Deep Learning Neural Networks on Mobile Platforms

eingereichtes
Hauptseminar
von

Andreas Plieninger

geb. am 10.02.1991
wohnhaft in:
Arcisstr. 21
80333 München

Lehrstuhl für
STEUERUNGS- und REGELUNGSTECHNIK
Technische Universität München

Prof. J. Conradt

Betreuer: Ph.D. Viviane Ghaderi
Beginn: 19.10.2015
Abgabe: 18.01.2016

TECHNISCHE UNIVERSITÄT MÜNCHEN

NEUROWISSENSCHAFTLICHE SYSTEMTHEORIE

PROF. DR. JÖRG CONRADT

2015-10-12

A D V A N C E D   S E M I N A R

for

Andreas Plieninger, Mat.-Nr. 3617584

Deep Learning Neural Networks on Mobile Platforms

Problem description:

A long-term goal of machine learning is to develop algorithms that achieve highly complex cognitive tasks, such as perception (recognizing scenes, objects), reasoning, and learning. Over the past few years, it has been demonstrated that deep learning neural networks are able to outperform traditional algorithms in image classification, as well as object and face recognition tasks. Deep learning neural networks are now able to classify objects in images with near-human-level performance. However, deep learning architectures require a lot of computational power; training the networks requires big data sets and computations on large CPUs or GPU clusters. The question now arises if deep learning algorithms can be scaled to achieve more complex tasks while achieving power consumption levels that make a mobile implementation feasible.

Tasks:

- What is deep learning?
- Compare deep learning to traditional algorithms. Why and when does deep learning perform better?
- What are the disadvantages of deep learning?
- Investigate possibilities for deep learning on mobile platforms.

Supervisor: Viviane Ghaderi, Ph.D.

(Jörg Conradt)
Professor

Abstract

One goal of machine learning is to solve highly complex cognitive tasks, such as perception. In the last decade, deep learning networks showed immense progress and can now even surpass human performance in image object recognition on the ImageNet dataset. This report will compare a deep convolutional neural network to a traditional algorithm, why it outperforms them and what the challenges for deep learning are. The last part will investigate the possibilities and difficulties of using deep learning on mobile platforms.

Contents

Chapter 1

Introduction

Machine learning studies algorithms that can solve difficult tasks by generalizing from examples. This involves learning a model from observed data, which in turn helps to make predictions and decisions. In comparison to manual programming, the goal is to implement algorithms which learn from example data how to solve the task. This is often more cost-effective compared to manual programming [Dom12]. And as more data is available, more complex problems can be tried to solve. Typical applications include highly complex cognitive tasks such as object and speech recognition as well as natural language processing. Research in the last decade demonstrated that deep learning networks can outperform traditional algorithms in this field, especially for object classification in images.
This report will compare one deep learning technique to a traditional algorithm, why and when it outperforms them and what the challenges for deep learning are. The last part will investigate the possibilities of using deep learning on mobile platforms.

Chapter 2

What is Deep Learning?

Taking the Wikipedia definition for example, deep learning is defined as alogrithms that model abstractions using multiple processing layers[1]. Another definition of deep learning is "(...) techniques that use supervised and/or unsupervised strategies to automatically learn hierarchical representations in deep architectures for classification" [CL14]. The two key aspects of deep learning are commonly defined as

1. "models consisting of multiple layers or stages of nonlinear information processing
2. methods for supervised or unsupervised learning of feature representation at successively higher, more abstract layers." [Dom12]

A typical structure of such a deep learning system is shown in Fig 2.1: The input data is processed in multiple layers, until the final layer will produce some output. In the case of Fig 2.1, every layer further abstracts the input data: The input is convolved with four different filters followed by a non-linear activation. The resulting feature maps will get sub-sampled before they will again be convolved with eight different filters. After another sub-sampling step the 2D output is fed into a fully connected 1D neural network which produces the final output. The number of layers in deep learning techniques can vary from three to five layers and get as big as 22 layers in GoogLeNet [SLJ^{+}14].
Deeper networks can represent a larger class of functions using less parameters than shallow ones. In detail, a n-layer network with a number of hidden units polynomial in the number of inputs can represent a specific class of functions, whereas a (n-1)-layer network needs an exponentially larger number of hidden units to represent the same functions[2] [Cyb89, HSW89, Hor91]. This universal approximation theorem states that a three layer network (one hidden layer) is a universal function approximator. How is this possible? The input layer divides the input space

