Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 51 (2020) 32–37

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021)
15-18 June 2021, Athens, Greece.

Cognitive Grasping System: A Grasping Solution for Industrial Robotic Manipulation using Convolutional Neural Network

Lucia Biagetti[a,*], Amrit Kochar[b], Cristina Cristalli[c], Simonetta Boria[a]

[a] *School of Sciences and Technologies, University of Camerino, Via Madonna delle Carceri 9, 62032 Camerino, Italy*
[b] *Computer Science & Information Systems, Birla Institute of Technology and Science Pilani, 333031 Pilani, India*
[c] *Research for Innovation, Loccioni, Via Fiume 16, 60030 Angeli di Rosora, Italy*

Abstract

In the modern era, object grasping has thousands of use cases across industries and loads of manual effort is devoted to repetitive tasks. Automating this task is important and the use of robots with embedded artificial intelligence is the key for improving grasping operations. Over the years many researches have been working on object grasping to make this operation as flexible as possible. Starting from the latest results of the use of Convolutional Neural Network, the proposed work aims at optimizing the results of the grasping tasks to make it reliable for an industrial use. Limitations are analyzed and new parameters are defined in order to make the manipulation task repeatable in terms of robot grasp position. In fact, in an automated production line, this is an important problem to consider because in many situations the object has to be positioned always in the same position with the same orientation.

Keywords: Industry 4.0; Robotic Manipulation; Robotics; Collaborative Robots; Industrial Automation; Deep Learning; Convolutional Neural Networks;

1. Introduction

The Industry 4.0 paradigm [1] is essentially a world-wide effort to redefine how the production processes will be in the future. Despite all the advancements in building the 4.0 framework, the task of picking up a multitude of different work pieces every hour on large scale production lines is done manually or by re-programming a robotic arm each time a new object is encountered. Automated universal picking has been a topic of active research in the last decade with the main objective being to obtain good results for grasping of objects with high repeatability, accuracy, and speed without any a priori information with respect the object to grab.

There have been various approaches studied by researchers all around the world to get the most efficient solution for the robotic grasping problem, from a Support Vector Machine (SVM) powered solution [2], to Google cloud-based solution [3], to data-driven solutions [4]. Across most of the work carried out in the past, the aim was always to avoid the reprogramming of the robot every time there is a new object to grab.

GraspIt [5] for example has been used to simulate grasps and to generate datasets, but no real life grasps are acquired to build the dataset. Despite the simulator takes into account a lot of factors, several other practical parameters have not been considered in its methodology. For example, in the solution where SVM has been used to train the model, the authors choose simulated grasps by having only 16 distinct starting positions for the gripper and then vary the angles to populate the training set. It would always be an utopian vision to have a truly generalized training dataset generation using a simulation software. In every proposed solution, the training and testing data points are limited and this influenced the quality of the result. For example, in the SVM model, only 1350 grasps have been used to train the model and another 150 have been used for test set [2]. Over the years, there have been a few interesting approaches to solve the grasping problem, as suggested by [6], proposing a learning approach that uses visual features to compute the grasp point for an object just based on 2D images, eliminating the need to build 3D models. In the last few years, some very innovative and efficient solutions for the robotic grasping prob-

* Corresponding author. Tel.: +39-3349996309
E-mail address: lucia.biagetti@unicam.it (Lucia Biagetti).

