Heterogeneous Learning on Big Multimodal Data

ADVANCED SEMINAR

submitted by
Sebastian Schlecht

NEUROSCIENTIFIC SYSTEM THEORY
Technische Universität München

Prof. Dr Jörg Conradt

Supervisor: Cristian Axenie
Final Submission: 18.01.2016

In your final hardback copy, replace this page with the signed exercise sheet.

Abstract

The Big Data era brought up many new possibilities and challenges to developers and researchers. With today's mobile platforms, which are equipped with a multitude of sensors, data have become abundant.However, algorithms and processes have yet to be developed to successfully leverage all this information efficiently as well as to tackle remaining problems. This report discusses two heterogeneous machine learning techniques, *Transfer Learning* and *Active Learning*, which can be valuable tools in scenarios where only few and/or unlabelled data are available. This can be especially beneficial in the mobile context where resources are normally limited. After a theoretical overview of both techniques, two exemplary use-cases are shown in which these approaches significantly help to obtain better results. In both cases, large amounts of available data from the same or a similar problem domain are used effectively in order to increase both accuracy and robustness of a predictive model. In the end, a short outlook is given how machine learning may develop on mobile platforms, given the plans of established companies as well as recent results obtained by researchers.

Zusammenfassung

In Zeiten von Big Data bieten sich viele neue Möglichkeiten und Herausforderungen für Entwickler und Wissenschaftler. Mobile Geräte, ausgestattet mit einer Vielzahl an Sensoren, produzieren große Mengen an Daten aus verschieden Modalitäten. Algorithmen müssen jedoch auch in der Lage sein, daraus Nutzen zu ziehen um bisherige Probleme zu lösen und dadurch an Leistung zu gewinnen. Dieser Bericht zeigt zwei Ansätze aus dem Bereich Heterogenes Maschinenlernen auf, *Transfer Learning* und *Active Learning*, die dabei helfen können auch in schwierigen Situationen die Genauigkeit bisheriger Algorithmen zu erhöhen und Nutzen aus großen Datenmengen zu ziehen. Gerade für mobile Geräte kann dies von großem Nutzen sein, da dort die vorhandenen Ressourcen meist beschränkt sind. Nach einem theoretischen Überblick über beide Ansätze werden zudem zwei exemplarische Anwendungen dargestellt, die sich dieser heterogenen Methoden bedienen. Es wird gezeigt, dass dadurch die Genauigkeit der Algorithmen in beiden Fällen verbessert, sowie die Robustheit der Modelle erhöht werden kann. Grundlage dafür ist das effektive Nutzen der verfügbaren Datenquellen, sowohl aus der eigenen als auch aus verwandten Problemstellungen. Im Anschluss werden aktuelle Ergebnisse aus Forschung und Entwicklung diskutiert, die dazu beitragen können maschinelles Lernen auf mobilen Geräten voranzutreiben.

Contents

Chapter 1

The Big Data context

Big Data The roots of Big Data are in the late 2000s, when mostly the terms *Business Intelligence and Analytics (BI)* were used for data visualisation, dashboarding, and reporting. However, with the growing number of devices which are producing data, the possibilities of how to make use of that vast amount of information increased heavily. Mobile phones, equipped with a multitude of sensors, exceeded the number of PCs and laptops in 2011. [CCS12] According to Gartner [Incb], 3.8 billion connected devices were in use in the end of 2014. All these devices produce massive amounts of data, allowing for new ways of location-, context-, or person-based analytics by incorporating this information into BI and other methods for data-analysis. Reports [CCS12] claim that the number of connected devices will heavily increase further while the amount of information transferred via the Internet is growing exponentially [BF10]. Therefore, new ways and algorithms are needed to store, process and transfer these large amounts of data.

Multimodality Current state-of-the-art smartphones feature accelerometers and gyroscopes, as well as light, location, image, and novel fingerprint sensors. [Sch] With modern operating systems like Android and iOS, these different sensor-types became easily accessible for application developers to use (e.g. [Incc]) because of the commonly established interface between hardware and software.

This large amount of data from various modalities available also poses some challenges for application developers and researchers. Due to large variations in both number and types of sensors, the heterogeneity of acquired signals can be quite high amongst different types of mobile platforms [SBB+15]. In addition, even in the Big Data era, there still are major problems related to available data:
It can still occur that there is just not enough data available to train a learning system or that there is a lot of information available, yet unlabelled. For this purpose, this report discusses two heterogeneous learning techniques, *Active Learning* and *Transfer Learning*, which can help to improve a learner's performance by efficiently leveraging today's abundance of data.

