GIST Feature Extraction From Shallow Convolutional Neural Networks

Reducing complex Convolutional Neural Networks while maintaining scene feature generation

Johannes Patzschke
Otto-von-Guericke-Universität Magdeburg
Mat.-Nr.: 191564
Email: johannes.patzschke@st.ovgu.de

Tim Gerrits
Otto-von-Guericke-Universität Magdeburg
Mat.-Nr.: 191920
Email: tim.gerrits@st.ovgu.de

***Abstract*—Despite its complexity, Convolutional Neural Networks (CNNs) have grown to become a leading technology in object recognition for research and industry purposes. This is achieved by using huge datasets like Imagenet and long training periods of network models with several hidden layers to generate high level features for classification. However, these so called deep networks were found to be inefficient for scene classification so other databases like the Places database were introduced to train networks for this purpose. The high complexity of common CNN topologies as well as the huge size of training sets for generating features renders CNNs to be impractical. In this paper, we present our work on generating two-class scene classification features with fast and simple CNNs. To do so, we implemented different network models with varying but reduced complexity, trained on the same dataset and compared the results in terms of accuracy and time used while increasing the complexity gradually. We were able to show, that a shallow network can compete with state-of-the-art deep networks or hand-generated features. This was done in the context of a scientific team project supervised by Prof. Tönnies at the Otto-von-Guericke University of Magdeburg.**

I. Introduction

Fast and easy scene recognition as done by the human visual system is a widely discussed field in Computer Vision and has produced a huge variety of techniques and approaches. Having a first, very coarse classification of an image into scene categories like *indoor*, *beach* or *forest*, also known as getting the GIST [18] of the scene, might be used for further refinement of recognition or classification tasks or simply sorting the data.

A standard classification task in general, either a very detailed description of a part of the image or a coarse scene classification, follows a simple pipeline, where image features are extracted, described by feature descriptors, which are then used in a feature vector. This description of an image is then used by an arbitrary classification method that results in a predicted label for that image, depending on the training set.
A simple, fast and reliable way to do so, is to decide on features, that are well suited to the given classification problem. This, however, requires an understanding of the image and which features are actually capturing the meaning of the given image best. However, it might seem a better idea to mimic the human brain and its structure to solve this problem, which inspired the work on Convolutional Neural Networks (CNNs). Deep CNNs trained on large databases have proven to yield extraordinary results in recognition tasks [12], [13] and also in scene classification, when an appropriate database was used [29]. They are, however, very time consuming and complex. It is interesting to investigate, whether the topology of those networks can be reduced to save time and resources and might therefore make it usable for embedded hardware or mobile devices.

In this report, we investigate CNN structures on their ability to generate scene classification features, while using only shallow Convolutional Nerual Networks as high-level features might not be necessary to capture the overall GIST of the scene. We therefore compare well known deep CNN structures with our reduced or shallow network and see, if the reduction of complexity still yields acceptable results. We then add additional layers and investigate, whether speed and accuracy are affected.

II. Related Work

For many years, using hand-generated or state-of-the-art features like SIFT, SURF or HOG was the standard way to go in classification tasks. The pioneering work of Oliva and Torralba [20] introduced special hand-designed features, ranging from low-level to high-level information, which they said would capture the most important features, or as they called it the *spatial envelope* of a scene. Based on this work, several improvements in accuracy and speed have been introduced [24]. For our work, it is however more interesting to look at research that has been done about Convolutional Neural Networks and their role in scene classification tasks.
Changing parameters of a CNN for specific classification tasks has led to powerful new solutions in very specific areas like pedestrian detection [28], traffic sign detection [3] or scene recognition. Most of these approaches use large databases, several convolution and subsampling layers and are mostly relying on a ”deep” architecture. Zhou et al. [29] however state, that a CNN trained on the ImageNet database is not well suited for the task of scene classification and therefore introduce a large scene-centric dataset called PLACES. This means that a network has to be trained on the new image set, which is a very time consuming process. As models, already trained on ImageNet are available to download for a variety of CNN tools, other research work tries to bypass the need to

train anew [5]. Different ways are proposed to transfer mid-level features, generated with these models, and use them with other techniques like spatial pyramid or statistical analysis [1], [25].
Ba and Caruana show [2], that a shallow network is capable of achieving similar accuracies to deep networks, even though deep networks make learning easier. McDonnell and Vladusich [16] however introduce a shallow Convolutional Neural Network that yields results remarkably close to state-of-the-art deep networks while drastically reducing training time.

