Ravensburg-Weingarten University

Computer Vision

Image Classifier

Fabian Wicker 32235

5. November 2020

HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

1 Abstract

Without any or enough training data there is no way for deep learning to make a robust classification. In this report, google images are used for training and validated with several datasets and classes. In addition, there is a Gradient-weighted Class Activation Mapping (Grad-CAM) (section 3.11) implemented for localization of the predicted classes. The main goal of the project was to find out whether sufficient accuracy for image classification can be achieved with image search engines like google and bing images. Images from search engines can also contain other classes, therefore a threshold was used to get only the most reliable predictions of the network. For easier usability, a complete toolchain is implemented, the user only needs to type in the required classes, the path to the images to be predicted and optional the desired threshold value. The program automatically searches locally for pre-trained networks, if this is not the case the program loads about 600 pictures for every class and starts to learn the new class with transfer learning. After transfer learning, the network will be fine tuned for a few epochs. Finally the newly trained network will start to predict all images in the given path. The results can be used for pre-labelling as well as a more precisely dataset for a different network training.

HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

Inhaltsverzeichnis

List of abbreviations

Grad-CAM	Gradient-weighted Class Activation Mapping
ResNet50	Residual Networks 50
ResNet	Residual Networks
API	application programming interface
ReLU	Rectified Linear Unit
HTML	Hypertext Markup Language
JPEG	Joint Photographic Experts Group
RGB	red, green, and blue color space
RGBA	red, green, blue and alpha color space
PNG	Portable Network Graphics
RAM	Random-Access Memory

2 Introduction

One of the most time consuming, difficult and expensive thing on machine learning is to create a dataset. Why shouldn‘t use Google or Bing images to get a huge amount of images pre-labeled with the required classes? The answer is not that easy, for example if you are searching for a car within a search engine, you would get sets of images of the interior of a car as well as of the exterior. This is one argument against using search engines as data source, another argument is the handling with licenses of the images or to specific classes like not very well known things. Nevertheless to find out, how accurate and more or less useful it can be, to use search engines for datasets. The toolchain contains all parts of loading images, resize, reformat, transfer learning, fine tuning and predict the user given classes on images. In addition the tool can make also a Grad-CAM for localize the classes within the images.

2.1 Google Images

Google Images (LLC, 2001) was released in July 2001 and indexing billions of images (Siegler, 2010) in the web. The Google search algorithm gives a good selection of the searched classes. For the search results correct images are not that good for a dataset. But for example, you want to train a network to detect cars, but Google Images will give you car images from inside and outside. The images of the car interieur are a big problem for training a network for the class car.

2.2 Bing

Bing, launched on June 3 2009, is also a search engine like Google Images (LLC, 2001) but developed and hosted by Microsoft. The search engine Bing is also used as data source for images in this project to generate a greater variation of images. The same advantages and disadvantages apply to Bing as to the google image search engine.

2.3 ImageNet

ImageNet is a open dataset which provides over 14,197,122 labelled images and 21841 synsets indexed. (Stanford Vision Lab, 2009)

> “There are more than 100,000 synsets in WordNet, majority of them are nouns (80,000+). In ImageNet, we aim to provide on average 1000 images to illustrate each synset.”(Stanford Vision Lab, 2009)

2.4 Keras

Keras was used as framework in this project, because its very easy to use and well documented with python.

> "Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides."(Chollet et al., 2015)

2.5 Tensorflow

Tensorflow is used in this project as backend for machine learning with high-level Keras-application programming interface (API). (Martín Abadi et al., 2015)

> "TensorFlow is an interface for expressing machine learning algorithms and an implementation for executing such algorithms. A computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards."(Martín Abadi et al., 2015)

2.6 ResNet

In 2015 the model ResNet (Kaiming et al., 2015) was the winner of ImageNet challenge (Russakovsky et al., 2015). ResNet was one of the first extremely deep networks with up to 152 in relation to the winner of the ImageNet challenge 2012 AlexNet (Krizhevsky et al., 2012), which has only 8 convolutional layers. For this project Residual Networks 50 (ResNet50) was chosen, because the relationship between accuracy and speed is given. To avoid starting from scratch a pre-trained ResNet50 with ImageNet weights was used and do transfer learning and fine tuning the model with the user given classes.

