2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics
International Conference (PRASA-RobMech)
Bloemfontein, South Africa, November 29 - December 1, 2017

Automated Parking Space Detection Using Convolutional Neural Networks

Julien Nyambal
School of Computer Science and
Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
Email: julien.nyambal1@students.wits.ac.za

Richard Klein
School of Computer Science and
Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
Email: richard.klein@wits.ac.za

***Abstract*—Finding a parking space nowadays becomes an issue that is not to be neglected, it consumes time and energy. We have used computer vision techniques to infer the state of the parking lot given the data collected from the University of The Witwatersrand. This paper presents an approach for a real-time parking space classification based on Convolutional Neural Networks (CNN) using Caffe and Nvidia DiGITS framework. The training process has been done using DiGITS and the output is a caffemodel used for predictions to detect vacant and occupied parking spots. The system checks a defined area whether a parking spot (bounding boxes defined at initialization of the system) is containing a car or not (occupied or vacant). Those bounding boxes coordinates are saved from a frame of the video of the parking lot in a JSON format, to be later used by the system for sequential prediction on each parking spot.**
The system has been trained using the LeNet network with the Nesterov's Accelerated Gradient as solver and the AlexNet network with the Stochastic Gradient Descent as solver. We were able to get an accuracy on the validation set of 99% for both networks. The accuracy on a foreign dataset(PKLot) returned as well 99%. Those are experimental results based on the training set show how robust the system can be when the prediction has to take place in a different parking space.

I. Introduction

As the population grows, the number of private vehicles increases as well. But the number of parking spaces most of the time remains.

. Sometimes, and most of the time, there are vacant spots, but the drivers do not have any information about them. It could be, either the free spot is far from them, or it is hidden by some other cars or any other objects big enough to hide the spot. In the past, and maybe at some places now, parking spaces are managed by some persons in the parking lot who might not have the total view of the next available parking space. Sometimes the driver him/herself has to check for a vacant space by circling in the parking lot, and another driver will come and many losses are generated: time, fuel and maybe temper. Some researchers came up with different parking space detection algorithms using some gadgets like video cameras or sensors to detect a vacant spot when needed. Those algorithms combined gave birth to automated parking detection.

The target of the system is mostly open areas like shopping mall parking lots or university parking lots. Implementing a sensor-based approach to automating the parking spot detection will require paperwork, time will be wasted and more trouble to the users when trying to park their vehicles, while installing those devices on the ground. On the other hand, those parking lots are monitored by video cameras for security purposes, which is the case in most shopping malls as well. Video-based detection for detection is being used in many areas like obstacle detection, human detection [1].

Convolutional Neural Networks architecture (CNN) are similar to the human neural network build with synapses (weights) and neurons. From this point of view, complex tasks can be learned through the network. This uses CNN with preexisting architectures to detect in real-time the vacancy of a parking spot.

II. Related Work

A sensor-based approach can be used to detect a moving object entering the parking lot. The size of the object and the time it takes to cross the gate where the sensor is located infers the occupation of a spot by the car [2]. Due to the cost linked to the installation and maintenance of sensors, their use is broad but restricted when budget is an issue. Another big issue of the sensors, they cannot be implemented outdoors, because there is no roof to install a sensor.

The approach in [3] presents a video-based system for parking space classification that has been tested and resulted in a success when it has been also working on another parking lot without re-training the model. To solve the problem of occupancy of a parking spot, color histograms and Difference-of-Gaussian (DoG) have been used for feature extraction, and three machine learning algorithms have used for classifying between a free or occupied parking spot. Those three learning algorithms are: K-Nearest Neighbor (k-NN), Linear Discriminant Analysis (LDA) and the Support Vector Machines (SVM).

To avoid errors during the classification process, temporal integration exponential smoothing has been applied to the classification results. DoG and color histograms (features extractors) have been combined to the three machine learning algorithms in a one-to-one relationship, to produce the best performing combination of features extractors and machine learning algorithms used. The system has been tested with a

978-1-5386-2313-8/17/$31.00 ©2017 IEEE

success rate of more than 92% on an unseen parking lot, which is a success based on the problem stated by the author [3].

