


An Evaluation of VGG16 Binary Classifier Deep Neural Network for Noise and Blur Corrupted Images

 Devrim Akgün¹

¹Sakarya University, Faculty of Computer and Information Sciences, dakgun@sakarya.edu.tr;

Received 22 April 2020; Revised 30 November 2020; Accepted 02 December 2020; Published online 30 December 2020

Abstract

Deep learning networks has become an important tool for image classification applications. Distortions on images may cause the performance of a classifier to decrease significantly. In the present paper, a comparative investigation for binary classification performance of VGG16 network under corrupted inputs has been presented. For this purpose, images corrupted at various levels and fixed levels with Gaussian noise, Salt and Pepper noise and blur effect were used for testing. Convolutional layers of the VGG16 were frozen except the last three convolutional layers and a dense layer for binary classification was added. According to experimental results, as the effect of distortion is increased, performance of the deep learning classifier drops significantly. In the case of augmented training with distortion effects, the results were improved significantly.

Keywords: deep neural network, VGG16, pretrained networks, fine tuning, augmentation, corrupted images

VGG16 İkili Sınıflandırıcı Derin Sinir Ağının Gürültülü ve Bulanık Görüntüler için Değerlendirilmesi

Öz

Derin öğrenme ağları, görüntü sınıflandırma uygulamaları için önemli bir araç haline gelmiştir. Görüntülerdeki bozulmalar, sınıflandırıcının performansının önemli ölçüde düşmesine neden olabilir. Bu makalede, bozuk girişler altında VGG16 ağının ikili sınıflandırma performansı için karşılaştırmalı bir araştırma sunulmuştur. Bu amaçla, çeşitli seviyelerde bozulmuş görüntüler ve Gauss gürültüsü, Tuz ve Biber gürültüsü ve bulanıklık etkisi ile sabit seviyelerde görüntüler test için kullanılmıştır. VGG16'nın evrişimli katmanları, son üç evrişimli katman hariç dondurulmuştur ve ikili sınıflandırma için yoğun bir katman eklenmiştir. Deneysel sonuçlara göre, bozulmanın etkisi arttıkça, derin öğrenme sınıflandırıcısının performansı önemli ölçüde düşmektedir. Bozulma etkilerini içeren artırılmış eğitim durumunda, sonuçlar önemli ölçüde iyileştirilmiştir.

Anahtar Kelimeler: derin sinir ağı, VGG16, önceden eğitilmiş ağlar, ince ayar, zenginleştirme, bozuk görüntüler

1. Introduction

Deep learning based methods have been used widely in various fields such as computer vision[1], natural language processing [2], audio signal processing [3], and medical diagnosis [4]. This is mainly the result of availability of larger datasets, developing GPU technology, frameworks and toolkits such as Keras [5], Tensorflow [6], CNTK [7] and Theano [8]. One of the important subfields of deep learning is the Convolutional Neural Networks (ConvNets) which were initially used for handwritten number recognition [9]. Deep convolutional networks are successful in image classification applications due to their success in generalization. On the other hand training such networks requires datasets large enough to exemplify possible cases of classified images. Although increasing the size of the dataset, increases the generalization capability of ConvNets, it is not easy to find required number of images that provides the required generality. One of the approaches to increase to generality to cover these effects is to apply augmentation on images [10]. Examples usually include synthetic effects such as shift, zoom, blur, contrast, illumination, flip or shear operations. Especially including naturally occurring common

corruptions such as blur effect, Gaussian noise or Salt and Pepper noise effects in datasets gains importance for increasing the accuracy of the deep learning classifiers. An input image with these corruptions can be misclassified by convolutional neural network if these aren't considered during training. In literature, various corruptions were studied by numerous authors to improve naturally occurring corruptions. Dodge and Karam considered five types of quality distortions: blur, Gaussian noise together with contrast, JPEG, and JPEG2000 compression [11]. In another study, they considered blur and Gaussian noise for improving the performance of ConvNets and compared with human detection accuracy [12]. Vasiljevic et al. improved the performance of ConvNets by fine-tuning for blur degradation [13]. Yin et al. studied the relations between frequency of a various corruption types and network performance for data augmentation [14]. Rusak et al. used data augmentation with Gaussian and Speckle noise and improved the performance of ResNet50 against corruptions on ImageNet-C [15]. Kamann et al. realized detailed evaluations for semantic segmentation models regarding realistic image corruptions [16].