3 Methods

3.1 Starting Point

At the beginning the user must start the programm with the terminal. The programm itself takes two mandatory arguments »*--classes <keyword_1,keyword_2>*« (in short: »*-c <keyword_1,keyword_2 >*«) for the detecting classes and »*--prediction_path <path_to_predicting_images>*« (in short: »*-p <path_to_predicting_images>*«) with the path to predicting images. Another optional argument is the threshold for the predicted images, which is very important for the accuracy of the results. The threshold has the default value 0.99, which means only results with 99% certainty are saved in results, images below the threshold would be skipped. In order to the default value of the threshold you must use the argument »--threshold <threshold_value>«(in short: »-t <threshold_value>«). In order to start the program with all arguments, the command must look like this:

```
1         python classifier_training.py -c "cat,dog" -p "~/eiwomisau/test/mixed
    " -t 0.98
```

Launch of the toolchain

Now the toolchain is looking for an already trained network, if a suitable network is available, the network will be loaded and starts predicting the user given images. If there is no suitable network, the toolchain starts to download images with the desired keywords.

3.2 Downloading Images

In order to download images by search engines like in this project with Google Images and Bing, there are several options. The simplest option would be to download images of the user given class by hand, but this is not practical. There is a big problem with scraping search enginges. They do not want to be read out automatically, because that means unnecessary server load without advertising. The result of this is, that they have costs but no earnings. To prevent to be read out automatically they often change their Hypertext Markup Language (HTML) tags. This means it‘s a cat and mouse game. During this project, the first library »Google Images Download» (Vasa, 2018-08-11) downloading Google and Bing Images was deprecated for Google Images in the middle of June because they changed their HTML tags. An alternative scraping library was found in »Image-Downloader» (Zheng, 2017-01-11). With this library it was possible to download up to 800 Google Images and 1000 Bing Images per keyword.

3.3 Remove of small images

After downloading all images of every keyword smaller than the input size of the modified network (224x224 pixels) will be deleted, because they are useless for training.

3.4 Convert Compression

Since we have no control of how the images were recorded and saved, we need to convert all unsuitable compressions. The toolchain uses red, green, and blue color space (RGB) and the Joint Photographic Experts Group (JPEG)-compression but there are countless other formats like red, green, blue and alpha color space (RGBA) or Portable Network Graphics (PNG)-compression.

3.5 Resize Images

As soon as small images were deleted, images that are too large must be converted to the size of 224 x 224 pixels, as shown in figure 1.

Figure 1: Resizing resulting image with 224 x 224 pixels

3.6 Remove Duplicates

Identical images do not bring any advantage and are deleted by images hash values. With this method duplicates can be found very quickly, because not every picture has to be opened or cached several times while comparing. The image hashes are not memory-hungry so it‘s no problem to store them in Random-Access Memory (RAM).

3.7 Equal distribution of images

All keywords are based on the keyword with the fewest images, so the images in the classes that are too many are deleted. This is done because an unequal distribution of the number of images are not good for training.

3.8 Create Dataset

Before we can start with training, we need to create the dataset from the previously downloaded and edited images. The number of images per keyword after the previous is normally around 600. Now the images are splitted into a testing set (80%) and a validation set (20%). With a smaller amount of images per keyword, like in this project, there is a high risk for overfitting. A way to prevent this, is to augment the small number of images in and only in the test dataset with zooming, rotation, width shift range, height shift range, shear range and horizontal flip.

