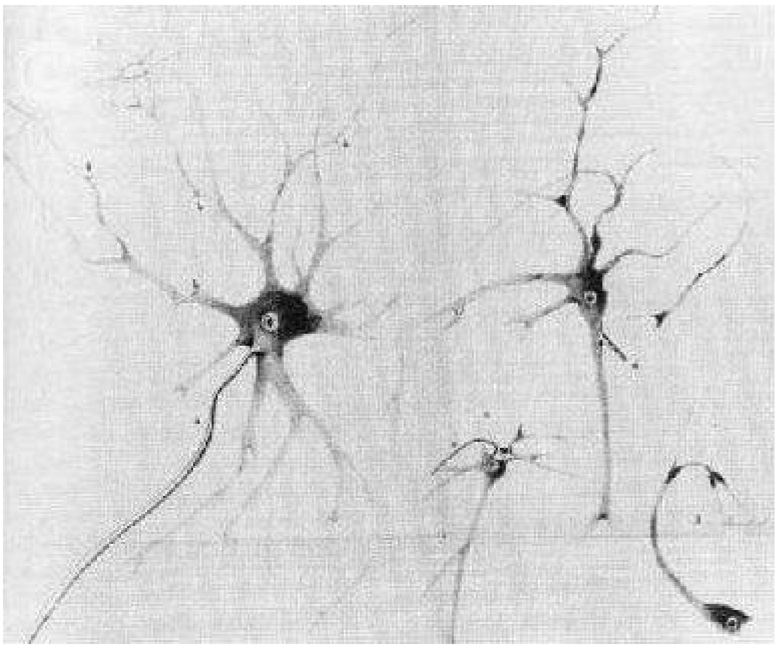

Sorting batteries with deep neural networks

Using deep belief networks and convolutionary neural networks for image classification.

Master of Science Thesis

Intelligent System Design

RASMUS JOHANSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, January 2011

Sorting batteries with deep neural networks
Using deep belief networks and convolutionary neural networks for image classification.

RASMUS JOHANSSON

Examiner: CLAES STRANNEGÅRD

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:
Drawings of what inspired the creation of artificiall neural networks: The neuron.
(Deiters 1865).

Department of Computer Science and Engineering
Göteborg, Sweden January 2011

Abstract

This master thesis evaluates the possibilities of using different versions of deep Artificial Neural Networks for identifying batteries at a battery sorting facility. GPU implementations of different versions of Deep Belief Networks and Convolutionary Neural Networks are evaluated on data sets made up by images of batteries. Different ways to get higher performance from the nets, in terms of percent correct classified, are then explored. The results suggest that the methods could be used to get a fast, and therefore industrial usable system, that can handle the identification of a large number of battery classes, with an error rate within the demands of the industry.

Contents

Terms

Accuracy :	Percent correct classified images
Error rate :	Percent incorrect classified images
ANN:	Artificial Neural Network
DBN :	Deep Belief Network
RBM :	Restricted Boltzmann Machine
CNN :	Convolutionary Neural Network
Class:	The fraction the batteries should be sorted into (eg: lead, alkaline, nickel metal hydride)
Sub-class :	A certain kind of battery within a class (eg : Phillips power alkaline, Duracell plus)
Instance :	A specific physical battery of a sub-class

1 Introduction

This master thesis was carried out as part of the Master program Intelligent System Design at Chalmers University of Technology and Optisort AB in Gothenburg Sweden. The goal was to explore if Deep belief nets or Convolutionary networks could offer an alternative or supplement to the methods used by Optisort.

1.1 Background

In Sweden alone, the amount of depleted portable batteries that is collected each year, is estimated to be around 1450 ton [2]. There is a lot of hazardous material in batteries and they must be sorted according to these to be properly recycled. Batteries are today separated into 8 different classes when sorted for recycling. Sealed lead acid, alkaline/zinc carbon, nickel metal hydride(Ni-MH), lithium primary, mercury, lithium ion(Li-ion) and nickel cadmium. The purity of these fractions must exceed 95 % but many recyclers demand even higher purity depending on the fraction [14].

There exists several approaches to sorting the batteries for recycling, but manual sorting, sorting by weight and sorting by electrical conductivity are the most used[14]. These methods are problematic in terms of both cost and accuracy[14], and therefore it becomes interesting to look for alternative methods.

Machine vision has the potential of working as a stand alone system for sorting, but can also be a part of a larger system, which also takes into account some of the above mentioned properties.

What further makes a visual system interesting, is the extra information it would be able to extract from the batteries. While weight and electrical conductivity basically is the same for all batteries made from the same materials, the label can give information on both the producer and in what time period the battery was put on the market. This is information that the industry is interested in, but for the moment has no good way of acquiring[14].

The obvious way to visually deduce what class the battery belongs to from the label, would be to read the informative text. This is because of dirt and damages often hard to do and the text is also only visible from some specific angles. The approach taken in this thesis is therefore instead to train the networks to identify the battery from all of what can be seen of the label. After finding out the exact identity(which will here after be called *sub-class*) of the battery instance, it will be possible to retrieve what class it belongs to by looking this up in a database.

-

It is estimated that sorting batteries in an industrial setting with machine vision, would mean to discriminate between around 2400 different sub classes[14].

Something important to notice when considering the large number of sub classes to be sorted, is the fact that not all sub classes will be occurring equally frequent. In Sweden the relatively few sub classes produced by big brands like

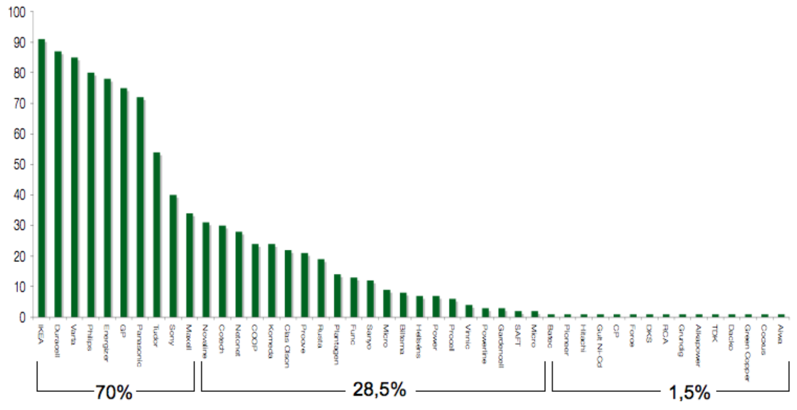

Figure 1: Frequencies of different brands[14]

GP and Duracell, together with big house brands like IKEA and Claes Ohlsson, makes up the absolute majority of the batteries (see figure 1). An accuracy of 95% in an industrial setting can therefore be accomplished by achieving a slightly higher accuracy on a much smaller number of sub classes.

1.2 Purpose

The main goal of this thesis is to explore with what accuracy Artificial Neural Networks(ANN) can manage to classify images of batteries, and with the use of different techniques try to achieve an as high accuracy as possible.

Because of the difficulties it means to acquire data sets of the thousands of sub classes that exists on the market, the networks will not be tested on that many sub classes in this thesis. The hope is instead that it will be possible to arrive at an informed guess on how the system would perform, *if* it was tested on such a data set, by noticing how it performs on some smaller sets.

The networks that will be examined are two versions of ANNs called Convolutionary Neural Networks(CNN) and Deep Belief Networks(DBN). To be able to run as many tests as possible all tests will be done with the help of a Graphical Processing Unit (GPU).

1.3 Questions

The following questions have by the author been identified as important, and will be examined in this thesis.

What kind of network should be used to get the best results when sorting batteries?

There exists benchmarks for DBNs and CNNs on image recognition tasks, but with the data sets and the networks differing from the ones that will be used in this thesis, these results don't give the final answer to what will work best for sorting batteries.

What kind of data set should be used for the training?

