvidia.com/digits.
10. B. Erickson, P. Korfiatis, Z. Akkus, T. Kline, and K. Philbrick, "Toolkits and libraries for deep learning," *Journal of Digital Imaging vol. 30 pp. 400-405*, 2017.
11. I. Goodfellow and Y. Bengio, *Deep Learning.* Cambridge, Massachusetts: The MIT Press, 2016.
12. T. M. Mitchell, *Machine Learning.* McGraw-Hill, New York, 1997.
13. I. Goodfellow, N. Koenig, N. Muja, C. Pantofaru, and A. Sorokin, "Help me help you: Interfaces for personal robots," 2010.
14. L. Bottou, "Large-scale machine learning with stochastic gradient descent," *Proceedings of COMPSTAT 2010.*
15. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Deeper with convolutions," Available from: http://arxiv.org/abs/1409.4842.