Chapter 2

Theoretical background on heterogeneous learning

The following section should provide a theoretical overview about heterogeneous machine learning techniques. When presenting the different approaches, there are two common terms used in the literature which describe a single machine learning problem: a domain and a task. A domain D consists of a feature space X and the respective marginal probability distribution $P(X)$. Regarding a specific domain, a task T consists of both a label space Y and an objective predictive function f, whereas Y contains all the labels which are possible in a specific machine learning setting. f maps a certain feature vector in X to a certain label in Y, thereby connecting domain and task. In most examples, X will be a n dimensional space, in which each data-instance can be represented by a point in that space. [PY10]

2.1 Transfer learning

2.1.1 Scenarios for transfer learning

Transfer learning is a machine learning technique which, compared to traditional machine learning algorithms, does not address isolated tasks only. It much more leverages knowledge, which has been acquired from a certain task to solve another similar issue. In essence, it is very much comparable to the way how human beings learn to solve a previously unknown problem - taking knowledge from previous tasks and applying them to the new objective.
Transfer learning can mostly be seen as an extension to various traditional machine learning methods rather than an entirely different approach – it works well with common classification and inference algorithms. When looking at a common learning curve of a certain machine learning task, which is plotting the prediction performance over the training time, transfer learning can be seen as a possibility to increase the learning performance. The three characteristics of the graph in figure 2.1 show this relationship: Increased initial starting performance, higher slope and higher

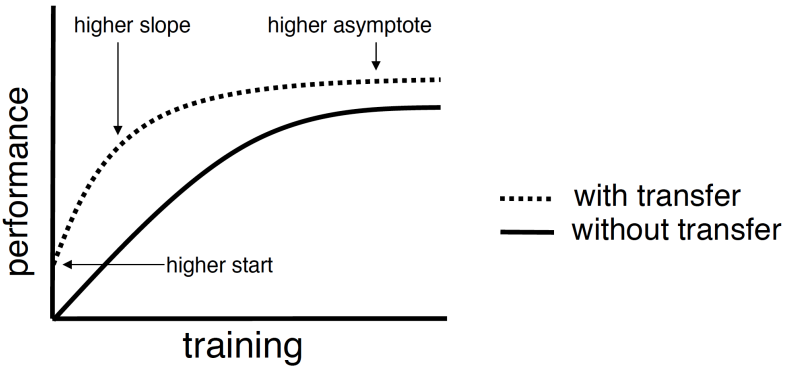

Figure 2.1: Learning process with and without knowledge transfer. Reprinted from [TS09, p.2]

.

asymptotical peak performance [TS09].
This possibility to learn a certain target task T_t faster is especially attractive for problems where only few or even not enough training data are available to learn the task of interest with high accuracy. If there is a similar task T_s which in contrary provides a lot of training data in its domain D_s, it might be used for transfer learning in order to boost the learning performance of the target task. This does also work for unsupervised learning – in case obtaining (unlabelled) data from the target domain D_t is very expensive, there might exist a related source domain D_s for which it is easier, the knowledge can be transferred. [PY10, p.1]

Further categorisation Since transfer learning can be difficult to grasp and highly depends on the target machine learning context, it makes sense to further categorise the settings as shown in [PY10, p.3–4] into *inductive transfer learning*, *transductive transfer learning* and *unsupervised transfer learning*.
Inductive transfer learning implies that source and target tasks are different from each other, regardless of how their domains are related. However, in *transductive transfer learning* source and target task are the same, yet their domains are different (either feature space X or marginal probability distribution $P(X)$). Thus, this setting is often related to domain adaption in various machine learning environments. Finally, *unsupervised transfer learning* is very similar to *inductive transfer learning*, but in an unsupervised context, which means that there is no labelled data available from either domain. [PY10]

Transferring knowledge

According to [PY10], knowledge can be transferred in the following ways, each to be applied in the machine learning context of choice:

Transfer of instances Labels from the source domain are to be re-labeled eventually and then re-used in the target domain

Transfer of feature-representations Valid feature-representations (often of lower dimensionality) are shared to increase the relatedness of the domains

Transfer of parameters Specific parameters are shared to increase the performance of the target setting and give a head-start during learning

Transfer of relational knowledge Generate knowledge to map from one domain to the other and use that mapping to reach an increase in performance

2.1.2 Negative transfer

One major problem for transfer learning is negative transfer. This means that a particular transfer method not only fails to increase the performance of the learner but may even decrease it. This situation can arise in cases where source and target context are not similar enough, the mapping between the source properties and target-properties is done in a wrong way, or the transfer method itself is not applicable. [PY10]
Some methods address negative transfer by using a more cautious way to transfer knowledge – this means that the knowledge gained from the source setting is only transferred in part. Therefore, the impact of the transfer is being lowered, yet it also lowers eventual positive effects. An additional possibility is to reject bad information during transfer as done in *option-based transfer* for reinforcement-learning [CDB08]. Thereby, the learner can evaluate transferred knowledge and drop data-instances once they yield bad results. In case multiple source tasks are available to gather knowledge from, there are further possibilities: (1) a single task within the group is chosen which fits best to the target task. (2) all the tasks involved can be explicitly modelled in order to evaluate their transferability [EL$^+$08].

