A Polite Robot: Visual Handshake Recognition Using Deep Learning

Liutauras Butkus, Mantas Lukoševičius
Faculty of Informatics
Kaunas University of Technology
Kaunas, Lithuania
liutauras.butkus@ktu.edu, mantas.lukosevicius@ktu.lt

***Abstract*—Our project was to create a demo system where a small humanoid robot accepts an offered handshake when it sees it. The visual handshake recognition, which is the main part of the system proved to be not an easy task. Here we describe how and how well we solved it using deep learning. In contrast to most gesture recognition research we did not use depth information or videos, but did this on static images. We wanted to use a simple camera and our gesture is rather static. We have collected a special dataset for this task. Different configurations and learning algorithms of convolutional neural networks were tried. However, the biggest breakthrough came when we could eliminate the background and make the model concentrate on the person in front. In addition to our experiment results we can also share our dataset.**

Keywords—image recognition, computer vision, deep learning, convolutional neural networks, robotics

I. Introduction

The goal of this project is to create a robot that can visually recognize an offered handshake and accept it. When the robot sees a man offering a handshake, it responds by stretching its arm too. This serves as a visual and interactive demonstration, which would get students more interested in machine learning and robotics.

For this purpose we used a small humanoid robot, a simple camera mounted on it, and deep convolutional neural networks for image recognition. The recognition, as well as training of it, were done on a PC and the command to raise the arm was sent back to the robot.

This article mainly shares our experience in developing and training the visual handshake recognition system, which proved to not be trivial. In particular, we will discuss how images were collected, preprocessed, what architecture of convolutional neural networks was used, how it was trained and tested; what gave good and what not so good results.

This document is divided into several sections. Section II reviews existing solution to similar problem. Section III introduces our method for this project. Section IV describes the data set used in this study. Section V emphasizes importance of data preprocessing before training. Section VI describes robot interface. Sections VII and VIII provide analysis of results and conclusions.

Related Work

Virtually all vision-based hand gesture recognition systems described in literature use (a) image sequences (videos) with (b) depth information in them, see [1] for a good recent survey. Microsoft Kinect [2] and Leap Motion [3] are two examples of popular sensors specifically designed for gesture and posture 3D tracking. While clearly both temporal (a) and depth (b) aspects are helpful in recognizing hand gestures, our system uses neither of the two. We (b) used an inexpensive camera for simple RGB image acquisition to make the system more accessible and the algorithms more widely applicable, e.g., in smartphones, natural lighting. We also (a) used single frames to recognize the extended hand for the handshake, since the gesture is rather static – just holding the extended hand still – and could perhaps be called "posture". This makes the recognition problem considerably harder.

A bit similar project to ours called "Gesture Recognition System using Deep Learning" was presented in PyData Warsaw 2017 conference [4]. The author introduced a Python-based, deep learning gesture recognition model that is deployed on an embedded system, works in real-time and can recognize 25 different hand gestures from a simple webcam stream. The development of the system included: a large-scale crowd-sourcing operation to collect over 150,000 short video clips, a process to decide which deep learning framework to use, the development of a network architecture that allows for classifications of video clips solely with RGB input frames, the iterations necessary to make the neural network run in real-time on embedding devices, and lastly, the discovery and development of playful gesture-based applications. Their approach is still different from our approach in that they used video samples as their input (several frames at a time) and tried to recognize moving gestures.

There is considerable literature similar to our approach in both (a) and (b) for recognizing sign language hand gestures (or rather postures) from RGB images, including using deep learning [5]. These approaches, however, usually work with images of a single hand on a uniform background where the hand can be cropped from the image using thresholding [5], skin color [6], or relying on the subject wearing a brightly-colored glove [7].

III. OUR METHOD

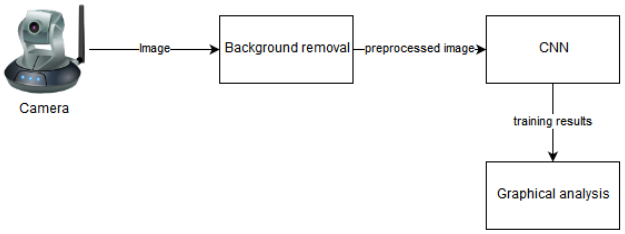

Fig. 1. System training model.

