Project: Robotic Inference - Thomas Hatting

1. Abstract

Children learn to speak through ordinary everyday interaction. This paper proposes a speech and language development system for toddlers (children aged 1-3 years). The idea is that the child holds up an ordinary (and safe) household object in front of the camera. The robotic inference system identifies the object and activates a simple conversational interface to engage the child in a conversation. Several deep neural network architectures are considered. The resulting model is generated using the GoogLeNet deep neural network architecture and provides a high accuracy. It is successfully tested on a separate batch of images.

2. Introduction

Children learn to speak through ordinary everyday interaction: the child is spoken and listened to, his or her questions are answered and questions are posed to the child. Speech and language development is also supported by singing, playing, and reading. Between ages 1.5 and 2 years, the child has usually built up a small vocabulary: a few words at first and then gradually more. Around the age of 2, the child usually knows well over 200 words. At the age of 2 and a half years, the child learns new words every day and uses mostly nouns and verbs. At the age of 3 years, the child is interested in word plays and nursery rhymes, and the development of linguistic awareness begins (Helsinki 2018; Wikipedia 2018).

This paper proposes a speech and language development system for toddlers (children aged 1-3 years). The idea is that the child holds up an ordinary (and safe to handle) household object in front of the camera. The robotic inference system identifies the object and activates a conversational interface, engaging the child in a simple conversation, for example, “Is the cup full or empty?” or “What do you use a phone for?”. The system can help the child become familiar with objects in the home and to develop vocabulary and sentence structure. Recent studies have shown that the more words spoken to a child in early stages of development, the better are the chances of the child performing well at school. In the early years, children need to hear approximately 21,000 words per day or more for their vocabulary to develop at an appropriate pace (Psychology Today 2014; Eghan 2016).

This report focuses only on robotic inference, but later stages could involve development of an appropriate conversational interface that can be activated by the inference system. Also, this system is not meant as a substitute but rather as a supplement to everyday parent-child interaction.

3. Background / Formulation

Three deep neural network architectures were considered: LeNet (Lecun et al. 1998), AlexNet (Krizhevsky et al. 2012) and GoogLeNet (Szegedy et al. 2014). The first network architecture, LeNet, has been developed primarily to recognize handwritten digits 0-9. Therefore, it was decided to consider one of the other two networks, namely AlexNet or GoogLeNet.

GoogLeNet generates models with a higher accuracy than AlexNet (Canziani et al. 2017). For this reason, it was decided to use the GoogLeNet network architecture for the supplied data. However, it is important to note that there exists a hyperbolic relationship between model accuracy and inference time (Canziani et al. 2017).

Due to the hyperbolic relationship between accuracy and inference time, this increased accuracy provided by GoogLeNet does lead to a higher inference time. However, as long as inference time stays within set limits, the GoogLeNet architecture appears to be a better choice than AlexNet. Due to higher accuracy, GoogleLeNet also appears to be a good choice for the proposed robotic inference idea.

In terms of parameters, the required number of epochs for the model can be estimated by observing at what point the learning rate descends to a value close to zero in the DIGITS GUI. Also, it is important to use a portion of collected data for validation. Typically, 25-30% of collected data should be used for validation.

4. Data Acquisition

The images were collected using an integrated webcam on a Dell laptop. This solution was chosen because it provided a quick and efficient way of collecting data. Four categories were defined: fruit, cup, phone and nothing. In all 1500 images were collected for each category using a Python script (see appendix below). The images were in grayscale and had a resolution of 256 x 256 pixels. It was decided to use grayscale rather than RGB, as there does not appear to be any particular benefit of using color images. A resolution of 256x256 was used, as this resolution is the intended image size of the chosen deep neural network. Figure 1 shows a sample of images.

Figure 1: Images collected by means of an integrated webcam on a laptop. Images were in grayscale and had a resolution of 256 x 256 pixels.

In figure 2 below, the number of collected images for the robotic inference idea are shown. They are divided into test, validation and evaluation.

	Test	Validation (25%)	Evaluation
Fruit	1110	370	20
Cup	1110	370	20
Phone	1110	370	20
Nothing	1110	370	20

© Thomas Hatting

Figure 2: Number of images collected to create and test the model

5. Results

The results for the pre-supplied data and the robotic inference idea are shown below, including graphs of accuracy and learning rate. Furthermore, the final model for the inference idea is tested with a separate batch of images.