How good the network in the end will be able to perform, is besides the network architecture and the method of training the network, very dependent on what kind of data it is trained on. It is especially important to know how a data set should be constructed to enable the trained network to correctly classify instances that, while being of the same sub class, still may look quite different because of dirt and corrosion.

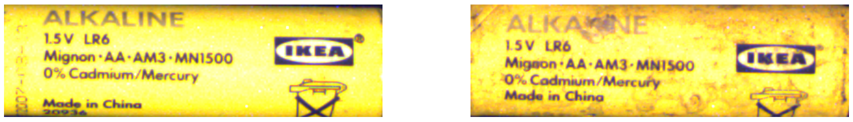

Figure 2: Different instances of the same sub-class

How does the performance of the networks relate to the number of categories?

Although networks similar to the ones that will be investigated, has performed well on some well known data sets [25], the data sets consists of quite few categories. It is therefore important to see how well the network performs when handling more classes and if something could be done to handle more classes in a better way.

How will the option of using several images for classification affect the accuracy?

With more cameras, several images of the same battery, but from different angles, could be used to classify it. This should have the potential to give a much higher accuracy.

How should the output of the networks be processed?

The outputs from the networks will be numbers that represent the probability of the image to belong to a certain sub-class. By inspecting these probabilities it might be possible to reduce the number of false classifications.

1.4 Limitations

To stay on a level reachable for everyone with some experience with neural networks, this thesis will not explain all of the mathematical justifications of the examined ANNs.

The methods for training the networks that were tested in this thesis, were not developed from scratch by the author, but based on the ones explored in Yoshua Bengio's IFT6266 course at Montreal university Canada [8, 7]. The alterations that were made to the networks, and motivations for these alterations are described in chapter 8.

Even with most calculations done on the GPU, experiments on ANNs still takes quite a lot of time. A choice was therefore made to test as many versions of net architectures and data sets as possible instead of optimizing the specific architectures. This means that the results presented in this thesis must be seen as a lower, rather than upper, bound on the performance that is possible to get.

Because of limitations when it comes to data sets, it has not been possible to test the networks on the 2400 sub classes that it is estimated it would need to discriminate between. Effort has instead been made to get an idea of what kind of performance that would be possible to get on such a data set.

Recognizing different sizes of batteries (AAA/AA) can be considered as an "easy" task, in comparison to discriminating between the batteries of the same size, this is therefore not discussed in this project.

The images of the same sub classes will in all data sets always have the same orientation. While this will not be the case in a sorting facility, it will be assumed that this problem can be solved. Either by including images of both orientations when it is possible to acquire bigger data sets, or by constructing one net for one of the orientations, and another for the opposite orientation.

2 Method

The project has been divided into four phases but with much iterations between them.

Literature survey

The most studied literature, has been papers by the inventors of the networks and methods to train these networks, that are investigated in this thesis, Geoffrey E Hinton[3] and Yann LeCun[1]. The different researchers and researching groups that have made contributions to the field often use different software. This meant that finding out what programing languages and librarys that were most practical to use (chapter 7), became a major part of this stage.

Acquisition of data sets

Through out the project, several data sets were produced (chapter 9). They were used to evaluate the relative performance of the different ANNs, but also to see how the performance is depending on properties of the data set.

Construction of networks

Implementations of the studied networks were adapted to fit to the format of the battery images. With these implementations as base, different architectures were developed (chapter 8).

Testing networks and data sets

The different networks were tested on the data sets and the results were used to identify what networks should be further explored (chapter 10).

Iterations, iterations, iterations...

Because it was necessary to test the networks to get an idea of which were worth exploring further, the literacy survey had to be made iteratively with the testing. In the same way, the adaptation of the algorithms and data sets, were made iteratively with the experiments on the performance of the networks.

Part I
Survey of theory

This section describes the networks examined in this thesis. It will explain the basics of Artificial Neural Networks and the core concepts of Deep Belief Networks and Convolutionary Neural Networks. This includes why it is important to make the networks deep and how it with DBNs and CNNs is possible to train such deep architectures.

3 Artificial Neural Networks

Artificial Neural Networks are interconnected sets of model neurons that simulate the function of biological neural networks . The neuron models consist of weights, bias and an activation function [28].

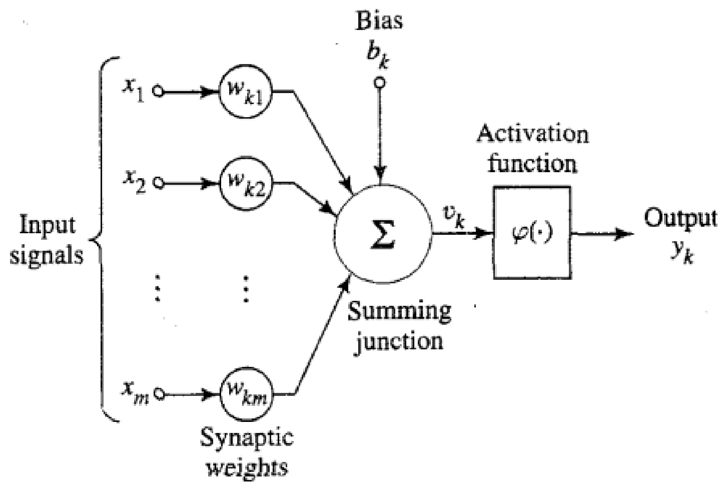

Figure 3: A neuron model [27]

For each neuron, the input signals are multiplied with the weights and the products are summed up together with the bias. The output of the neuron is then calculated as the activation function applied to this sum[28].

neuron output = activation function (input vector * weight vector + bias)

The activation functions most used are the sigmoid function and hyperbolic tangent (tanh), they are both non linear and causes the output to vary between 0 and 1,and -1 and 1.

In a typical *feed forward neural network* (see fig 4) the neurons are ordered in layers and send their output to the layer on top of itself[28]. All the networks that are tested in chapter 10 are instances of such networks. When using such nets for classification, each neuron in the output layer is typically associated with one class. The goal is to by adjusting the weights and biases in the net, get a network that activates the core ct neuron in the output layer, according to the label of the input.

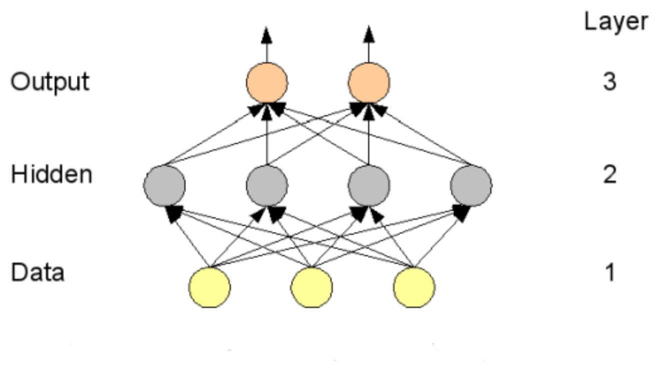

Figure 4: A fully connected feed forward network with one hidden layer

3.1 Depth of the network

It might be sufficient to use nets of two layers of neurons to do a classification task, but the same function can usually be represented in a much more compact way with a deeper net [6]. The neurons then form a hierarchy of progressively more complicated feature detectors and it has been argued that it is better to do the classification with such a net [21]. Single cell recordings in the visual systems of mammals [11], have also shown that this is the way that brains process visual input.

3.2 Training with back propagation

ANNs have been around for a long time, but there was not known of any good way to training them before the *back propagation algorithm* [30].

The back propagation algorithm consists of one *forward pass* and one *backward pass*. In the forward pass the outputs of the neurons in each layer are computed one layer after another, starting from the lowest layer which has the data vector as input. The outputs of the final layer is then compared to the desired output vector to get an error for each neuron in the final layer. Derivatives of those errors are then propagated backwards through the weights to form the error derivatives of the neurons in the preceding layer. Once the error derivative is computed for a neuron the weights leading to that neuron can be changed so that the error is minimized.

