Image classification with convolutional neural networks

Alzahraa Salman
alzahraa@kth.se

Ludvig Kindberg
lki@kth.se

Rouwayd Hanna
rouwayd@kth.se

Abstract

Convolutional neural networks (CNN) are powerful visual models that use filters to find patterns or features. CNNs are considered to be the architecture of choice when it comes to image classification. In this project, a convolutional neural network, with a similar architecture to AlexNet, was implemented to perform image classification using the CIFAR-10 dataset. The open-source neural-network Python library Keras was used to build the network. Different techniques such as dropout, data agumentation and batch normalization were applied to the network to study the effect on its performance. Our best model achieved an accuracy of around 83% with an AlexNet insipred structure of the network.

Repository with the code `https://github.com/luddz/Project_Deepen_DD2424`

1 Introduction

For us humans, classification between objects is a trivial task but this same task has proven to be quite a complex problem for machines. One of the classic tasks in computer vision is image classification. Image classification refers to the task of labelling of images into one of several predefined categories [1]. Image classification is important in several different areas such as face recognition, robotics and object detection. Neural networks are a powerful technology for image classification since they are able to recognize patterns. Generally, artificial neural networks are computing systems, inspired by the human brain, that are able to process complex data inputs. By feeding the network inputs together with the expected output, the network is able to learn by attempting to minimize the computed error. Convolutional Neural Networks are very similar to ordinary Neural Networks with the difference that CNN architectures make the explicit assumption that the inputs are images. This allows encoding certain properties into the architecture which can make the forward function more efficient to implement [2]. This also allows the different layers to learn specific features of the classes, from a more generic filtering at the first layers and to more fine tune and very specific filters for the final layers.

In this paper, a convolutional neural network is implemented to perform image classification using the CIFAR-10 dataset. The architecture of the network is similar to the one of AlexNet [3]. Our model, with similar architecture to AlexNet, was compared to a simple ConvNet model provided by Keras [4]. We ran both networks for 10 epochs and our network achieved far greater results. Moreover, we studied the performance of our network when using different regularization techniques such as batch normalization, dropout and data augmentation. Both dropout and data augmentation addressed the issue of over-fitting so it was interesting to combine them and study the outcome. It turned out that dropout combined with data augmentation did improve the results and decreased over-fitting which made us use these techniques when training our best model.

2 Related Work

For a long time the structure of convolutional neural networks has been the same: stacked convolutional layers (optionally followed by max-pooling) that are followed by one or more fully-connected layers [5]. There have been different variants of this basic design in the image classification literature and research. These models have yielded some great results and even some of the best results on MNIST, CIFAR and especially on the ImageNet dataset classification. The results of Alex Krizhevsky, Geoffrey Hinton and Ilya Sutskever work were a breakthrough for the computer vision community [3]. They created a neural network architecture, called 'AlexNet', which was trained to classify 1.2 million high-resolution images into 1000 different classes. The network consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers. Their network won the ImageNet LSVRC contest in 2012. The results of their work shows that a large deep convolutional neural network can achieve record breaking results even on highly challenging datasets. In their work, the authors describe how the use of methods such as data augmentation and dropout helped in reducing over-fitting and improving the performance of the network.

Two years later, a new design of a deep convolutional neural network architecture was proposed in the ImageNet LSVRC 2014 [6]. This architecture achieved the new state-of-the-art results for classification and detection. What differs this architecture from previous models is the improved utilization of the computing resources inside the network. Both depth, number of network levels, and width, number of units at each level, of the network were increased. This work shows that one of the most straightforward and effective ways of improving the performance of deep neural networks is by increasing their size. Moreover, this architecture used 12 times fewer parameters than the architecture of Krizhevsky et al [3] while being significantly more accurate.

3 Data

The data used in the project is the CIFAR-10 dataset, collected by A. Krizhevsky and G. Hinton [7]. The CIFAR-10 is an established computer-vision dataset used for object recognition. The dataset consists of 60,000 (32×32) RGB colored images in 10 classes, with 6000 images per class. It is divided into 50,000 training images and 10,000 test images.

A simple preprocessing of the data was made. We converted our image data to float format and divided it by 255 to convert the data to be between 0 and 1. No further preprocessing or other special treatments of the data were made.