III. Experimental Setup

A. Datasets

A well trained CNN needs a suitable database with labels. Preparing this data is a cumbersome task and requires a lot of time. Fortunately, a variety of different, free-to-use image classification databases were published over the years. Amongst other image collections like CIFAR-10/100 [11], or Caltech-101/256 [6], [7], ImageNet [4] is the most famous one, including over 15 million labelled high-resolution images in over 22.000 categories. They are, as Zhou et al. [29] state, much too detailed for scene classification, which is why there are databases specifically created for this task like SUN [26], PLACES [29] or the Environmental Scene Database [19].

Fig. 1: Training data: Hand picked images from the SUN [26] database, split into indoor and outdoor categories.

Unfortunately, we were not able to work with the PLACES database, as it is not publicly available and therefore used a hand picked selection of the SUN2012 dataset, providing about 3933 color images in the two categories *indoor* and *outdoor* for training. We chose to only use two instead of several scene classes, to reduce the focus on the database but concentrate on the network's architecture. Both categories were randomly divided into training, validation and test data with a proportion of 88/4/8 inspired by LSVRC 2014 [17]. While the training and validation sets were used for training, the test set served as the final evaluation of the trained model. Other preprocessing steps included rescaling all images to 256x256 pixels and subtracting the mean activity from each pixel [12].

B. CNN Tools

As Convolutional Neural Networks are just a specific kind of Neural Network, that process data in a grid-like topology, many implementations and tools were published to help building nets with only a few lines of code. In general, in these tools or frameworks, a user defines a set of variables like number of hidden layers, learning rate or filter size. It was surprisingly difficult to find a fast, easy but powerful framework, as most of the tools like EBLearn [21] or ConvnetJS [9] offer already trained models, often based on ImageNet or MNIST [14] database but are hard to train with your own data. Even after contacting some of the developers, we were not able to get information on how own data must be prepared to be used by these networks for training. Cuda Convnet [10] specializes on speed but can only be used on specific, rather expensive graphics cards, which we could not get hold of. In the end, we decided to use the well known and fairly well documented deep learning network Caffe [8], which profits from its huge and very active community. The immense number of different branches can be overwhelming at times but enabled us to quickly find some guidelines for network optimization or visualization of network parameters.

C. CNN architecture

Caffe supports a variety of different layers for building Convolutional Neural Networks. As stated before, it was our objective to create a shallow network which eventually contains only one convolutional layer. For this purpose we started with a network containing only the most necessary parts which is illustrated in figure 2. Starting with the input layer for processing the incoming images, the second layer applies different filter kernels on each image. For simulating the neuron activity we decided to use Rectified Linear Units (ReLUs) which train several times faster then other activation units and tend to reduce overfitting [12]. These activations are sub-sampled by a pooling layer followed by a fully-connected layer with two outputs for our two-class problem. In later experimental phases a Local Response Normalization (LRN) layer was applied to the sup-sampled values. This resulted in drastically better results due to higher learning rates and a more stable network behaviour. Furthermore, we tested the application of an additional dropout layer as this promised less overfitting and inhibit co-adaptations of neurons [23].

D. Parametrization

A well known challenge of CNNs is to find the right parametrization for the sheer number of different parameters. Even though a shallow network structure reduces the number already, during the course of this work we were confronted with three different kinds of parameters. The first group was only subject to the technical setup. This implied the batch size which determines how many images can be processed simultaneously in one iteration. Group two consisted of parameters that can vary greatly between different networks but should be consistent in our experiments. Therefore these parameters were based on other scientific publications and user experiences with the framework. We concluded to use a convolutional filter with the size of 7x7 pixels and a stride of 3 as this improves the classification performance [27] [22]. For the pooling layer a sub-sampling over a 3x3 pixels region with stride 2 was used

