Seminar WS 2018/19: Smart Augmentation Learning an Optimal Data Augmentation Strategy

Fabian Kahl

Supervisor: Lilli Bruckschen

January 22, 2019

Abstract

This written report is a summary and critical analysis of the paper "Smart Augmentation: Learning an Optimal Data Augmentation Strategy" written by Joseph Lemley, Shabab Bazrafkan and Peter Corcoran [4] as part of the Humanoid Robotics Seminar at the University of Bonn during the winter semester 2018/19.

Nowadays neural networks are an absorbing part in the area of machine learning. An often occurring problem by training of a deep neural network is that there are not enough data available to maximize the generalization capacity of the network. This most commonly results in overfitting, what means that the generated network is biased to the training data and is not able to generalize the problem. [4]

To handle this problem they suggest their so called Smart Augmentation. It is a method to find the best augmentation strategy for the given data for training a deep neural network. Therefor they use a second network which generates new samples by means of blending several input images and considering the loss of the deep neural network. [4]

1 Introduction

A typical problem by training a deep neural network is that there are not enough data available to maximize the generalization capacity of the network. As a consequence, overfitting of the generated network happens most commonly. Suchlike trained networks are useless because they are biased to their training data and are not able to generalize their learned model to the underlying problem. They can only handle with the known input data from the learning process usefully. They will not be able to classify new data, which were unknown for the network during its training process but similar to the input data, with a satisfactory miscalculation. [4]

To handle with overfitting there are several different approaches in today's research. Popular approaches are for example different regularization techniques or data augmentation strategies [4]. This written report focuses on the approach of data augmentation.

1.1 Augmentation

1.1.1 Augmentation Approaches

There are in general two different approaches to augment data. The first one is unsupervised augmentation where data expansion is done regardless of the samples and with no feedback about the performance. Classical techniques to do unsupervised augmentations are for example blurring, rotating, translating, flipping and scaling of input data. For this kind of problem solving there are many different methods available. For a more detailed and deeper view in this kind of augmentation see publication [7] and [2]. [4]

The second approach is supervised augmentation which means any kind of controlled augmentation with a teacher. Such a method is the so called Smart Augmentation from Joseph Lemley, Shabab Bazrafkan and Peter Corcoran [4]. Their method will be summarized and critically analyzed in section 2, 3, 4 and 5. [4]

1.1.2 Augmentation Strategies

Not every augmentation strategy is suitable for every problem. There is no "right" augmentation strategy suitable for every problem. [4]

Hence it would be nice if Augmentation Strategies were predictable. But Augmentation Strategies are not predictable. There is no universal method to predict the result of a specific augmentation strategy on a specific data set before training the network with the data by the chosen strategy. To see the efficiency of an augmentation strategy the network must be trained with these augmented data. But the training of networks is time-consuming. Therefore it is not efficient to test all possible or a sufficiently large number of augmentation strategies for the given problem. [4]

2 Smart Augmentation

Smart Augmentation is a method to find the best augmentation strategy for the given dataset to train the deep neural network optimally. This method maximizes the achievement of the training process of the network. It reduces the probability of overfitting and increases the network's accuracy. [4]

Therefor it needs two neural networks. A neural network A generates new samples based on the given dataset and a deep neural network B solves the actual specific task. Network A learns how to generate augmented data during the training process of network B for network B. For that, network A gets the loss of network B as feedback to adjust its augmentation strategy. [4]

The main goal of the method is to train network B as well as possible while there are not enough samples in the given dataset to train without augmentations. The augmented data created by network A is the result of intelligent blending of features between two or more samples from the same class in the given dataset. So this augmented data reduces the loss of network B. Moreover it increases the accuracy of network B during the training process and reduces the probability of overfitting by network B. [4]

The explanation of this method is theoretical because it can be used in many modifications and there is no "right" implementation. It can be used in combination with other regularization methods, as well as with classical augmentation techniques. [4]

2.1 Smart Augmentation in a Diagrammatic Way

It is possible to implement Smart Augmentation with one single network for network A or with several networks for network A. The diagrammatic process of Smart Augmentation is shown in its both variants in figure 1 and in figure 2.