2.2 Active learning

Active learning is the second heterogeneous learning technique covered by this report which can be used in machine learning to improve the learning process, especially its speed. The core idea of active learning is that an algorithm can choose a certain unlabelled sample from a given pool in order to obtain the label for it. This sample is chosen on the basis of how *valuable* it would be for the learning process. Therefore, labels are obtained in a way that the learner progresses faster compared to using

randomly chosen data-instances. Once selected, the algorithm can query the label of the chosen sample from an *oracle*, often realised as a human. It is assumed that the *oracle* knows the context of the problem and can therefore label data-instances correctly such that they can be used for training. Since the sample has been selected based on how *valuable* it is for the learner, the algorithm's performance can be increased while reducing the learning effort. There are many frameworks which provide rulesets and algorithms to actually select a single sample from a set of data, some of which will be discussed in this section.

2.2.1 Use-cases for active learning

Active-learning is especially useful when the target scenario provides a lot of data, for which it is difficult to obtain labels for. In that case, the learner could query certain samples which, once labelled, provide a more significant gain in learning performance. Thereby, the overall number of labelled samples which are needed to achieve a certain target accuracy is reduced. [Set12]

2.2.2 Obtaining queries

The selection process itself, as well as the definitions of *value* and *utility* highly depend on the query selection framework. There are various frameworks which describe how to obtain data-instances covered by literature. Common approaches are summarised below.

Query synthesis Query synthesis is a method in which the learner selects an unlabelled data-point within the feature-space with high *utility* and presents it to the *oracle*. This also includes samples, which have not been obtained previously but still lie in that feature space, because the learner is allowed to construct samples itself. The definition of *utility* again depends on the measure of choice (section 2.2.3). There are approaches using a region of uncertainty, explicit utility measures or a model-committee. More details on utility measures are discussed in the next subsection. [Ang88] It has been shown that query synthesis can improve the performance by reducing the amount of samples which are required to reach a certain accuracy, for instance for regression [CGJ96] and finite problem domains [Ang04].

Selective sampling Selective sampling can be a very useful method in settings where unlabelled data is pretty much abundant. In comparison to synthesised queries, the data which is obtained always consists of real-world data-instances, not synthesised ones. This guarantees that the data which might be staged to be labelled in a next step is actually sensible because their source is a real-world distribution. This approach is sometimes also called *Stream-based selective sampling* because data are obtained as a stream, not at once. [Set12]

Pool-based sampling This approach is very similar to selective sampling, with a major difference however that multiple data-items are available at the same time. The core of the idea is that an algorithm can essentially choose between unlabelled data-instances from a large pool of data, again using a utility measure to select the data-instances to query the labels for. Under normal conditions, this pool is closed (static) but this is not a hard requirement. [Set12]

2.2.3 Utility measures

There are many methods under discussion in the literature (e.g. [Set12]) on how to evaluate data-points (i.e. their *value*) and thus detect the next sample which should be queried by the learner. Common strategies are shown below.

Uncertainty sampling For this strategy, the one data-instance is queried whose predicted label or output y is the least confident within the feature space X. A simple example for a support vector-machine (SVM) would be that a data-point is considered the most informative one for which the euclidian distance between its feature vector and the hyperplane separating two classes in the feature space X is the lowest. [CNKK14, TC01]
The uncertainty measure can be extended to focus on the *margin* which could essentially take the information from the posterior distribution into account by looking at the first and second most likely predictions under a certain model. In essence, it assumes that data-points which are very near to each other must be very informative since the possibilities to separate them are fewer.
A very general strategy to determine utility of a data-point is to measure its *entropy* and thereby its information content. *Entropy* is used very often in machine-learning in general, for instance as a log-loss interpretation. [Set12]

Query by committee Another popular approach is called query-by-committee which operates in the version space V, representing multiple different *hypothesis* – themselves consistent with the training data [FSST97, Set12]. In this strategy an ensemble of classifiers or approximators is used to determine a data-instance with a high rate of error, meaning the ambiguity of that data-instance is high when looking at the outputs of the committee. To form the committee, one could either use multiple different types of classifiers or approximators in parallel, or use the same type more than once and train each model on different input data from the input space. [CCT10, KV$^+$95]

Chapter 3

Applications and implementations for mobile devices

3.1 Maschine learning on mobile devices

Especially in machine-learning, algorithms are often resource-hungry and have thus been limited to devices with high computational capabilities. With the increase of computing power on mobile platforms, these devices have now become a potential target to run various algorithms which have been constrained to desktop or static systems beforehand. Although their performance increases steadily, there are still aspects which have to be considered by an developers and researchers to make algorithms fit the mobile context:

Computational capabilities Although the computational capabilities of mobile platforms are steadily increasing with the gain in overall CPU performance, they will mostly fall behind their static counterparts. Mobile computation systems, in form of System-on-chips (SoCs) or even dedicated computing extensions like mobile GPUs in notebooks, are mostly making a tradeoff between energy consumption and performance. The reason is that hardware developers have to keep in mind limitations in both package size and availability of power on a mobile platform. This not only affects the computational capabilities in form of calculation speed but also in form of application memory (RAM).
Although these devices have to make sure that energy is consumed in a controlled manner, performance tends to reach a level in which high CPU speeds and even mobile high-performance computing (Mobile HPC) is possible. [RCG+13]

Power Power is generally limited on mobile platforms due to limitations in battery charge. This not only affects the hardware which can be deployed on these devices, but also the software which is running on that hardware. Both hardware, firmware, operating system and applications have to make sure to reduce their CPU time to the necessary minimum to keep the power consumption low and thus the battery lifetime

high. Especially for algorithms with a long runtime at high CPU load, this can be a major problem. Considering multiple training epochs of a classifier for instance, which may take up to a few hours on mobile hardware with an associated high compute load would bring a major disadvantage for the battery lifetime, especially when executed frequently.