2351-9789
10.1016/j.promfg.2020.10.006

lem, have been proposed ranging from machine learning based solutions, to vision based systems, to building a system using hand-eye coordination [7–14].
The training of a model remains an important aspect to deal with and there is a pertinent literature that contains various examples of how the training of a model with bigger datasets can produce high quality results, with one of the most popular examples being ImageNet [15]. This trend of using big data to train new-age models is growing, thanks to the tremendous improvement in quality owing to a bigger dataset [16, 17]. One of the biggest datasets for the robotic grasping problem has been created by Berkeleys AUTOLab team and it is called the Dexterity Network (Dex-Net). It has over 6 million data point clouds, and grasp metrics generated using over 1000 objects [18]. This data repository has been the state-of-the-art since it was published and has been used to get a big improvement in the quality of solutions as compared to previous solutions. This is thanks to the fact that a classification model capable of recognizing objects regardless of their shape or material has been introduced. More specifically the classification model is based on a particular type of Convolutional Neural Network called Grasp Quality Convolutional Neural Network, GQCNN, were the grasping problem is solved without reprogramming the robot or providing a different dataset made of different objects each time [10].
However, the positions where the object is grabbed are not highly repeatable and this aspect in an industrial automatic system is not negligible. In fact, in the majority of the applications the object has to be moved from one position to another one avoiding multiple manipulations. For this reason, starting from the work done on GQCNN, the proposed Cognitive Grasping System aims at optimizing the grasp position, i.e. the grasp has to be as close as possible to the center of the object, and the grasp repeatability, i.e. the grasp has to be always in the same position independently from the orientation of the object. The paper, after a description of the main parameters that characterize the results of the proposed GQCNN, presents the influence of different parameters on the quality of the grasp, such as: 2D images, the hyper-parameters, the network architecture. Extensive tests have been performed in real industrial use cases in order to measure the improvements of the grasping based on the aforementioned criteria.
The paper is articulated in the following sections: next section provides the problem statement for the proposed work; section 3 describes the methodology followed for this study; section 4 presents the experimental setup; section 5 provides achieved results; section 6 contains some concluding discussion and remarks.

2. Problem Statement

In the industrial environment, the need to automatize the repetitive tasks in order to optimize the time and resource usage is more and more important. For this reason, a so called Cognitive Grasping System has been developed starting from the research performed on the GQCNN that has been properly modified and optimized. The system is composed of a 3D camera, a six-axis robot, a two finger gripper and an embedded board where the modified GQCNN is running.
The Convolutional Neural Network has the following architecture:

- The network input consists of a depth image centered on the grasp center pixel and aligned with the grasp axis orientation, acquired through the camera.
- 4 convolutional layers in pair of two separated by ReLU activation functions. This type of function stands for rectified linear units and it is commonly used in Convolutional Neural Networks [19].
- 3 fully connected layers.
- A further fully connected layer has been added with a further input representing the distance of the gripper from the camera.

The output consists of a pose, which represents the position in the object's gripping space, and the estimation of the probability of a grasp success $Q_\theta \in [0, 1]$.
This network has been already trained on the Dex-Net 2.0 dataset [10], providing a pretrained model. In this work the architecture of the network is the same as mentioned above, while the Hyperparameters used and the dataset on which the network was trained have been changed. In particular, starting from the pretrained model and modifying some of the fundamental parameters, a fine-tuning of the network has been performed on a smaller dataset, consisting of a part of the Dex-Net database and a small group of grasp images of the two objects used for this work. The aim is that the optimal grasp has to be as close as possible to the center of the object, the angle has to be as parallel as possible to the object and the grasp has to take place in the same conditions regardless of the object taken into account; this to ensure higher repeatability from the industrial point of view. The function to be optimized is the robust grasping policy, described as

$$\pi(y) = \underset{u \in C}{\operatorname{argmax}}\, Q(u, y) \tag{1}$$

where the set C is a discrete set of antipodal candidate grasps, u is the parallel-jaw grasp in the 3D space, y is the depth image taken from the camera, Q is the robustness function. Each grasp candidate is evaluated by the GQCNN, and the most robust grasp is executed.

3. Methodology

The base structure of GQCNN's is used as a starting point. After experimenting with several modifications a final model is prepared which obtains high quality grasps. A study of the Hyperparameters, that characterize the neural network, has been done and different solutions have been found.

The Hyperparameters taken into consideration are the following:

- Loss Function,
- Optimizer,
- Momentum rate,
- Learning rate,
- Regularizer,
- Weight decay rate.