In the presented paper a comparative inspection for binary classification performance of fine-tuned VGG16 deep neural network under corruption effects which may occur naturally. For this purpose, the pretrained weights on ImageNet were used for the first four block of VGG16. Fifth block, where three successive convolutional layers and a dense layer for binary classification were trained using Cat vs. Dog dataset from Kaggle competition. Performance of the binary classifier were measured for various levels of Gaussian noise, Salt and Pepper noise and blur effects in datasets. Binary classifier were trained using augmentations with Gaussian noise, Salt and Pepper noise and blur effects to see the improvements in accuracy results. In the following section, background information about ConvNets and VGG16 are given. In section three, properties of fine-tuned VGG16 as binary Classifier are given. In the fourth section comparative experimental results are given. Finally the results of the study is summarized.

2. Background

Convolutional neural networks provide a good means for extracting image features for deep learning classification applications. Due to its success, in the past several years ConvNets have frequently been used in sequence processing applications as well as computer vision problems. ConvNets usually consist of a convolutional layer, pooling layer, and fully connected layer as shown by Figure 1. There can be a number of convolutional layers according to complexity of the problem and pooling layers between the convolutional layers. Convolutional layers usually extracts image features and usually dense layers follow convolutional layers for classifying the selected features. Hence, the last layer of the network usually is a softmax layer or a sigmoid layer. In practice convolutional layers are usually implemented for one dimensional (1D), two dimensional (2D) and three dimensional (3D). ConvNets for computer vision problems are usually implemented using 2D convolutional layers.

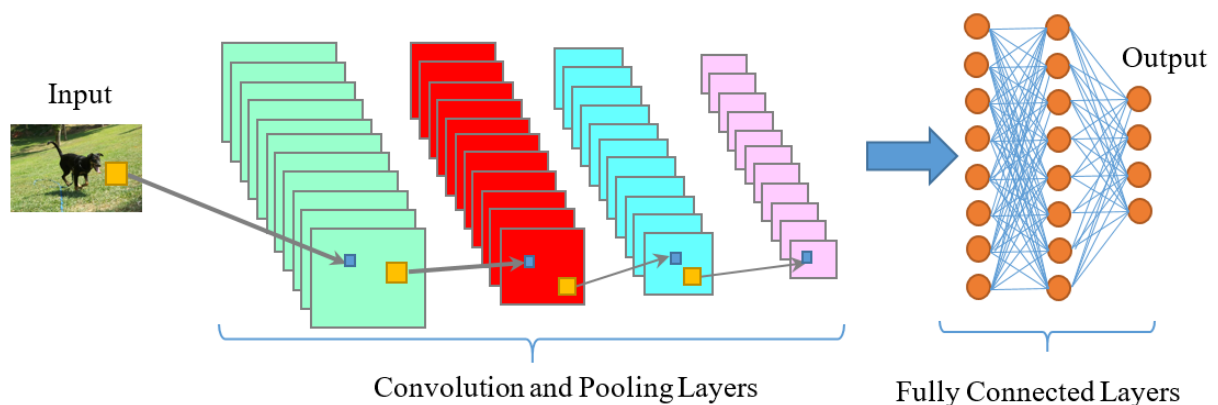


Figure 1 Basic idea of ConvNet architecture

A more detailed illustration of a convolutional layer of ConvNet is given by Figure 1. A 2D convolutional layer apply a number of trainable kernels to the input images. The number of outputs

depends on the number of kernels and they select some features of input image according to kernel weights. Each of the output images are then applied to pooling operation or directly sent to another convolutional layer. Kernel weights and biases are the trainable parameters of the convolutional part of the ConvNet. Training ConvNets for large set of images usually demands intense computational power and it sometimes takes days or weeks to train a deep learning model even on a powerful computer with GPU support. As an alternative to training whole deep learning network, pre-trained networks are of importance in deep learning applications. VGG16 model was developed by K. Simonyan and A. Zisserman and it was submitted to Large Scale Visual Recognition Challenge 2014 (ILSVRC2014) [17]. VGG 16 is a pretrained network and it has 13 convolutional layer and dense layers for multiclass classification of 1000 classes as shown by Figure 2. This model was fine-tuned as binary classifier and used in the experimental evaluations.