[1]Wikipedia, last access: Dec. 2015, `https://en.wikipedia.org/wiki/Deep_learning`

[2]Stanford Deep Learning Wiki, last access: Dec. 2015, `http://deeplearning.stanford.edu/wiki/index.php/Deep_Networks:_Overview`

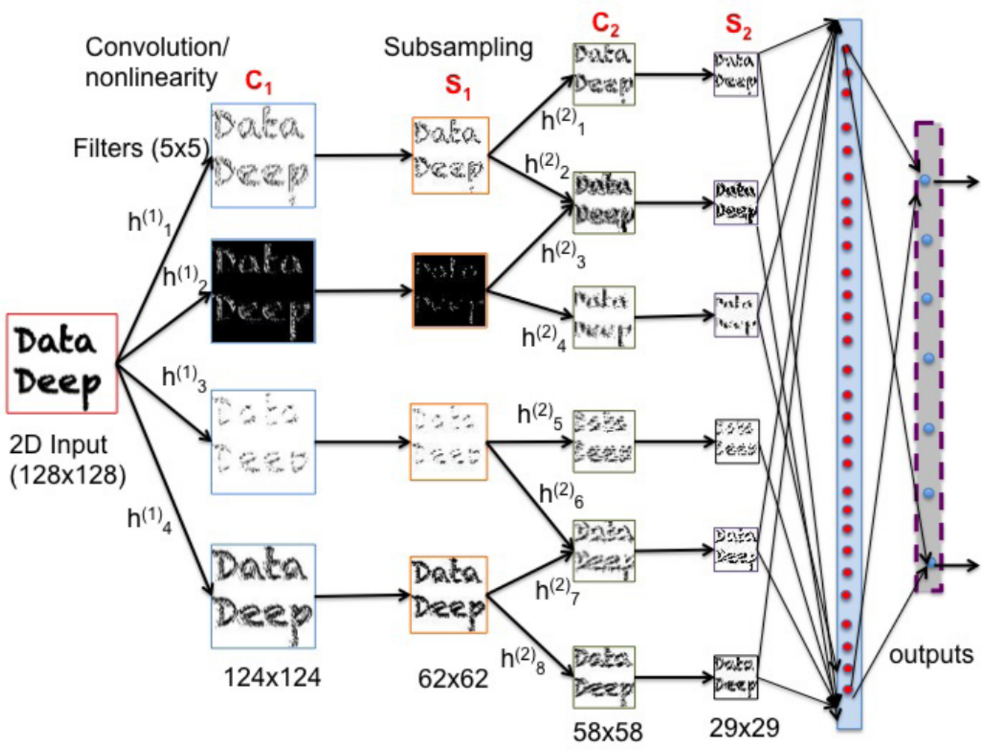

Figure 2.1: Example of a CNN structure for pattern classification in images [CL14]

in many segments, and each segment is approximated by one hidden neuron. But the exponentially growing number of necessary neurons makes it exponentially inefficient for training and practical applications. There lies the advantage of deep networks, which can be thought of constant-depth threshold circuits. These circuits are know to be able to compute a number of interesting problems, for example a small 3-hidden layer network can sort N N-bit numbers (parameter size polynomially bounded in n) [SBKH93].
Although practical experiments suggest that at least some layers are beneficial, up to today, the number of layers necessary for a specific class of problem is not always proven. A recent paper shows some progress. It explores the representational power of two and three layer deep networks for another class of problems [KW15].
There is an additional intuitive argument, why deeper networks should perform well in several perception tasks: The human brain needs about 100 ms to recognize a picture, which implies that it only uses about 10 layers for the answer [HB07]. Similarly, the cerebral cortex has six layers[3], which suggests that even for complex tasks it might not be necessary to have as many as 22 layers.

[3]Wikipedia, last access: Dec. 2015, `https://en.wikipedia.org/wiki/Cerebral_cortex#Layered_structure`

Also the architectures under the marketing name “deep learning” greatly vary. Depending on the application, the following are, among others, used: Deep belief network (DBN), deep autoencoder and deep neural network (DNN). For every use case, the architecture is further adapted.

