CS-E4600
Programming project

Team JSolo
Jonatan von Martens, 476139,
jonatan.vonmartens@aalto.fi

Project repository
`https://github.com/jonatanvm/deep-learning-project`

May 19, 2019

Abstract

This project paper reviews different convolutional neural networks (CNN) to assess their usefullness in classifying images of the *tiny ImageNet* dataset.
Out of the networks reviewed, pretrained versions of AlexNet, SqueezeNet and ResNet perform the best. Out of all networks ResNet-18 gives the best result with an accuracy of 66%.

1 Introduction

During the course *CS-E4890 - Deep Learning* we've many times explored Image classification. The classification has however dealt with a very small number of training samples and I want to explore how methods discussed during this course preform on larger image-datasets, and if they can give competitive results. I'll also study methods which weren't examined during this course and compare them to methods which were. I hope comparing the methods will give insight in to how good the methods discussed during this course are and hope that I will learn more about deep learning by studying methods not discussed during this course.

This paper gives an overview of the methods and models used, as well as results and conclusions. For a more detailed look on the model parameters and on how the models were run visit the project repository, which address you can find on the cover of this project paper.

2 Data

As data I used *tiny ImageNet*, which is a subset *ImageNet* and part of the Stanford course CS231n [4, 3, 5]. *ImageNet* is image database, which consists of more than 100 000 "synsets" or "synonym sets". A synonym set is a collection of images described by words or word phrases. In total the ImageNet database consists of more than 14 million images.
The smaller *tiny Imagenet* only has 200 classes, each containing 500 images, with an additional 50 validation images and 50 test images for each of the 200 classes. This only makes a total of 120000 images, or approximately 0.8% of ImageNet.
The images in *tiny ImageNet* are 64x64 pixel RGB images. This means that the data has $64 \cdot 64 \cdot 3 = 12288$ dimensions.

Because only the validation set was annotated with the correct labels, I discarded the test set and split the validation set in to two sets, one of which became the test set and one of which became the validation set. Table 1 summarizes the used training, validation and test-set split.

Set	Classes	Images/Class	Total number of images
training	200	500	100000
validation	200	25	5000
test	200	25	5000

Table 1: Dataset split [4]

3 Methods and Models

I decided to use convolutional neural networks (CNNs), as many were discussed during the course and many have performed well on ImageNet. [1, 11, 10]
Of the methods discussed during this course I decided to use LeNet and ResNet. Of methods not discussed during this course I used AlexNet and SqueezeNet. Sections 3.1, 3.2, 3.3 and 3.1 give an overview of how these methods work and what their structure is. Notebooks *lenet.ipynb*, *run_models.ipynb* and *run_models2.ipynb*, in the project repository, give details about all the network structures.

3.1 LeNet-5

LeNet-5 is a CCN developed by Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner. LeNet was originally designed for computer generated and handwritten character recognition.[15, 12]

Network structure

LeNet has two convolution layers followed by three fully connected layers. Each convolution layer is followed by a max pooling layer. ReLU is applied to all convolution and fully connected layers. The LeNet network structure is summarized in figure 1.

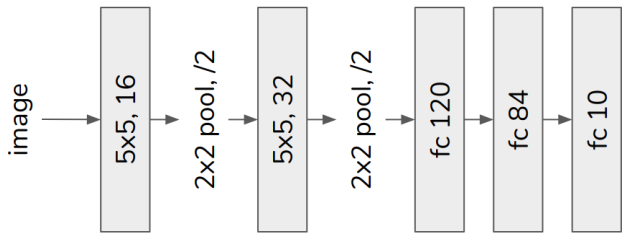

Figure 1: LeNet structure [9]

ResNet-18

A residual neural network (ResNet) is a neural network, in which the network consists of groups of blocks, where blocks have so-called skip connections to avoid the problem of vanishing or exploding gradients [7]. A block can consist of different numbers of convolutional layers depending on the implementation, but most typically the number is either two or three [10].

Network structure

An example of a ResNet structure can be seen in figure 2.

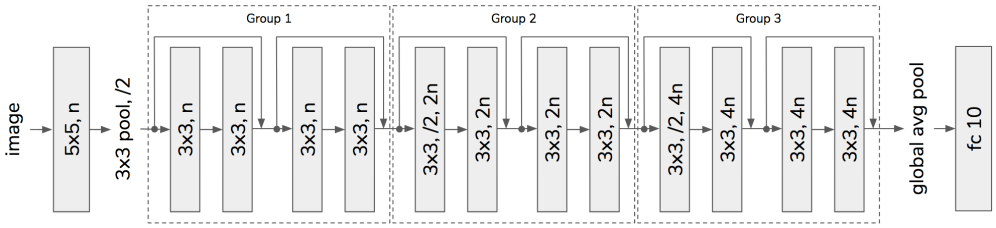

Figure 2: Example ResNet structure [9]