Many research papers claim to have achieved state-of-the-art results on the CIFAR-10 dataset. In the paper *Convolutional Deep Belief Networks on CIFAR-10* [8] (2010), Alex Krizhevsky trained a two-layer convolutional Deep Belief Network. Their best model, with 64 9x9 filters, classifies the CIFAR-10 test set with 78.9% accuracy (21.1% error rate). In 2016, in the paper *Densely Connected Convolutional Networks* [9], Dense Convolutional Network (DenseNet) was introduced. DenseNet connects each layer to every other layer in a feed-forward fashion. The results of this work shows that DenseNet outperforms the then current state-of-the-art consistently on all the CIFAR datasets. An error-rate of 5.19% for CIFAR-10 was achieved. In 2017 a research paper, *Improved regularization of convolutional neural networks with cutout*, exploring how regularization techniques can improve the performance of convolutional neural networks was published [10]. The results of this work shows that a simple regularization technique of randomly masking out square regions of input during training, cutout, can improve the overall performance of a convolutional neural network. This new architecture achieved state-of-the-art performance on the CIFAR-10 dataset with 2.56% error rate. Also, one of the hot topics for research and upping the accuracy for the CIFAR dataset seems to be data augmentation. In 2018, Ekin D. Cubuk et al described a procedure called AutoAugment to automatically search for improved data augmentation policies [10]. Their search algorithm found the best data augmentation policy so that the neural network yields the highest validation accuracy on a target dataset. They achieved an error rate of 1.5% on CIFAR-10.

4 Methods

Our main approach to tackle the image classification task is to use convolutional neural networks. This is because of the continous successful results they have achieved in this field [3, 6, 10]. The fact that convolutional neural networks use filters and are designed to recognize patterns makes them a great candidate for solving image classification tasks.

For implementing the network we chose to use some well-known open-source neural-network libraries. This is due to the time and experience limitations and also since we got approval to use high-level libraries from the feedback of our project proposal. After some research we opted to use Tensorflow which is a well-known library for dataflow and neural networks. Tensorflow was developed and used by Google and is Open source. Also, we chose to use Keras, a high-level neural networks API written in Python and capable of running on top of TensorFlow [4], since it made it easier to build the network architecture. Keras is user-friendly and it is quite easy to learn how to use it for those with less experience. Having worked on the assignments of the course made it easier to understand the concepts and interface of these high-level APIs.

We compared our network to a standard simple convolutional neural network provided by Keras. We wanted to study the benefits of an AlexNet inspired CNN over a simpler one. Moreover, during the labs in the course, we experienced that Batch normalization increases the ability to learn deeper networks more reliably. Therefore we chose to apply batch normalization to our network to see how it effects the performance of it. Also since over-fitting is a common problem during training, we decided to explore the dropout and data augmentation techniques to reduce over-fitting. We compared the accuracies of networks with and without the use of these methods, closely analyzing the gap between training accuracy and validation accuracy.

Due to time constraints we have not had the ability to search for every optimal hyper-parameter variable combination such as the percentage of dropout to use between the different layers. Hence the number used, 40%, was the result of some searching on the internet of different values used which is slightly lower than the 50% used in the original AlexNet.

Furthermore the number of epochs that were used to compare the different parameters towards each other was 10. This number of epochs proved to be sufficient for yielding comparable results without taking too long to train. The time it took to train each model, for 10 epochs, was around 20 minutes.

We also noticed the importance of the computer that was preforming the training and testing. With our own laptops the results obtained were strange and not as expected, while running everything on computers in the computer halls at KTH gave significant improvements in both speed and accuracy as well as aligning with the expected results.