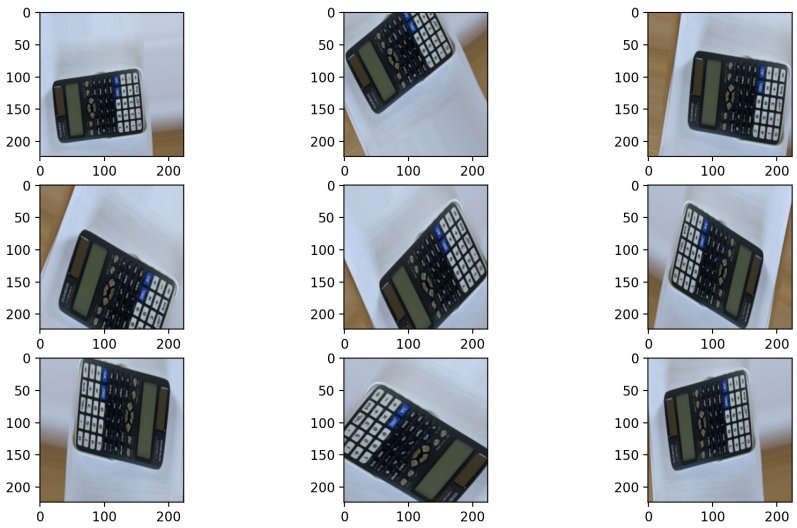


Figure 2: Image augmentation

3.9 Transfer Learning

Transfer learning is a method where a pre-trained network is reused for another task. Therefore you need to import the pre-trained network, freeze all layers and replace the output layer, which is only trainable now. So most of the trainable weights are non-trainable except the weights of the output layer, for example in this project the model has only 4,098 of 23,591,810 trainable parameters. Now we can start with the training for maximum 100 epochs, but early stopper was implemented to avoid unnecessary training time. There is also a model checker, which only save the solution of each epoch if the validation loss is smaller than the saved model.

3.10 Fine Tuning

Fine tuning is used in this project to get the last small improvements of a network. For that, all layers of the network are set to trainable and the learning is set to very low (1e-5). The low learning rate prevents the excessive change of weights in the network and is good to get the final optimization for the network, because with higher learning rates the weights oscillates to much. This implementation is only experimental and the fine tuned model will be only saved, if the validation loss is better than the saved model. On my tested keywords there was no benefit with fine tuning for example as shown in figure 3, but it cannot cause any damage and if it worser than the pre-saved transfer learning model, it has no effect on the model.

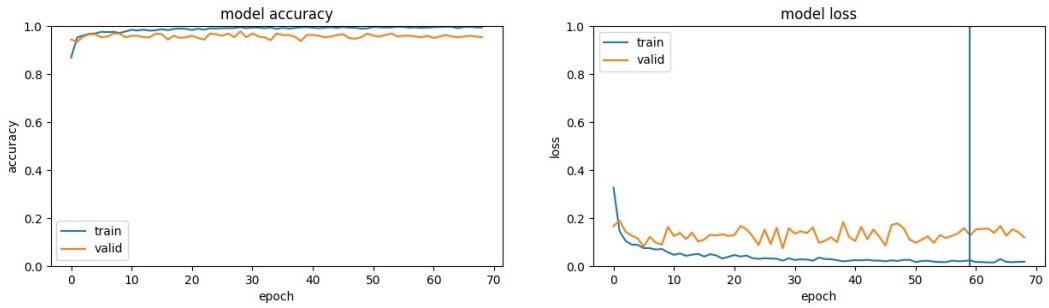


Figure 3: Training result of keywords calculator, pen and triangle ruler

3.11 Grad-CAM

After the prediction an attempt was made to localize the detected class with Grad-CAM, which visualize areas of interest in the last convolutional layer. The idea behind that is the areas where the network was activated must match with the detected object. As result we got a heatmap, which matches with the detected object in the image. The modified ResNet50 uses images with the width and height of 224 pixels, as result of this the last convolutional layer has 7 x 7 pixel.

> "To obtain class-discriminative localization map $L^c_{\text{Grad-CAM}} \in \mathbb{R}^{u \times v}$ with the width u and height v of 7 and Z is the number of pixels in the feature map, we first compute the gradient of the score for class c, y^c, with respect to feature map activations A^k of a convolutional layer $\frac{\partial y^c}{\partial A^k}$. These gradients flowing back are global averagepooled over the width and height dimensions (indexed by i and j respectively) to obtain the neuron importance weights a^c_k:"(Selvaraju et al., 2019-12-3)

$$a^c_k = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{global\ average\ pooling} \underbrace{\frac{\partial y^c}{\partial A^k_{ij}}}_{gradients\ via\ backprop} \tag{1}$$

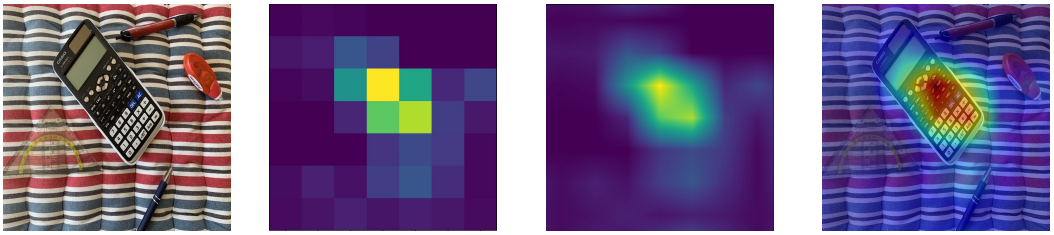
(Selvaraju et al., 2019-12-3)