Storage As shown in [KAU12], application speed is affected by the devices' storage capabilities due to database access and other applications' I/O. This may not directly affect every algorithm that is being executed, but definitely a whole application's overall performance when running next to the operating system and other instances. This is especially important when the application relies on its own file I/O such as reading from a database. The paper states that, especially for smartphones, this can be a highly relevant factor due to high variance for storage devices.

3.2 Implementation examples

3.2.1 Transfer learning for deep convolutional networks

In recent years, deep neural networks (DNNs) got high attention after new algorithms were developed to more effectively train deep networks on state-of-the art computing hardware [EBC+10]. Deep neural networks are structured in a hierarchical way (layers) and have proven to be particularly effective for certain tasks, e.g. image classification [SLJ+14].
In this sub-section, a workflow is shown in which a pre-trained model of a deep convolutional neural network can be taken and adapted (fine-tuned) to a different task of a similar domain, in this example image-classification. The adapted model can then be deployed on an execution device, e.g. a phone, which does the evaluation of samples based on the fine-tuned model. This results in less (computational-) effort for the actual use-case, as well as in better results when only few data are available. In a simple machine learning setting one might have the task to decide to which class an image belongs to, which has been captured by a smartphone camera and solve this problem with a DNN. Instead of starting to train from scratch on the phone itself using a multi-layer neural network one could use a model that has been pre-trained and fine-tune it towards the target task on a more powerful hardware platform (off-device). To do so, learned parameters are transferred in an *inductive* manner. Afterwards one could deploy the model on the phone where mainly inference is done and eventually further adjustments/training (on-device). Ideally, the model to start with has been trained on a source task T_s being similar to the target task T_t. Pre-trained DNN models often use a network architecture which has been particularly successful in a certain problem domain and have been trained on powerful hardware [KSH12]. These models are available online(e.g. [VCb]) for various frameworks like *Caffe* [VCa].

It has been shown that for image classification, the first layers of a deep convolutional network always learn similar features, i.e. Gabor Filters or color blobs. Since these features are general throughout different classification tasks, they provide potential to be transferred to a new task and enable a head-start for the target learning process. This is especially useful in case only few training data are available for the desired target task because the network can preserve the transferred lower levels' general features and focus on tuning the upper, more specific layers toward the target task. The degree to which the transferred network is kept depends on the setting of the target task itself - the less training data are available, the more layers should be kept or *frozen* and thus leveraged during the transfer to avoid overfitting. The more training data are available, the more layers can be fine-tuned because for higher amounts of data, the risk of overfitting is generally lower. [YCBL14]

To quickly compare the training results of training a network from scratch versus a transfer learning approach, a dataset from Caltech [FFFP07] can be used for training both a randomly initialised and a pre-trained version of the AlexNet [KSH12] network. The experimental data-set from Caltech features 101 classes of random objects, whereas the number of images per class was reduced to 10 images each, in order to create a scenario with a very low amount of training data.
The pre-trained model can be obtained from the Caffe Model Repository [VCb], which resembles the snapshot at iteration 360,000. It was pre-trained on the ImageNet data-set, consisting of around 1.2M images distributed across 1000 categories.
Training has been conducted using NVIDIA DIGITS as a front-end for the Caffe Deep-Learning framework [Incd].
Figure 3.1 shows the comparison between the pre-trained and randomly initialised model in regard of training accuracy (orange) and validation loss (green). The experiment has been conducted several times with similar results while training for 60 epochs each time.
It can be seen that the pre-trained network not only reaches a higher overall accuracy after 60 epochs, the slope of the accuracy curve is also much higher in the beginning while reaching the asymptotical peak faster. Similar result were obtained in [ZF14].
Once the fine-tuning has been finished, the model can be deployed on a mobile device and used for inference in the target task's context. Fine-tuning with given parameters would also give a head-start for pure on-device training, thus immensely reducing computational effort which is required to reach the desired accuracy. Training off-device can on the other hand save both battery-lifetime and filesystem I/O due to less frequent snapshots being made of the trained model and parameters. Figure 3.2 shows the process from knowledge transfer (off-device) to deployed inference on a mobile device (on-device).
With the growing computational capabilities of smartphones (GPUs, CPUs and even dedicated hardware) and ongoing optimisations of deep neural networks, these models can become an attractive option for classifying data in real-time, even at