The *PyTorch* ResNet-18 structure that I used, has a similar structure to the one shown in table 2. The difference is that ResNet-18 consists of four groups, containing four blocks, each block containing two convolutional layers. The convolution layers have a resolution of three, 2d batch normalization after each layer, and ReLU normalization between the layers. [14]

3.2 AlexNet

AlexNet is a CNN designed by Alex Krizhevsky, which won the ImageNet LSVRC-2010 contest on September 30, 2012.[6]

Network structure

The network consists of five convolution layers followed by three fully connected layers. ReLU is applied to each layer and layers 1, 2 and 5 are followed by max pooling. All convolution layers have a resolution of 3 except for the first one which has a resolution of 11.[11]
The structure of AlexNet can be seen in figure 3. The AlexNet implementation in PyTorch, which I used, also has a 2d adaptive pooling layer between the last convolutional layer and the first fully connected layer.

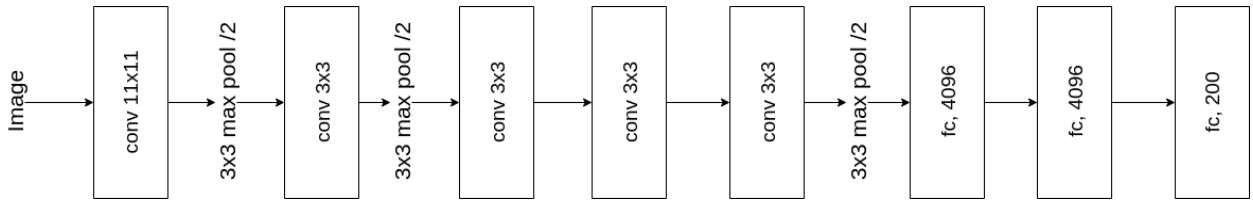

Figure 3: AlexNet structure

3.3 SqueezeNet

SqueezeNet is a network which was developed to combat the high memory requirements of neural networks, by reducing the number of parameters.[8]

Network structure

SqueezeNet mainly consists of successive "Fire modules". A Fire module consists of two *squeeze* convolution layers followed by a "normal" convolution layer with a resolution of three. The *squeeze* simply means that the convolution layer has a resolution of one. Each of the tree convolution layers are followed by ReLU activations. The full network structure of SqueezeNet can be seen in figure 4.

When testing I used SqueezeNet v1.1 which reduces the the required computation by multiple of 2.4 compared to SqueezeNet v1.0, while maintaining the accuracy.
The difference in the implementation between SqueezeNet v1.1 and v1.0 is that in v1.1 the first convolutional layers has 64 filters instead of 96 and a resolution of 3 instead of 7 and that pooling is performed after layers 1,3 and 5 instead of layers 1,4 and 8. [14][13]

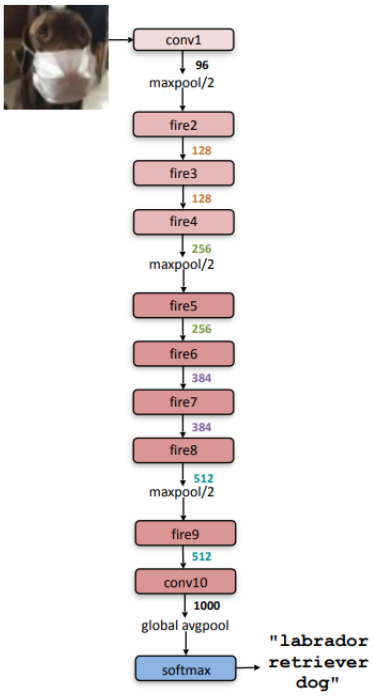

Figure 4: SqueezeNet v1.0 structure [8]

4 Training

As LeNet requires 28x28 single channelled greyscale images, I will use it as a benchmark results as I don't expect it to give very promising results given that the dimensionality is greatly reduced and I have less data to train the network with.
After that I'll train AlexNet, SqueezeNet and ResNet, on the full 64x64 RGB images, expecting to get better results than LeNet.
Finally we will use AlexNet, SqueezeNet and ResNet which are pretrained on ImageNet. In order to train the pretrained models, the data needs to be upscaled to 224x244 pixel images, because that is what the pretrained data dimensionality is. This means however that the training will be slower as the dimensionality of the data is increased.

When running the different methods I used cross-entropy loss and stochastic gradient descent (SGD) to train the models, both of which are implemented in PyTorch [2]. For SGD I used 0.001 as the learning rate and 0.9 for momentum. I varied batch sizes between 32, 50 and 100, and epochs between 10, 40 and 50.

5 Results

The results are divided in to two tables, table 2 and table 3. Table 2, shows the results of the models which weren't pretrained with ImageNet table 3 shows the results of the models which were pretrained with ImageNet. The pretrained networks performed the best, which was to be expected, and ResNet-18 was the overall winner with an accuracy of 66.5%.