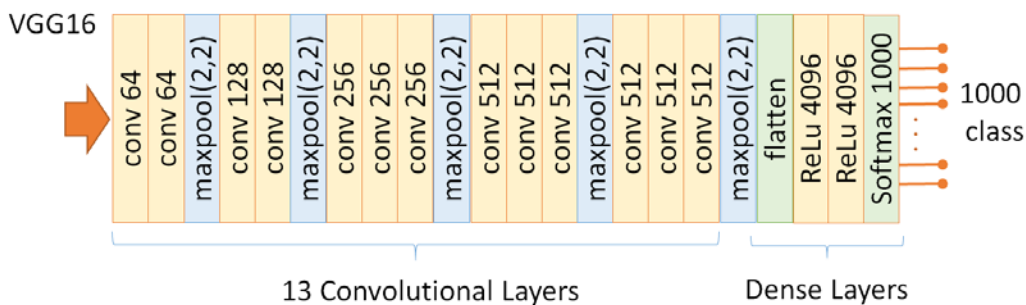


Figure 2 Layers of VGG16 model

3. Binary Classifier and Experimental Setup

In the present study, VGG16 deep neural network were trained as binary classifiers. For this purpose, fully connected layers of both models are replaced with a sigmoid layer that has one unit for binary classification. All ConvNets layers were frozen except the last three ConvNets layers of VGG16. Figure 3 shows the block diagram of model used in the study. Dense layer contains a sigmoid layer for binary classification. Table 1 summarizes the trainable and non-trainable parameters of the model.

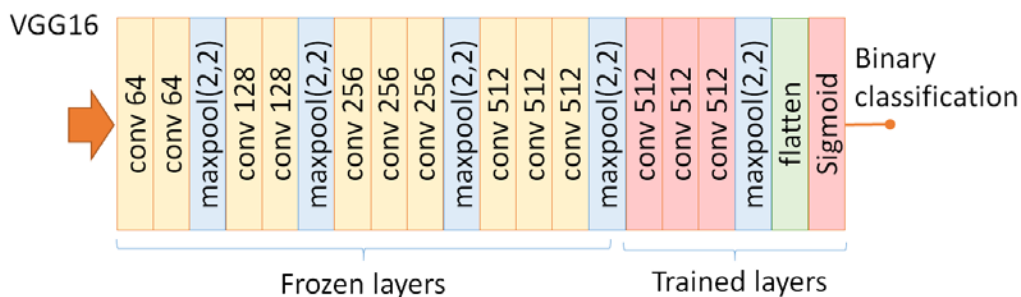


Figure 3 Binary classifier using VGG16 the last ConvNets

Table 1 Binary classifier model parameters

Trainable parameters	10,290,945
Non-trainable parameters	7,635,264
Total parameters	17,926,209
Number of Trained Conv2D layers	3
Number of Frozen Conv2D layers	10

Models were trained using Dogs vs. Cats classification dataset from Kaggle for training deep learning models. In the experiments, 4000 images were used for training and 2000 images were used for validation. Data Augmentation using ImageDataGenerator was applied where rotation range is set to 30% and width shift range, height shift range, shear range and zoom range are all set to 20%. Horizontal and vertical flip options are all set to true. The models were trained using 150 epoch and 200 steps per

epochs. Validation during training were realized using 100 steps. In each case batch size is set to 20. In order to train a reference model, no corruption augmentation was done apart from the augmentations described above. Figure 4 shows the accuracy and validation results for the trained binary classifier and the losses in each epoch for training and validation. It provided about 97% validation accuracy for the images with no corruption augmentation.