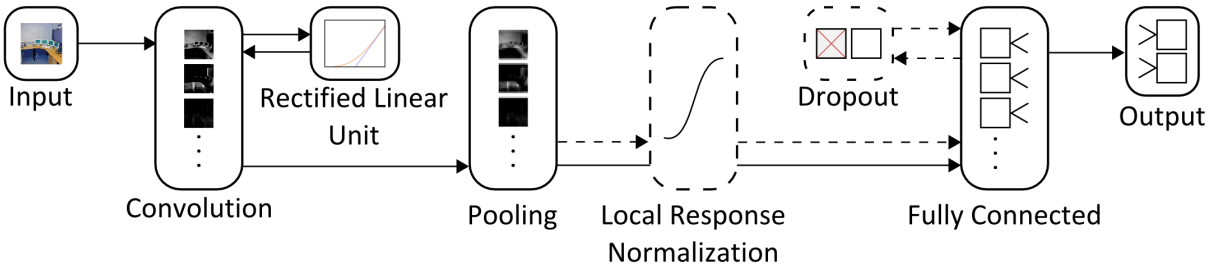

Fig. 2: CNN layout: Depicting the layer structure whereas dotted components are optional and were applied in later stages.

which normally provide the best results [12]. The last group of parameters were different for each training routine and some even had to be changed during the course of training. Therefore these parameters were the object to experiment with. Number of filters, basic learning rate and the existence of a LNR or dropout layer were defined before each training. During training, the learning rate had to be adjusted manually.

E. Computer Setup

All routines were performed on the latest stable Ubuntu system version 14.04. The Computer running these procedures contained an Intel Core i7-2600K processor, clocking at 4.20 GHz, 8 GB DRR3 RAM and a NVidia GeForce GTX 590 graphics card, which had 2 cores and an overall memory of 3 GB. This is, however, not a high-end setup, but emphasizes the idea of simplifying the network to run on affordable hardware setups. Within Caffe, training was done with GPU-support to fasten the process, which is considered to be state-of-the-art.

IV. Experiment

A. Models

We started with the most simple model without a LRN layer and 24 different filters. Specification of base learning rate and its rate of changing had to be adjusted throughout different training runs for each individual model. These adoptions are explained in subsection IV-C. Furthermore, models with 42 and 96 filters were tested and all their counterparts with the additional LRN layer and dropout layer resulting in 9 different models to experiment with. Throughout this paper we will refer to them as F24, F24 LRN, F24 LRN DO, F42 and so on, whereas the first term shows the number of different filter kernels and the other two acronyms indicate if this specific layer is utilized.

B. General Procedure

Each model was trained on the same training set. To artificially enhance their number, two data augmentation techniques [12] were used. Input samples were randomly mirrored to represent a different image. In addition the input layer didn't use the whole image but randomly cropped a 227x227 pixels area using it as training samples. One forward pass processed a fixed number, which was dependent on the batch size, of images. Afterwards the weights of the Convolutional Network were altered with gradient descent. After the model had converged we tested it's capabilities with the help of the testing set. At the end of this process, we recorded the time needed for convergence as well as the final accuracy of the model.

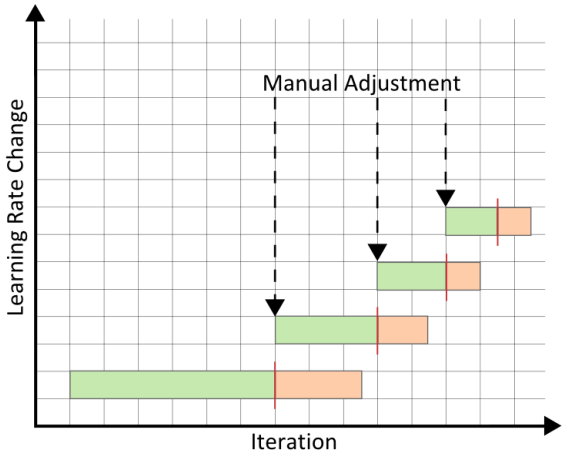

Fig. 3: Learning rate adjustment: Green indicating parts of the final trained network model and red representing discarded changes.

C. Adjustments

During training process, we had to observe two different parameters. The first parameter was the measured accuracy on the validation set which was fetched between a fixed number of iterations steps. Secondly we monitored the calculated loss which described how good the predictions matched the calculated output. The goal is basically an exponentially increasing accuracy and an exponentially decreasing loss value [15]. Since Caffe doesn't support such behaviour by itself we had to manually adjust the learning rate. The framework offered only a limited number of possibilities for changing parameters during the course of the training without interrupting it. Even though there is a huge community, the framework itself isn't that well arranged. So adding own routines into specific parts of the program were not easy to realize. Besides, Caffe depends on different programming languages. Figure 3 illustrates how the training for one model took place. Due to these adjustments the training process posed as a cumbersome task. You had to keep an eye on the accuracy and loss value to prevent overfitting or an overstating change for either one of them but simultaneously keep an adequate learning rate so that the model still improves over time.