Although back-propagation performs well on some problems it performs poorly on many other were it doesn't manage to get better results with more layers[21]. Back-propagation on multiple layered ANNs also have a tendency to get stuck in local optimum at the same time as the learning rate becomes quite slow [21]. Despite the above mentioned advantages of deeper architectures, they have until quite recently because of a lack of a better training method not been discussed much in the machine learning literature[6].

4 Deep Belief Networks

"*Deep belief nets are probabilistic generative models that are composed of multiple layers of stochastic, latent variables*" [18].

This means that in contrast to the above described feed forward networks, which produce an output from a given input, Deep Belief Nets (DBNs) generates output without the need for an input. After training a DBN on images, the outputs it generates will be images similar to the ones it has been trained on.

The method of training a DBN, which will be described below, has showed to be very useful also as a way of training an ordinary deep ANN [21].

4.1 Training with Restricted Boltzmann Machines

A way of training DBNs was found by considering two neighboring layers of neurons as a *Restricted Boltzmann Machine*(RBM)[12].

A Boltzmann Machine is "*a network of symmetrically connected, neuron-like units that make stochastic decisions about whether to be on or off* "[19]. In a RBM these neuron-like units are assumed to be divided in to two layers with no connections between the units of the same layer. The lower layer, which in training is initialized to the data vector (eg an image) is called "visible" and the second layer "hidden". While the connections in a feed forward network are assumed to go in only one direction (from the lower layer to the higher) the connections in RBM are bidirectional.

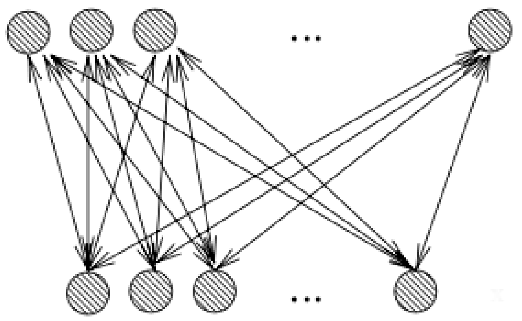

Figure 5: a Restricted Boltzmann machine

The learning of a RBM consists of changing the weights so that the model gets higher probability of generating the data. Theoretically, this would involve multiple iterations of parallel updating of the visible and hidden units, to get an unbiased sample of the model states. However, a much faster way of training the RBM was found, when discovering [19] that the learning works well in practice also if a sample of a reconstruction is used.

Hinton calls this learning method "contrastive divergence" and is described in Algorithm 1.

Algorithm 1 Contrastive divergence [19, 21]

```
1.Starting with a data vector on the visible units, update all of the hidden units in parallel.
2.Update all of the visible units in parallel to get a "reconstruction".
3.Update all of the hidden units again.
$\Delta weight_{ij} =< v_i h_j >_{data} - < v_i h_j >_{reconstruction}$
Were $< v_i h_j >_{data/reconstruction}$ is the probability for unit $i$ in the visible layer. to be active at the same time as unit $j$ in the hidden layer. This while the model is driven by data, respective driven by a reconstruction of the data.
```

More informal: When driven by data (1.) the weight between a unit in the visible and hidden layer should be strengthened if both units are on. When driven by a reconstruction (3.) the weight between a unit in the visible and hidden layer should be weakened if both units are on. In this way the RBM learns to generate the data, instead of the reconstruction. If the reconstruction is identical to the data there is no need to change the weights, and the changes cancel each other. By leaving out the h_j respective v_i term, the same procedure can be used to adjust the biases [19].

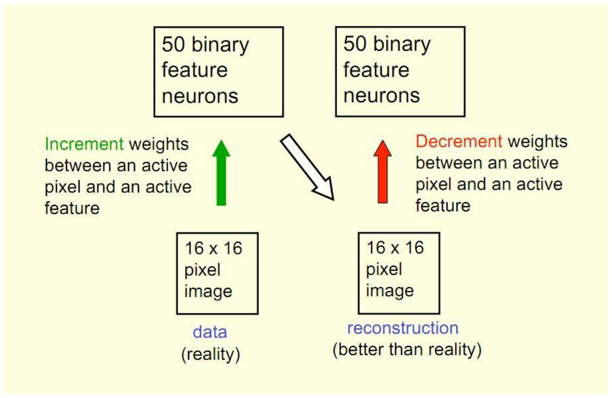
50 binary feature neurons
50 binary feature neurons
Increment weights between an active pixel and an active feature
Decrement weights between an active pixel and an active feature
16 x 16 pixel image
16 x 16 pixel image
data (reality)
reconstruction (better than reality)

Figure 6: Example of the three steps of contrastive divergence[20]

4.2 Training multiple layers

Once one layer is trained, a new RBM can be added on top of the old net. The topmost layer of the old net will be used as the visible layer of the new RBM, and this visible layer will get its pre-training instantiation by propagating the input data from the lowest layer to the new visible layer (see figure 7).

This procedure can be repeated to create as many layers as wanted and in this way constitutes a way to train a deep net in a fast and greedy manner. By running the two topmost layers as a RBM and the rest of the network as a feed forward net (see figure 8), a Deep Belief Network is constructed which will generate data similar to the data it has been trained on.

Hinton claims that :"*It can be proved that each time we add another layer of features we improve a variational lower bound on the log probability of the training data*" [16].

In other words, the new multi-layer model that is created by adding another hidden layer has a better lower bound on the probability of generating the training data than the previous multi-layer model[21].

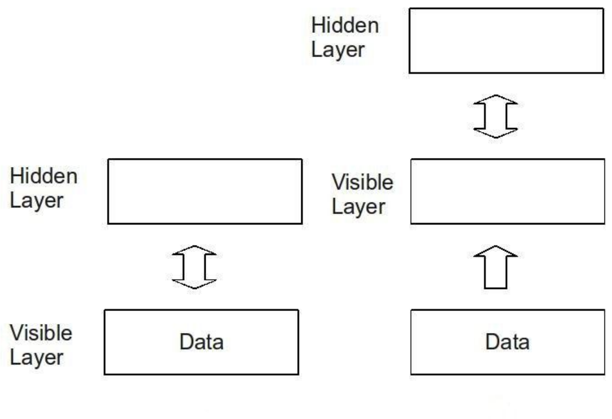

Figure 7: Training of the first and second layers of a DBN

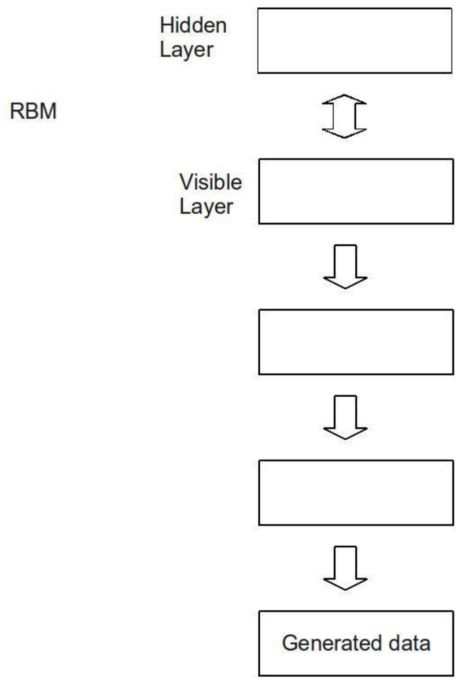

Figure 8: Generating data with a DBN

4.3 Using Deep Belief Networks to classify data.

While there are several ways to use a DBN for classification[17], the best results have been achieved when the weights of the trained network have been used as initializations before fine-tuning with back propagation [21]. Usage of the back propagation algorithm, will now less often end up in a local optimum, as the network already is close to a good solution. There is therefore only need for a local search.

5 Convolutionary Neural Networks

Before the contrastive divergence algorithm, there was no good known way to train deeper versions of fully connected feed forward ANNs. Good results had however been reported with deep networks called *Convolutionary Neural Networks*[24]. These are designed especially for classification of images.