In figure 1 the Smart Augmentation variant with just one single network for representing network A is shown. Network A gets several data samples I_1 until I_k from the same class as input. Their selection happens as the user wishes, e.g. randomly or by clustering. It creates an output sample Out_1, which is from the same class like the input samples. The loss function L_A calculates the loss for the created sample Out_1 by means of a sample from the same class from the dataset. Then a selection function selects the input for network B by means of the result of L_A, Out_1 and the selected sample from the dataset. Now network B can be trained with this sample to accomplish its actual specific task. By means of the output Out_2 of network B, the loss function L_B calculates the loss for network B. This will be the feedback for network A in its next iteration step. The total loss can be calculated with L_A and L_B. [4]

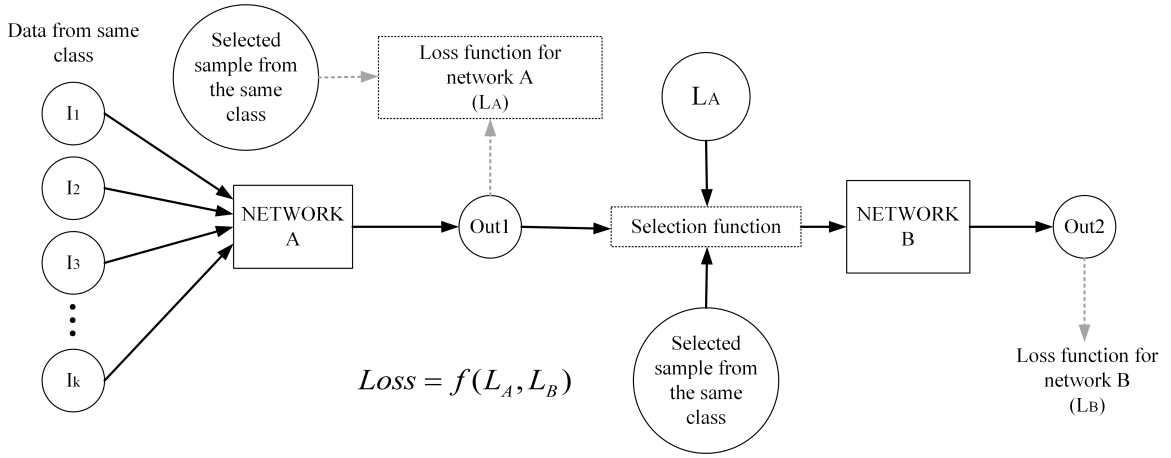

Figure 1: Smart Augmentation with just one network A [4]

In figure 2 the Smart Augmentation variant with more than one network for representing network A is shown. In the contrary to the version with just one network for representing network A, there is a class mapper in this version. It gets several data samples I_1 until I_k from the same class and their class label as input. The selection of the samples happens how by just having one network for representing network A. The class mapper forwards the samples to the right network, responsible for this class label. This network creates the output sample Out_1, which is from the same class as the input samples. From now on all steps are the same as by just having one network for representing network A, explained in the paragraph right above. [4]