Chapter 3

Convolutional Neural Network for Image Classification

This chapter will describe one kind of deep learning architecture, a deep convolutional neural network (CNN) for image classification. The described architecture was developed by Krizhevsky et al. [KSH12] and won the imagenet competition 2012, where algorithms have to label the objects visible in images[1] [Sch15]. It was the first time that a deep learning approach won this contest.

3.1 The input and output of the task

Fig 2.1 describes the algorithm input and expected output. The input is a raw image, the output should be the label of the image. In the imagenet competition, each image has five labels, ordered by priority. Therefore the algorithm can also output five labels, each with a priority of how sure it is that the label correctly classifies the image.

3.2 Feature Representation

The algorithm usually performs two steps. The first step is to extract features from the input, in this case the raw image. The pixel values are not a good representation of what the image shows. As example, good features for a picture of a motorbike could be: “Does it show handlebars?”, “Does it show wheels?” or on a lower level “Are there horizontal edges?”.

[1]Team Supervision `http://image-net.org/challenges/LSVRC/2012/results.html`

[2]Deep Learning, Andrew Ng, last access: Dec. 2015, `http://research.microsoft.com/en-us/events/fs2013/andrew-ng_machinelearning.pdf`

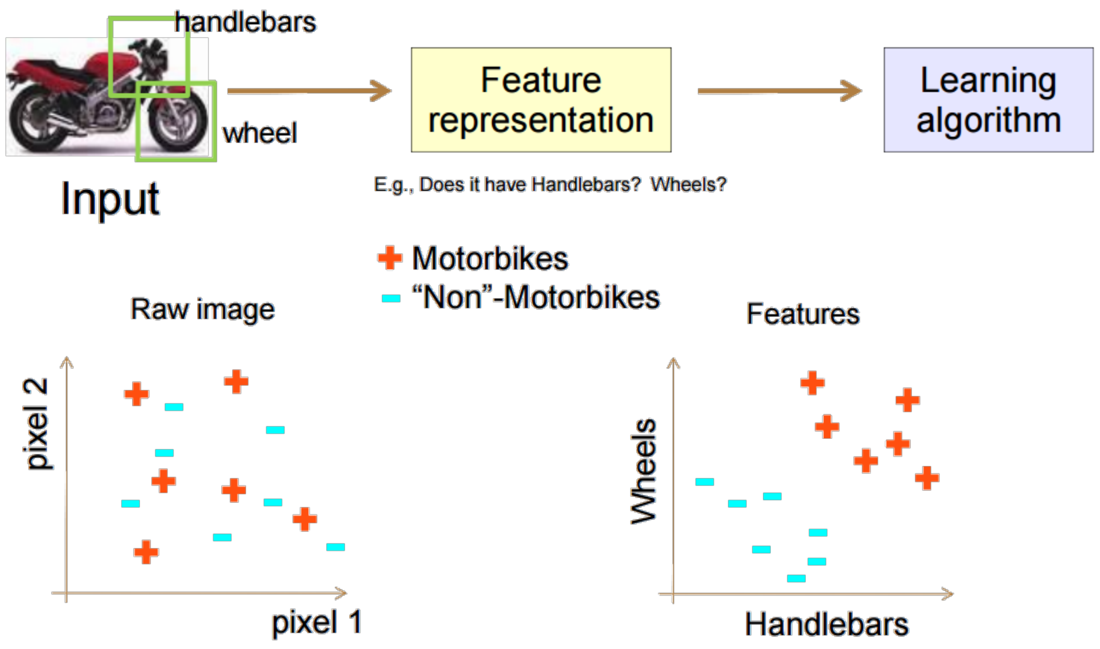

Figure 3.1: Input data and feature representation for a classifier, Source:[2]

3.3 Classification

The second step of the process is a learning algorithm with the goal to classify the input. As input the feature representation is used. The expected output are the correct labels. This part of the algorithm tries to learn to map the feature vector to the labels. If the feature vectors are good enough, the classification is much easier than with the raw image. Fig 3.1 illustrates this example. For the raw image (left side), the two input classes are mixed together and are hard to classify, whereas after the feature representation (right side) the two classes of images can be separated more easily. Therefore, in order to get good classification results, an excellent feature representation is very important.