Starting from the description of the loss function, which is the penalty obtained from a bad prediction, the best function to apply to this multi-class classification problem has been the Softmax CrossEntropy loss function with logits. The term logits stands for the vector of raw predictions that a classification model generates, which is the input of the Softmax function. Introducing the definition of the Softmax Cross Entropy loss function, it consists of a Softmax activation function plus a CrossEntropy loss function and it is used to train a Convolutional Neural Network, obtaining as output a probability over the number of classes for each image.
Let C be the number of classes, the Softmax function is defined as

$$\text{softmax}(h)_i = \frac{\exp(h_i)}{\sum_{c=1}^{C} \exp(h_i)}, \quad c \in [C] \tag{2}$$

where h is termed as logit. Considering a CNN learning a nonlinear mapping from the input $x \in \mathbb{R}^p$ to the feature $z = Z(x) \in \mathbb{R}^d$, the Softmax CrossEntropy (SCE) loss is defined as follow:

$$L_{SCE}(Z(x), y) = -1_y^T \log[\text{softmax}(Wz + b)], \tag{3}$$

for a single input-label pair (x, y), where 1_y is the one-hot encoding of y and the logarithm is defined as element-wise. Here W and b are the weight matrix and bias vector of the SCE loss, respectively.
The Sparse Softmax Cross Entropy loss function with logits performs the same cross-entropy calculation of error, without requiring that the target variable should be one hot encoded prior to training and it is widely used in multi-class classification problems and for this reason it has been chosen [20].
Regarding the optimizer, a comparison between the Stochastic Gradient Descent (SGD) with momentum [21] and the Adam optimizer [22] has been performed.
The SGD is an optimization algorithm which performs a series of steps to minimize the loss function. It oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum. Momentum is a method that helps accelerate SGD in the relevant direction and dampens oscillations. It does this by adding a fraction γ of the updated vector of the past time step to the current update vector. Given a loss function $J(\theta)$ to be minimized, classical momentum is given by:

$$v_{t+1} = \gamma v_t - \eta \nabla_\theta J(\theta_t) \tag{4}$$

$$\theta_{t+1} = \theta_t + v_{t+1} \tag{5}$$

where θ is the parameter, so the weights, biases or activation, v_t is the last update of θ, $\eta > 0$ the learning rate, γ the momentum term, and $\nabla_\theta J(\theta)$ is the gradient at θ_t. The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, faster convergence and reduced oscillation are obtained. After trying Adam Optimizer also, it results that Adam in terms of optimization and speed is better than SGD with momentum, but in some areas it did not converge to an optimal solution. So for some tasks, state-of-the-art results were achieved by applying SGD with momentum.
Regarding the choice of the value of the momentum term, common values in practice are 0.5, 0.9, and 0.99. After all of them were tried, momentum term has been set equal to 0.9, because a small difference on the performance of the network has been noticed in term of good and not good grasps.
About the learning rate η [23], smaller learning rates require more training epochs, conversely larger learning rates require fewer training epochs. Further, smaller batch sizes are better suited to smaller learning rates given the noisy estimate of the error gradient. By this consideration the learning rate has been set to $\eta = 0.1$ for 25 epochs with a batch size of 64.
To avoid overfitting a L2 regularizer [24] has been inserted, that is one of the most common type of regularization. This updates the general cost function by adding another term known as the regularization term. Due to the addition of this regularization term, the values of weights matrices decrease because a neural network with smaller weight matrices leads to simpler models. Therefore, the L2 regularizer will also reduce overfitting to quite an extent, as following:

$$\text{Cost function} = Loss + \lambda \sum \|w\|^2 \tag{6}$$

where the regularization term is the sum of the square of all feature weights multiplied by the regularization parameter λ, which determines how much penalizes the weights and it has been set to $\lambda = 0.0005$, by the fact that weight decay has been also implemented.
The idea of weight decay [25] is simple and it is used to prevent overfitting: every time the weight w is updated with the gradient ∇J with respect to w, it is also subtracted from it the quantity $\mu \cdot w$, hence the product of the weight decay rate and the weight itself. This gives the weights to decay towards zero. In fact,

$$w_{t+1} = w_t - \lambda \nabla_w J - \mu w_t \tag{7}$$

Theoretically, large datasets and deeper architectures seem to require smaller values of the decay rate hence it was chosen the value of 0.66.

With every change to any of the Hyperparameters, the network was re-trained to observe how it effects the performance. By repeating this exercise for every parameter, the study ensures a final trained model, which guarantees better grasp coordinates, as desired. Moreover to ensure replicability as robust as possible from the industrial point of view, a threshold for the Q-value was given. In this way all the potential grasps that had a probability lower than this threshold are not taken into consideration and a re-sampling of the network is carried out.