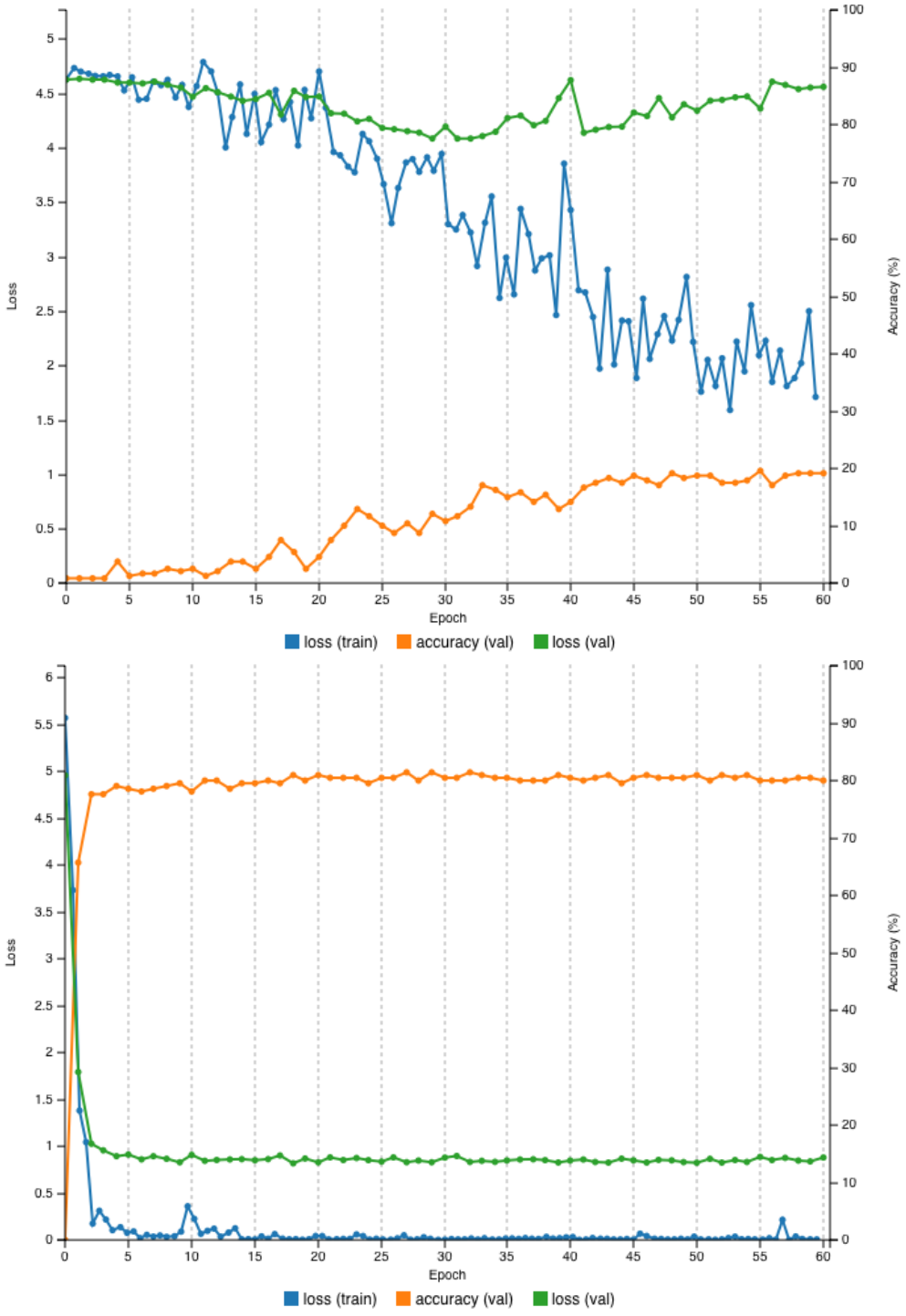

Figure 3.1: Learning curve of a randomly initialised AlexNet (top) and a fine-tuned AlexNet (bottom). Both networks have been trained on a very low amount of training-data.

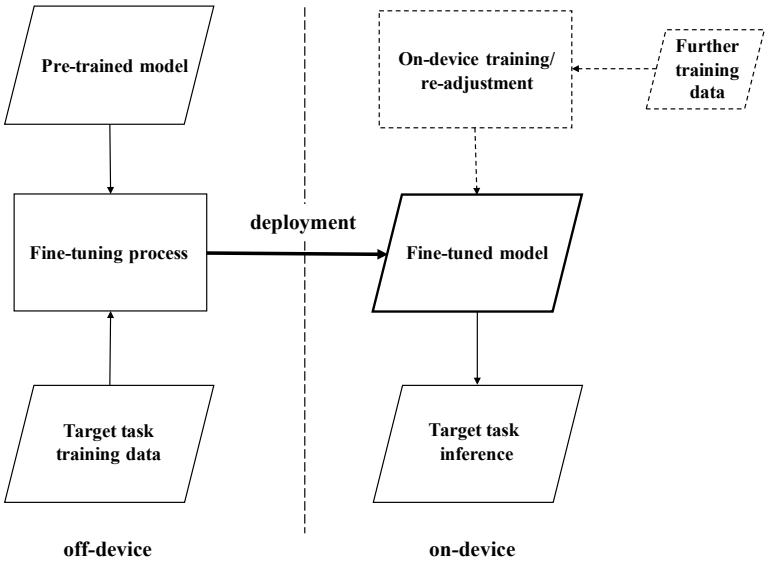

Figure 3.2: DNN transfer-learning and deployment process overview

large size [CW11, Incf, DYDA12]. Current compute platforms as mentioned in the last chapter can run these networks already at high speed due to hardware support for parallelised computations.