Now we can calculate the weighted combination of activation maps to follow it by Rectified Linear Unit (ReLU), because we are only interested in positive influenced pixels for the class of interest. Negative pixels belong to other classes in the image (Selvaraju et al., 2019-12-3).

$$L^c_{Grad-CAM} = \text{ReLU} \underbrace{(\sum_k a^c_k A^k)}_{linear\ combination} \tag{2}$$

(Selvaraju et al., 2019-12-3)

As result of this calculations we receive a 7×7 pixels heatmap with the same size as that of the last convolutional layer in the modified ResNet50. The most important steps are illustrated in figure 4.



(a) original image (b) heatmap activation (c) heatmap resized (d) superimposed

Figure 4: Steps of Grad-CAM calculation

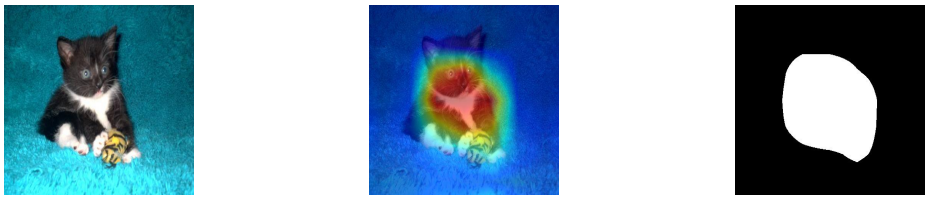
4 Results

To get an overview of the classifier accuracy a network with the keywords cat and dog was trained, because there is a testing set with this keywords on www.kaggle.com. The results with search engines trained network is shown in table 1 below.

total images	2025
threshold images	1843
threshold missed images	182
correctly classified images	1838
misclassified images	5
total accuracy	90.77%
threshold accuracy	99.73%

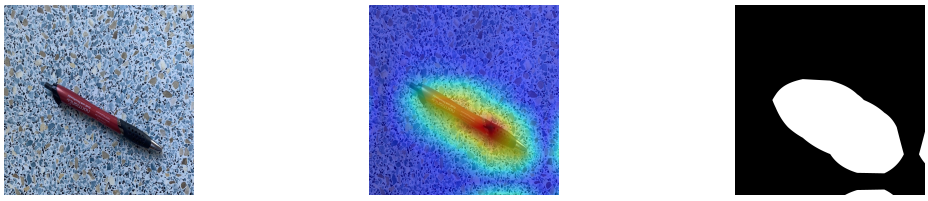
Table 1: Results of cat and dog testing set

From 2025 pictures remain 1843 predicted images with a threshold over 99%, all of the other images counts as misclassified and shown in the total accuracy of 90.77%. But the accuracy of threshold images is at 99.73%. This result shows the importance of a high threshold.
Furthermore, for detecting the localization the toolchain applies automatically Grad-CAM as shown in figures 5 and 6.



(a) original image (b) superimposed image (c) segmentation

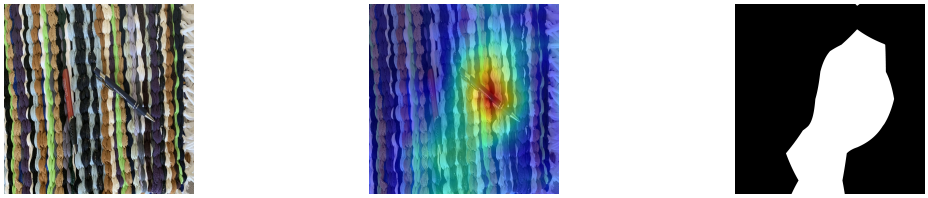
Figure 5: Results of Grad-CAM on one image from dog and cat testing set