Method	Image scaling	epochs	batch size	Training accuracy	Validation Accuracy
LeNet-5	28x28 grayscale	40	32	0.218	0.220
AlexNet	64x64 RGB	40	50	0.237	0.240
SqueezeNet	64x64 RGB	40	50	0.233	0.230
ResNet-18	64x64 RGB	40	50	0.228	0.237

Table 2: Not pretrained

Method	Image scaling	epochs	batch size	Training accuracy	Validation Accuracy
AlexNet	224x224 RGB	10	100	0.562	0.573
SqueezeNet	224x224 RGB	10	100	0.589	0.586
ResNet-18	224x224 RGB	10	100	0.628	0.625
AlexNet	224x224 RGB	50	100	0.590	0.587
SqueezeNet	224x224 RGB	50	100	0.544	0.546
ResNet-18	224x224 RGB	50	100	0.660	0.665

Table 3: Pretrained

Stanford has a leaderboard of the best results achieved on the Tiny-ImageNet dataset which we can compare our results to. [4]

The results, as of 14 May 2019 (15:54) where:

#	Name	Accuracy
1	Avati,Anand	0.732
2	Kim,Hansohl Eliott	0.689
3	Qian,Junyang	0.662
4	Liu,Fei	0.661
5	Zhai,Andrew Huan	0.554
6	Shen,William	0.548
7	Shcherbina,Anna	0.494
8	Ebrahimi,Mohammad Sadegh	0.439
9	Ting,Jason Ming	0.384
10	Random Guesser	0.005
11	Khosla,Vani	0.005

Table 4: Best Stanford-student results

We see that the results we have achieved are in line with the results released by Stanford on their leaderboard. If we however compare our results to the top results in the world, which have been achieved on the complete ImageNet shown in table 5, we see that we are quite far behind.

#	Name	Accuracy	Year
1	GPIPE	84.3%	2018
2	AmoebaNet-A	83.9%	2018
3	MultiGrain PNASNet @ 500px	83.6%	2019

Table 5: Best ImageNet results [1]

6 Conclusion

It's very surprising that LeNet performed almost as well as non-pretrained AlexNet, SqueezeNet and ResNet-18, even though LeNet was trained with downscaled, greyscale images. This could however simply be because I didn't fine tune the parameters enough, or run AlexNet, SqueezeNet and ResNet-18 for enough epochs.
The pretrained models performed well and I'm happy with the results. They aren't the best in the world, but that is to be expected given the limited computational resources I had. The pretrained models do however have accuracies which are in line with the results released by Stanford as a part of their competition.

References

[1] Papers With Code. *Image Classification on ImageNet*. [Online; accessed 19 May 2019]. 2019. URL: https://paperswithcode.com/sota/image-classification-on-imagenet.

[2] Torch Contributors. *PYTORCH DOCUMENTATION*. [Online; accessed 19 May 2019]. 2018. URL: https://pytorch.org/docs/stable/index.html.

[3] J. Deng et al. *ImageNet: A Large-Scale Hierarchical Image Database*. [Online; accessed 13 May 2019]. 2018. URL: http://image-net.org/.

[4] Andrej Karpathy Fei-Fei Li and Justin Johnson. *CS231n: Convolutional Neural Networks for Visual Recognition*. [Online; accessed 13 May 2019]. 2019.

[5] Andrej Karpathy Fei-Fei Li and Justin Johnson. *Tiny ImageNet Visual Recognition Challenge*. [Online; accessed 13 May 2019]. 2019. URL: https://tiny-imagenet.herokuapp.com/.

[6] Hao Gaos. *A Walk-through of AlexNet*. [Online; accessed 18 May 2019]. 2017. URL: https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637.

[7] Kaiming He et al. *Deep Residual Learning for Image Recognition*. [Online; accessed 18 May 2019]. 2015. URL: https://arxiv.org/abs/1512.03385.

[8] Forrest N. Iandola et al. *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size*. [Online; accessed 18 May 2019]. 2016. URL: https://arxiv.org/abs/1602.07360.

[9] Alexander Ilin. *CS-E4890 - Deep Learning, course materials*. 2019. URL: https://mycourses.aalto.fi/course/view.php?id=20606.

[10] Prakash Jay. "Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image Classification: From Microsoft to Facebook [Part 1]". In: (Feb. 2018). [Online; accessed 18 May 2019]. URL: https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[12] Yann LeCun. *LeNet-5 fffdfffdfffd A Classic CNN Architecture*. [Online; accessed 19 May 2019]. URL: http://yann.lecun.com/exdb/lenet/.

[13] *Official SqueezeNet repo*. [Online; accessed 18 May 2019]. 2019. URL: https://github.com/DeepScale/SqueezeNet.

[14] Adam Paszke et al. *Automatic differentiation in PyTorch*. 2017.

[15] Muhammad Rizwan. *LeNet-5 fffdfffdfffd A Classic CNN Architecture*. [Online; accessed 14 May 2019]. 2018. URL: https://engmrk.com/lenet-5-a-classic-cnn-architecture/.