3.4 Overall Network Structure

As described before, there are many different network topologies. In this example of the ImageNet, the network input consists of raw RGB values of the 250 pixel x 250 pixel input images. The first five convolutional layers are sometimes followed by a max-pooling layer. These lead into three fully-connected layers ending in a final 1000-way softmax. The complete network has about 60 million parameters and 650,00 neurons. The interested reader can find a more detailed description of the exact network parameters in the original publication [KSH12].

3.5 Training

There are different ways to train a neural network. In this specific example, the model was trained using stochastic gradient descent (SGD). The updating rule includes weight decay. For training, batches of 128 input examples were used. As gradient direction, the average of each batch at each neuron was used. Furthermore, the learning rate was manually changed to smaller values after the validation error stopped improving. For the initialization of the weights, a zero-mean Gaussian distribution was used. Additionally, the bias was set to either zero or one, depending on the layer. For the interested reader, the original paper can provide a more detailed description [KSH12].
After about 90 cycles over the 1.2 million input images, the learning on two graphic cards, which took about five days, was finished.

3.6 Classification Results of Trained Network

The described network achieved a top-5 error rate of 18.2% on the popular ILSVRC-2012 test set. This was even further improved by adding an additional sixth convolutional layer and averaging the result of seven separately trained networks. Additional improvement was gained by pre-training the network with the ImageNet 2011 release. This achieved the best submission and winning the contest with a top-5 error rate of 15.3%. The second best submission, not a deep network, achieved a top-5 error rate of 26.2%. In the following contest the year after, the ILSVRC-2013, almost all submissions used deep learning techniques inspired by the described CNN[3] [4] [RDS+15].

3.7 Comparison to Traditional Algorithms or Why did the CNN win?

The second best submission to the ILSVRC-2012[5] competition was a "traditional" algorithm. Traditional will be used in this report to reference to algorithms which do not use deep neural networks. In detail, the algorithm of the team called "ISI" used the weighted sum of scores of several classifiers trained on Fisher Vectors [SP11] computed on feature descriptions provided e.g. by SFIT, CSFIT, LBP and GIST [RDS+15].

[3]Deep Learning, Andrew Ng, last access: Dec. 2015, `http://cs229.stanford.edu/materials/CS229-DeepLearning.pdf`

[4]ILSVRC-2013 results, last access: Dec. 2015, `http://www.image-net.org/challenges/LSVRC/2013/results.php`

[5]ILSVRC-2012 results, last access: Dec. 2015, `http://image-net.org/challenges/LSVRC/2012/results.html`

SIFT, as example, is a feature descriptor, developed by Lowe and published in 2004 [Low04]. It is arguably the most cited paper in the computer vision community of the last twenty years with about 32.000 quotes[6]. The algorithm can detect and describe local features independent of the rotation and scale. These extracted key points are usually corners or edges. Several key points combined can be used to recognize larger objects. The whole algorithm was developed manually by Lowe. He hand-engineered and changed the algorithm in order to improve the feature descriptors. The final result, the so-called SIFT algorithm, was not a product of a machine learning algorithm or deep learning system, but of thoughtful human effort. Why is this distinction important? SIFT extracts the features of the image, which are then processed and used for the classification. Therefore it is a key part of the system. On the other hand, a deep learning based approach like the CNN described earlier, does not use any predefined feature algorithms. Instead, it learns the feature extraction algorithm from the data. Only the in comparison simple learning algorithm for the CNN has to be developed by a human. The final feature algorithm is then learned by the network. Arguably the algorithm can use more data and is faster to improve and adapt its feature representation than a human updating his algorithm[7] [8].
An illustrative example is the classification of images of galaxies: Because SIFT was developed with natural scenes in mind, it does not fit well to describe galaxies. Whereas a CNN can learn its features based on the input data and optimize to extract useful information out of galaxy images[9]
Another advantage of this CNN is, that more available training data leads to better results. The performance scales better with the size of the available data set. These kind of algorithms, as shown in Fig 3.2, start outperforming other algorithms after a certain amount of data was used for training[7] [8]. Furthermore, the past successes of deep learning were primarily achieved by better hardware and larger training sets and not by improved algorithms [Dom12].