(a) original image (b) superimposed image (c) segmentation

Figure 6: Results of Grad-CAM with a pen

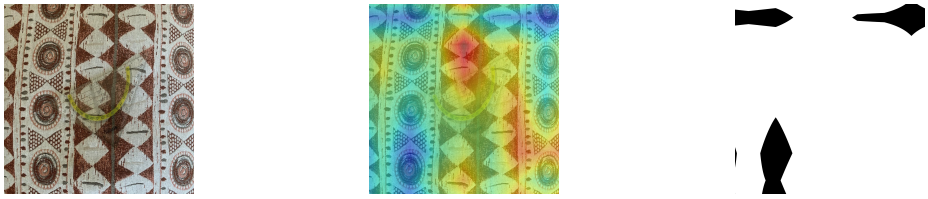
For every predicted image, the original, the superimposed and the segmentation image are stored in the results directory. As shown in figure 5 the localization is quite well interpreted, but the results of the localization are very dependent on the background as shown in figure 7.



(a) original image (b) superimposed image (c) segmentation

Figure 7: Results of Grad-CAM on an image with two pens with a rug in background

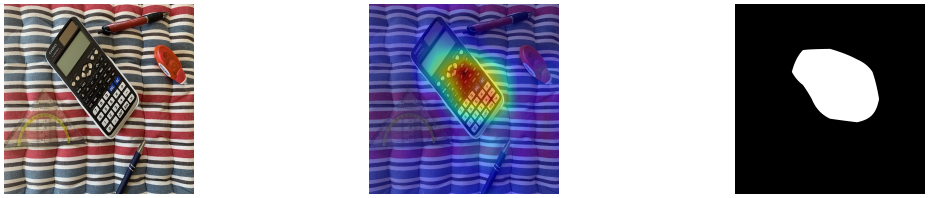
Also it‘s very difficult to locate transparent things like a triangle ruler with a background pattern as shown in figure 8.



(a) original image (b) superimposed image (c) segmentation

Figure 8: Results of Grad-CAM on an image with two pens with a rug in background

Finally a prediction was made with several learned classes to test what happens with the prediction and localization. The toolchain automatically select the class with the highest prediction accuracy like in figure 9 the calculator.



(a) original image (b) superimposed image (c) segmentation

Figure 9: Results of Grad-CAM on an image with multiple trained classes

Despite the vulnerability with localization, all four images were correctly classified.

5 Conclusion

Finally a toolchain was created, which will automatically train with images from search engines as training and validation dataset. After training a prediction for the selected classes can be made with a reasonably accurate localization. The positive result of the classification was surprising, even if the localization cannot keep up with the performance of the classification. For pre-labelling or as a „better“ dataset, this approach could mean an improvement, at least on very common classes. For the future could still be found out how a network trained with the data set generated in this project compares to the „original“ network trained with search engines. It could also be a benefit for labelling for a new labelling company, which is not yet able to use its own labeled images to train a network for pre-labelling.

Figures

Tables

Literatur

Chollet, F. et al. (2015). Keras. https://github.com/fchollet/keras

Kaiming, H., Zhang, X., Ren, S. & Sun, J. (2015, 10. Dezember). Deep Residual Learning for Image Recognition [Accessed: 2020-06-26]. https://arxiv.org/abs/1512.03385

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012, 30. September). ImageNet Classification with Deep Convolutional Neural Networks [Accessed: 2020-06-26]. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

LLC, G. (2001, 1. Juli). Google Images [Accessed: 2020-06-26]. https://www.google.com/imghp?hl=EN

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, … Xiaoqiang Zheng. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems [Software available from tensorflow.org]. https://www.tensorflow.org/

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. & Batra, D. (2019-12-3). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization [Accessed: 2020-06-26]. https://arxiv.org/pdf/1610.02391.pdf

Siegler, M. (2010, 20. Juli). Google Image Search: Over 10 Billion Images, 1 Billion Pageviews A Day [Accessed: 2020-06-26]. https://techcrunch.com/2010/07/20/google-image-search/

Stanford Vision Lab, P. U., Stanford University. (2009). ImageNet [Accessed: 2020-06-26]. http://image-net.org/about-overview

Vasa, H. (2018-08-11). Google Images Download [Accessed: 2020-06-26]. https://github.com/hardikvasa/google-images-download

Zheng, Y. (2017-01-11). Image Downloader [Accessed: 2020-06-26]. https://github.com/sczhengyabin/Image-Downloader

6 Appendix

The LaTeX files can be found on: Latex Sourcecode.
The sourcecode of this project was added to git as submodule but can also be found on: Project Sourcecode.