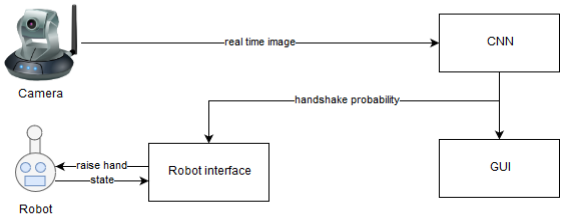

Fig. 2. System running model.

The system model consists of several parts showed in Figure 1 and 2, including camera, camera images pre-processing, convolutional neural network's training using deep learning, graphical user interface, robot interface and robot itself.

At first camera was used to collect the image dataset. This is described in section IV. After later research, which is described in section VI, the images in the dataset had to be pre-processed to be able to train the model, which is the next part of our system. Using Keras library the model was created, compiled and finally trained with the images (this process explained in section VI). The final part is to run the model to recognize new live images. For this reason camera's interface was programmed to take photos at every 0.5 second, the model gets those images as an input and returns probability of seeing an offered handshake as an output result. If this result is above a certain threshold, a robot interface sends a command to robot to perform a corresponding task. This part more deeply described in section VI.

A. *Choice of using deep learning libraries*

Deep learning [8] (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised.

Deep learning models are loosely related to information processing and communication patterns in a biological nervous system, such as neural coding that attempts to define a relationship between various stimuli and associated neuronal responses in the brain.

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks [9] have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics and drug design, where they have produced results comparable to and in some cases superior to human experts.

Convolutional networks [8], also known as convolutional neural networks, or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

Keras [10] is a high-level deep learning library written in Python and capable of running on top of either TensorFlow or Theano deep learning libraries. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research. Keras deep learning library allows for easy and fast prototyping (through total modularity, minimalism, and extensibility). It supports both convolutional networks (we used in our solution) and recurrent networks, as well as combinations of the two. Keras also supports arbitrary connectivity schemes (including multi-input and multi-output training) and runs seamlessly on CPU and GPU. The core data structure of Keras is a model, a way to organize layers. The main type of model is the Sequential model, a linear stack of layers. Keras' Guiding principles include Modularity. A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that users can combine to create new models. Each module should be kept short and simple. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.

B. *Our convolutional neural network model*

The convolutional neural network model that we used is specified in Figure 3.

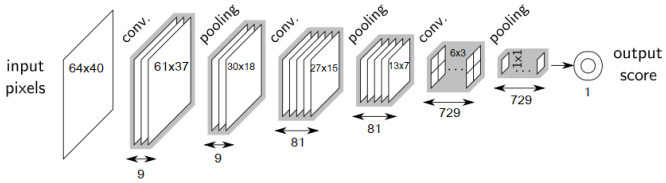

Fig. 3. Our convolutional neural network model.

It takes 64x40 resolution images as inputs, consists of three convolutional layers, each followed by pooling, and has a single node output. We use rectified linear units in all layers except for the output node where it is sigmoid.

C. *Training process*

Before starting to train the model there are several parameters which describe training details. The first parameter is epochs count. Epoch itself is an arbitrary milestone, generally defined as “one pass over the entire dataset”, used to separate training into distinct phases, which is useful for logging and periodic evaluation. In general it means how many times the process will go through the training set.

Second parameter is batch size. Batch size defines number of samples that going to be propagated through the network. For instance, there are 200 training samples and we want to set up batch size equal to 30. Algorithm takes first 30 samples from the training dataset and trains network. Next it takes second 30 samples and trains network again. The procedure can be done until we propagate through the networks all samples. However, the problem usually happens with the last set of samples. In this example the last 20 samples which is not divisible by 30 without remainder. The simplest solution is just to get final 20 samples and train the network.

We have tried different loss functions and training optimization methods. The ones that worked reasonably well in the end are reported in Section VII.

Train accuracy and train loss are calculated on the go, during training. Figures in Section VII show how well our network is doing on the data it is being trained. Training accuracy usually keeps increasing throughout training.