4. Experimental Setup

The core idea was to develop an industrial-grade solution, so all the tests were run on two industrial objects different in shape. Results obtained for each object individually are compared and presented. The tests have been performed having three scenarios in mind:

- Single Object grasping - single object in the viewing area of the camera (varied by its location in terms of angle and position).
- Homogeneous Multi-object grasping - multiple objects of the same type in the viewing area of the camera. For this, the number of objects placed at any point is varied between 4 to 6 and also the pattern of placement is varied between a well-organized one and a random one.
- Non-Homogeneous Multi-Object grasping - multiple objects of both types (non-homogeneous) in the viewing area of the camera. For this, the number of objects placed at any point is varied between 4 to 6 and also the pattern of placement is varied between a well-organized one and a random one.

For each test scenario, 100 grasps were sampled, and all of these grasps were distributed all around the field of view of the camera.
The Cognitive Grasping System is composed of four modules:

- Embedded board
- 3D Camera
- Robot
- Gripper

The embedded board used is the Nvidia Jetson TX2 with Ubuntu 16.04, where Python programming language has been installed with Tensorflow library. The Convolutional Neural Network has been implemented on this board.
The 3D camera is capable of providing a Point Cloud as an output and these data are then processed to obtain a depth image. The camera used in this work is an Asus Primesense Camera, with a resolution of 640×480 pixel and a distance from the surface of 800 mm. Its drivers were installed on the embedded board in order to get better performances. Machine vision algorithms have been used to standardize and normalize camera acquisitions to improve the acquired image.
The robot used is an Universal Robot UR10, a six axis cobot, equipped with an optimum parallel-jaw gripper for the grasp. Once the network has calculated the best pose, transformations are applied to obtain the grasp coordinates to pass to the robot. In Figure 1 the robot, the gripper and the camera are shown.
Tests had been performed using first the original GQCNN algorithm (Version 1) and then the optimized one (Version 2).

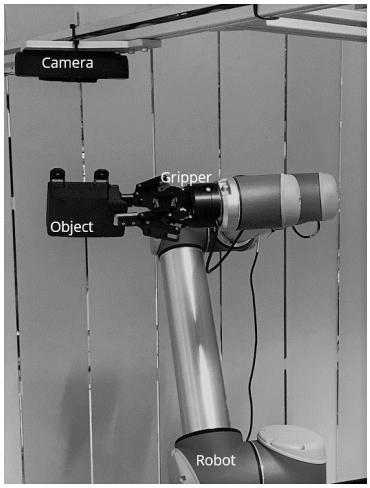

Fig. 1: Cognitive Grasping System

5. Results

The results obtained for all 3 scenarios have been compared between the two versions of the solution and are presented in tables 1, 2 and 3. The two objects have been evaluated in different ways due to their different shapes: the parameters used to evaluate the accuracy of the grasp for the first object are the distance of the gripper from the center and its angle, while for the second object, due to its protruding part, the distance from the protruding part and the depth of the grasp are considered. The images of the two objects used for the inference test are shown below (Fig. 2), with an example of calculated grasp.
All the distance values are in cm, angle values are in degrees, and the depth values are in percentage of good grasp, it means the percentage of objects grasped straight versus objects grasped just by the protruding part which leads it to be held in a tilted manner.
In the case of the second type of object, from 10 to 15 % of grasps fail, because of the structure of the object and more due to the limitations of camera detection. The two small dials on

Table 1: Single Object Grasping

RESULTS	Object 1		Object 2	
	Distance	Angle	Distance	Depth
Version 1	1.47	4-10	1.36	~ 35% of good grasps
Version 2	0.61	4-10	0.44	~ 85% of good grasps

Fig. 2: Objects shape

Table 2: Homogeneous Multi-Object Grasping

RESULTS	Object 1		Object 2	
	Distance	Angle	Distance	Depth
Version 1	0.789	2-5	0.888	~ 80% of good grasps
Version 2	0.711	2-5	0.41	~ 80% of good grasps