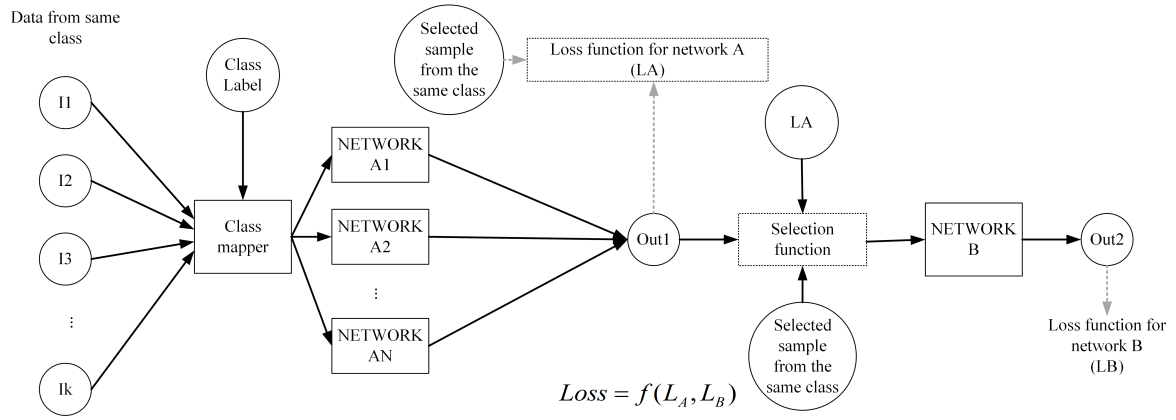

Figure 2: Smart Augmentation with more than one network for "network A" [4]

Using just one single network for representing network A has the advantage that only two networks have to be managed, network A and B. Using multiple networks for representing network A has the advantage that every network can learn its own class-specific augmentation strategies independent from the other networks, which may work well for one class, but not for all classes. [4]

2.2 Functioning of Network A

Network A is a neural network. The precondition for network A is that all images in the input dataset have the same shape, dimension and type. The selection of the input images for network A or for the class mapper is up to the user. It can be done e.g. randomly or by clustering. [4]

Network A gets two or more samples of the same class of data as input. It creates a new sample by intelligent blending of the features between the input images. The generated image approximates data from that class. The loss function (L_A) gets the output of network A and an image from the same class out of the data as input to ensure the output from network A is similar to other members of its class. Network A gets the loss from network B as feedback to adjust its intelligent blending strategy. So it converges to the best augmentations to train network B. [4]

The behavior of the above described network A can be implemented by a single network or by multiple networks. This will be explained in section 2.1. [4]

2.3 How the Augmentation Works

The augmentation step will be done by network A. Therefor it gets two or more samples from the same class in the dataset as input. By means of the loss of network B network A learns the best intelligent blending data augmentation strategy for training network B. With this, network A generates the best blended sample for the specific underlying problem handled by network B. For handling that task the quality of the generated sample is not important, it can be worse than the quality of the input samples. Important is that the generated sample contains features from the class of the input samples, but is not exactly one of its samples. So that network B is able to better generalize its network. [4]

In figure 3 an example for the augmentation of network A is shown. Network B is supposed to recognize genders. The two images on the right side are the input images for network A from the class "male" from the dataset and the image on the left side is the generated output of network A. The image on the left side is generated by a learned combination of the features of the two images on the right side. It represents features from the class "male" but it is not part of the underlying dataset. Its quality is worse than the quality of the two images on the right side but it contains features of the class "male". With this features this image helps network B to better generalize the concept of gender, to reduce overfitting and to increase its accuracy. [4]

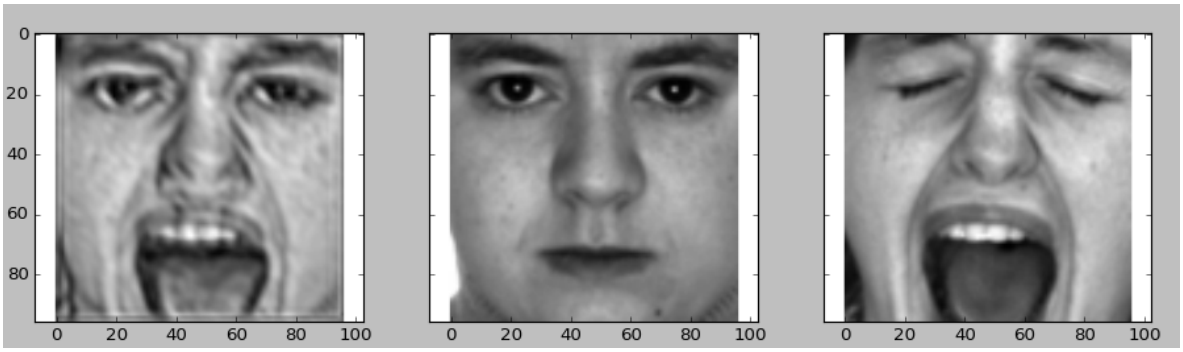

Figure 3: Example for input and output of network A [4]