5.1 Results for supplied data

A model was generated using the supplied data in folder /data/P1_data in the DIGITS Workspace and using a GoogLeNet network architecture. The number of epochs were set to 6, and percentage share of validation data was set to 30% (slightly above default value). The results are shown in figure 3. As can be seen, the model accuracy is 75.4% and inference time is between 5 and 5.7 ms.

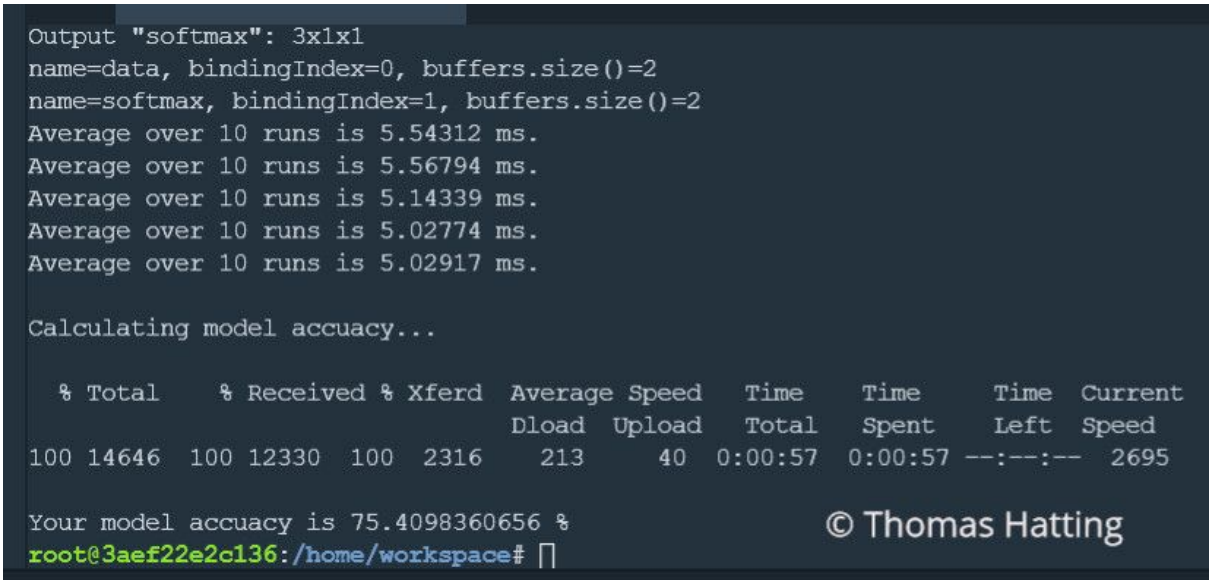

Figure 3: Results achieved for supplied data using GoogLeNet with epoch=6 and validation data percentage share set to 30%

5.2 Results for the robotic inference idea

The DIGITS GUI was used to generate a model for the robotic inference idea using the collected data (see section 4). In order to compare accuracy, both AlexNet and GoogLeNet architectures were tested.

Figure 4 and 5 show the accuracy and learning rate for AlexNet. The accuracy reached 94.2% with number of epochs set to 5 and validation data percentage share set to 25% (default value).