D. *Validation process*

To validate the model we need to have new dataset this new images, which has not been used in training process. Validation is usually carried out together with training. After every epoch, the model is tested against a validation set, and validation loss and accuracy are calculated. These numbers tell you how good your model is at predicting outputs for inputs it has never seen before. Validation accuracy increases initially and drops as you over fit. Overfitting happens when our model fits too well to the training set. It then becomes difficult for the model to generalize to new examples that were not in the training set. For example, our model recognizes specific images in your training set instead of general patterns. Our training accuracy will be higher than the accuracy on the validation/test set.

E. *Testing process*

To test the model we need another new dataset. Testing usually is run manually by giving an image from dataset for trained model to get a result. And the result is a percent value that shows probability on each output option

IV. Data Collection Preparation

As we mentioned in the previous section a collection of image data was needed to implement this project. As the system only recognizes greetings, only two results are possible: greetings are recognized or not. During the development of the whole project, more than 4,000 different images were collected for the training of the neural network. Approximately 2000 for each category. Single-image resolution is 318x198.

We can see in Figure 4, that in the image, one person was usually with his hand stuck or not. It was also tried to capture images in as many different environments as possible. Human clothing was also varied trying to capture as diverse as possible colors. This is important in order to ensure that recognition is not restricted to a particular specific situation.

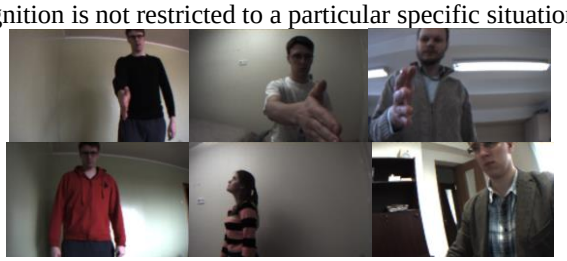

Fig. 4. Image samples: top positive, bottom negative.

The pictures were divided into three sets: training, validation and testing. The neural network is taught with training data. It is then validated with validation data to verify that a well-trained neural network performs recognition with new examples. The test data is intended to validate the final neural network's capability to obtain the final true recognition. In addition, data augmentation [11] was used during training, in which various small transformations were made to the images before training on them (rotation, translation, color-shift, up- /down-scaling).
If there are people who are interested in this task, we could share the data with everyone who wants it.

V. BACKGROUND REMOVAL

Initially, we tried to train the neural network with the data obtained directly from the camera without preprocessing them. However, it has been noticed that the model with the best attempt reached 78 percent training accuracy and about 64 percent validation accuracy followed by overfitting, during which the error rate increased significantly. For this reason, it was necessary to look for solutions on how to avoid overfitting and how to increase the validation accuracy of the model. To achieve this, attempts were made to change the model's parameters, but this did not improve result as much as it was expected. Then it was decided to process the data itself. From previous experiments, we were able to get the impression that overfitting appears due to the excessive color gamut and color of the images. For this reason, we have decided to try removing background images and training a neural network with pictures without background. However, that causes a new problem. How to detect where the background is and where is an object (in this case a human)? For this problem, we decided to take the first image without a human and claim that it is a background and all other images are objects with backgrounds. Though, in this case camera had to be in fixed position. Then we were able to subtract two images and get image without a background. Usually, after subtraction some noise always had

left in images. To reduce it, we set a permissible error for pixel RGB values. You can see those images in Figure 5.

Fig. 5. Same images with (left) and without (right) background.

The result was obvious. It's possible to achieve 91 percent accuracy with a smooth natural background. This means that the model is quite precise enough to recognize the extended hand when the background behind the human is equal and does not need to be removed.

In order to obtain this result, first we needed to draw up a test plan, which would make it clear how the training is most appropriate. We've identified three methods (regular training, training with removed background, training with attached backgrounds), and five types of data (when the background is a specific color, when the background is smooth and natural, when the background is static color, when the background is changing and when the background is with a few outsiders). All test results are presented in the results Section VII.

VI. Interfacing Robot

Fig. 6. Photo of our robot and use case.

For this project HR-OS1 Humanoid Endoskeleton robot [12] was used. It is showed in Figure 6. It has integrated onboard Linux computer with Intel Atom processor, which gives all the processing power to run robot. The HR-OS1 is a hackable, modular, humanoid robot development platform designed from the ground up with customization and modification in mind. It has built in software which invokes robot actions. You can see robot's software interface in Figure 7.