3 Evaluation of Smart Augmentation

3.1 Used Datasets

For their experiments they used five datasets with different characteristics to evaluate the performance of their method. They have only measured the impact of their method, they have not compared their method with other methods. [4]

1. AR:

   Their first dataset is from the AR face database [5]. It contains 4000 frontal face images of female and male subjects. They converted all images to 96×96 grayscale. The pixel values were normalized between 0 and 1. The images were split into 70% training images, 20% validation images and 10% testing images. [4]

2. AR augmented:

   This dataset is the same as the above one [5] with the difference that it contains every combination of flipping, blurring and rotation with -5°, -2°, 0°, 2° and 5° around the center of the image. This results in 48360 images. [4]

3. FERET:

   Their third dataset is from the FERET dataset [6]. They used all FERET images with gender labels. All images were converted to 100×100 grayscale. The pixel values were normalized between 0 and 1. The images were split into 70% training images, 20% validation images and 10% testing images. [4]

4. Adience:

   Their fourth dataset is Adience. They converted all images to 100×100 grayscale. The pixel values were normalized between 0 and 1. The images were split into 70% training images, 20% validation images and 10% testing images. [4]

5. MIT:

   Their fifth dataset is from the MIT places dataset [8]. They used only images from the classes "Abbey" and "Airport". All images were rescaled to 256×256 with 3 color channels. The pixel values were normalized between 0 and 1. [4]

3.2 Experiments and Results

All experiments were performed for 1000 epochs and were compared then. They have made 30 experiments with different parameters by the 5 datasets which are explained in section 3.1 "Used Datasets". Therefor they have used several neural network architectures with varied parameters for network A and network B. [4]

The Smart Augmentation experiments with just one network for network A have resulted in reducing the rate of overfitting and increasing the accuracy significantly. It has improved from 83.52% to 88.46% at the FERET Dataset and from 70.02% to 76.06% at the Adience dataset. There was no noticeable pattern recognizable when varying the number of inputs for network A. [4]

The experiments with just using traditional augmentation methods have improved the accuracy a little bit from 88.15% to 89.08%. By adding Smart Augmentation to traditional augmentation methods the accuracy has improved further to 95.66%. From this it follows that Smart Augmentation can be used in combination with traditional augmentation strategies without negative consequences. [4]

The experiments with using two networks for network A were predominantly better than using just one network for network A. They have improved from 92.49% to 93.35% in the average. These networks were not better in every case, sometimes they have produced a network with the equal accuracy rate as the experiments with just one single network for representing the Augmentation Network. This happens when class-specific augmentation strategies have not existed. That supports the assumption that network A should consist of one network for every class to enable the learning of class-specific augmentation strategies. [4]

4 Discussion

Smart Augmentation by Lemley et al. [4] is a useful and reliable method to train a workable network out of not enough samples to train without augmentations. Moreover it increases the accuracy of a neural network and reduces its overfitting. But this method can only solve classification problems, it can not solve all kinds of problems [1]. But they have only evaluated their method on classification problems. Therefore it is not possible to come to a final assessment without implementing new experiments. So Smart Augmentation should just used blind if the problem is a classification problem. An example for non-classification problems that can be solved by neural networks are the regression problems.

A regression problem is to try to predict a value for an input based on previous information. Therefor it estimates a function that maps input to output. In the contrary to classification problems, it estimates an actual value and not just a class. An example for a regression problem is to estimate someones credit score based on information about the person like its past payments. [3]

Regression problems can be solves by neural networks but it is unknown with effect Smart Augmentation have on the trained networks. It is not clear whether the usage of Smart Augmentation can worsen the resulting neural network or not. This applies for all non-classification problems. So Smart Augmentation should not used blind if the kind of problem is unknown, it should be ponder whether the problem is a classification problem or not.