3.2.2 Active learning with context recognition on smartphones

Context-awareness is recently seen as a highly important aspect for fields like ubiquitous computing, personal life-assistance, collaborative e-learning and many other applications in the mobile world. Context itself has become much more than just a single value of a sensor, for example ambient temperature. It rather evolved to a high-dimensional space, created by the surrounding environment of the device and adhered by a multitude of sensors which can be found in mostly all modern smartphones. [SL06,Gay09] Continuous-sensing can be a key-enabler for retrieving context because the smartphone can continuously retrieve and evaluate incoming sensor data throughout its uptime. Modern operating systems also support this kind of sensing methodology via multi-tasking and parallelism capabilities, enabling the sensing application to run in the background without being interrupted. [LML+10] In addition, hardware manufacturers strive to provide a platform which drives continuous sensing on a hardware-level. ARM [SLS] for example is investigating in a setup called *Sensor Hub*, which provides higher-level access to the devices' sensors while wrapping lower-level embedded software to make real-time data-acquisition faster,

more energy-efficient and easier to access.
Active Learning can be a suitable tool to improve learning performance in Human Activity Recognition (HAR) and context recognition tasks because of the following: Unlabelled data is pretty much abundant. In a continuous-sensing setup, sensors are producing large amounts of unlabelled data for each activity that is carried out by the user. However, obtaining labels can indeed be expensive since sensor data can vary with either changes of the location of the sensor (e.g. smartphone in different pockets) [KLLK10] or varying sensor hardware used in different smartphones [SBB+15] which both induce high variance in the training data.
Amazon Mechanical Turk [Inca] can be a way to obtain labels for data-sets in a large scale but there is an inherent problem with labels for activity recognition: in case samples are labelled by a human *oracle*, it is not guaranteed that the labels are indeed correct. Sensor data can be difficult to read and understand by a human, thus labels can be prone to error as presented in [ZSS11]. The paper addresses the difficulties for labelling data using workers from Mechanical Turk by reducing label-noise through re-labelling. Therefore, highly certain labels can be acquired for the activity recognition scenario which is supported by active learning. The data is then classified by a SVM, which is also used to retrieve the next unknown data-instance to query the true label for. For sampling, a modified version of the max-margin approach was used. It also incorporates further information of the underlying distribution by taking the unknown data-vectors' euclidian distance to a known instance, weighted with a factor based on the known instance's euclidian distance to the next data-cluster centroid in the feature space. The centroids have been computed by k-means clustering. The data which is used in the paper is based on different modalities, generated by multiple accelerometers, gyroscopes and magnetometers which are all combined into one big feature-vector (and thus classified using one SVM).
Having data from different modalities offers the additional possibility to combine those sources of information using multiple models. As shown in [SVLS08], multiple sensors of a different type can boost the classification performance when used with multiple classifiers in an ensemble. The paper uses two active learning approaches in order to obtain labels for certain data-points to train the classifiers. The algorithm searches through a pool of samples (pool-based sampling) and selects those data-points which should be queried from the user. For sampling functions, two approaches were used - uncertainty sampling and query-by-committee. According to [SVLS08], active learning together with combined classifiers for multiple sources of sensor-data outperforms supervised learning as well as other semi-supervised approaches to some extent. Especially uncertainty sampling performs pretty well in such an environment.
Both papers show that active learning can be a highly valuable strategy for HAR and context-recognition tasks since the nature of the problem fits well to the strengths of active learning. The multitude of sensors available on today's smartphones are well suited for multiple classifiers used in an ensemble and the progressing support

of continuous-sensing is making it simpler for application developers to use all the device's resources in a standardised manner.

Chapter 4

Developments on mobile phones and their implications on machine learning

4.1 Compute-load distribution on heterogeneous computing architectures

Hardware-manufacturers are working on more powerful API support to distribute compute load to the various computing units on a mobile/heterogeneous platform. This enables applications to offload computations which are normally carried out by the CPU to a massively parallelised computing architecture to improve in speed. To do so, the algorithms themselves have to be able to benefit from a parallelised architecture which in turn depends on their design. Especially data-level parallelism is important to make an application benefit from a parallel API because computations are mostly carried out by running the same code on multiple compute nodes with potentially different input or parameters per node. [K$^+$07, GO11]
The GPU manufacturer NVIDIA released their GPGPU API toolkit CUDA for ARM chipsets in 2013 [Incf] which are primarily used in smartphones and alike. This also enables GPUs in mobile SoCs to run GPGPU computations which have previously been restricted to desktop environments. Similar to this proprietary solution from NVIDIA is the open programming framework OpenCL [Gro] which not only focuses on GPGPU programming, but rather on distributing the algorithm on more heterogeneous computing units. OpenCL offers the possibility to run the code on a computing unit different to a GPU, e.g. DSPs or other specialised hardware, which is often found in mobile SoCs.
Both OpenCL and CUDA could serve as a compilation target for higher level compilers in a next step. These compilers decide which task has to run on which computing unit and thus generate necessary code (as shown in [DBB07] or [Wol10]). Works like [SY15] show that these APIs can successfully boost the performance of ma-

chine learning algorithms once running on a heterogeneous architecture. Also, as a quite specific example, the CUDA-enabled NVIDIA Jetson TX1 [Ince] board can run a deep convolutional neural network comparable to the previously mentioned transfer-learning scenario at very high speeds using the Caffe DNN framework and the CUDA API.