5.1 Exploiting the properties of an image

Edges, corners and other interesting features that can be used for classification of images, are most often local. This is exploited in many computer vision systems, eg. in the widely used SIFT features [26].

CNNs takes advantage of this by limiting the neurons of the first layer, to only be connected to pixels of a smaller area in the image. A neuron is in this way extracting a local feature, and because the same features usually should be extracted in the whole image, neurons that connect to different areas can share the same weights. To be able to extract different features from the image, the neurons are divided into different sets, were weights only are shared within the sets. The result is a number of convolutions of the image, were the shared weights of the different sets are forming the convolution kernels. Convolution of the image with the kernels, creates one *feature map* for each kernel. These feature maps are connected to a sub sampling layer that reduces the size of the feature maps(see 9). In the CNN named le net 5 [24], this sub sampling is done by connecting each non overlapping 2*2 region of the feature maps to one neuron in the sub sampling layer. Each of these neurons computes the average of its inputs and ad this to a trainable bias. The sum is then passed through a sigmoid function. The process can then be repeated by convolving these down sampled feature maps once more, this time with neurons that are connected to the same small regions in all feature maps. The top layer feature maps are connected to an ordinary feed forward net. The full setup of the convolutionary network Le net [24] is shown in figure 9.

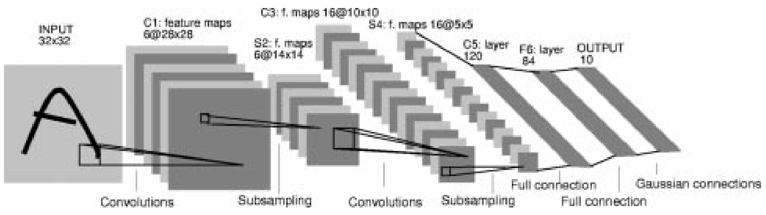

Figure 9: The architecture of the convolutionary network Le net 5 [24]

While it is not totally clear why deep CNNs work so well, two (non excluding) theories have been mentioned [6]. It might be that by limiting the number of inputs to the neurons, the error derivatives don't get as spread out as they would otherwise become (the cause of the good/bad performance doesn't become as unclear). But with CNNs perform well, even with untrained random weights in the first layers, this can not be the whole truth. Another theory is that the hierarchical local connectivity structure by setting all the non-local connections to zero instantiates the network in a setting close to the global optimum.

6 The state of the art

There exists a number of data sets that are used for benchmarking image classification methods. One of the most used is the MNIST database of handwritten digits [25]. It consists of images of handwritten zero to nine digits, divided into a training set of 60,000 examples and a test set of 10,000 examples.

It should be mentioned that handwritten digits can be considered to be rather different from images of batteries. They are not as rigid in their form as batteries are and while the images of MNIST are gray-scale, batteries come in many colors. This is never the less one of the most used data sets, and good performance on this set is a good indication of a method working well also on other image classification tasks.

Ranking	error rate %	method
1	0.35	6-layer NN (on GPU) [elastic distortions]
2	0.39	large conv. net, unsup pretraining [elastic distortions]
3	0.40	Convolutional net, cross-entropy [elastic distortions]
4	0.52	K-NN with non-linear deformation (P2DHMDM)
5	0.53	large conv. net, unsup pretraining [no distortions]
6	0.54	Trainable feature extractor + SVMs [affine distortions]
7	0.54	K-NN with non-linear deformation (IDM)
8	0.56	Virtual SVM, deg-9 poly, 2-pixel jittered

Table 1: The 8 best performing methods on the MNIST dataset of handwritten digits[25]

It can be hard to compare the results because the images are sometimes preprocessed by other methods. As can be seen in table 1 the three most successful methods all used elastic distortions to enlarge the training set. (more of this in chapter 10.5). That it is the deep and convolutionary neural networks that gets the best results, should be seen as a motivation for examining such nets in this thesis.

7 Using the GPU with the Theano library

Even with the reduction in parameters that one gets get with CNNs, and the speed-ups that greedy RBM-training gives, experimenting with neural networks is still a very time-consuming business. To do as much computation as possible on a GPU to speed things up further, was in this thesis therefore identified as crucial. To avoid spending to much time on implementing the methods, an ANN-library that had support for both RBM-training and convolutionary networks was sought for. Comparisons was therefore made between some of the GPU-librarys for Matlab and the python libraries Theano [4] and Gnumpy[29]. Theano was finally chosen because of its ease of use, frequent use for deep neural networks [8], and finally because it unlike Matlab is free .

Theano generates automatically optimized c-code. This means that also without use of the GPU, programs written in Theano can be very fast. It is however claimed that when using a GPU, it can execute instructions up to 140 times faster than on a CPU[4]. Another thing that makes Theano extra useful for optimization tasks like the training of an ANN, is that it automatically computes derivatives of functions. With the error of the network defined as a function, it therefore becomes straightforward to train networks with gradient descent, without ever having to actually implement the back propagation algorithm[4].

Part II
Sorting batteries: Implementing the methods and analyzing the results

8 Evaluated networks

This section describes the setup of the different kinds of CNNs and DBNs that were tested. The networks were adopted from the ones taught at Yoshua Bengio's IFT6266 course [7, 8]. As explained earlier, the network's top layer is used for deciding to which sub class the network assigns an image. Each neuron is associated with a specific sub-class and if the neuron which is associated with the correct class also is the one that has the highest output, then the network has done a correct classification.

While the nets follow the descriptions in part I, the detailed settings are as following.

Deep Belief Network

A DBN as presented in chapter 4 is adopted to images of size 28 * 84 gray scale pixels, with as many output neurons as sub classes in the experiment. It is pre-trained with RBMs and then fine-tuned with gradient descent. The number of neurons are set to 1500 in the first layer, 1500 in the second, 1000 in the third and the same number of neurons as the number of sub classes in the fourth.

Composite Deep Belief Network

The idea behind the construction of the network, which the author has chosen to call *Composite Deep Belief Network*, is to train one DBN for each sub class. This should make the top neurons of these networks become detectors for high level features that are common in images of those sub classes. A shared output layer that connects to the top units of all networks should then be able to use them for discrimination(see figure 10). Hinton[17] explains a similar approach but the author has not been able to find any evidence for this method being used with several class specific DBNs before. The setup of the DBNs were identical to the one explained above, with the difference that the final layer was removed. After unsupervised training of the class specific DBNs, the final layer was connected to the top neurons of those networks. The weights of this final layer was then trained with gradient descent. The main advantage of this kind of network, is that it would be possible to extend it for handling more sub classes, without re-training the whole network.

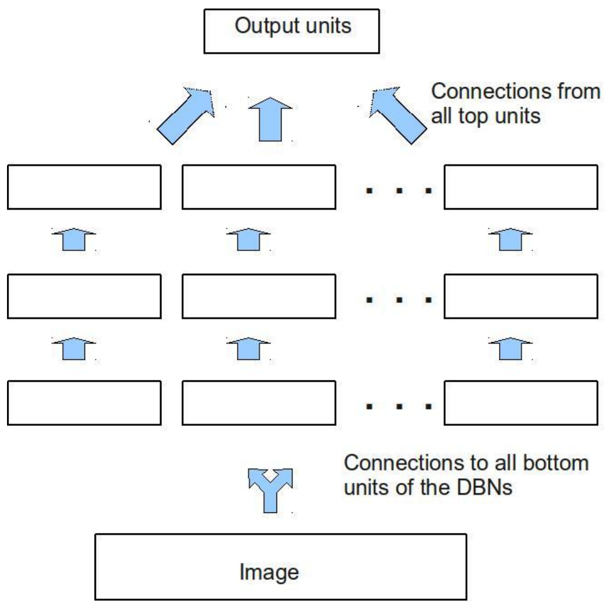
Output units
Connections from all top units
Connections to all bottom units of the DBNs
Image

Figure 10: Composite Deep Belief Network

Convolutionary Neural Network

A network with almost identical architecture to the Le-net 5 presented in fig 9, was adopted to images of size 84* 28 with as many output neurons as sub classes in the experiment. It was trained and evaluated on gray-scale versions of the images in the data set. The network uses two convolutionary layers with following sub sampling steps. On top of this are two layers of fully connected neurons. 20 filters of size 5*5 were used in the first layer and 50 filters of the same size were used in the second. The layers of fully connected neurons consists of one layer of 500 neurons and a final layer of one neuron for each sub class. One of the differences from the Le-net 5 is that the sub sampling is done by using the maximum value instead of the average. More details on the net and how it differs from the original Le-net can be found in [5].

