Robustness of Deep Learning Architectures with Respect to Training Data Variation

Andreas Bartschat[1], Tim Unger[1], Tim Scherr[1], Johannes Stegmaier[2], Ralf Mikut[1], Markus Reischl[1]

[1] Institute for Automation and Applied Informatics,
Karlsruhe Institute of Technology, Germany
[2] Institute of Imaging and Computer Vision
RWTH Aachen University, Aachen, Germany
E-Mail: andreas.bartschat@kit.edu

1 Motivation

The use of deep neural networks (DNN) revolutionized machine learning tasks like image classification and segmentation [6]. However, DNNs are black box models developed with the aim to generalize a given training set and thus to process unknown data. Due to the high amount of model parameters, it is yet impossible to fully understand the decision making process. The identification of parameters with poor generalization is hard and manual improvement is impossible. Thus, robustness is subject of consideration.

So called adversarial attacks evaluate robustness of trained DNNs by modifying data examples to enforce misclassification [2, 5]. In image classification, the required changes prove to be very subtle and original and modified examples differ only in minimal noise, which would be easily ignored by the human eye [9]. This leads to the question of the robustness of DNN architectures against modifications like noise or blur. Dodge et al. [1] evaluated modifications of the test images. However, the influence of modifications in the training samples remains unclear.

In this contribution we investigate the influence of modifications in the training samples to small DNN architectures with respect to their robustness against modifications in test samples. Furthermore, by introducing three different ar-

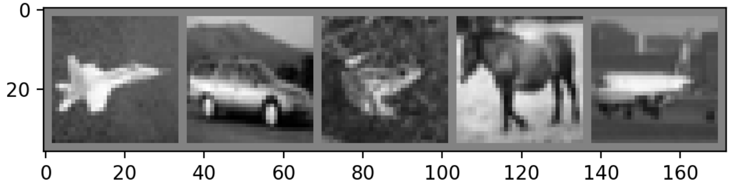

Figure 1: Five CIFAR-10 example images. Classes from left to right: plane, car, frog, horse and plane

chitectures inspired by the popular architectures ResNet [3], Inception-v3 [8] and ResNeXt [10], we investigate the influence of these architectures to their ability to generalize well against modifications. Furthermore, by modification of the training data we investigate the influence of noise and blur in the training process to the robustness of the trained model.

2 Methods

2.1 Data Set

For training and evaluation the CIFAR-10[2] data set is used [4]. It consists of 60.000 color (RGB) images with a resolution of 32×32 pixel and a pixel depth of 24 bit (8 bit per color). The data set comprises 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The images are evenly distributed over the classes and a split of 50.000 images is used for training and 10.000 for validation. Figure 1 shows five example images of the data set.

2.2 Network Structures

Many different network architectures and modules have been proposed over the last years. Most of these architectures consist of multiple stacked modules with small variations. A module describes a combination of different layers such as

[2] `https://www.cs.toronto.edu/~kriz/cifar.html`

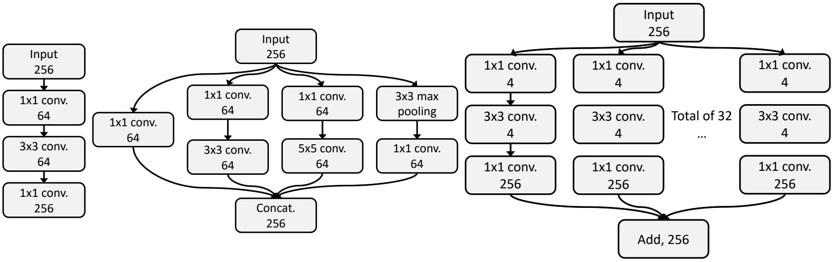

Figure 2: Modules for network architectures: Left: ResNet, middle: Inception, right: ResNeXt. The second line defines the number of feature maps produced by the module.

convolutional, activation, pooling and batch normalization layers. Modules vary in their parameters as well as in the arrangement of layers. Here, we compare three modules based on the popular architectures ResNet [3], Inception-v3 [8] and ResNeXt [10].

To allow a comparison of these modules, we use the same pre- and post-processing and the same amount of modules. The pre-processing consists of two subsequent convolution modules (with a convolutional layer, a rectified linear unit activation function and a batch normalization). After these modules, three stacked modules of the proposed architectures follow. All modules use a residual connection [3] and are followed by a max pooling for downsampling [3]. The post-processing consists of another convolution module, a dense layer with ten output neurons, one for each class of the CIFAR-10 data set and a softmax function. All architectures are trained with the stochastic gradient descent algorithm based on a categorical cross entropy loss of for 90 epochs.