4.2 Direct hardware support for machine learning

The execution environment of a machine learning algorithm can be taken one step further by using dedicated hardware. Some papers [LV13, NMH15] already investigated this possibility, mainly using FPGAs to implement an algorithm (e.g. SVM classification) in hardware to accelerate its execution. The company Qualcomm approaches this solution by announcing Zeroth [Qua], a cognitive platform to support machine learning with the next Snapdragon 820 processor. Qualcomm wants to use a distinct processor called NPU (Neural Processing Unit) for dedicated machine learning tasks. [Ara14] The new chip should be designed to compute on a spiking neuron basis which thus enables a hardware-based computing platform for spiking neural networks [JG].
Combining these new frameworks with specialised hardware could therefore highly improve the performance of current algorithms. Together with a multitude of sensory input, the increased throughput could enable more complex and accurate systems on mobile platforms. Once machine learning sets foot in a large number of mobile applications, hardware manufacturers may find it worth integrating direct hardware acceleration into their architectures.

Chapter 5

Conclusion

The report shows that the Big Data era provides many possibilities for data-analysis and machine-learning applications in business and research contexts. Especially on mobile devices, the abundance of multimodal data in form of sensory- and user-collected information can be the basis of many new applications. However, mobile platforms pose some constraints on existing algorithms in form of compute performance, power and storage limitations. In addition, heterogeneity of available sensor hardware, as well as high velocity and variety of data can be a challenge for developers and researchers. Transfer learning on the one hand can be very beneficial in case only low amounts of (training-) data are available in the target domain and it is possible that knowledge can be leveraged from a similar domain. Active learning on the other hand can be very useful in settings where a lot of unlabelled data are present in the target domain, yet it is expensive to obtain labels for them. An exemplary application for either technique shows how these approaches were applied successfully in order to boost the learning performance while tackling aforementioned problems. Convolutional deep neural networks can improve significantly with transfer learning once only few training data are available. For context-recognition on the other hand, active learning can provide a framework to effectively deal with large amounts of unlabelled data. Current technology trends show that there is high potential for machine learning algorithms on mobile platforms to improve in performance in the future. Heterogeneous hardware architectures and corresponding APIs, as well as specialised machine learning hardware could provide a boost for common approaches in data-mining and machine learning. For large scales however, it will be very interesting to see whether hardware manufacturers see it worth implementing those features (especially distinct machine learning hardware) in their mass market products.

List of Figures

Bibliography

[Ang88] Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

[Ang04] Dana Angluin. Queries revisited. *Theoretical Computer Science*, 313(2):175–194, 2004.

[Ara14] Karim Arabi. Low power design techniques in mobile processors. 2014.

[BF10] David Bollier and Charles M Firestone. *The promise and peril of big data*. Aspen Institute, Communications and Society Program Washington, DC, USA, 2010.

[CCS12] Hsinchun Chen, Roger HL Chiang, and Veda C Storey. Business intelligence and analytics: From big data to big impact. *MIS quarterly*, 36(4):1165–1188, 2012.

[CCT10] Crystal Chao, Maya Cakmak, and Andrea L Thomaz. Transparent active learning for robots. In *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, pages 317–324. IEEE, 2010.

[CDB08] Tom Croonenborghs, Kurt Driessens, and Maurice Bruynooghe. Learning relational options for inductive transfer in relational reinforcement learning. In *Inductive Logic Programming*, pages 88–97. Springer, 2008.

[CGJ96] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 1996.

[CNKK14] Elisavet Chatzilari, Spiros Nikolopoulos, Yiannis Kompatsiaris, and Josef Kittler. Active learning in social context for image classification. In *9th International Conference on Computer Vision Theory and Applications, VISAPP*, 2014.

[CW11] Kwang-Ting Cheng and Yi-Chu Wang. Using mobile gpu for general-purpose computing–a case study of face recognition on smartphones. In *VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on*, pages 1–4. IEEE, 2011.

[DBB07] Romain Dolbeau, Stéphane Bihan, and François Bodin. Hmpp: A hybrid multi-core parallel programming environment. In *Workshop on General Purpose Processing on Graphics Processing Units (GPGPU 2007)*, 2007.

[DYDA12] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 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.

[EL+08] Eric Eaton, Terran Lane, et al. Modeling transfer relationships between learning tasks for improved inductive transfer. In *Machine Learning and Knowledge Discovery in Databases*, pages 317–332. Springer, 2008.