CNN on RGB valued images

Batteries are quite colorful and it seems reasonable that using that information should give a higher performance. By considering each RGB band as a separate feature map (see figure 9), the architecture above could be used with the single adjustment of connecting the first filters to three RGB bands instead of gray scale images.

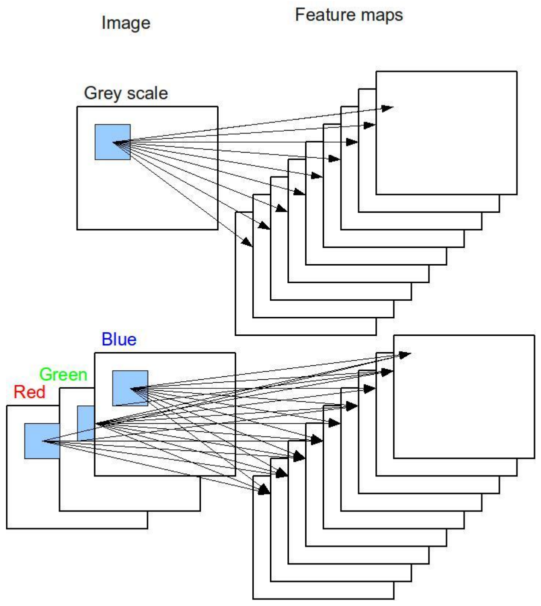

Figure 11: Adjusting the convolutionary network to work with RGB valued images

CNN with down-sampled RGB image as extra input

The setup of this network was inspired by the good performance of a similar network presented in [23]. The constructed network is identical to the examined gray-scale version above, with the difference that an extra hidden layer of 40 neurons is connected to the topmost layer. This extra layer gets its input from the RGB image down-sampled to size 9 *3 pixels.

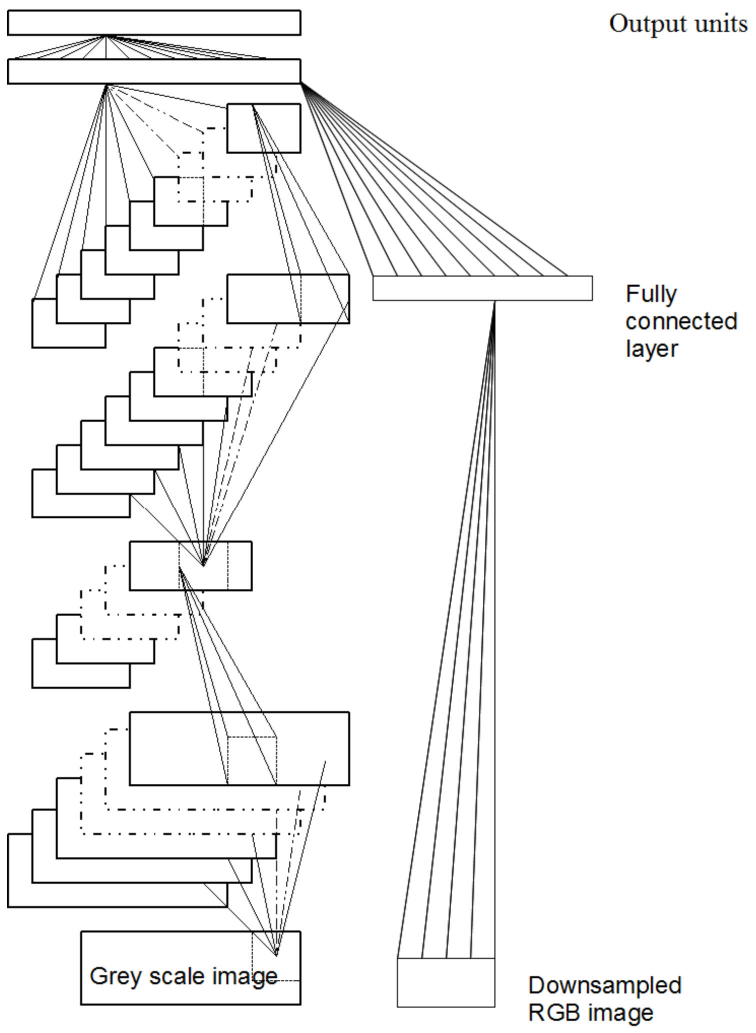

Figure 12: CNN with down-sampled RGB image as extra input

9 Construction of the data sets

When training machine learning algorithms, the goal is to learn from data by exploiting the statistical regularities present in the signal . This said, the trained model should also be able to generalize from this data, so that it can also perform well on data it hasn't been trained on[10].

To do this, images of batteries were divided into one training set and one test set. The images were divided in such a way, that images taken of a special instance only end up in one of the sets. Training is done with one set, and the testing is done with the other. A model which is only performing well on the training set is said to be over fitted [10] and to avoid this, a third validation set is used. By stopping the training when an evaluation on the validation set starts to give worse results, some of this over fitting can be avoided. In some of the tests below, the lack of data made it impractical to divide the data in three parts and it will then be mentioned that the test set also is used as valid set. This is not ideal but must be seen as an acceptable compromise, as the performance on the valid and test sets through all the experiments have been very similar.

The number of images in the training, validation and test sets will in all the experiments always be 10 000. This will be unconditional of the number of different instances the images are taken from. This is accomplished by creating copies of the images in the different sets until the limit is reached.

The images used in this thesis were created by automatically rotating a battery while 100 photos were taken from different angles. While most of the images were provided by Optisort, it was for experiment 10.4 necessarily to go through this procedure to create a specialized data set. To be able to use fewer weights in the first layers (which greatly reduces the time it takes to train the networks), these images were downscaled from the original 318 * 100 pixels to 84 * 28 pixels. As mentioned above, some of the nets used the original RGB images while others used gray scale versions.

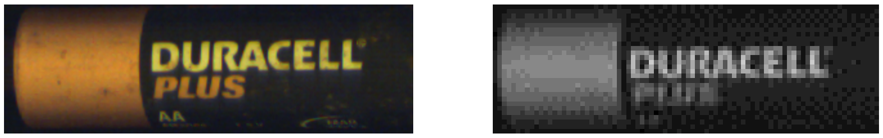

Figure 13: Images of 318 * 100 and 84 * 28 pixels

10 Experiments

In this section the different ANNs are tested for performance and the most promising are chosen for further experiments.

These experiments tests if training on distorted images gives higher performance, and how to best combine the classifications of different images of an instance. As a way of easily handle more sub classes, it is also tested if networks trained on different subsets of the sub classes can recognize which images they were not trained on. Each test, is repeated three times and the best accuracy achieved is presented.With no optimizations made concerning the number of neurons, filters and layers, a network must perform significantly worse than the others to be excluded from further tests.

10.1 Comparing the networks Part I

To compare the different networks described in chapter 8 they were all tested on the same data sets, consisting of images belonging to 10 different sub classes.

10.1.1 Data set

The data set consists of images taken of 10 different battery instances for each sub-class. For each sub-class, the images of two instances were added to the test set, the images of two instances to the valid set, and the images of 6 instances to the training set.