Wu et al. [4] proposed a method for detecting available parking space using both Support Vector Machines and a Markov Random Field framework. The input is a video, and from the images, the system distinguishes between empty spaces from occupied spaces in a parking lot. The system is trained to recognize empty parking spaces by applying a machine learning algorithm rather than segmenting the image of the vehicle from the parking space. The system is composed of 4 parts: pre-processing, ground model feature extraction, multi-class Support Vector Machines recognition, and Markov Random Field based correction. Features are extracted from the color histogram, 50 features are selected after dimension reduction. Support vector machines are used to classify the occupancy of a parking spot based on the patches of the parking lot manually determined. Markov Random Field (MRF) is used to improve the performance done by the Support Vector Machines. The results of the experiments show that Support Vector Machines used without Markov Random Field produce 84.35% using a single space for detection and 85.57% using 3 spots. When using Markov Random Field, the accuracy climbs to 93.53%. The accuracy increases with the training sample, which shows the strength of the algorithm used.

A robust dataset of images of parking spaces under different weather conditions can be used to better generalize classification of the state of a spot [5]. The authors use of the Local Binary Pattern (LPB) and the Local Phase Quantization (LPQ) textual descriptors for features extraction. For their classifier, they combined the Support Vector Machines with some variants of the LPQ and the LPB which are the LBP Rotation Invariant, LPQ with Gaussian window, and the LPQ gaussian derivative quadrature filter pair to achieve a classification rate of 99.64%.

Amato et al. [6] present an approach of detecting parking spot using deep learning techniques. The authors trained their classifier on two datasets: PKLot [5], and their own dataset from the National Research Council in Pira (CNRPark). Two network configurations have been used: mAlexNet: is a mini version of the AlexNet network and mLeNet: is the mini version of the LeNet network.

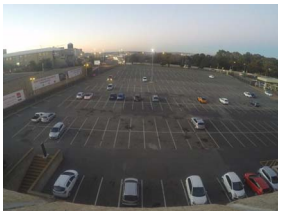
Those mini-networks are modified well-known CNN architectures (AlexNet and LeNet respectively) to accelerate the process of classification, as the original ones are very expensive in time when resources are not equivalent to the big size of the network. This method produces an accuracy greater than 90% inter and intra-datasets.

III. The dataset

The dataset is made of 782 images of parking lots. Those parking spots have been extracted from 2 parking lots of the University of the Witwatersrand.

A. Data Collection

The data collection, using a GoPro Hero 4 Black camera, took a working week (5 days), from 6 am to approximately 8 am at the rate of one frame per minute. During that period of time, the parking gets filled actively and rapid change of illumination is captured. The camera was set to take pictures at 5 MegaPixels which is in a resolution of 2560 x 1920 pixels. Due to the "*Wide*" setting of the GoPro, severe radial distortion appears on all the frames. It is of type Barrel due to the fisheye lens of the camera [7], which affects the straight lines of the parking spots as they tend to be curved. The position of the camera and the choice of the buildings have been chosen in such a way that a car from a parking spot does not hide more than 60% of the next car around.



(a) Original image with radial distortion of type Barrel.

(b) Image with radial distortion corrected.

Fig. 1: Correction of radial distortion (Chamber of Mines, University of the Witwatersrand).

The radial distortion (Barel) has been manually fixed using *distort* and the size of the images manually fixed using *convert*. *Distort* and *convert* are both Unix utilities [8]. Each frame has been resized from 2560 x 1920 pixels to 1000 x 1000 pixels. Fig. 1a is the original frame captured by the camera that is distorted due to the fisheye lens of the camera. Fig. 1b is the result of all the manipulations on the original images coming directly from the camera to correct the distortion.

B. Labeling

The process used is to crop from each image of a parking lot each individual parking spot so that a label (occupied or vacant) can be manually assigned to it. To do so, the cartesian coordinates of each spot stored in an XML file generated by a labeling tool will help to extract them from the image.

That tool extracts the patches of all the manually selected parking spots, using the XML file generated from a single image. Since the parking lot does not move, the same XML file has been reused on the other images to get the multiple patches [9], [10]. Fig. 2a and Fig. 2b represent respectively the cropped image of an occupied and vacant parking spot.