Summary In summary, deep learning algorithms try to model the feature representation and their classifiers in one step. If enough computational power and training data is available, they outperform traditional combinations of feature descriptors and classifier, and in some cases, even humans [KSH12, SLJ$^+$14, HZRS15, KFCRB15]. Table 3.1 summarizes the mentioned key differences.

[6]Google Scholar, last access: Dec. 2015, `https://scholar.google.com`

[7]Deep Learning, Andrew Ng, last access: Dec. 2015, `http://cs229.stanford.edu/materials/CS229-DeepLearning.pdf`

[8]RSS2014: Invited Talk: Andrew Ng, Deep Learning, last access: Dec. 2015, `https://www.youtube.com/watch?v=W15K9PegQt0`

[9]Sander Dieleman, Galaxy Zoo challenge, last access: Jan. 2015, `http://benanne.github.io/2014/04/05/galaxy-zoo.html`

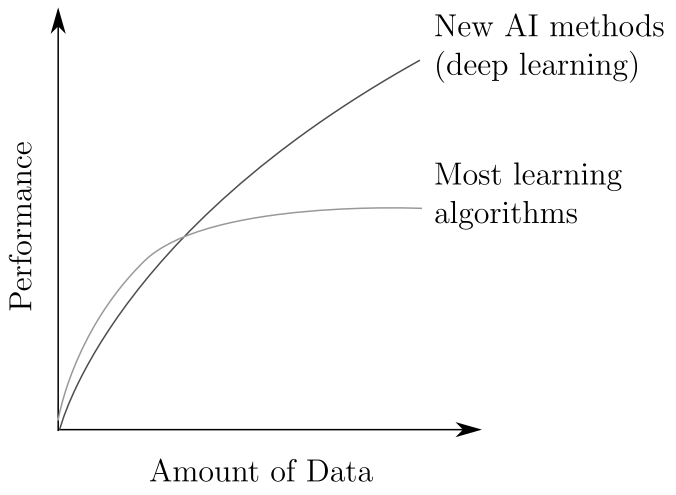

Figure 3.2: Performance of algorithms over amount of available data

	Deep learning	Conventional Algorithms
Feature representation	learned	hand engineered
Classification	learned	learned
More training data	continues improving	stops improving
Necessary training data	+++	+
Computation requirements	+++	+

Table 3.1: Comparison of key properties between deep learning and traditional algorithms

Chapter 4

Challenges for Deep Learning

4.1 Training Data

As more training data leads to better results, a lot of data is necessary. Because it is too much work to label all of it, the so-called pre-training uses unlabeled data (e.g. pictures downloaded from the Internet). Afterwards, for the specific task, a smaller labelled data set is used for fine tuning (e.g. images labelled as cat or dog) [Le13]. This two-step progress significantly improves the performance of the final model [EBC$^+$10].

4.2 Network Size

More complicated problems seem to need larger networks to tackle them. Therefore, the amount of network parameter increases. This leads to more necessary computations, which in turn either takes more time or needs more hardware.

4.3 Scaling

In order to increase the network size and reduce the necessary computation time, more hardware is needed. To achieve this, the deep learning algorithm has to run on distributed hardware. Another possible solution is to use improved hardware. For example, in the last years high-end consumer graphic cards were more and more used to speed up the CNN training.

4.4 Improved Algorithms

Until now, as mentioned in the introduction, it is not really mathematically understood, why deep learning works comparatively well. It is not even proven, what the influence of the number of layers and neurons per layer is [KW15]. Additionally, further work is necessary to characterize the problems for which e.g. SGD is efficient [GV14]. Furthermore, research is needed to show if perhaps improved algorithms could allow us to use smaller models [GV14].

Chapter 5

Deep Learning on Mobile Platforms

The following chapter investigates the possibilities for deep learning on mobile platforms. Several aspects are discussed. The key constraints on a mobile device compared to a data center facility are (1) the size of the device and therefore limit computational power and (2) energy, as the devices are often battery powered. A mobile implementation of a deep learning algorithm needs to consider these constraints.