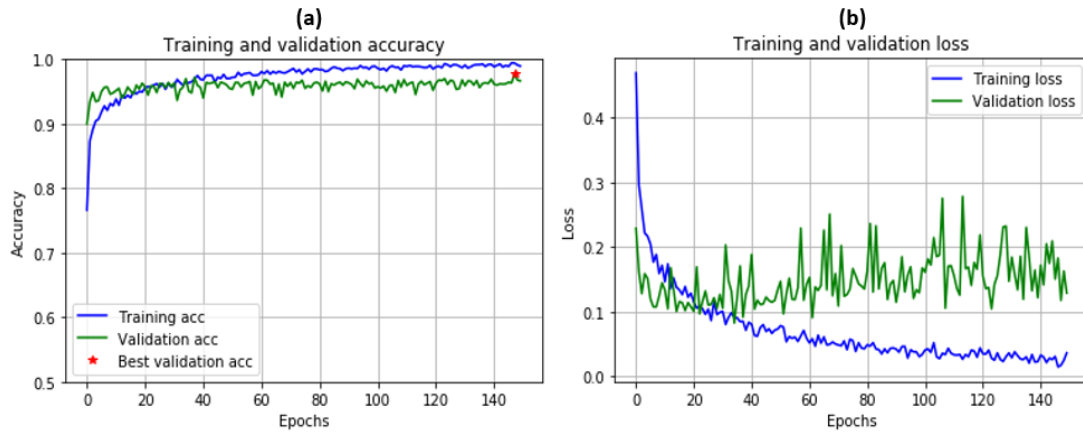


Figure 4 Accuracy and validation results for VGG16 binary classifier without augmentation for corruptions