After the labeling, vacant and occupied parking spots patches have been separated into different folders, vacant and occupied folders respectively. The raw dataset contains 782 images of parking lots.

(a) Occupied parking spot.

(b) Vacant parking spot.

Fig. 2: Samples of vacant and occupied parking spot.

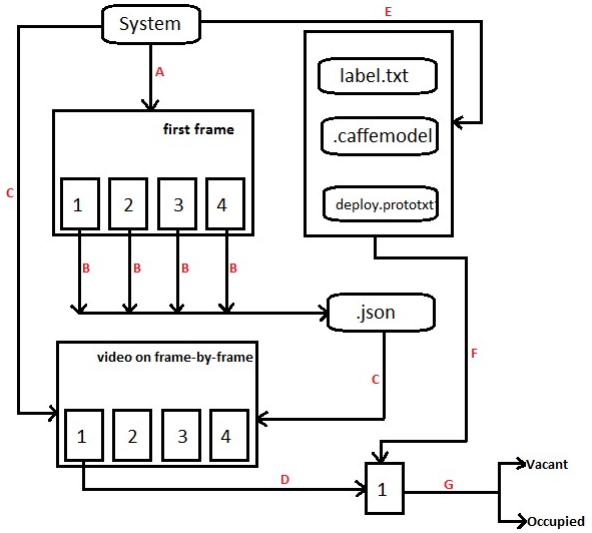


Fig. 3: System Processing Flow.

IV. Deep Learning with Caffe and Nvidia DIGITS

A. The System

The system built works on the fact that the images of the parking spot will represent a fixed parking lot.

The system analyzes the parking lot after every fifth frame. A frame represents a state of the parking spot at a precise moment. The system works on live feed or on a recorded video. Fig. 3 represents the flow of the data from raw images from video frames to take a decision on the vacancy of a parking spot.

On Fig. 3, *first frame* and *video on frame-by-frame* represent a state of the parking lot and "1", "2", "3", "4" represent the parking spots on each frame.

A- The System launches the frame required to define the parking spots,
B- The coordinates of the parking spots are stored in a JSON file (.json), will get generated if it does not exist,
C- The System loads the video and the JSON file (.json),
D- Fig. 3 corresponds to one iteration of the loop in the number of parking spots. The first iteration will crop the spot one from the frame *n* of the loaded video,
E- The System loads the labels file, the caffemodel, and the deploy file corresponding to the caffemodel,
F- Those three arguments on the system are applied to the cropped image of the parking lot, save on the disk. In Fig. 3, the cropped image is the parking spot "1",
G- The result of the prediction from the System with the arguments applied on the frame *n*. A *red* or *green* rectangle corresponding respectively to either an occupied or vacant parking spot will surround the cropped image after the classification.

B. Configuration

Nvidia Deep Learning package DIGITS, which a web server that manages the activities of the GPU (Nvidia only) to efficiently perform the classification tasks. The background heavy tasks are done with Caffe. was used for both the training and testing phases. It has been used for training and testing.

The CPU could also be used for training and testing but the performances will be relatively slower if a single CPU is to be used.

1) Hardware and software: The implementation of the system has been done on a desktop with an Intel i5 processor at 3.20GHz with 4 cores. The GPU used is a Nvidia GeForce GTX 750 Ti with 2GB video RAM.

The system is trained under Ubuntu 14.04 with Nvidia DIGITS 3, using cuDNN [11] which has been shown to accelerate CNN training and testing processes [12].

2) Different Convolutional Neural Networks Configurations Used: Two CNNs architectures were used and compared, AlexNet [13] and LeNet [14]. AlexNet is very good for image classification, first due to the depth of the architecture, and other factors like dropouts. Another important factor to note, AlexNet has been trained on 1000 classes with more than 1 million images, which implies that the weights are the depth of the network could be reused for finetuning. with some tweaking LeNet in its parameters performs better.

LeNet architecture takes as input grayscale images of size 28x28, whereas AlexNet takes as input color images of size 256x256. All images are normalized by subtracting the training mean from all the images before the training begins.

AlexNet has been slightly modified as follow: five convolutional layers, followed by max pooling, norm, ReLU layers and three fully connected layers.

LeNet as well has been slightly modified as follows: two convolutional layers, followed by max-pooling layers and two inner products layers.