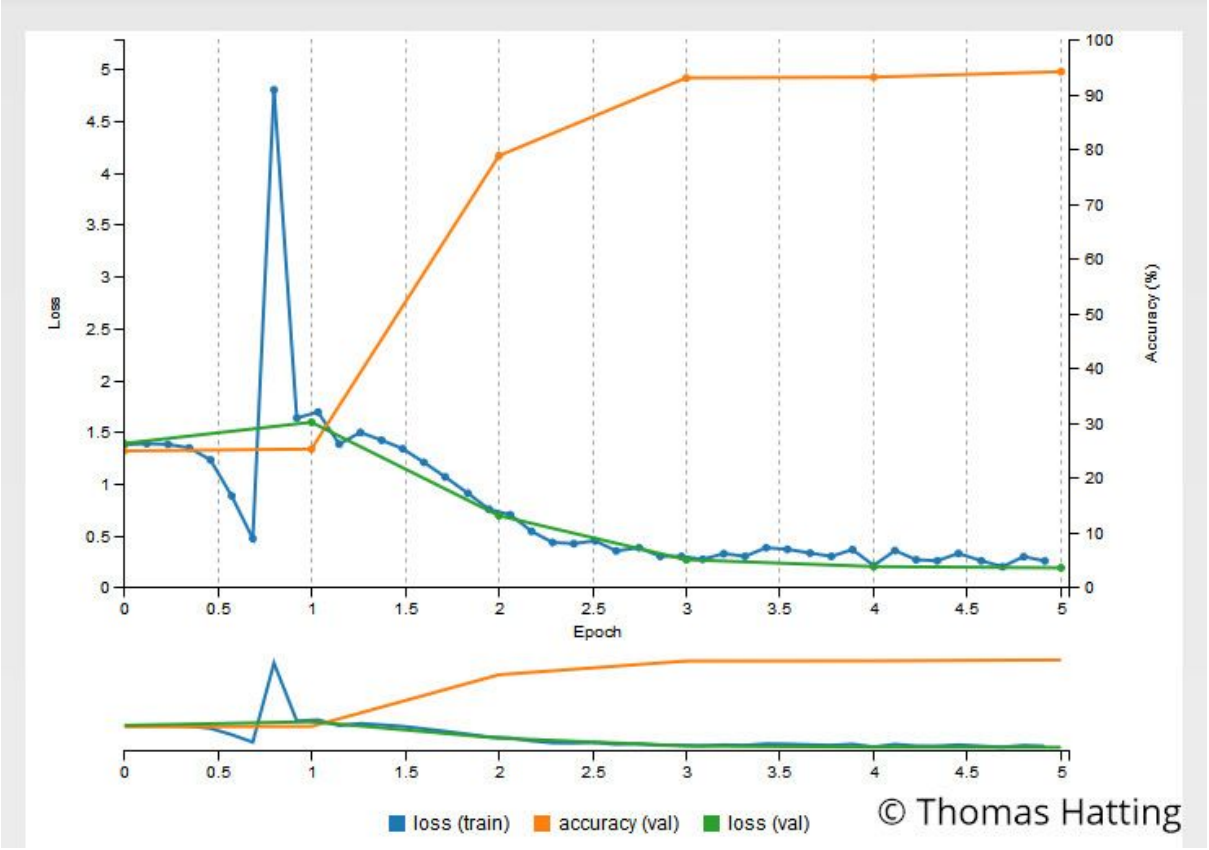

Figure 4: Loss & accuracy for generated model using AlexNet (epoch=5, validation data = 25%)

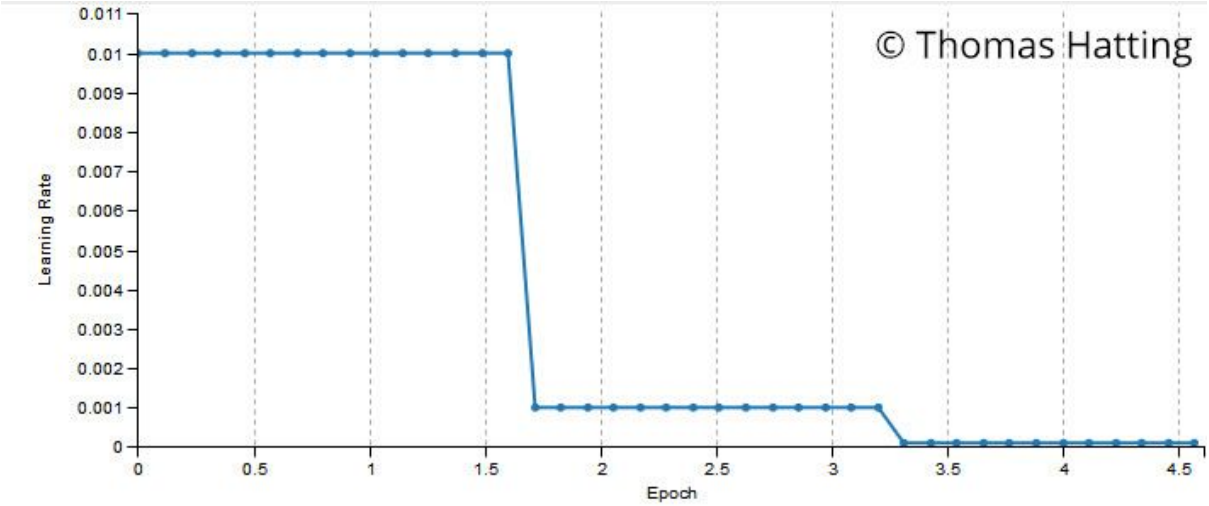

Figure 5: Learning rate for generated model using AlexNet

The number of epochs were chosen by observing at which point the learning rate descends to a minimum.