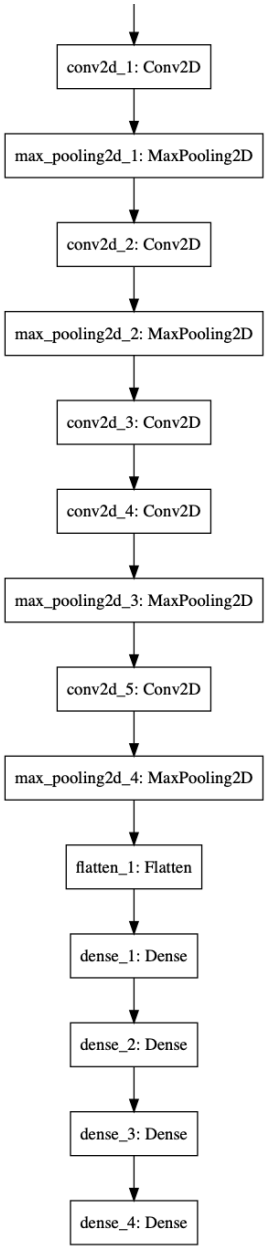

Figure 1: Overview of the architecture

4.1 Architecture

Our network architecture is inspired by AlexNet, a challenge winning convolutional neural network and also one of the proposed architectures for the project idea. The extent to the similarity of our network architecture and the original AlexNet architecture is that both have five convolutional layers, all of which except the third convolutional layer are followed by maxpooling and three fully connected layers. Also, our network attach ReLu activation after every convolutional layer and fully connected layer.

Since the original AlexNet is run with high-definition images (224 x 224) and our dataset contains images of size (32x32), some adjustments to parameters had to be made. For example the filter sizes and pooling windows were changed to better fit the lower input size.

The architecture for the CNN provided by Keras is simpler. The network consists of four convolutional layers and only one fully connected layer. It has two maxpooling layers, one after the second and one after the fourth convolutional layer. The full source code for this network can be found in Keras Documentation webpage [4].

5 Experiments

5.1 Standard Keras CNN vs AlexNet

We compared the performance of a simple CNN provided by Keras and our CNN with similar architecture to AlexNet. We trained both networks for 10 epochs and plotted the accuracy functions on the training and validation data. The final test accuracy for Keras CNN is 66.84% while our network achieved 73.62%. The AlexNet inspired network is deeper and more complex than the one that Keras has. AlexNet has showed that depth and more filters per convolutional layer has a positive impact on the overall performance of a network.

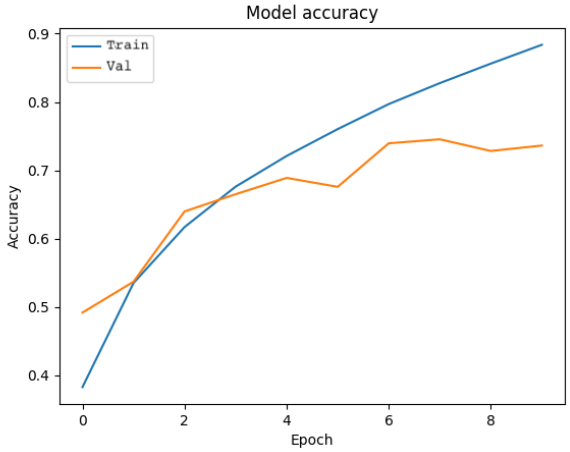

Figure 2: Accuracy function of train and validation data over 10 epochs for our network

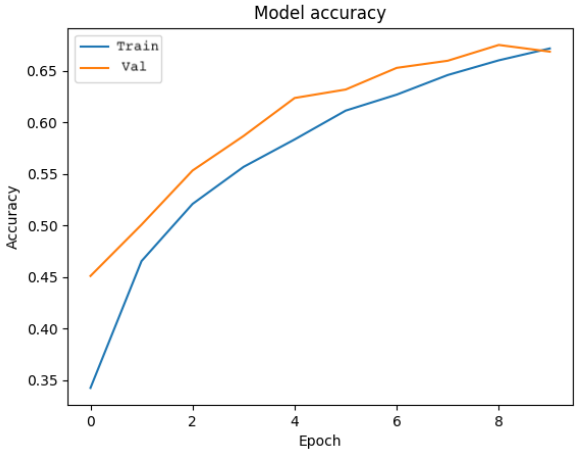

Figure 3: Accuracy function of train and validation data over 10 epochs for Keras provided network

The image classification problem has been explored earlier within the assignments of this course but only with fully connected networks. The results above show that convolutional neural networks preforms much better than only fully connected networks. The best test accuracy achieved by fully connected networks, in the assignments, was around 53.5% which was obtained when using techniques including batch normalization. Our results show that CNNs can achieve quite higher accuracies even without the techniques that made fully connected networks perform their best.

5.2 Improving our model

In this section we study and analyze different techniques to improve the performance of our AlexNet inspired network.