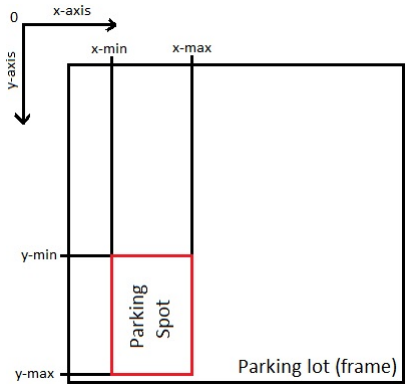


Fig. 4: Definition of x-min, x-max, y-min, y-max of a single spot. Those coordinates are relative to the current image of the parking lot and will not change since the setup of the frame will remain static.

C. Processing and Prediction

We show some details of the processing that happens in the system, from the image acquisition to the prediction of the specific status of the selected parking spot.

1) Image Acquisition: The images are taken from a video of a parking lot stored on a local workstation. Using OpenCV libraries, the video is read and generates frames to perform the predictions.

2) Definition of Parking Spots: The first frame of the footage (or a frame that contains mostly occupied parking spots) is used to define what a parking spot is in that current frame, when the program starts. This technique of defining a parking spot on one frame relies on the fact that the camera will not move during the whole process of prediction and logically the parking lot does not move, only the cars are moving in or out the different defined parking spots. Therefore this definition applies for the subsequent frames. The user defines the spot by clicking on two corners of the parking spot (the upper corner left and the lower corner right). After that set of clicks, the parking spot is defined and classified on the go.

3) Storage of Coordinates on JSON file: A JSON file is created by the system to hold the coordinates of the different parking spots defined by the user. The JSON coordinates (terminologies used in the JSON format are explicitly explained in Fig. 4) are stored in the following format:

```
{
"spot_1": {              /*Spot identifier*/
"name": "spot_1",        /*Spot identifier*/
"xmax": 448,  /*Spot maximum x-coordinate*/
"xmin": 345,  /*Spot minimum x-coordinate*/
"ymax": 542,  /*Spot maximum y-coordinate*/
"ymin": 478   /*Spot minimum y-coordinate*/
}, ...
```

4) Prediction and classification: The JSON file previously generated is used to perform the classification of each spot.

Algorithm 1 Classification

Input: model, deploy_file, cropped_image, label_file
Output: The actual class of the cropped image (**string**)

```
status = { }; Dictionary holding each class and their corresponding confidence level
i = 0
result_status = " ", actual name of the class.
while all the labels and confidences of each predictions do
    status[confidence(i)] = confidence
    status[label(i)] = label
    i ← i + 1
end while
if status[confidence0] > status[confidence1] then
    result_status = status[label0]
else
    result_status = status[label1]
end if
return result_status
```

The following set of operations will take place:

1) **Cropping**: OpenCV allows processing the spots on a frame-by-frame basis. The system take as parameter the JSON file that contains the set of coordinates and the video. Parking spots will be cropped from each frame of the video based on the JSON file. The cropping is done sequentially and the corresponding image is stored on the disk, in other words, only one cropped image is stored and the next picture overrides it later on.
2) **Prediction and classification**: After the training phase, the generated Convolutional Neural Network model comes along with two other files: *label.txt* containing the different label of a spot (vacant or occupied) and the *deploy.prototxt*, which is the network configuration used by the model. This file defines the type of input allowed by the model and the protocol followed to produce the output. The system takes as arguments the model created at the training phase, the corresponding labels, the network configuration (deploy.prototxt), and the cropped image of the parking (one at the time). Those arguments are processed through a borrowed function written by Nvidia, classify[15], which is a part of DiGITS. Modifications were made only in the classify function as shown in Algorithm 1. Those modifications allow to visually determine whether a spot is either vacant (green rectangle) or occupied (red rectangle), based on the score or confidence level (result of the probability model computed) produced by the classification method. Those results are then streamed to the main system to produce the required output.

(a) Prediction on some spots 3. The system does not get triggered with a human passing through the classification area.

(b) Prediction on spot 2. The street sign does not trigger the classifier.

Fig. 5: System in action on predicting vacancy of parking spot.