Model	Base learning rate	Batch size	Epoch time	Convergence	Accuracy	
					Validation set	Test set
F24	$0{,}1^6$	128	2,81 s	103 min	83,53%	77,62%
F24 LRN	$0{,}1^3$	128	2,93 s	72 min	93,75%	85,58%
F24 LRN DO	$0{,}1^3$	128	2,92 s	71 min	93,85%	83,27%
F42	$0{,}1^6$	128	4,11 s	140 min	83,01%	71,72%
F42 LRN	$0{,}1^3$	128	4,21 s	103 min	92,84%	84,93%
F42 LRN DO	$0{,}1^3$	128	4,20 s	102 min	93,12%	83,55%
F96	$0{,}1^7$	32	6,56 s	50 min	82,56%	62,54%
F96 LRN	$0{,}1^3$	32	7,01 s	43 min	94,65%	84,60%
F96 LRN DO	$0{,}1^3$	32	7,07 s	44 min	94,65%	86,18%

TABLE I: Comparison of the different model parameters and their respective outcome.

V. Results and Discussion

A. Comparison of the different models

For comparing and analysing the different models we evaluated them by following parameters. The base learning rate influencing the amount of change during training. Batch size defines how many images can be processed simultaneously in one iteration. Epoch time represents how much time in seconds it took the model to train once on all images of the training set. Convergence states the time in minutes necessary to train the whole model. As figure 3 shows only the green parts are included in this time. Furthermore the time for validation which is mainly used for adjustments is excluded too. The accuracy values indicate the rate of successful classification for the validation and the test set. In addition we took a look at the different filter kernels. Table I summarizes the results of the performed experiments.
As previously explained batch size is subject to the capabilities of the computer and the required memory of the network. Therefore we had to use a smaller batch size for the CNNs with 96 filter kernels. Table I shows that the addition of a LRN layer has a significant impact on the output model. First of all it enabled us to use a higher base learning rate. In the standard model the loss value often concluded into an invalid value during early stages of the training process. This is due to the huge difference of value ranges between the activations and therefore increased the correction gradient further in every iteration. The LRN performs a local inhibition of values over multiple images resulting in a much more stable domain. This can even be seen in the resulting filter kernels visualized by figure 4. Although the normalization raised the epoch time, the model could converge a lot faster as the learning rate can be set to a higher initial value. Most important is the fact that the accuracy drastically increased for both data sets. Unfortunately, the effect of a dropout layer is less significant. It barely enhanced the resulting model and in two cases even worsened the accuracy on the test set. This could be due to the fact that we only have one fully-connected layer with an output of two values. For this setup, it practically only made the changes in one iteration half as effective as before. All models perform up to 6-20% worse when classifying the test set. The manual adjustments, the small image sets and the constant validation set are the cause of this difference. For better performance there should be a greater database and a randomly changing validation set. Besides, the alternation of the learning rate should occur more flexible and automatically controlled by the framework which is at the moment not able to do so. Although you could expect an increased amount of time for convergence similar to the increase from F24 LRN to F42 LRN the models with 96 filter kernels are faster to train. This results from the smaller batch size and the otherwise stable model parameters. Therefore the control of the learning rate happened in a similar fashion and manually influenced the learning curve. The last thing to mention is the impact of the number of different filter kernels. In our shallow net layout it had little to basically none influence at all on the different accuracies as our limitation to one convolutional layer is already the restriction for a better result.

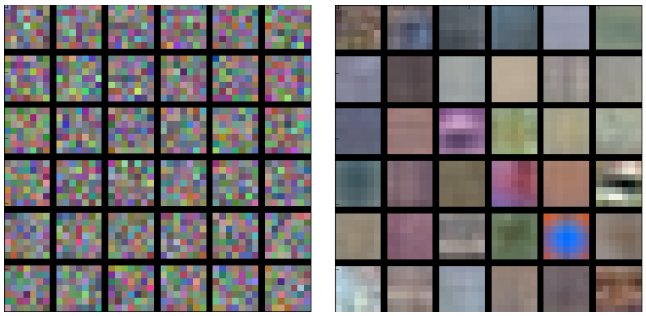

Fig. 4: Filter comparison of the first 36 filter kernels: Left: F96 model. Right: F96 LRN DO model.