The first module is the *ResNet* module, which is also called bottleneck module [3]. It downsamples the number of feature maps with a 1×1 convolution module, followed by a 3×3 convolution module for feature extraction and an upsampling of the feature maps by a 1×1 convolution module. An example can be seen in Figure 2, left. This module has shown to perform at least similar to a 3×3 convolution module while using less parameters. The given example in Figure 2, left uses 64×256 parameters for the first, $64 \times (3 \times 3 \times 64)$ parameters for the second and 256×64 for the third module, resulting in a total of 69.632 trained parameters. In contrast, a single 3×3 convolution requires $256 \times (3 \times 3 \times 256) = 589.824$ parameters.

The *Inception* module is the eponym of the Inception network. This module exists in several versions which can be combined to several versions of an Inception network. For simplicity we only investigated the module of the Inception-v3 network [8]. The module is visualized in Figure 2, middle and consists of four parallel paths with convolution modules of size 1×1, 3×3 and 5×5 as well as a max pooling layer. The idea behind the concept is that all paths can focus on features of a certain size and all results are concatenated at the output of the module. Since all of these paths work on the complete input, also the parameter count for such a module is higher. The shown module in the middle of Figure 2 contains 192.512 parameters.

The *ResNeXt* module is based on the ResNet module and introduces parallel paths. Instead of a single bottleneck, multiple bottlenecks with decreased depth are used. In the example shown in Figure 2 right, a total of 32 parallel paths are shown, each consisting of 256×4, $4 \times (3 \times 3 \times 4)$ and 256×4 parameters. This results in a total of 70.144 trainable parameters. The idea behind this module is that each bottleneck can focus on different features and each bottleneck can become an expert for different tasks.

2.3 Evaluation

To evaluate robustness against adversarial attacks, six models with noisy and six models for blurry inputs are trained for all three architectures. All models are evaluated in terms of accuracy and in terms of the minimal change in pixels to enforce a wrong classification of an image.

For the noisy input, a model without noise and six models with a noise level of $\sigma = \{0.01,\ 0.05,\ 0.1,\ 0.5,\ 1,\ 5\}$. σ specifies additive Gaussian noise for each pixel originating from a normal distribution with zero mean and the given σ. The noise is added to the normalized image with a range of $[-1, 1]$. Thus, also small changes are kept in the image and not discarded due to rounding. An example of two original images and the same images with noise are shown in Figure 3.

For the blurry input, the parameters are exactly the same. Here, the image is blurred with a Gaussian filter with standard deviation sigma.

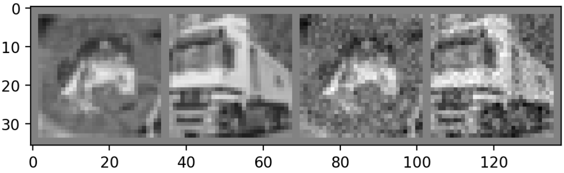

Figure 3: On the left: a frog and a truck from the original data set. On the right: the same frog and truck with added noise ($\sigma = 1$).

3 Results

In order to evaluate the influence of noise and blur to the training data, each trained model is evaluated either with all levels of noise or blur. Furthermore, for each model an adversarial attack is computed for 128 images of the original data set. This attack computes a minimal change of an image given a model by applying the backpropagation algorithm to modify the input image and not the model. Based on these gradients a minimal change to the image can be applied to enforce a wrong classification.

The results for the noisy inputs can be seen in Figure 4. The x-axis shows the noise level of the training set and is labeled by the σ of the models' training data. The plotted lines show the accuracy on the evaluation sets, where each model is evaluated with all noise levels.

For all architectures, the accuracy and their behavior for the different noise levels are similar. The data set with a noise level of $\sigma = 0.01$ behaves like the data set without noise for the training as well as the evaluation. Starting with a model trained on the noise level $\sigma = 0.1$, the evaluation set with the same noise level provides the best accuracy and the overall accuracy of the model starts to drop. Thus, the noise levels of the evaluation sets provide the best accuracy if the model is trained with a similar noise level. These observations are true for all architectures, i.e., none of the architectures generalizes particularly well against noise. Given these results, a small level of noise has no impact to noise-free images but the accuracy for a small level of noise increases. Furthermore, the results indicate that the model should be trained at least with the amount of noise that is expected in the application data.