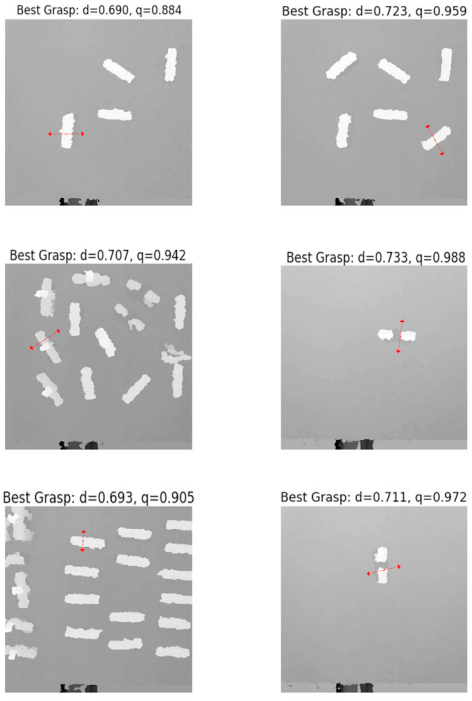

Fig. 3: Example grasp images

the front of the object, sometimes come in the way of the gripper when it is moving down to a certain depth, and when it hits the two dials, the grasp stops. The problem can be easy solved given that the gripper width could be increased. In figure 3 some

Table 3: Non-Homogeneous Multi-Object Grasping

RESULTS	Distance from the center
Version 1	1.24
Version 2	0.914

images obtained during the tests are shown.

Due to the fact that the margins are very small, the proposed solution has been modified to ensure grasps of a certain standard with high repeatability and confidence. Taking into consideration, the resolution errors of the camera (around ~4mm at the distance on which the test are made) and other factors (grippers physical property and lighting) the limit of the distance of the grasp from the center of the object has been fixed at 5mm. Any grasp which is far from the center of the object by more than 5mm would not be considered as a good grasp. After performing various modifications explained above like parameter changes, retraining, generating a new dataset, the threshold has been set. If at times, the grasp computed was not considered good, the program re-sampled the grasps.

6. Discussion and Conclusion

During the course of the tests the following considerations could be drawn:

- There were some grasps very far from the center of the object (15% of the grasps) which led to a higher average.
- The object sensing was not good towards the edges of the viewing frame, so edge grasps were lower in quality. This led to occlusion based errors (especially in terms of depth). This error was observed more with multiple objects grasping and did not affect when there was only one object in the field of view.
- Shadows affected the grasp quality a lot.

Regardless of the three scenarios analyzed, there was an improvement in terms of the distance of the grasp from the center of the objects, which is significant in the case of the presence of only one object in the field of view of the camera.
The results obtained have shown that the CNN network trained with a general database is able to grab different objects, without further customizing the model. The efforts made to bring the model into an industrial environment have involved optimizing the algorithm and the various phases of preparation of the input image and interfacing with the various components of the system. Particular attention has been paid to hardware where the CNN model and image preprocessing are run. This lead to enabling the Dex-Net database as a starting point for building a solution on it. Various modifications and additions are performed so as to come up with a solution with industry precision for single object grasping with 100% accuracy within 60% area of the field of view. 100% successful grasps within 75% area of the field of view has been also ensured.

References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, M. Hoffmann, Industry 4.0, Business and Information Systems Engineering: The International Journal of WIRTSCHAFTSINFORMATIK 6 (4) (2014) 239–242.

[2] R. Pelossof, A. Miller, P. Allen, T. Jebara, An svm learning approach to robotic grasping, in: IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, Vol. 4, 2004, pp. 3512–3518 Vol.4.