TABLE I: Configuration of the 2 datasets used for to train the system

Dataset	Dimensions	Size	Entries (train & val)
Parking(Colour)	256x256x3	84.3MB	2026 & 676
Parking(Grayscale)	28x28x1	2.16MB	2026 & 676

V. Experiments and Results

For the experiments, the comparison of LeNet and AlexNet allowed us to run the classification on a recorded video[1] containing a parking lot, with some modifications on the hardware and the weight of the inputs (images) that each network has to process.

The next section presents the different set of CNN and parameters employed and the corresponding performances on the earlier stated video.

A. Dataset: Input to the Convolutional Neural Networks

From Section III, a set of images of both statuses a parking spot: occupied or vacant. As shown in Table I, we have a very small number of images in the dataset for training compared to the number of images in the dataset from Section III. Many instances of identical pictures were removed to avoid any type of overfitting. Based on the two selected CNNs to develop the system, we shaped the data according to the input requirements of the concerned CNN.

For the AlexNet, all the images are resized to 256x256. Those images are in RGB.

For the LeNet, all the images are resized to 28x28. Those images are in grayscale.

From Fig. 6, the image mean is giving a clear approximation of all the status of a parking spot at any given time, including the parking lines.

Table I shows the split of the dataset used for training and validation processes with the ratio of 75% for training and 25% for testing.

[1]https://www.youtube.com/watch?v=U7HRKjlXK-Y

Fig. 6: Image Mean of the dataset: Vacant and Occupied spots.

TABLE II: All configurations for the system training using AlexNet and LeNet using different types of solvers

Network	Solver	Accuracy(Train)	Accuracy(Test)
AlexNet	SGD	0.99	0.95
AlexNet	NAG	0.99	Never Ended
AlexNet	AdaGrad	0.98	0.89
LeNet	SGD	0.99	0.92
LeNet	NAG	0.99	0.93
LeNet	AdaGrad	0.99	0.89

B. Configurations of Convolutional Neural Networks

Given the datasets generated from the Section V-A, we built some CNN configurations, based on LeNet and AlexNet, to use them in the system for classification. For all the network configurations, the Base Learning Rate or *Base lr* is initialized at 0.01. The Learning Rate Policy or *Policy lr* is the way the learning rate will change with time. The learning rate is be divided by 10 after each 33% portion of the training set. Three solvers for the classifier are mainly used to compare the results. The three solvers used are:

1) Stochastic Gradient Descent (SGD)
2) Adaptive Gradient Descent (AdaGrad)
3) Nesterov's Accelerated Gradient (NAG)

We have made 30 passes through the training set (epochs). Table II is the summary of all the custom configurations to get the best accuracy with the validation set along with the speed performance. The validation set is made out of 25% of the entire set. The validation set is entirely independent of the training set.

C. Results

After the training phase, very high accuracy values are noted at validation level based on Table II (accuracy calculated during training with images that are closely similar to the training set, but not in the training set). For the accuracy of the system, PKLot dataset has been used [5] to assess the strength of the system to classify on a foreign parking lot.

Table II shows the testing results of the classifier after the training and validation process. The test data has been generated from unseen images of parking spots to the classifier. The performance of all different CNN configurations has been

tested against the test images. All those configurations received the same batch of test images at once to simulate a close to real-time heavy flow. In the same table, during testing AlexNet with the Nesterov's Accelerated Gradient, the batch of images took very long to terminate. Giving that the system is meant for a real-time purpose, time is an important factor to monitor and delays are considered as failure.

Fig. 5a and Fig. 5b represent the results produced by the system with AlexNet as classifier. The change of lighting, which is a big factor misbehavior of many classifiers, does not affect too much the classification done by the system trained on this relatively small dataset. The classification runs on a single frame rather than a stream of frames, where instead of 25 frames per second (fps), it classifies on 5 fps, which takes into account the 2GB of the Nvidia graphic memory.

VI. Conclusion

In this paper, we have shown with success that computer vision with a single camera, using Convolutional Neural Networks is having a success rate close to the traditional methods used in the past. The classifier made of LeNet architecture with the Nesterov's Accelerated Gradient as solver produced an accuracy of 93.64%, and the AlexNet architecture with the Stochastic Gradient Descent as solver produced an accuracy of 95.49% which are better than both Amato et al. [6] and Wu et al. [4] who produced respectively 90.4% and 93.52%. Given the hardware used for the research, the LeNet architecture with the Nesterov's Accelerated Gradient is faster than the AlexNet architecture with the Stochastic Gradient Descent when tested.