Figure 6 and 7 show accuracy and learning rate for GoogLeNet. The accuracy reached 98.5% with number of epochs set to 5 and validation data percentage share set to 25% (default value). **This model had a higher accuracy than the previous example and was therefore chosen as the final model of this project.**

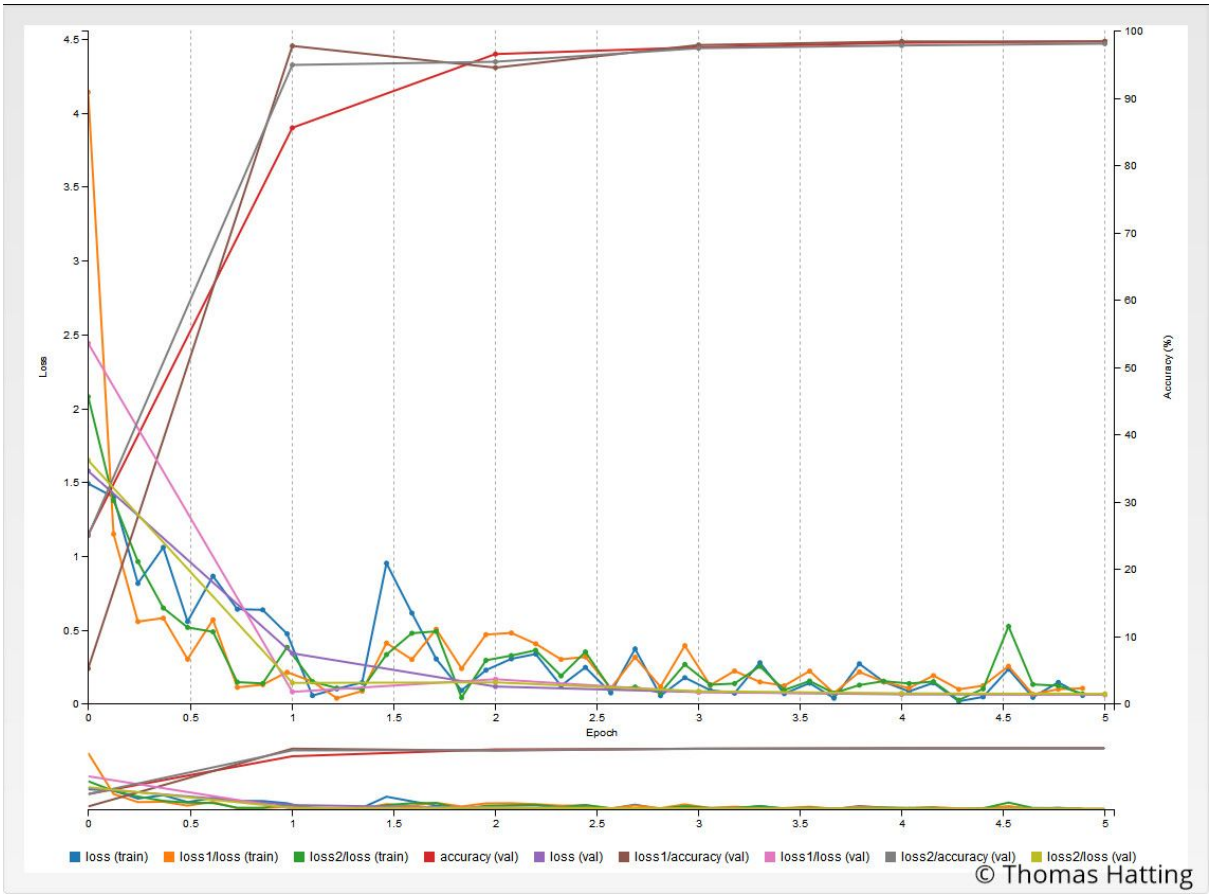

Figure 6: Loss & accuracy of generated model using GoogLeNet (epoch=5, validation data = 25%)