Neurons in a fully connected layer develop co-dependencies amongst each other during the training which curbs the individual power of each neuron leading to over-fitting of training data [3]. To reduce over-fitting in the fully-connected layers we employed the regularization method *dropout* [11] that proved to be quite effective. This technique involves setting the output of each hidden neuron to zero with some probability. These neurons which are "dropped out" do not contribute to the forward pass, neither the back-propagation.

We use dropout in all of the fully-connected layers of the network. Without dropout, our network suffers from over-fitting. Comparing Figure 2, where our network were run without dropout, with the figure below, Figure 4, shows that the gap between the training and validation accuracies is smaller when using dropout. Also, the test accuracy for the network when using dropout increased to 74.2% from previous 73.62%.

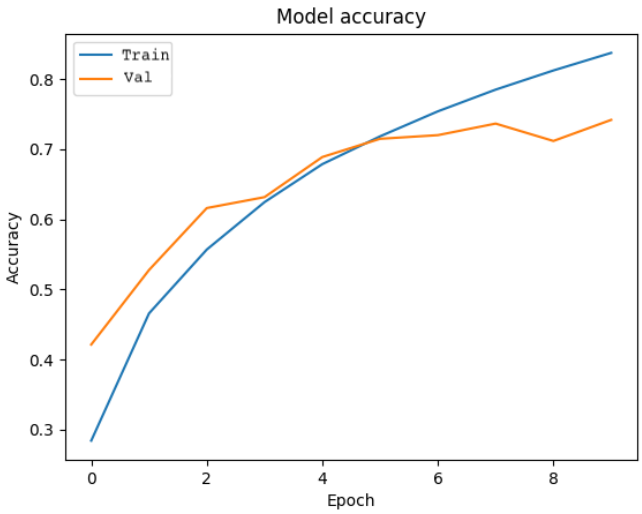

Figure 4: Accuracy function of the train and validation data over 10 epochs using dropout (40%)

Another well-known technique when dealing with image classification is *batch normalization*. Batch normalization is used for improving the performance, speed and stability of artificial neural networks [12]. We applied batch normalization to our network and ran it for 50 epochs with dropout of 40%. A test accuracy of 75.38% was achieved. However, it is obvious, looking at Figure 5, that this training still suffered from over-fitting, as the validation accuracy stopped increasing after about 15 epochs while the training accuracy maintained a stable increase throughout the whole training. As the training time increased (50 epochs) the network became more prone to over-fitting. Dropout alone was not enough for preventing over-fitting.

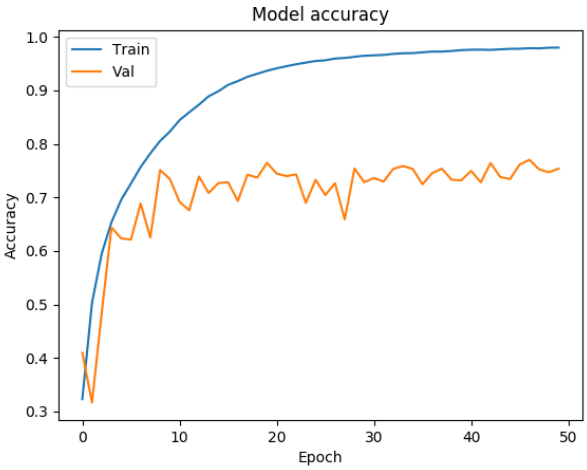

Figure 5: Accuracy function of the train and validation data over 50 epochs for the best model

Krizhevsky [3] pointed out that *data augmentation* is the easiest and most common method to reduce over-fitting on image data. Therefore we decided to employ data augmentation to our network to investigate whether it decreases over-fitting or not. Data augmentation resulted in significant improvements for our network, both in terms of accuracy and avoiding over-fitting. The form of the data augmentation consisted of randomly shifting the images vertically and horizontally and randomly flipping the images horizontally. The test accuracy achieved when combining dropout, data augmentation and batch normalization is 82.97% which is quite higher than the results obtained without data augmentation. Also, as can be seen in Figure 6 and 7 over-fitting has been decreased. The gap between the training and validation data is significantly smaller. This model with all the three techniques combined became our best model.