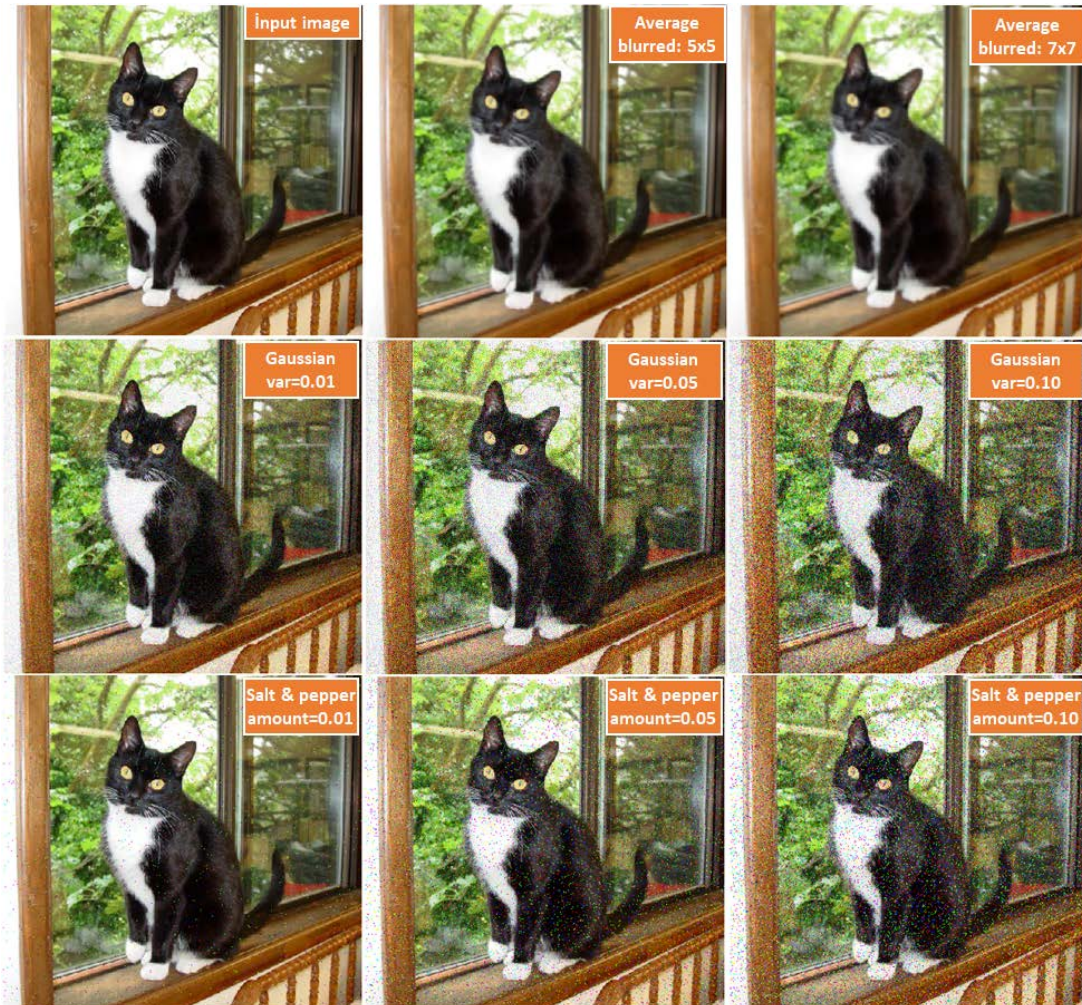


Figure 5 An example input image and blur effects, Gaussian noise and Salt and Pepper noise applied to it for the illustration