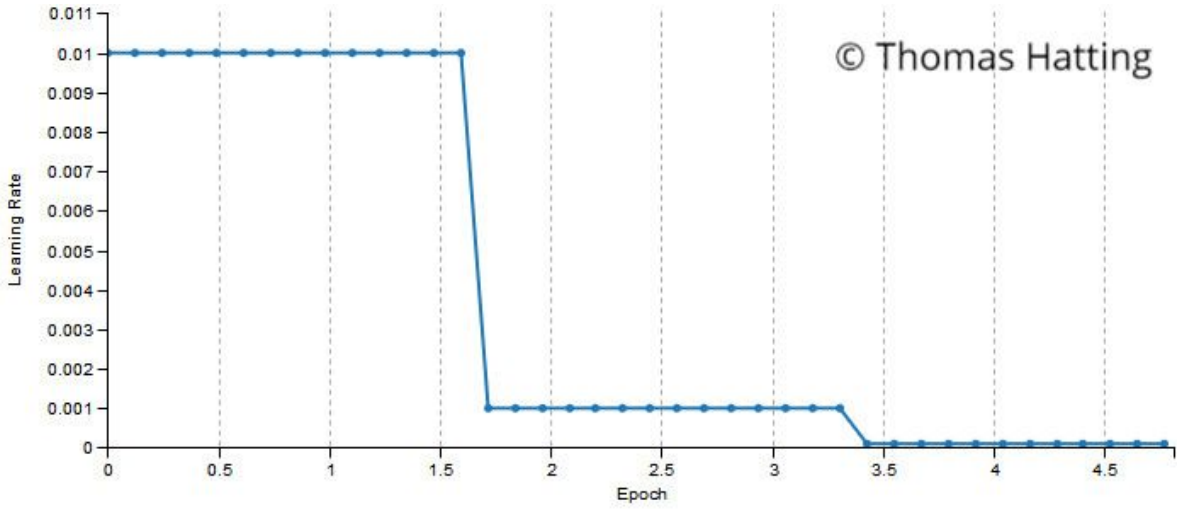

Figure 7: Learning rate for generated model using GoogLeNet

In order to test the final model, a separate collection of test images were generated (80 images). These images were tested by uploading an image list and using the "Classify Many" option in the DIGITS GUI. In all, 79 out of 80 images were correctly identified by the robotic inference system. There was only one incorrectly identified image (test_60.png). This image showed the phone but was incorrectly identified as a fruit (banana). For this particular image, the phone was located towards the edge of the image, and instead, the system identified the arm holding the phone as a fruit. Figure 8 on following pages shows the results of the "classify many" test:

All classifications

	Path	Top predictions							
1	/data/test_images/test_0.png	Banana	99.94%	Phone	0.04%	Cup	0.03%	Nothing	0.0%
2	/data/test_images/test_1.png	Banana	99.91%	Phone	0.06%	Cup	0.04%	Nothing	0.0%
3	/data/test_images/test_2.png	Banana	99.87%	Phone	0.08%	Cup	0.05%	Nothing	0.0%
4	/data/test_images/test_3.png	Banana	99.91%	Phone	0.06%	Cup	0.04%	Nothing	0.0%
5	/data/test_images/test_4.png	Banana	99.91%	Phone	0.05%	Cup	0.03%	Nothing	0.0%
6	/data/test_images/test_5.png	Banana	99.92%	Phone	0.05%	Cup	0.03%	Nothing	0.0%
7	/data/test_images/test_6.png	Banana	99.84%	Phone	0.08%	Cup	0.07%	Nothing	0.0%
8	/data/test_images/test_7.png	Banana	99.83%	Phone	0.09%	Cup	0.08%	Nothing	0.0%
9	/data/test_images/test_8.png	Banana	99.9%	Phone	0.06%	Cup	0.04%	Nothing	0.0%
10	/data/test_images/test_9.png	Banana	99.96%	Phone	0.03%	Cup	0.02%	Nothing	0.0%
11	/data/test_images/test_10.png	Banana	99.95%	Phone	0.03%	Cup	0.02%	Nothing	0.0%
12	/data/test_images/test_11.png	Banana	99.96%	Phone	0.03%	Cup	0.02%	Nothing	0.0%
13	/data/test_images/test_12.png	Banana	99.94%	Phone	0.04%	Cup	0.02%	Nothing	0.0%
14	/data/test_images/test_13.png	Banana	99.78%	Cup	0.11%	Phone	0.11%	Nothing	0.0%
15	/data/test_images/test_14.png	Banana	99.85%	Phone	0.08%	Cup	0.07%	Nothing	0.0%
16	/data/test_images/test_15.png	Banana	99.91%	Phone	0.05%	Cup	0.04%	Nothing	0.0%
17	/data/test_images/test_16.png	Banana	99.91%	Phone	0.05%	Cup	0.03%	Nothing	0.0%
18	/data/test_images/test_17.png	Banana	99.91%	Phone	0.06%	Cup	0.04%	Nothing	0.0%
19	/data/test_images/test_18.png	Banana	99.82%	Phone	0.09%	Cup	0.08%	Nothing	0.0%
20	/data/test_images/test_19.png	Banana	99.78%	Phone	0.11%	Cup	0.11%	Nothing	0.0%
21	/data/test_images/test_20.png	Cup	99.66%	Phone	0.28%	Banana	0.06%	Nothing	0.0%
22	/data/test_images/test_21.png	Cup	99.56%	Phone	0.38%	Banana	0.06%	Nothing	0.0%
23	/data/test_images/test_22.png	Cup	99.62%	Phone	0.31%	Banana	0.06%	Nothing	0.0%
24	/data/test_images/test_23.png	Cup	99.6%	Phone	0.33%	Banana	0.07%	Nothing	0.0%
25	/data/test_images/test_24.png	Cup	99.44%	Phone	0.49%	Banana	0.07%	Nothing	0.0%
26	/data/test_images/test_25.png	Cup	99.05%	Phone	0.84%	Banana	0.11%	Nothing	0.0%
27	/data/test_images/test_26.png	Cup	99.02%	Phone	0.8%	Banana	0.17%	Nothing	0.0%
28	/data/test_images/test_27.png	Cup	99.23%	Phone	0.67%	Banana	0.1%	Nothing	0.0%
29	/data/test_images/test_28.png	Cup	99.44%	Phone	0.47%	Banana	0.09%	Nothing	0.0%
30	/data/test_images/test_29.png	Cup	99.2%	Phone	0.69%	Banana	0.11%	Nothing	0.0%
31	/data/test_images/test_30.png	Cup	99.07%	Phone	0.79%	Banana	0.14%	Nothing	0.0%

DIGITS Model_objects_1 Classify Many thomas (Logout) Info About

34	/data/test_images/test_33.png	Cup	99.76%	Phone	0.19%	Banana	0.04%	Nothing	0.0%
35	/data/test_images/test_34.png	Cup	99.78%	Phone	0.19%	Banana	0.03%	Nothing	0.0%
36	/data/test_images/test_35.png	Cup	99.99%	Banana	0.0%	Phone	0.0%	Nothing	0.0%
37	/data/test_images/test_36.png	Cup	100.0%	Banana	0.0%	Phone	0.0%	Nothing	0.0%
38	/data/test_images/test_37.png	Cup	100.0%	Banana	0.0%	Phone	0.0%	Nothing	0.0%
39	/data/test_images/test_38.png	Cup	100.0%	Banana	0.0%	Phone	0.0%	Nothing	0.0%
40	/data/test_images/test_39.png	Cup	100.0%	Banana	0.0%	Phone	0.0%	Nothing	0.0%

© Thomas Hatting

41	/data/test_images/test_40.png	Nothing	97.9%	Banana	1.88%	Phone	0.12%	Cup 0.1%
42	/data/test_images/test_41.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
43	/data/test_images/test_42.png	Nothing	97.9%	Banana	1.88%	Phone	0.12%	Cup 0.1%
44	/data/test_images/test_43.png	Nothing	97.9%	Banana	1.88%	Phone	0.12%	Cup 0.1%
45	/data/test_images/test_44.png	Nothing	97.9%	Banana	1.87%	Phone	0.12%	Cup 0.1%
46	/data/test_images/test_45.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
47	/data/test_images/test_46.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
48	/data/test_images/test_47.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
49	/data/test_images/test_48.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
50	/data/test_images/test_49.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
51	/data/test_images/test_50.png	Nothing	97.9%	Banana	1.88%	Phone	0.12%	Cup 0.1%
52	/data/test_images/test_51.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
53	/data/test_images/test_52.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
54	/data/test_images/test_53.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
55	/data/test_images/test_54.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
56	/data/test_images/test_55.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%