5.1 Improved and Dedicated Hardware

One idea is to improve the hardware. In the last decade, hardware suppliers made considerable progress in improving and adapting GPUs for deep networks. Namely, the fourth generation of the CUDA neural network library (cuDNN) was just released[1] and also supports GPUs designed for mobile devices as the nVidia Tegra K1. Other companies start as well to support deep learning. Qualcomm started an initivative to also support artificial neural networks[2]. Their mobile Snapdragon processor offers a CPU and GPU to run deep networks[3]. It furthermore has a DSP (digital signal processor) which enables to run small neural networks on low-power [LG15].
On the other hand, if the progress of hardware improvements continues, at some point in time the computational power of mobile devices will match those of dedicated desktop hardware available now. Therefore, current neural networks which have been trained on multiple desktop GPUs, will be able to be trained on mobile devices in the future.

[1]nVidia cuDNN, last access: Dec 2015, `https://developer.nvidia.com/cudnn`

[2]Qualcomm, last access: Jan. 2015, `https://www.qualcomm.com/invention/cognitive-technologies/zeroth`

[3]Qualcomm Snapdragon development paltform, last access: Jan. 2015, `https://developer.qualcomm.com/hardware/mdp-810`

There is also research about utilizing FPGAs for RNNs in order to speed up the training time [CMC15].

5.2 Split Learning and Execution

To train a deep network, many iterations are necessary. The training takes very long compared to one forward execution. To speed up the process on mobile devices, the training is done beforehand using dedicated hardware. The finally trained model is then transferred to the mobile device. Depending on the exact process, the model can be further optimized. Examples are [Vas15, NAPP15]:

1. Less important weights can be set to zero
2. Weights can be deduplicated
3. Lower precision arithmetic can be applied
4. Range-constrained topologies can be used

Then, the mobile device can execute the model. One example of this approach is the spotting of audio keywords like “Ok, Google”, which uses a finished trained model on the device [CPH14].
Another improvement is the final fine-tuning of the model and adaption of some parameters on the target device[4]. For example, the model is pre-trained to recognize faces. Then, on the mobile device, the network is trained to recognize one specific face[5].
The two approaches of local and cloud computing can also be combined. Parts of the execution can be offloaded to the cloud, where it will be calculated and the result sent back to the mobile device [LG15].
The pre-training and cloud offloading are not always a viable solution: If the task is not know beforehand and the transfer of input data is not possible or severe tight constraints are given (for example a combination of deep and reinforcement learning to play games [MKS+15]), local training and learning seems to be the only feasible strategy.

5.3 Existing Deep Learning Frameworks

This section will investigate the compatibility of current deep learning frameworks with mobile devices. It will not compare other features or the performance, as there already exist benchmarks [BRSS15].

[4]Simon Guigue, last access: Dec. 2015, `http://www.atelier.net/en/trends/articles/training-your-smartphone-perform-visual-recognition-deep-learning_430700`

[5]Qualcomm, last access: Jan. 2015, `https://www.qualcomm.com/news/onq/2015/03/02/qualcomm-zeroth-advancing-deep-learning-devices-video`

5.3.1 Caffee

Caffee is a framework developed by the Berkley Vision and Learning Center and community contributors [JSD+14]. As examples demonstrate, the software can be used to train a state of the art neural networks using the ImageNet data set. It utilizes only the available CPUs, or, if compatible, also GPUs. For the mobile nVidia Tegra K1 GPU, the support was recently added[6].

5.3.2 Torch

Torch is another computing framework, which supports deep learning algorithms [CBM02]. It can use GPUs to seed up computation. Recent ports brought support to run on mobile devices like iOS, Android and FPGAs.

5.3.3 Theano

Theano also supports deep learning algorithms with efficient calculation of multi-dimensional arrays [BLP+12]. The python library includes GPU support as well as symbolic differentiation. Support for mobile devices has not been a core feature of the software yet.

5.3.4 Deeplearning4j

Deeplearning4j tries to solve some issues of the other frameworks, namely Java as a more robust and portable language and providing commercial support[7]. This enables the framework to run on any platform with Java support. Additionally, strong support to scale over many parallel GPUs or CPUs is implemented.

5.3.5 Tensorflow

Tensorflow is an open-source framework for numerical computations, which was recently released by Google [AAB+]. It facilitates large-scale machine learning on distributed systems. Without changing code, it can run on multiple CPUs or GPUs in a server or mobile device.