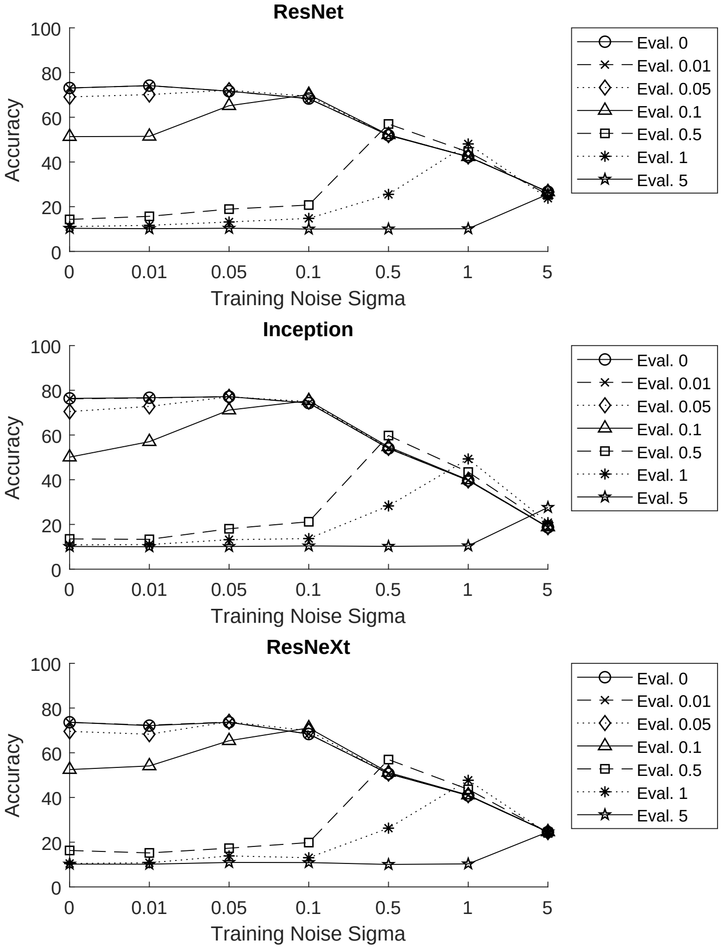

Figure 4: Evaluation of the influence of different noise levels for training (*x*-axis) on the performance of different noise levels for evaluation data sets (lines). The number behind the evaluation data sets is the σ. Three architectures are shown, the top one is based on ResNet, the middle one on Inception and the bottom one on ResNeXt.

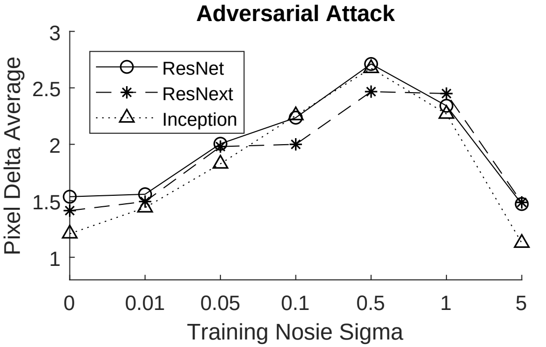

Figure 5: The average amount of required change in pixels to create an adversarial image for different architectures. The *x*-axis shows the noise level of the used training set.

The evaluation of the adversarial attacks in Figure 5 shows the noise level of the models' training set on the *x*-axis and the average delta in pixels required to change the classification of an example image on the *y*-axis. This means that on average every pixel of an image has to be modified by the given delta to change the predicted class of that image. Again, the architectures behave similar and all architectures can be tricked by adversarial attacks with a similar change in pixel values.

It is notable that models trained with noisy data are slightly more robust against adversarial attacks, since the average change in pixels must be higher. For models with a very high level of noise like $\sigma = 1$ or $\sigma = 5$ the required amount of change in the pixels starts to drop. However, these models also have a significantly lower accuracy and do not generalize well to new examples.

Overall, a small level of noise increases the robustness against adversarial attacks and the performance for noise in the evaluation data. Thus, in this case a training with $\sigma = 0.1$ seems to be beneficial for the model.

To put the results on adversarial attacks in perspective, even a change of 2.5 of the pixel value is only a change of 1% in the 8 bit images. Thus, all of the models and architectures are easily tricked by adversarial examples and none of the architectures generalizes well against such attacks.