[FFFP07] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[FSST97] Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2-3):133–168, 1997.

[Gay09] Geri Gay. Context-aware mobile computing: affordances of space, social awareness, and social influence. *Synthesis Lectures on human-centered informatics*, 2(1):1–62, 2009.

[GO11] Dominik Grewe and Michael FP O?Boyle. A static task partitioning approach for heterogeneous systems using opencl. In *Compiler Construction*, pages 286–305. Springer, 2011.

[Gro] Khronos Group. Opencl. Accessed 03.01.2016. URL: `https://www.khronos.org/opencl/`.

[Inca] Amazon Inc. Amazon inc. Accessed 14.01.2016. URL: `https://www.mturk.com/mturk/welcome`.

[Incb] Gartner Inc. www.gartner.com. Accessed 29.12.2015. URL: `http://www.gartner.com/newsroom/id/2905717`.

[Incc] Google Inc. Android sensor api. Accessed 29.12.2015. URL: `http://developer.android.com/guide/topics/sensors/sensors_overview.html`.

[Incd] NVIDIA Inc. Nvidia digits. Accessed 05.12.2015. URL: http://devblogs.nvidia.com/parallelforall/easy-multi-gpu-deep-learning-digits-2/.

[Ince] NVIDIA Inc. Nvidia jetson tx1. Accessed 28.12.2015. URL: http://devblogs.nvidia.com/parallelforall/nvidia-jetson-tx1-supercomputer-on-module-drives-next-wave-of-autonomous-machines/.

[Incf] NVIDIA Inc. Nvidia releases cuda for arm. Accessed 03.01.2016. URL: http://devblogs.nvidia.com/parallelforall/cuda-arm-platforms-now-available/.

[JG] Qualcomm Jeff Gehlhaar. Qualcomm low power soc. Accessed 04.01.2016. URL: https://www.cs.utah.edu/asplos14/files/Jeff_Gehlhaar_ASPLOS_Keynote.pdf.

[K+07] David Kirk et al. Nvidia cuda software and gpu parallel computing architecture. In *ISMM*, volume 7, pages 103–104, 2007.

[KAU12] Hyojun Kim, Nitin Agrawal, and Cristian Ungureanu. Revisiting storage for smartphones. *ACM Transactions on Storage (TOS)*, 8(4):14, 2012.

[KLLK10] Adil Mehmood Khan, Y-K Lee, SY Lee, and T-S Kim. Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis. In *Future Information Technology (FutureTech), 2010 5th International Conference on*, pages 1–6. IEEE, 2010.

[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.

[KV+95] Anders Krogh, Jesper Vedelsby, et al. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7:231–238, 1995.

[LML+10] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[LV13] Kyong Ho Lee and Naveen Verma. A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals. *Solid-State Circuits, IEEE Journal of*, 48(7):1625–1637, 2013.

[NMH15] Katayoun Neshatpour, Maria Malik, and Houman Homayoun. Accelerating machine learning kernel in hadoop using fpgas. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pages 1151–1154. IEEE, 2015.

[PY10] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[Qua] Qualcomm. Zeroth cognitive platform. Accessed 04.01.2016. URL: `https://www.qualcomm.com/invention/cognitive-technologies/zeroth`.

[RCG+13] Nikola Rajovic, Paul M Carpenter, Isaac Gelado, Nikola Puzovic, Adrian Ramirez, and MR Valero. Supercomputing with commodity cpus: are mobile socs ready for hpc? In *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*, pages 1–12. IEEE, 2013.

[SBB+15] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 127–140. ACM, 2015.

[Sch] M Sc Maximilian Schirmer. Smartphone Hardware Sensors.

[Set12] Burr Settles. Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, jun 2012.

[SL06] Thomas R Roth-Berghofer Stefan Schulz and David B Leake. Modeling and retrieval of context. 2006.

[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.

[SLS] Steve Scheirey, Hillcrest Labs, and Diya Soubra. Sensor Fusion , Sensor Hubs and the Future of Smartphone Intelligence.

[SVLS08] Maja Stikic, Kristof Van Laerhoven, and Bernt Schiele. Exploring semi-supervised and active learning for activity recognition. In *Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on*, pages 81–88. IEEE, 2008.

[SY15] Min Gyung Song and Dongweon Yoon. Optimization of a Machine Learning Algorithm on the Heterogeneous system using OpenCL. 2015.

[TC01] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118. ACM, 2001.

[TS09] Lisa Torrey and Jude Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009.

[VCa] Berkeley Vision and Learning Center. Caffe. Accessed 09.01.2016. URL: `http://caffe.berkeleyvision.org/`.

[VCb] Berkeley Vision and Learning Center. Caffe model zoo repository. Accessed 09.01.2016. URL: `https://github.com/BVLC/caffe/tree/master/models`.

[Wol10] Michael Wolfe. Implementing the pgi accelerator model. In *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, pages 43–50. ACM, 2010.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[ZF14] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[ZSS11] Liyue Zhao, Gita Sukthankar, and Rahul Sukthankar. Robust active learning using crowdsourced annotations for activity recognition. In *Human Computation*, 2011.

License