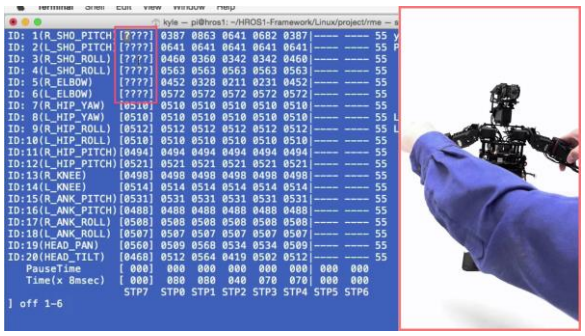

Fig. 7. Robot joints management interface.

Robot interface is used when the model is started to predict new images. After CNN return probability of the image it is sent to robot interface. Then robot interface reads the input value and if it is true interface runs command for a robot to raise its hand.

VII. Experiment Results and Analysis

In this section we will explain in detail what experiments were done and what results were achieved.

As we mentioned in the previous section the first experiments were carried out using dataset with non-removed background images for training and validation. The other parameters were:

- Image width: 64
- Image height: 40
- Training dataset samples: 421
- Validation dataset samples: 122
- Epochs: 30
- Batch size: 32
- Model loss function: mean squared error
- Model optimizer: stochastic gradient descent
- Model metrics: accuracy

After training we have got the results, which are shown in Figure 8, and they after final iteration were:

- Training accuracy: 78%
- Training loss: 0.16
- Validation accuracy: 64%
- Validation loss: 0.19

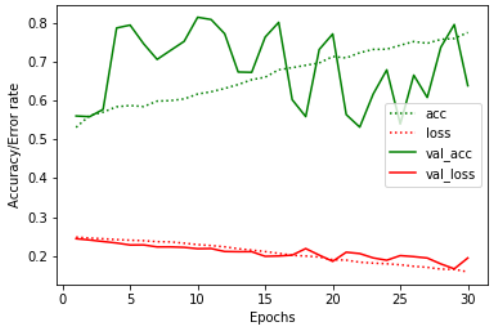

Fig. 8. Training without removing background images results graph.

The result shows that model validates new images by 64% accuracy. However, after testing manually this model with images which very different environments, the result have been even worse.

The next experiment were held by training model with removed background. The model parameters were:

- Image width: 64
- Image height: 40
- Training dataset samples: 421
- Validation dataset samples: 121
- Epochs: 50
- Batch size: 32
- Model loss function: mean squared error
- Model optimizer: stochastic gradient descent
- Model metrics: accuracy

After training we got the results, which are shown in Figure 9, and they after final iteration were:

- Training accuracy: 91%
- Training loss: 0.07
- Validation accuracy: 82%
- Validation loss: 0.12

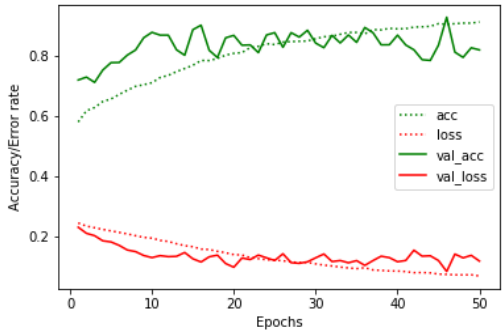

Fig. 9. Training with removed background images results graph.

This time the result shows that model validates new images by 82% accuracy and after testing manually this model with images which very different environments, the result shown about the same accuracy.

All our tests and their results are shown in the table below. Some of the most sophisticated training experiments have not been completed, as it was not immediately meaningful to perform them due to poor results from simple training.

30 epochs	**A**	**B**	**C**
1. *Background of a specific color*	-	91/82	86/77
2. *A smooth natural background*	81/77	92/80	81/70
3. *Static background (colorful)*	52/54	62/55	58/52
4. *Changing background*	40/50	53/50	Not tried

TABLE I. DIFFERENT TRAINING VALIDATION TABLE

Columns of the table represents training of the model, rows – how model was validated:

A - Simple training on original images;

B - Trained with removed backgrounds;

C - Trained with background replacements;

x/y – training accuracy / validation accuracy

The results showed that the removal of the background significantly improves the accuracy of the model recognition. The best results were from experiments where training took place with removed background pictures and validating with also removed background images or smooth background images.