DIGITS Model_objects_1 Classify Many thomas (Logout) Info About

58	/data/test_images/test_57.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
59	/data/test_images/test_58.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
60	/data/test_images/test_59.png	Nothing	97.91%	Banana	1.87%	Phone	0.12%	Cup 0.1%
61	/data/test_images/test_60.png	Banana	63.75%	Cup	25.32%	Phone	10.93%	Nothing 0.01%
62	/data/test_images/test_61.png	Phone	95.47%	Cup	4.5%	Banana	0.03%	Nothing 0.0%
63	/data/test_images/test_62.png	Phone	98.74%	Cup	1.26%	Banana	0.0%	Nothing 0.0%
64	/data/test_images/test_63.png	Phone	98.8%	Cup	1.2%	Banana	0.0%	Nothing 0.0%
65	/data/test_images/test_64.png	Phone	99.38%	Cup	0.62%	Banana	0.0%	Nothing 0.0%
66	/data/test_images/test_65.png	Phone	99.61%	Cup	0.39%	Banana	0.0%	Nothing 0.0%
67	/data/test_images/test_66.png	Phone	99.78%	Cup	0.22%	Banana	0.0%	Nothing 0.0%
68	/data/test_images/test_67.png	Phone	99.81%	Cup	0.19%	Banana	0.0%	Nothing 0.0%
69	/data/test_images/test_68.png	Phone	99.69%	Cup	0.31%	Banana	0.0%	Nothing 0.0%
70	/data/test_images/test_69.png	Phone	97.75%	Cup	2.25%	Banana	0.0%	Nothing 0.0%
71	/data/test_images/test_70.png	Phone	79.91%	Cup	20.07%	Banana	0.02%	Nothing 0.0%
72	/data/test_images/test_71.png	Phone	73.4%	Cup	26.57%	Banana	0.03%	Nothing 0.0%
73	/data/test_images/test_72.png	Phone	96.91%	Cup	3.08%	Banana	0.0%	Nothing 0.0%
74	/data/test_images/test_73.png	Phone	97.59%	Cup	2.41%	Banana	0.0%	Nothing 0.0%
75	/data/test_images/test_74.png	Phone	97.52%	Cup	2.48%	Banana	0.01%	Nothing 0.0%
76	/data/test_images/test_75.png	Phone	98.1%	Cup	1.89%	Banana	0.0%	Nothing 0.0%
77	/data/test_images/test_76.png	Phone	97.01%	Cup	2.99%	Banana	0.01%	Nothing 0.0%
78	/data/test_images/test_77.png	Phone	94.31%	Cup	5.68%	Banana	0.01%	Nothing 0.0%
79	/data/test_images/test_78.png	Phone	98.72%	Cup	1.28%	Banana	0.0%	Nothing 0.0%
80	/data/test_images/test_79.png	Phone	98.86%	Cup	1.14%	Banana	0.0%	Nothing 0.0%

© Thomas Hatting

Figure 8: Test of the model (evaluation) using “Classify Many” option in the DIGITS GUI

6. Discussion

In general, the generated models in this project were highly accurate. This is likely due to a combination of good quality data and choosing the appropriate deep neural network as well as using suitable parameter values.

In this project, accuracy seems more important than inference time. After all, it is more important that the inference system correctly identifies the object early on. If the object is identified incorrectly, the conversational interface would consequently engage in a conversation with the child, which may prove meaningless and confusing. A high inference time would cause a delay in the system, however this delay might be less annoying than a system engaging in incorrect conversations.

In any case, the inference time in this project was not particularly high. It was not possible to measure the inference time directly (as the evaluate command in the DIGITS Workspace did not work for the collected data). However, when a similar model was generated with the pre-supplied data (/data/P1_data), it showed an inference time of 5 to 5.7 ms, which is within acceptable limits.