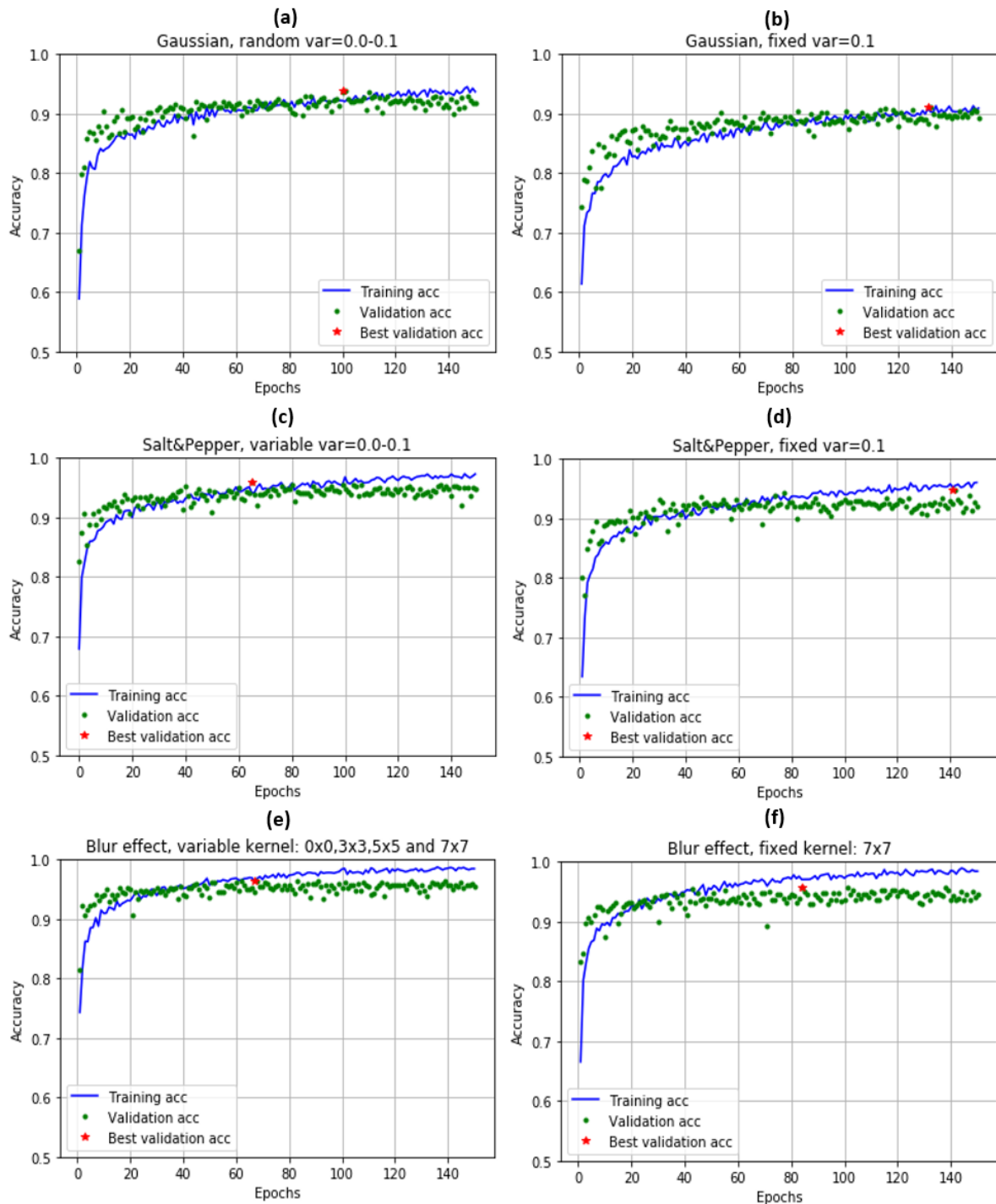


Figure 6 Training and validation accuracies: a) variable Gaussian noise corruption, b) fixed Gaussian noise corruption c) variable Salt and Pepper noise corruption, d) fixed Salt and Pepper noise corruption e) variable Blur corruption, f) fixed Blur corruption

4. Experimental results

Figure 5 shows an example input image and example corrupted images of various qualities were synthesized using the original input. These corruptions include blur effect, Gaussian noise and Salt and Pepper noise examples used in augmentation during training and testing. Summary of training and validation accuracies for all the cases used in the experiments are shown by Figure 6. Figure 6a and 6b show the training with variable and fixed Gaussian noise augmentations respectively. Gaussian noise level for variable corruption was varied between 0 and 0.1 using random numbers. The same is also true

for Salt and Pepper noise where Figure 6c and 6d show the training with variable and fixed noise augmentations respectively. Figure 6e and Figure 7f show the results for augmentation with variable and fixed Blur effects. Filter kernel size was selected as 7x7 for fixed augmentation and filter kernel size was selected randomly as no filtering, 3x3, 5x5 and 7x7 for variable implementation.

Table 2 show the validation accuracies for the cases where the model trained with and without Gaussian noise corruption. The results for model where augmentation was done without Gaussian noise corruption shows significant drops approximately to 53% as the Gaussian noise variance level increased to 0.1. This is increased approximately to 90% when the model trained with fixed Gaussian noise corruption where variance level is set to 0. However the results for images with corruption drops approximately to 72%. The best results were obtained for the model trained with variable Gaussian noise corruption where variance level is selected randomly. For the cases where Salt and Pepper noise corruption was used similar behavior was observed as shown by Table 3. The best results for this case were also obtained using the model trained with corruptions with random levels. Table 3 shows the results for input images corrupted with blur effects using average filter reduces the accuracy as the level of blur using average filter is increased. For the case without augmentation, the accuracy drops from 97% to 96% for 3x3 blur. Although this can be assumed to be small decrease, the accuracy further drops approximately to 93% for 5x5 blur and 87% for 7x7 blur. These were improved when fixed blur corruption using 7x7 window included in augmentation during training. In general, augmentation with fixed or random blur corruptions produced close results and they provided better accuracy over training with no blur augmentation.

Table 2 Validation accuracies with and without Gaussian noise corruption

Gaussian noise corruption variance	Training without Gaussian noise corruption	Augmentation with fixed Gaussian noise (var=0.1)	Augmentation with random Gaussian noise (var=0.0-0.1)
0.000	0.9710	0.7180	0.9647
0.005	0.9457	0.7848	0.9546
0.010	0.9147	0.8196	0.9490
0.050	0.6538	0.8995	0.9263
0.100	0.5388	0.9027	0.8988
Mean	0.8048	0.8249	0.9387

Table 3 Validation accuracies with and without Salt and Pepper noise corruption

Salt and Pepper noise corruption variance	Training without Salt and Pepper noise corruption	Augmentation with fixed Salt and Pepper noise (var=0.1)	Augmentation with random Salt and Pepper noise (var=0.0-0.1)
0.000	0.9710	0.9026	0.9734
0.005	0.9647	0.9153	0.9668
0.010	0.9622	0.9165	0.9645
0.050	0.8775	0.9382	0.9501
0.100	0.7145	0.9361	0.9343
Mean	0.89798	0.9217	0.9578