10.1.2 Results

	classification error	training time (minutes)	Classification time (seconds)
DBN	2.89 %	147.04	0.000865
Composite DBN	13.27%	1325.03	not measured
CNN	0.08%	51.92	0.000997
CNN full RGB	0.08%	854.40	0.000995
CNN down sampled RGB	0.18%	120.03	0.000946

Table 2: Error rates for the different nets on a data set of 10 sub classes

10.1.3 Conclusion

All CNNs performed well with error rates close to zero. It was therefore concluded that these methods would have to be compared on a data set of more classes to make a choice on which network-architecture to experiment further with.

10.2 Comparing the networks Part II

The initial experiments did not reveal any clear differences in performance between the different convolutionary networks, and the experiments were extended to test how the networks performed when varying the number of sub classes present in the data set. By testing on data sets of different number of sub classes, it is at the same time tested how the performance is dependent on the number of sub classes.

10.2.1 Data set

Among the images provided by Optisort, there were only a few sub classes that had images from as many as ten scanned instances. To be able to create a data set of enough sub classes, the number of scanned batteries of each class had to be limited to 4 . For each class, one instance was used in the test set and the other three in the training set. Because of the limitations of instances, the test set was also used as validation set.

Three data sets were constructed. One of 10, one of 20 and one of 30 sub classes.

10.2.2 Results

To easier be able to compare the results only the percent wrongly classified is presented. The classification time and training did not change note-worthily from the last experiment.

	10 sub classes	20 sub classes	30 sub classes
DBN **rejected**	-	-	-
Composite DBN **rejected**	-	-	-
CNN	3.19%	10.60%	14.78%
CNN full RGB	4.90%	13.85%	13.84%
CNN down sampled RGB	3.28%	11.60%	14.74%

Table 3: Error rates for the different nets on data sets of 10, 20 and 30 sub classes

10.2.3 Conclusion

None of the versions of convolutionary networks that has been tested performs significantly better than any of the other. When choosing between the networks, the gray scale versions seems preferable since it is the fastest to train and leaves the color information untouched to be used as base for other classifiers. The performance is also getting worse the more sub classes that are included in the data set.

10.3 Examining the relation between accuracy and number of instances in the training set

When comparing the results of 10.1.2 with those of 10.2.2 it seems like more instances in the training set makes the performance go up. To further investigate how the performance is related to the number of instances in the training set, experiments were made with varying numbers of instances on both the data set of 10 and 30 sub classes.

10.3.1 Data set

In the data set of 10 sub classes, the number of instances is varied from 1 to 6.

In the data set of 30 sub classes the number of instances is varied from 1 to 3.

10.3.2 Results

10.3.3 Conclusion

In the results it can be seen that adding a third battery to the training set of 30 classes, actually reduce the accuracy. This is not what would have been expected and doesn't fit with the rest of the results. An explanation might be

number of instance used for training set	1	2	3	4	5	6
CNN 10 sub classes	8.45	6.39	3.32	0.48	0.0	0.08
CNN 30 sub classes	28.63%	12.02%	14.78%	-	-	-

Table 4: Error rates on data sets of 10 and 30 sub classes

that this behavior arose because some of the third instances that were added were heavily damaged or in some other way causing the images to divert from the instance used for the test set.

However, the overall conclusion must be that it is very important to have many instances of each sub class to get good performance.

10.4 Examining the influence of clean instances in the training set

While images from a large number of instances gives better results, it might not be practical to find many instances of the same battery in an industrial setting. It would therefore be better if it was enough with a smaller number of instances. While having several instances in the training set avoids making the network too dependent on the dirt and damages in specific instances, this should also be possible to accomplish by training on a single relatively "clean" instance of each sub-class. A new data set was therefore created to see how well the networks would perform if it was only given images from one, by hand cleaned instance, of each battery-class for the training set.

10.4.1 Data set

The data set included images from one "clean" and one "dirty" instance of 30 sub classes. The clean is used for training and the dirty is used for validation and test set. As there were no such "clean" instances in Optisorts images, the data set had to be constructed from scratch.

10.4.2 Results

	classification error	time to train(minutes)
CNN down sampled RGB	6.20%	8.26
CNN	6.26%	8.24
CNN RGB	5.24%	916.45

Table 5: Error rates on a data set of 30 sub classes were images in training set are from one clean instance for each sub class

10.4.3 Conclusions

Using relatively clean instances for the training set proves to give a significantly higher performance when compared to the results in 10.3.2. This means that care must not only be taken to use enough instances for the training set, but that accuracy can be improved by using relatively clean instances to make sure that the instances are representative for the sub-class as a whole.

10.5 Using simulated dirt

Using one relatively clean instance for each sub-class , avoids some of the over fitting of training the networks to recognize the specific dirt and damages of the specific instances in training set. This said, the network might still become over-fitted in such a way that it only recognizes relatively clean instances. To avoid this, experiments were done to see if introducing noise in the images of clean instances in the training set would give better results. It has been shown [9]that creating deformed copies of images, if done in the right way, can greatly increase the accuracy of the network by enlarging the data set. This experiment should be seen as an attempt to emulate soiling and bleaching of the batteries. With the number of images of a data set, in all experiments being 10 000, the actual number of images is not larger for this experiment than the other. But the size of the data set can still be said to have increased in the way that the number of exact copies in the data set is decreased.

10.5.1 Data set

For this experiment the data set that was described in 10.4.1 was used. The only difference is that each image in the training set was processed by a function that added noise to the image. Besides adding a random amount of speckle noise at random positions in each image, the function also made the image darker or or lighter to a random degree. No effort was spent on trying to optimize the frequency of speckles, how big and distorting they should be, or how much the light in the image should be allowed to change.

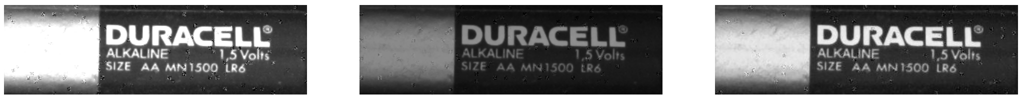

Figure 14: Three noisy versions of the same image

10.5.2 Results

10.5.3 Conclusions

Adding noise to the clean images seems to reduce the number of errors for all of the networks. With no attempt made to optimize the noise to look more like genuine dirt, this should be interpreted as a proof of concept and not as a limit to

	classification error
CNN 30 classes	5.17%
CNN down sampled 30 classes	5.50%
CNN RGB	5.10%

Table 6: Error rates on a the 10.4.1 data set were the training images were distorted by noise

how much the results can be improved. With perfectly clean instances together with a better function for generating artificial noise, it should be possible to achieve much better results.

10.6 Discriminating between more sub classes

In 1.1 it is stated that the network not only should be able to categorize the batteries with an error rate below 5 percent, but that it also must be able to do this while handling up to 2400 sub classes.

With no way of acquiring a data set of this many sub classes, experiments were done to with data sets of fewer sub classes still get an idea of the performance on such a data set. Experiments are here done to see how the error rate follows from the number of sub classes that the net should be able to handle.

10.6.1 Data set

The earlier experiments has shown the importance of the number of instances represented in the training set. When using enough instances the error rates approached zero. The problem in testing how the error rates depend on the number of sub classes to discriminate between, is that it would take to much time to create a data set of that many sub classes, without decreasing the number of instances of each sub class used for the training set. To avoid this problem an artificial data set was created. Instead of using several instances for each sub class only one instance was used, and the difference between images taken of different instances, was instead simulated by adding the same kind of artificial noise that was used in 10.5.

This means that the images in the test set and training set originally looked exactly the same, but the different kind of noise added to the images makes them different in much the same way as images of different instances of the same subclass look different. With different noise added to every image a “perfect” data set of size 10 000 is simulated, were each image seems to be taken from a different instance.

number of sub classes	10	50	100
CNN	23.01%	4.07 %	0.08 %

Table 7: Error rates on the test set with artificial data sets of 10 ,50 and 100 sub classes

10.6.2 Results

10.6.3 Conclusion

The network delivers impressive results when trained and evaluated on 100 sub classes, but gets far worse results on 50 and 10 sub classes. As the data sets are artificial it is hard to do any reliable conclusions from this, but the fact that the networks manges to classify an artificial data set of this many sub classes this well is still encouraging.