VIII. DISCUSSION AND FUTURE WORK

In this work several different training experiments were performed, watching and studying accuracy of the trained models. The experiments showed that the results depended more not on the model used and its parameters, but on the transformation of the images. To achieve the best results image preprocessing played a key role in this experiment. The best result was reached by removing background before training.

Our interpretation of the results is that removing the background reduces the variation in the data and makes the machine learning model focus on the person in the image. Without the background removal the models are prone to overfitting, probably basing their decision on wrong features of the image. Might be that similar accuracy can be achieved without background removal, but with much more data,

training, and probably more powerful models. In that case the models have to infer that the person in the foreground is the most important object in the images and learn how to distinguish it on its own. Motion or depth information, which is used in many gesture recognition systems, would also make separation of the person in front from the background easier and likely not necessary to be done explicitly.

This is consistent with results discussed in related work (Section II) where other authors either use motion and/or depth information, or also crop the foreground from the background in some way.

In our approach it is not necessary to remove the background during testing/valuation. The best validation results are on data with smooth natural backgrounds. The accuracy of this validation data reached 92%. A reasonable future work would be to attempt to create a model that can better recognize offered handshakes in a wider range of environments.

REFERENCES

[1] Maryam Asadi-Aghbolaghi, Albert Clapes, Marco Bellantonio, Hugo Jair Escalante, “A survey on deep learning based approaches for action and gesture recognition in image sequences”, 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), 2017 http://sunai.uoc.edu/~vponcel/doc/survey-deep-learning_fg2017.pdf,

[2] Microsoft Robotics, “Kinect Sensor”, accessed on 05 - 2018 https://msdn.microsoft.com/en-us/library/hh438998.aspx

[3] Leap Motion, “Leap Motion – Developer”, accessed on 05 - 2018 https://developer.leapmotion.com/

[4] Joanna Materzynska, “Building a Gesture Recognition System using Deep Learning”, PyData Warsaw 2017, https://medium.com/twentybn/building-a-gesture-recognition-system-using-deep-learning-video-d24f13053a1

[5] Oyedotun, O.K. & Khashman, “Deep learning in vision-based static hand gesture recognition“, Neural Computing and Applications (2017) 28: 3941. https://doi.org/10.1007/s00521-016-2294-8

[6] Dennis Núñez Fernández, Bogdan Kwolek, “Hand Posture Recognition Using Convolutional Neural Network” Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. CIARP 2017. Lecture Notes in Computer Science, vol 10657. Springer, Cham , 2018 http://home.agh.edu.pl/~bkw/research/pdf/2017/FernandezKwolek_CIARP2017.pdf

[7] Rosalina, L. Yusnita, N. Hadisukmana, R. B. Wahyu, R. Roestam and Y. Wahyu, "Implementation of real-time static hand gesture recognition using artificial neural network," 2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT), Kuta Bali, 2017 http://journal.binus.ac.id/index.php/commit/article/viewFile/2282/3245

[8] Ian Goodfellow, Yoshua Bengio, Aaron Courville, “Deep Learning”, MIT Press, 2016. http://www.deeplearningbook.org/

[9] Denny Britz, „Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs“, accessed on 05 - 2018 http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[10] Keras documentation, “Why use Keras?”, accessed on 05 - 2018 https://keras.io/why-use-keras/

[11] Prasad Pai, “Data Augmentation Techniques in CNN using Tensorflow”, accessed on 05 - 2018 https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9

[12] Tossen Robotics, “HR-OS1 Humanoid Endoskeleton spescifications”, accessed on 05 - 2018 http://www.trossenrobotics.com/HR-OS1

[13] Połap, Dawid, Marcin Woźniak, Christian Napoli, Emiliano Tramontana, and Robertas Damaševičius. "Is the colony of ants able to recognize graphic objects?." In International Conference on Information and Software Technologies, pp. 376-387. Springer, 2015.

[14] Woźniak, Marcin, Dawid Połap, Christian Napoli, and Emiliano Tramontana. "Graphic object feature extraction system based on cuckoo search algorithm." Expert Systems with Applications, vol. 66, pp. 20-31, 2016.