5.3.6 Summary

As the development shows, the focus on adding support to run deep neural networks on mobile devices is rising. Without changing code, the same model and software can now run on dedicated servers as well as on a mobile devices.

[6]Pete Warden, Caffee on Mobile GPU, last access: Dec. 2015, `http://petewarden.com/2014/10/25/how-to-run-the-caffe-deep-learning-vision-library-on-nvidias-jetson-mobile-gpu-board/`

[7]Deeplearning4j, last access: Dec. 2014, `http://deeplearning4j.org/`

5.4 Improved Algorithms

Another way to overcome the computation constraints on mobile devices, is to develop improved algorithms. Many researchers try to optimize and improve the performance of an algorithm for specific problems. One example of such an novel approach in computer vision is to combine classification trees with CNNs [KFCRB15]. The first results are promising, as the top5-error rate is lower than the best GoogLeNet architecture. Furthermore, using branching nodes in the network could reduce the number of necessary calculations for a forward execution. Compared to a CNN, not the whole layer, only the child nodes of that branching node have to be evaluated. This could allow to run the same network on mobile devices without loosing accuracy.
Another example is an improved initialization of weights and momentum for stochastic gradient descent, which leads to an overall performance improvement [SMDH13].

5.5 Summary

Currently, deep learning is in parts already used on mobile platforms. The model is forward-executed on the device in order to classify the input. This already allows to solve complex tasks while achieving reasonable power consumption levels. The training of the model is usually performed on server hardware, as computational power is limited on a mobile platforms.
Improved specialized hardware, learning algorithms and software will enable to tackle more complex tasks with even lower power consumption than today.
Once it is possible to reasonably train deep networks on mobile platforms, another question will get more important: How can a deep network be trained completely without human supervision for new problems? Currently, a lot of parameter tuning and many trial runs with the help of scientists are necessary, until a deep learning model is achieving acceptable results. I am personally certain that some researchers will soon be able to answer this question.

List of Figures

Bibliography

[AAB$^+$] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org.*

[BLP$^+$12] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

[BRSS15] Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott, and Mohak Shah. Comparative study of caffe, neon, theano, and torch for deep learning. *arXiv preprint arXiv:1511.06435*, 2015.

[CBM02] Ronan Collobert, Samy Bengio, and Johnny Mariéthoz. Torch: a modular machine learning software library. Technical report, IDIAP, 2002.

[CL14] Xue-Wen Chen and Xiaotong Lin. Big data deep learning: Challenges and perspectives. *Access, IEEE*, 2:514–525, 2014.

[CMC15] Andre Xian Ming Chang, Berin Martini, and Eugenio Culurciello. Recurrent neural networks hardware implementation on fpga. *arXiv preprint arXiv:1511.05552*, 2015.

[CPH14] Guoguo Chen, Carlos Parada, and Georg Heigold. Small-footprint keyword spotting using deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4087–4091. IEEE, 2014.

[Cyb89] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[Dom12] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[EBC+10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[GV14] Ian J Goodfellow and Oriol Vinyals. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

[HB07] Jeff Hawkins and Sandra Blakeslee. *On intelligence*. Macmillan, 2007.

[Hor91] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[KFCRB15] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. June 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KW15] Daniel M Kane and Ryan Williams. Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits. *arXiv preprint arXiv:1511.07860*, 2015.

[Le13] Quoc V Le. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE, 2013.

[LG15] Nicholas D Lane and Petko Georgiev. Can deep learning revolutionize mobile sensing? In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pages 117–122. ACM, 2015.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[NAPP15] Preetum Nakkiran, Raziel Alvarez, Rohit Prabhavalkar, and Carolina Parada. Compressing deep neural networks using a rank-constrained topology. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. `doi:10.1007/s11263-015-0816-y`.

[SBKH93] Kai Yeung Siu, Jehoshua Bruck, Thomas Kailath, and Thomas Hofmeister. Depth efficient neural networks for division and related problems. *Information Theory, IEEE Transactions on*, 39(3):946–956, 1993.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[SLJ+14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[SMDH13] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147, 2013.

[SP11] Jorge Sánchez and Florent Perronnin. High-dimensional signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1665–1672. IEEE, 2011.

[Vas15] Artem Vasilyev. CNN optimizations for embedded systems and FFT. 2015.