It might at first seem contradictory to the earlier conclusions that the result gets worse for fewer classes, but an inspection of the intermediate results from the training gives an explanation. From the table 8, which shows the classification error on the valid set for each training epoch, it is clear that the behavior is caused by early over fitting. Over fitting can be associated with to much modeling capacity [10] which causes the network to model the noise and therefore loses its capability to generalize. Considering that a net needs less modeling capacity, the fewer categories it needs to discriminate between, the results seems more reasonable. It is harder to explain why over fitting happens much earlier here, than when discrimination between the same number of sub classes in 10.1.2. One explanation could be that the images in the artificial data set, even after the noise has been added, are to similar to each other. The training of the network then doesn't spend so much time trying to find the similarities between the images of the same sub class, and therefore starts to modeling the noise earlier.

From the above reasoning it follows that it might become important to adjust the number of layers and filters depending on how many sub classes the net should be able to handle. The function for introducing noise should also be altered to get more trustworthy results on further tests.

Results on artificial data sets of 10 sub classes						
epoch	1	2	3	4	5	6
validation error	80.37%	58.06%	54.02%	22.92%	89.11%	90.10%

Table 8: Error rates of the network during training shows that the network is becoming over fitted

10.7 Experiments with classifying "unknown" instances

While an ideal network would have a 0% error rate, this might in practice be impossible. The instance in the image might be heavily corroded or in other ways so damaged that its impossible to see what sub-class it belongs to. The battery might also be to new, or for some other reason not been added to the training set.

To handle this it will be explored how well a net that is only trained on a subset of the sub classes in a data set, manages to classify the ones it was trained on while acknowledge the ones it doesn't recognize as "unclassifiable".

If it turns out that the net manages to classifies the absolute majority the unknown instances as "unclassifiable" it might be a good idea to create a composite net similar to the DBN model explained in 8. The sub classes could then be divided into different partitions with one net trained on each partition which would eliminate the need to test the net on larger number of sub classes to get an idea of what kind of error rates that would be possible to achieve on the wanted 2400 sub classes. If the net on the other hand doesn't manage to do this it would be a better solution to use the same net for all classes.

An image is considered as "unclassifiable", if the neuron with the second highest activation in the output layer, has an activation that is higher than a factor times the activation of the neuron with the highest activation in the same layer. Setting the factor to 1 means that no images will be classified as unknown while a smaller factor will cause more unknowns. An alternative would be to set a threshold value that the highest activation must be over, but based on the results of [26] the first method was judged as superior, and therefore used in the experiments.

10.7.1 Data sets

The data sets used for this experiment are the same sets of 50 sub classes as was used in 10.6. The only difference is that for this experiment, images from another 50 sub classes were added to the test set. These images were all labeled so as to belong to the same sub-class. As this label was not included in the training set, all images of this extra sub-class would be wrongly classified if they were not considered as "unknown". This means that the best classification that can be done on the 50 sub classes data set would assign around 50% of the images as "unclassifiable" while having an error rate of 0%.

10.7.2 Results

factor	net trained on 50 sub classes	
	unclassifiable	classification error
0.3	72.31%	34.67%
0.5	54.91%	44.97%
0.7	35.20%	52.69%
0.9	12.66%	59.68%
1.0	0%	63.14%

Table 9: Error rates as a function of reliability of the classification

10.7.3 Conclusions

While a lower factor helps to give lower error rates, the percent unclassifiable gets much higher than the percentage of the images that comes from "unknown" images. That this happens while the error rate still is far from the low values in 10.6 , shows that a lot of the images that would otherwise have been correctly classified are now being judged to be "unclassifiable".

The conclusion must bee that it is better to use one single net that is trained on all sub classes than to divide the sub classes as suggested.

10.8 Using images from different angles for classification

The time for a net to process one image was in 10.1.2 measured to be around one millisecond. With sorting frequency not expected to be more than 5 instances per second in a sorting facility line[14], this leaves plenty of time that can be used for increasing the accuracy. One way to do this would be to train several nets and combine their classifications of the same image, for instance by letting them vote. While this probably would produce a more reliable classification compered with using a single net, a method that is more specialized for battery sorting will here be examined.

An image of an instance may look very different depending on which side of the battery that is shown. Using several cameras to acquire images from different angles of the battery should therefore be a very promising way of increasing the accuracy. In this experiment this is tested by repeating experiment 10.4, while this time making the classification based on a series of images of the same instance. To see how the networks top layer neuron activities should best be combined with those for another image, two methods are evaluated. The methods that are evaluated are to either compute the class-prediction as the sum of the activities generated by the different images, or to compute it as the product of these activities. This can be compared to what Hinton[15] calls "mixture of experts" and "product of experts" .

To get more information on the kind of mistakes the net makes, there was for this experiment also created a confusion matrix.

10.8.1 Data set

The experiment was performed on the data set of 30 sub classes used in 10.4. To easier make the classification based on several images, the test set had to be changed so that random images, but from the same instance, was tested after each other.

10.8.2 Results

number of images used	1	2	3	4	5
with the top layer activities multiplied	5.43%	3.43%	2.53%	2.32%	1.96%
with the top layer activities added	5.43%	3.54%	3.35%	3.05%	2.80%

Table 10: Error rates when using using several images for classification

sub class	1	2	3	7	9	10	11	19	20	23	25	29	30
1						16						46	
2									10				
3	4			1	4			3			1		14
9						4	2						
12												17	
20		4						22					
21				3						3			
30			2		1								

Table 11: The number of false matches for images of different sub classes when basing the classification on three images. (that a sub class not is present means that it always was classified correctly)

10.8.3 Conclusions

Using more images for the classification manages to reduce the error rate substantially and it is the multiplicative method that works best. To have in mind is that the images are from a random sides of the battery and therefore may overlap so that less than an optimal part of the batteries label is covered. By positioning the cameras so that they cover as large part of the label as possible the result should improve.

Inspecting table 11 gives that 66 out of the total 154 misclassifications involves the same image nr 1, and by inspecting the images in training and test set it becomes obvious that there has been made a mistake when this data set was created. The instance that was used to create the training set is of a rather different version than the one that was used to create the test set. A proper training set must include images of every version of a sub class and the result of this not being the case is higher error rates.

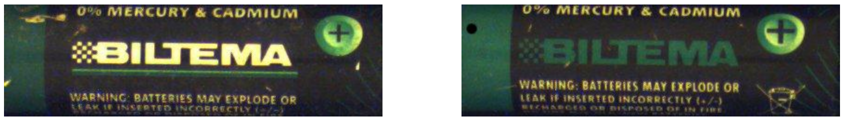
0% MERCURY & CADMIUM
BILTEMA
WARNING: BATTERIES MAY EXPLODE OR
0% MERCURY & CADMIUM
BILTEMA
WARNING: BATTERIES MAY EXPLODE OR
LEAK IF INSERTED INCORRECTLY (+/-)

Figure 15: Images showing the accidental difference between two instances used for the test and the training set

11 Conclusions

This section presents a summarized version of the conclusions that has been drawn from the tests.

11.1 The best network to use

The convolutionary networks seems to be the preferable choice, but no tested version performs significantly much better than the others. The versions that uses color information did not give significantly better results than the simple gray scale version.

11.2 How to create a good training set

Three conclusions concerning the training set was made in this thesis.

1. Using images from as many instances as possible is very important to avoid over fitting and achieve good accuracy.

2. Training the nets on images of relatively clean instances improves the accuracy further.

4. Adding virtual dirt to images of clean instances, allows for bigger training sets without using as many instances and is another way of improving accuracy.

11.3 Techniques that helps to give better accuracy

Experiment 10.7 showed that it does not work to train a CNN on a sub set of the sub classes, and trust that the network will assign low probabilities to all sub classes, when classifying an instance of a sub class it has not trained on. This means that the accuracy not can be increased by specializing certain networks on smaller sub sets of the sub classes.