Table 4 Validation accuracies with and without Blur corruption

Blur corruption size (average filter)	Training without Blur corruption	Augmentation with fixed Blur corruption (average filter: 7x7)	Augmentation with random Blur corruption (average filter: no filtering, 3x3, 5x5 and 7x7)
no filtering	0.9710	0.9480	0.9630
3x3	0.9607	0.9555	0.9577
5x5	0.9337	0.9567	0.9492
7x7	0.8662	0.9545	0.9335
Mean	0.9329	0.9536	0.9508

5. Conclusions

Performance of deep learning based classifiers may reduce significantly under corrupted input conditions. These can be compensated with augmentation using corrupted images during training to some extent. In the presented paper the performance of binary classifier using VGG16 deep neural network under corruption effects was investigated. In the experiments, the effects of Gaussian noise, Salt and Pepper noise and blur with average filter were investigated comparatively. According to the results, augmentation with the Gaussian noise, Salt and Pepper noise and blur effect improves the resistance to these types of corruptions. Experiments in this paper were carried out for binary classifier using VGG16 trained with ImageNet. Last three layers and a dense layer were trained in the experiments. Performance of binary classifier trained with augmentation without no corruption deteriorates significantly as Gaussian noise corruption at the input image is increased. In general, augmentation with fixed Gaussian noise produced worse results than the variable one. Similar behavior was also observed for Salt and Pepper noise corruption. Corruption with blur effects on input images using 3x3, 5x5 and 7x7 average filters have significant effect on the accuracy. Training using augmentation with random blur corruption showed better performance than fixed corruption for no blur effect or blur effect 3x3 on the input images. On the other hand training with fixed blur effect produced better results for blur effects 5x5 and 7x7 on the input images. In general, augmented training with corruption effects improved the results for corrupted images which may occur naturally. Presented comparative results here give insights into a binary classifier under corruption effects which may occur naturally. Although the experiments here were implemented for VGG16 these can be repeated for other models. Also these results can be extended for other corruption types.

References

- [1] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Computational Intelligence and Neuroscience*, vol. 2018. Hindawi Limited, 2018.
- [2] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, 2018.
- [3] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, and T. Sainath, "Deep learning for audio signal processing," *IEEE J. Sel. Top. Signal Process.*, vol. 13, no. 2, pp. 206–219, 2019.
- [4] F. Altaf, S. M. S. Islam, N. Akhtar, and N. K. Janjua, "Going Deep in Medical Image Analysis: Concepts, Methods, Challenges, and Future Directions," *IEEE Access*, vol. 7, pp. 99540–99572, 2019.
- [5] F. Chollet, "Keras," *GitHub repository*. GitHub, 2015.
- [6] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv Prepr. arXiv1603.04467*, 2016.
- [7] D. Yu *et al.*, "An introduction to computational networks and the computational network toolkit," 2014.
- [8] R. Al-Rfou *et al.*, "Theano: A {Python} framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.0, May 2016.

- [9] Y. LeCun *et al.*, “Handwritten digit recognition with a back-propagation network,” in *Advances in neural information processing systems*, 1990, pp. 396–404.
- [10] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J. Big Data*, vol. 6, no. 1, p. 60, 2019.
- [11] S. Dodge and L. Karam, “Understanding how image quality affects deep neural networks,” in *2016 eighth international conference on quality of multimedia experience (QoMEX)*, 2016, pp. 1–6.
- [12] S. Dodge and L. Karam, “A study and comparison of human and deep learning recognition performance under visual distortions,” in *2017 26th international conference on computer communication and networks (ICCCN)*, 2017, pp. 1–7.
- [13] I. Vasiljevic, A. Chakrabarti, and G. Shakhnarovich, “Examining the impact of blur on recognition by convolutional networks,” *arXiv Prepr. arXiv1611.05760*, 2016.
- [14] D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, “A fourier perspective on model robustness in computer vision,” in *Advances in Neural Information Processing Systems*, 2019, pp. 13255–13265.
- [15] E. Rusak *et al.*, “Increasing the robustness of DNNs against image corruptions by playing the Game of Noise,” *arXiv Prepr. arXiv2001.06057*, 2020.
- [16] C. Kamann and C. Rother, “Benchmarking the robustness of semantic segmentation models with respect to common corruptions,” *Int. J. Comput. Vis.*, pp. 1–22, 2020.
- [17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv Prepr. arXiv1409.1556*, 2014.