7. Conclusion

Children learn to speak through ordinary everyday interaction: the child is spoken and listened to, his or her questions are answered and questions are posed. This paper proposes a speech and language development system for toddlers (children aged 1-3 years). The idea is that the child holds up an ordinary (and safe to handle) household object in front of the camera. The robotic inference system identifies the object and initiates a simple conversation with the child.

The data were collected using an integrated webcam on a laptop. This solution was chosen because it provided a quick and efficient way of collecting data. Four object categories were defined: fruit, cup, phone and nothing. In all 1500 images were collected for each category using a Python script.

Three deep neural network architectures were considered to generate a suitable model: LeNet (Lecun et al. 1998), AlexNet (Krizhevsky et al. 2012) and GoogLeNet (Szegedy et al. 2014). In the end, a model was chosen using the GoogLeNet architecture. This model provided an accuracy of 98.5%, according to the DIGITS GUI. This model was tested with a separate batch of images. In all, 79 out of 80 images were identified correctly.

Future work includes implementing the robotic inference system on the Jetson TX2 platform. Due to time constraints, it was not possible to implement this at present moment. The Jetson platform provides an “at-the-edge” solution rather than a round-trip through the cloud. This can provide greater privacy and security compared to a cloud solution, which could be an important factor when parents decide whether or not to purchase the system.

Further work also includes developing a mechanism whereby the robotic inference system activates a conversational interface which engages the child in a conversation. The conversational interface could be implemented as a cloud solution using Amazon Lex (Amazon 2018), or, if an “at-the-edge” solution is preferred, it could be implemented with an an interface such as Snips (Snips 2018).

References

Amazon (2018). Amazon Lex Developer Guide available online: https://docs.aws.amazon.com/lex/latest/dg/what-is.html

Canziani, A., Culurciello,E. and Paszke,A. (2017) “An Analysis of Deep Neural Network Models for Practical Applications”. Weldon School of Biomedical Engineering, Purdue University, and Faculty of Mathematics, Informatics and Mechanics, University of Warsaw. Available online: https://arxiv.org/pdf/1605.07678.pdf

Eghan, A. (2016) “Bridge the word gap: speak 21,000 words to your preschooler daily”. February 25, 2016. GreatSchools. Available online: https://www.greatschools.org/gk/articles/word-gap-speak-more-words-to-your-preschooler-daily/

Helsinki (2018). Speech and Language Development. City of Helsinki, Finland. Available online:https://www.hel.fi/sote/perheentuki-en/1-6-year-olds/growth-and-development-of-children/speech-and-language-development/

Krizhevsky, A., Sutskever, I. and Hinton, G. (2012) “ImageNet Classification with Deep Convolutional Neural Networks”. University of Toronto. Available online: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998) “Gradient- Based Learning Applied to Document Recognition”. Available online: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

Psychology Today (2014). “Tackling the vocabulary gap between rich and poor children” https://www.psychologytoday.com/blog/the-athletes-way/201402/tackling-the-vocabulary-gap-between-rich-and-poor-children

Snips (2018). Snips Platform documentation available on Github: https://github.com/snipsco/snips-platform-documentation/wiki

Szegedy, C, Liu, W., Yangqing, J., Sermanet, P, Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2014) “Going deeper with convolutions”. Google Inc, University of North Carolina and University of Michigan. Available online: https://arxiv.org/pdf/1409.4842.pdf

Wikipedia (2018). “Language Development”. Article available online: https://en.wikipedia.org/wiki/Language_development

Appendix

Python script to collect grayscale images with a resolution of 256x256 (adaptation of script provided by Udacity):

```
import cv2
import time

# Run this script from the same directory as your Data folder

# Grab your webcam on local machine
cap = cv2.VideoCapture(0)

# Give image a name type
name_type = 'Cup'

print ("Photo capture enabled!")

for i in range (1500):

    time.sleep(0.5)

    # Read in single frame from webcam
    ret, frame = cap.read()

    # Use this line locally to display the current frame
    cv2.imshow('Color Picture', frame)

    # If you want them gray
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

    # If you want to resize the image
    gray_resize = cv2.resize(gray,(256,256), interpolation = cv2.INTER_NEAREST)

    # Save the image
    cv2.imwrite(name_type + "_" + str(i) + ".png", gray_resize)

    print ("Saving image number: " + str(i))

    #Press q to quit the program
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

© Thomas Hatting