Their method is very useful for solving classification problems. It improves the resulting neural network compared to using no augmentation strategies. Classification problems make the majority of today's problems by exploitation of neural networks. As a consequence Smart Augmentation is not the sensational breakthrough for the robotic community but still an important one for increases the accuracy of a neural network and reducing its overfitting by classification problems.

The authors have used a good style of writing. It is easy to follow their lines of thinking. Their explanations are well understandable and easy to follow. The graphics are sensible chosen and support the understanding of their method. The rough functionality of Smart Augmentation is, by means of their graphics and explanations, easy to understand. But the exact functionality of their method in detail was not explained. An explanation for, for example, how to blend intelligent is not given in the paper. Hence it will unfortunately not be possible to easily reproduce their experimental results or make further additional experiments to deeper evaluate their method.

The presented experiments and their results enhance the functionality of their method to increase the accuracy of a neural network and to reduce its overfitting by solving classification problems by training neural networks. But it would have been nice to see the effect of solving non classification problems with and without using Smart Augmentation. It seems probable that Smart Augmentation can ruin the trained network for such a problem because it can happen that it generates samples that are part of other classes but classify them wrong. It would be interesting to see if this misclassified samples can ruin the whole network or if they occur too rare to essentially affect the resulting network.

Unfortunately, in their discussion they do not specify the kinds of problems which their method can solve well and on which kinds of problems their method possibly even produces worse results compared to using traditional augmentation strategies or even to using no augmentation strategies and to using classical training for neural networks. So it is not clear whether the usage of Smart Augmentation can worsen results on non-classification problems.

It can make the reader believe that Smart Augmentation works on every kind of problems well and it improves the result definitely. Apart from that they have a good critical discussion on their own method and experiments.

5 Conclusion

Lemley et al. suggest Smart Augmentation, a very useful method to solve classification problems by training deep neural networks. It increases their accuracy and reduces its overfitting. Smart Augmentation does this by finding the best augmentation strategy for a given dataset to train a deep neural network B. Therefor it uses a second network A which is upstream of network B. Network A creates new samples for training network B. Therefor it gets several input samples from the given dataset and creates new samples by considering the loss of the deep neural network B. This samples will be used for the training of the actual network B that is supposed to solve the desired task.

Furthermore they have performed several experiments to confirm the usefulness of their method. They have made 30 experiments on 5 different datasets in total. Unfortunately, they only have evaluated their method on classification problems. They have compared the accuracy of the trained network by using their method with using traditional augmentation strategies and with using no augmentation strategies. They have not evaluated the effect of their method on non-classification problems (e.g. regression problems), hence its effect is unknown.

Smart Augmentation is well suitable for improving the network training for classification problems. It does not matter whether the available dataset is already extended with augmented samples which was created with other augmentation strategies or not. Using Smart Augmentation always improves the resulting network compared with using normal network training.

References

[1] Shabab Bazrafkan, Shejin Thavalengal, and Peter Corcoran. An end to end deep neural network for iris segmentation in unconstrained scenarios. *Neural Networks*, 106:79–95, 2018.

[2] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[3] Clustering Classification, Regression. https://campus.datacamp.com/courses/introduction-to-machine-learning-with-r/chapter-1-what-is-machine-learning?ex=6. Last accessed 13 January 2019.

[4] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869, 2017.

[5] Aleix M Martinez. The ar face database. *CVC Technical Report24*, 1998.

[6] P Jonathon Phillips, Hyeonjoon Moon, Syed A Rizvi, and Patrick J Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Transactions on pattern analysis and machine intelligence*, 22(10):1090–1104, 2000.

[7] Patrice Y Simard, Dave Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *null*, page 958. IEEE, 2003.

[8] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.