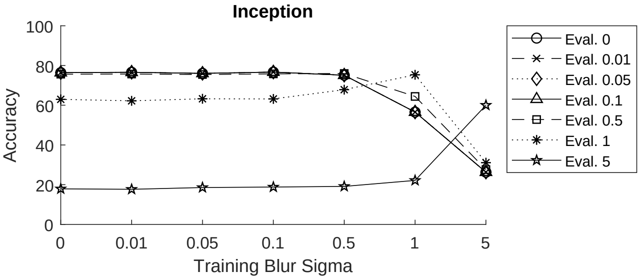

Figure 6: Evaluation of the influence of different blur levels for training (*x*-axis) on the performance of different noise level for evaluation. The figure shows the results for the Inception architecture, the results for ResNet and ResNeXt are similar (data not shown).

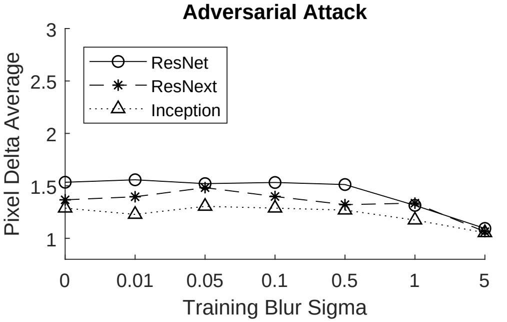

Figure 7: The average amount of required change in pixels to create an adversarial image for different architectures. The *x*-axis shows the blur level of the used training set.

The setup for the evaluation of blurry images is similar to the setup with noise (σ defines the size of a Gaussian kernel and thus the extend of blur introduced in the images). Since the results for the different architectures are similar, Figure 6 only shows the results of the best performing models, the Inception models. Up to a blur level of $\sigma = 0.5$ the models accuracy remains unchanged and starts to drop with an increase in blur.

Figure 7 shows the evaluation of adversarial attacks on models with different blur levels. The results show that training with blurry images has no effect on the amount of change that must be introduced to the pixels to create an adversarial example and again all architectures behave similar. For high blur levels of $\sigma = 1$ and $\sigma = 5$ the required change even starts to drop, indicating that models with low accuracy are more easily tricked. Overall, the training with blurry images does not have an effect for small blur levels and on adversarial attacks. Thus, it is not beneficial for this use-case.

4 Conclusions

This work investigated the influence of noise and blur to the accuracy and robustness of convolutional neural networks against adversarial attacks. For the training and evaluation the CIFAR-10 image data set was used and modules from three popular architectures are compared. To allow an easy comparison, the pre- and post-processing of the architectures used here are the same and only the intermediate layers are designed according to the proposed modules in the ResNet, Inception and ResNeXt architectures.

All models were trained several times with different levels of noise and blur. In the evaluation, the accuracy for different levels of noise and blur were evaluated as well as the amount of change in the pixels to create an example for an adversarial attack, meaning how much the pixels had to be changed on average to enforce a wrong classification.

The results show that training with a small noise level can increase the robustness against adversarial attacks and is beneficial for the performance of the model on noisy data. The introduction of blur however, has no beneficial effect on adversarial attacks up to a point where the models performance decreases significantly. Regarding the performance of the proposed modules no significant performance difference was observed.

Since augmentations of the training set are known to improve the model performance [7], also other augmentations like rotations and other transformations can be investigated regarding their effects on the robustness.

References

[1] S. Dodge and L. Karam. Understanding how Image Quality Affects Deep Neural Networks. In *2016 Eighth International Conference on Quality of Multimedia Experience*, pp. 1–6. IEEE, 2016.

[2] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[4] A. Krizhevsky and G. Hinton. Learning Multiple Layers of Features from Tiny Images. Technical Report, Citeseer, 2009.

[5] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial Examples in the Physical World. *arXiv preprint arXiv:1607.02533*, 2016.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3), pp. 211–252, 2015.

[7] T. Scherr, A. Bartschat, M. Reischl, J. Stegmaier, and R. Mikut. Best Practices in Deep Learning-Based Segmentation of Microscopy Images. In *Proc., 28. Workshop Computational Intelligence, Dortmund*, 2018.

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing Properties of Neural Networks. *arXiv preprint arXiv:1312.6199*, 2013.

[10] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated Residual Transformations for Deep Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5987–5995, IEEE, 2017.