B. Comparison to related work

Besides comparing the results within our different setups, it is interesting to see, how our best results compete against state-of-the-art approaches. As demonstrated by Zhou et al. [29] Deep Convolutional Neural Networks are capable of achieving astonishing results with accuracy up to 96% in scene classification tasks, which exceeds our results by far. They, however, used several days of training and a very large dataset. It is hard to really compare the results, without using the same training and validation data, but already gives an idea, of how much faster a shallow network is. When comparing results with hand-generated features, we can see some improvement in training time and accuracy. As stated by Vismanathan,

Siagian and Itti [24], these features yield accuracies about 80-86% while using about 2 hours to train a classifier. We were however able to use only 44 minutes of training to achieve 86% accuracy.

VI. Conclusion

In this work we investigated whether Shallow Convolutional Neural Networks are able to compete with deep CNN models and in their ability to correctly extract scene classification features from images and their performance compared to hand-generated GIST features. In our experiments we used CNNs only containing one convolutional layer, a sub-sampling layer and one fully-connected layer in a two-class problem and further increased the number of filters, added a normalisation layer and a dropout layer to see if the performance increased. We showed that a shallow network is able to produce GIST features in reduced time and with satisfactory accuracy, by adding only a Local Response Normalization layer and a Dropout Layer.

A. Future Work

Even though a shallow network has a limited amount of parameters and complexity, it still offers a large number of possible combinations, that could be investigated in further tests. Also, increasing the number of scene classes and the number of test images could lead to a better understanding of GIST features. Furthermore, manual adjustment of network parameters especially the learning rate are not sufficient enough for proper training. Future work should search for an alternative framework or dive deeper into Caffe to alter the code to fit it to their own purposes. Also having a deeper look into the GIST features and comparing them with hand-generated features in terms of their activation on images might be interesting.

References

[1] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. From generic to specific deep representations for visual recognition. *CoRR*, abs/1406.5774, 2014.

[2] L. J. Ba and R. Caurana. Do deep nets really need to be deep? *CoRR*, abs/1312.6184, 2013.

[3] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. A committee of neural networks for traffic sign classification. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1918–1921, July 2011.

[4] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[6] L. Fei-Fei, R. Fergus, and P. 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.

[7] G. Griffin, A. Holub, and P. Perona. Caltech-256 Object Category Dataset. Technical Report CNS-TR-2007-001, California Institute of Technology, 2007.

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[9] A. Karpathy. Convnetjs - deep learning in your browser, 2015.

[10] A. Krizhevsky. Cuda convet - high-performance c++/cuda implementation of convolutional neural networks, 2015.

[11] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, page 2012.

[13] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, Dec. 1989.

[14] Y. LeCun and C. Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[15] F.-F. Li and A. Karpathy. Cs231n convolutional neural networks for visual recognition, 2015.

[16] M. D. McDonnell and T. Vladusich. Enhanced image classification with a fast-learning shallow convolutional neural network. *arXiv preprint arXiv:1503.04596*, 2015.

[17] H. S. J. K. S. S. S. M. Z. H. A. K. A. K. M. B. A. C. B. Olga Russakovsky, Jia Deng and L. Fei-Fei. Large scale visual recognition challenge 2014, 2014.

[18] A. Oliva. Gist of the scene. *Neurobiology of attention*, 696(64):251–258, 2005.

[19] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42:145–175, 2001.

[20] A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. In *Progress in Brain Research*, page 2006, 2006.

[21] P. Sermanet, K. Kavukcuoglu, and Y. LeCun. Eblearn: Open-source energy-based learning in c++. In *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI'09)*. IEEE, 2009.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[24] M. Viswanathan, C. Siagian, and L. Itti. Comparisons of gist models in rapid scene categorization tasks.

[25] Y. Wang and Y. Wu. Scene classification with deep convolutional neural networks.

[26] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3485–3492, June 2010.

[27] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[28] X. Zeng, W. Ouyang, and X. Wang. Deep learning of scene-specific classifier for pedestrian detection. In *ECCV*, 2014.

[29] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning Deep Features for Scene Recognition using Places Database. *NIPS*, 2014.