The accuracy was in experiment 10.8 however greatly increased by basing the classification on several images of the instance to classify. Classifications based on five images taken of **random** sides of the battery reduced the error rate to almost a third (5,43/1,96 = 2,77).

11.4 How the accuracy depends on the number of sub classes

In 10.3 a CNN delivered error rates of 0.0% with training sets using images from 5 different sub classes. This did however not say much about how the network would perform when discriminating between more sub classes. Experiment 10.2 showed that the error rate goes up with the number of sub classes that is included in the data set. It became clear that sorting between the 2400 kinds of batteries, that is estimated to be sorted in a real battery sorting facility, is a demanding task.

Because of problems in acquiring large data sets, no reliable results on test sets of more sub classes than 30 was ever produced. For that data set however, error rates were by basing the classification on several images, achieved that are way below the 5% limit(see experiment 10.8). This was achieved even though the training set was made up of images from a single instance for each sub class, and the conclusions from 10.3 made it clear that data sets must be constructed by images from **many** instances to give good results.

The error rate of 0.08% that was achieved on an artificial data set of 100 sub classes, can because of the limitations in the way this data set was produced(see experiment 10.6) not be seen as totally reliable. It should however bee seen as a hint that it is possible to achieve very good accuracy, also when discriminating between many more sub classes than 30. Just as long as the images in the training set are representable for the test set.

11.5 Performance in a battery sorting facility

It has in this thesis been shown that CNNs that base a classification on several images of a battery, can achieve a very high accuracy when discriminating between up to 30 sub classes.

On the other hand it is still very unclear if it is possible to keep that kind of accuracy while sorting between thousands of sub classes. The final conclusion must be that bigger data sets must be constructed to get a stronger indication on how well a CNN would perform in a real sorting facility.

12 Further work

12.1 Create larger data sets

The tests conducted in this thesis doesn't give any clear indications on performance in a real setting. This is mostly because the data sets were to small to draw any conclusions about this. Creating data sets of thousands of sub classes will be extremely time consuming, but might in the end prove to be necessarily.

12.2 Combining the opinions of multiple networks

It was briefly mentioned in chapter 10.8 that training several nets on the same data set should be able to increase the accuracy if the networks opinions are averaged. This claim is repeated in [12] and is definitely something which should be investigated. While the networks that used color information didn't get lower error rates than the gray scale versions, it should be possible to use the color information by processing the color information in a separate net. By training a net on only the color histograms of the images, this network should be able to exclude all sub classes that doesn't resemble the image color wise, while the gray scale nets takes care of the 2-dim structure of the image.

12.3 Create more realistic distortions

Introducing artificial noise in the training set, in chapter 10.5 made it possible to reduce the error rates. But the improvement was not as substantial as the improvements that has been achieved with similar methods on other image recognition tasks [12].

Since both networks and recognition tasks are similar to the ones were the method has proved effective, the limited improvements should depend on the somewhat naive noise function. Refining the noise function to create more realistic images should therefore be a very promising way to achieve better results.

12.4 Use more modern versions of ANNs

Recent studies have with good results combined the structure of Convolutionary Neural Networks with the training methods of Deep Belief Networks[13]. The author has not been able to find any public available code for these nets, and they were therefore never investigated in this project. When code becomes more available, changing to these more modern nets should give lower error rates.

Another interesting architecture that should be kept under watch is the "Factored 3-Way Restricted Boltzmann Machines" which in [22] presents the so far best performance on the CIFAR-10 data set.

12.5 Optimizations

Besides optimizing the number of layers, filters, and filter sizes, there is also room for optimizations of the activation function used in the networks. Exchanging the usual tanh(X) with a more complex abs(g. tanh(X)) function, were g is a trainable parameter, has proved to substantially improve the networks performance [31].

12.6 Validate the classification

The experiments in 10.7 showed that the CNN wasn't very useful for judging when an image belong to a sub class it wasn't trained on.

Optisort has developed so called template matching methods where images are compared pixel by pixel with the battery's label. These methods have given good results in tests but shown to be to slow to be used on data sets of many sub classes. The method should however be a good way of validate the classification of a network and thereby solve the problem that the method examined in this thesis did not. Another way this method could be used, is by working as a post-classificator that decides which of the sub classes the networks picked as most likely, that should be the final choice.

References

[1] Yann lecun. URL `http://yann.lecun.com/`.

[2] http://www.batteriinsamlingen.se/.

[3] Geoffrey e hinton. URL `http://www.cs.toronto.edu/~hinton/`.

[4] theano. URL `http://deeplearning.net/software/theano/index.html`.

[5] a simplified version of lenet5, January 2011. URL `http://deeplearning.net/tutorial/lenet.html#lenet`.

[6] Yoshua Bengio. *Learning Deep Architectures for AI*. 2009.

[7] Yoshua Bengio. Yoshua bengio's teachings, 2010. URL `http://www.iro.umontreal.ca/~bengioy/yoshua_en/teaching.html`.

[8] Yoshua Bengio. Learning deep architectures for ai, December 2010. URL `http://deeplearning.net/tutorial/`.

[9] Luca Maria Gambardella Jurgen Schmidhuber Dan Claudiu Cires, Ueli Meier. Deep big simple neural nets excel on handwritten digit recognition. *http://arxiv.org/*, 2010.

[10] Guillaume Desjardins. Training deep convolutional architectures for vision. 2009.

[11] Felleman & Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47, 1991.

[12] Yee-Whye Teh Geoffrey E. Hinton Simon Osindero. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[13] R. Ranganath H. Lee, R. Grosse and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. 2009.

[14] CEO Optisort Hans eric Melin. Interview for the master thesis. November 2010.

[15] Geoffery E Hinton. Product of experts, . URL `HTTP://WWW.scholarpedia.org/article/Product_of_experts`.

[16] Geoffery E Hinton. Video lecture, . URL `http://carbon.videolectures.net/2009/singles/07/jul09_hinton_deeplearn/jul09_hinton_deeplearn.pdf`.

[17] Geoffrey Hinton. *A Practical Guide to Training Restricted Boltzmann Machine*. Department of Computer Science, University of Toronto, version 1 edition, August 2010.

[18] Geoffrey E. Hinton. Deep belief networks. *Scholarpedia, 4(5):5947.*, (Geoffrey E. Hinton (2009), Scholarpedia, 4(5):5947.), .

[19] Geoffrey E. Hinton. Boltzman machine. *Scholarpedia, 2(5):1668*, 2007.

[20] Geoffrey E Hinton. Neural information processing systems conference. 2007.

[21] Geoffrey E. Hinton. Learning to represent visual input. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 2009.

[22] Marc Aurelio Ranzato Alex Krizhevsky Geoffrey E Hinton. Factored 3- way restricted boltzmann machines for modeling natural images. *International Conference on Artificial Intelligence and statistic*, 2010.

[23] Alex Krizhevsky. Convolutional deep belief networks on cifar-10. 2010.

[24] Bottou L. Bengio Y. & Haffner LeCun, Y. Gradient-based learning applied to document recognition. 1998.

[25] Yann LeCun. The mnist database of handwritten digits, January 2010. URL `http://yann.lecun.com/exdb/mnist/%2010`.

[26] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[27] Haykin Simon. *Neural networks a comprehensive foundation.*

[28] Peter Norvig Stuart Russell. *Artificial Inteligence A modern Approach.* Russell Norvig, second edition edition, 2003.

[29] Tijmen Tieleman. *Gnumpy: an easy way to use GPUboards in Python.* Department of Computer Science, University of Toronto, July 2010.

[30] Paul J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University,, 1974.

[31] Koray Kavukcuoglu Yann LeCun and Clement Farabet. Convolutional networks and applications in vision. *Circuits and Systems (ISCAS), Proceedings of IEEE International Symposium on*, 2010.