[3] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, K. Goldberg, Cloud-based robot grasping with the google object recognition engine, in: IEEE Intl Conf. on Robotics and Automation, 2013, p. 8.
[4] J. Bohg, A. Morales, T. Asfour, D. Kragic, Data-driven grasp synthesis - A survey, CoRR abs/1309.2660 (2013). arXiv:1309.2660.
URL http://arxiv.org/abs/1309.2660
[5] A. T. Miller, P. K. Allen, Graspit!: A versatile simulator for grasp analysis, in: in Proc. of the ASME Dynamic Systems and Control Division, 2000, pp. 1251–1258.
[6] A. Saxena, J. Driemeyer, J. Kearns, A. Y. Ng, Robotic grasping of novel objects, in: Proceedings of the 19th International Conference on Neural Information Processing Systems, NIPS06, MIT Press, Cambridge, MA, USA, 2006, p. 12091216.
[7] O. Kroemer, R. Detry, J. Piater, J. Peters, Combining active learning and reactive control for robot grasping, Robotics and Autonomous Systems 58 (9) (2010) 1105–1116.
[8] I. Lenz, H. Lee, A. Saxena, Deep learning for detecting robotic grasps, Int. J. Rob. Res. 34 (45) (2015) 705724. doi:10.1177/0278364914549607.
URL https://doi.org/10.1177/0278364914549607
[9] L. Pinto, A. Gupta, Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours, CoRR abs/1509.06825 (2015). arXiv:1509.06825.
URL http://arxiv.org/abs/1509.06825
[10] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, K. Goldberg, Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics, CoRR abs/1703.09312 (2017). arXiv:1703.09312.
URL http://arxiv.org/abs/1703.09312
[11] S. Levine, P. Pastor, A. Krizhevsky, D. Quillen, Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, CoRR abs/1603.02199 (2016). arXiv:1603.02199.
URL http://arxiv.org/abs/1603.02199
[12] J. Yu, K. Weng, G. Liang, G. Xie, A vision-based robotic grasping system using deep learning for 3d object recognition and pose estimation, 2013, pp. 1175–1180. doi:10.1109/ROBIO.2013.6739623.
[13] J. Varley, J. Weisz, J. Weiss, P. Allen, Generating multi-fingered robotic grasps via deep learning, 2015, pp. 4415–4420. doi:10.1109/IROS.2015.7354004.
[14] A. Saxena, J. Driemeyer, A. Ng, Robotic grasping of novel objects using vision, I. J. Robotic Res. 27 (2008) 157–173. doi:10.1177/0278364907087172.
[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database, in: CVPR09, 2009.
[16] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft COCO: common objects in context, CoRR abs/1405.0312 (2014). arXiv:1405.0312.
URL http://arxiv.org/abs/1405.0312
[17] L. Deng, The mnist database of handwritten digit images for machine learning research [best of the web]., IEEE Signal Process. Mag. 29 (6) (2012) 141–142.
URL http://dblp.uni-trier.de/db/journals/spm/spm29.html#Deng12
[18] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kroeger, J. Kuffner, K. Goldberg, Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards, 2016.
URL http://dx.doi.org/10.1109/ICRA.2016.7487342
[19] A. F. Agarap, Deep learning using rectified linear units (relu), CoRR abs/1803.08375 (2018). arXiv:1803.08375.
URL http://arxiv.org/abs/1803.08375
[20] T. Pang, K. Xu, Y. Dong, C. Du, N. Chen, J. Zhu, Rethinking softmax cross-entropy loss for adversarial robustness, ArXiv abs/1905.10626 (2019).
[21] S. Ruder, An overview of gradient descent optimization algorithms, CoRR abs/1609.04747 (2016). arXiv:1609.04747.
URL http://arxiv.org/abs/1609.04747
[22] D. Kingma, J. Ba, Adam: A method for stochastic optimization, International Conference on Learning Representations (12 2014).
[23] Y. Wu, L. Liu, J. Bae, K.-H. Chow, A. Iyengar, C. Pu, W. Wei, L. Yu, Q. Zhang, Demystifying learning rate polices for high accuracy training of deep neural networks, ArXiv abs/1908.06477 (2019).
[24] B. Bilgic, I. Chatnuntawech, A. Fan, K. Setsompop, S. Cauley, L. Wald, E. Adalsteinsson, Fast image reconstruction with l2-regularization, Journal of magnetic resonance imaging : JMRI 40 (07 2014). doi:10.1002/jmri.24365.
[25] A. Krogh, J. A. Hertz, A simple weight decay can improve generalization, in: Proceedings of the 4th International Conference on Neural Information Processing Systems, NIPS91, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991, p. 950957.