The use of Convolutional Neural Networks facilitates the process of building a classifier as they automatically extract and use the features from the dataset. As shown in Fig. 5, the system is able to produce an accurate output based on the situation of the current parking spot the camera is facing, given that spot is well defined by the user when launching the system.

We have noticed during the research the presence of noise due to either the light, some elements stuck on the ground (oil leaks, cracks ...) or humans passing across parking spots. Those events have been corrected by improving the dataset based on the event of false classification due to noise. We have obtained the best combination in terms of parameters of the classifier by having the Nesterov's Accelerated Gradient as solver and LeNet as network architecture, especially for the speed of execution compared to the better accuracy produced by the AlexNet architecture with the Stochastic Gradient, but with poor results when it comes to speed (speed is entirely relative to the hardware used). The classification on video is faster and accurate enough to be deployed in real life situation, on a parking that does not have any image appearing in the training set. It implies that the system can be used to a new parking without retraining of the classifier (given that the amount of noise is under control. E.g.: lighting, oil or any type of leaks on the ground...).

As future work, we are planning on first improving the positioning of the camera so that we get a clear view of each car on the parking lot without a car hiding any part of the next car. Given that the availability of the parking spots is deduced from a well-defined parking lot, the system could direct the driver to the nearest available spot using some artificial intelligence techniques.

Acknowledgment

I would like to thank the Council for Scientific and Industrial Research (CSIR), Adrian [16], Prof. Turgay Çelik and the PIMD (Property and Infrastructure Management Division) at the University of the Witwatersrand for their contribution in this research project.

References

[1] I. Ohya, A. Kosaka, and A. Kak, "Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 969–978, Dec 1998.

[2] S. Lee, D. Yoon, and A. Ghosh, "Intelligent parking lot application using wireless sensor networks," in *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*, pp. 48–57, May 2008.

[3] M. Tschentscher, C. Koch, M. Knig, J. Salmen, and M. Schlipsing, "Scalable real-time parking lot classification: An evaluation of image features and supervised learning algorithms," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2015.

[4] Q. Wu, C. Huang, S. y. Wang, W. c. Chiu, and T. Chen, "Robust parking space detection considering inter-space correlation," in *2007 IEEE International Conference on Multimedia and Expo*, pp. 659–662, July 2007.

[5] P. R. de Almeida, L. S. Oliveira, A. S. B. Jr., E. J. S. Jr., and A. L. Koerich, "{PKLot} a robust dataset for parking lot classification," *Expert Systems with Applications*, vol. 42, no. 11, pp. 4937 – 4949, 2015.

[6] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Car parking occupancy detection using smart camera networks and deep learning," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, pp. 1212–1217, June 2016.

[7] S. Shah and J. K. Aggarwal, "A simple calibration procedure for fish-eye (high distortion) lens camera," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 3422–3427 vol.4, May 1994.

[8] ImageMagick, "Features and capabilities." http://www.imagemagick.org/script/index.php, June 2017. (Accessed on 10/01/2016).

[9] TzuTaLin, "tzutalin/labelimg: Labelimg is a graphical image annotation tool and label object bounding boxes in images." https://github.com/tzutalin/labelImg, June 2017. (Accessed on 09/30/2016).

[10] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1, pp. 157–173, 2008.

[11] NVIDIA, "Nvidia cudnn - gpu accelerated deep learning." https://developer.nvidia.com/cudnn, June 2016. (Accessed on 10/01/2016).

[12] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *CoRR*, vol. abs/1410.0759, 2014.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] NVIDIA, "Classification - nvidia." https://github.com/NVIDIA/DIGITS/blob/master/examples/classification/example.py, June 2016. (Accessed on 10/10/2017).

[16] Rosebrock, "Capturing mouse click events with python and opencv." http://www.pyimagesearch.com/2015/03/09/capturing-mouse-click-events-with-python-and-opencv/, October 2016. (Accessed on 10/01/2016).