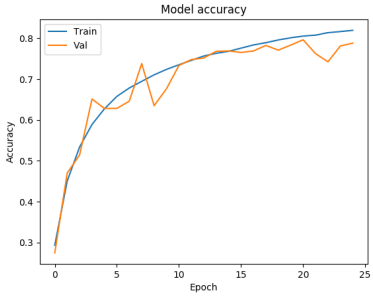

Figure 6: Accuracy function of train and validation data over epoch 1-25 for our best model

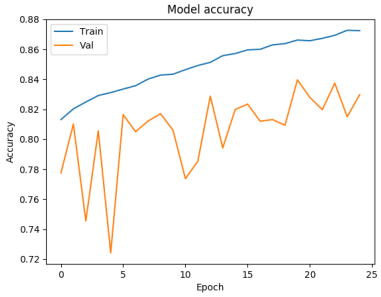

Figure 7: Accuracy function of train and validation data over epoch 26-50 for our best model. Notice smaller y-axis

6 Conclusion

This paper gives evidence that convolutional neural networks are a powerful tool for solving image classification tasks. A CNN was implemented to perform image classification on the CIFAR-10 dataset. The network had a similar architecture to the well-known AlexNet. A comparison of our netwrok and a simple CNN, provided by Keras, was made where our network yielded better results. When training both networks for 10 epochs, the test accuracy of our network was 73.62% in comparison to 66.84% for the Keras CNN. Furthermore, dropout and data augmentation were employed and showed a positive effect in reducing over-fitting, as well as increasing the test accuracy. Our best model, which combined dropout, data augmentation and batch normalization, achieved a test accuracy of 82.97%, when trained for 50 epochs. Future work could involve exploring different types of architectures and/or optimizers and study their affect on the performance of the network.

References

[1] Pooja Kamavisdar, Sonam Saluja, and Sonu Agrawal. A survey on image classification approaches and techniques. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(1):1005–1009, 2013.

[2] Andrej Karpathy. Cs231n convolutional neural networks for visual recognition.

[3] 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.

[4] François Chollet et al. Keras. `https://keras.io`, 2015.

[5] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[7] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[8] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7), 2010.

[9] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[10] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[11] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

A Skills acquired

A.1 Alzahraa Salman

- ConvNets
  I gained a deeper understanding of how CNNs work and why they are useful for image classification by studying the related literature of this topic. Also, I learned how different techniques can be applied to a network to improve its performance. All this made it possible to understand how to implement and work with a convolutional neural network.
- AlexNet
  CNNs can have different architectures. We chose to implement a CNN inspired by the well-known AlexNet. The research paper *Imagenet classification with deep convolutional neural networks* described how this architecture was built for the ImageNet dataset. Learning about this architecture, made it possible to adapt it to the CIFAR-10 dataset.
- Keras
  In this project we used Keras which is a Python API for neural networks. By checking their documentation page and source code examples, I learned how to implement our CNN with Keras.

A.2 Ludvig Kindberg

- The use of high-level software packages and APIs for implementing CNNs.
  I learned this through the project due to that was the code we wrote. The learning process was similar when using any new libraries or programming language, check the documentation and implementing and testing different architectures.
- Different Architectures in a CNN:
  During this project I gain insight in some well known Architectures, such as AlexNet, but also during the research for the state of the art architectures gain the insight in how the different layers work together and that a deeper network in a necessity to reach the highest accuracy.
- The importance of addressing the topics of over-fitting:
  This became very obvious when we trained the network for a longer period of time, compared to the assignments and just a few epochs. So what I learned was that just one approach was not enough, that to really solve the problem all three of them needed to be combined when training for a longer time.

A.3 Rouwayd Hanna

- Using high-level packages such as Tensorflow/Keras. This skill was acquired through internet tutorials and looking at example code. Also, the concepts of the APIs were easy to understand because of previous experience with nerual networks from previous assignments. The evidence lies in the code we have written in Python for building and training the network.
- Better understanding of CNN, what differs CNN from regular ANN, and why CNN is so powerful for image classification. This skill was acquired by studying research papers involving CNNs and their results on image classification. Also, I learned some of the current state-of-the art neural networks for image classification and their architectures, such as AlexNet. See Related Work section.
- Popular techniques for decreasing overfitting and improving performance of a CNN for classifying images. This skill was acquired through running experiments and analyzing results. Proof lies in the Experiments section.