CORRUPTION DEPTH: ANALYSIS OF DNN DEPTH FOR MISCLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Many large and complex deep neural networks have been shown to provide higher accuracy. However, very little is known about the relationship between the complexity of the input data along with the type of noise and the depth needed for correct classification. Existing studies do not address the issue of common corruption adequately, especially in understanding what impact these corruptions leave on the individual part of a deep neural network. Therefore, we can safely assume that the classification (or misclassification) might be happening at a particular layer(s) of a network that accumulates to draw a final correct or incorrect prediction. In this paper, we introduce a novel concept called **corruption depth**, which identifies the location of the network layer/depth until the misclassification persists. We assert that the identification of such layers will help in better design of the network by pruning certain layers in comparison to the purification of the entire network which is computationally heavy to do. Through our extensive experiments, we present a coherent study in comparison to the existing studies which are diverse in understanding the processing of examples through the network. Our approach also illustrates different philosophies of example memorization and a one-dimensional view of sample or query difficulty. We believe that the understanding of the corruption depth can **open a new dimension of model explainability**, where in place of just visualizing the attention map, the classification progress can be seen throughout the network.

1 INTRODUCTION

Deep networks are composed of several layers which learn different kinds of features that vary from the beginning to the end of the network. For instance, generally initial few layers learn the low-level image features such as edges and lines, middle layers learn the shape and structure of the object, and deeper layers learn the high-level features which form the entire example/sample Keshari et al. (2018); Zeiler & Fergus (2014). While these networks yield state-of-the-art performances on cleaner and good quality images; they are found to be vulnerable to corrupted images Hendrycks & Dietterich (2019). The corruption often can be caused by a different type of noise in the input. Therefore an understanding of how the corrupted data passes through these feature processing pipelines is essential. By understanding the impact of the corruption in data, it will help in reducing the additional computational load needed to perform the image denoising from complex architectures such as generative networks Lee et al. (2017); Song et al. (2017) or building a defense where we might not be required to train the entire new network for attack detection Agarwal et al. (2021a;b). Recent studies aim to understand the broad behavior of the networks from the point of view of geometry and Eigenvalues of the Hessian matrix Ghorbani et al. (2019); Yao et al. (2020), generalization effectiveness Jiang et al. (2019); Unterthiner et al. (2020), and stochastic optimization Smith et al. (2021). These existing studies aim to find confidence in the prediction when the images are not part of a training set Jiang et al. (2021) and how complex the clean data is Agarwal et al. (2020b); Baldock et al. (2021); Hooker et al. (2019); Toneva et al. (2018). Therefore, they only understand the average behavior of the networks and are not effective in general from a robustness point of view. Based on the example characteristics, these studies see the understanding of deep neural networks from two views: (i) statistical and (ii) learning. However, the behavior of the deep networks in presence of the corruption in the data is not considered and only presents a one-dimensional understanding Carlini et al. (2019). These existing studies encapsulate that the easier examples exist in domains such as

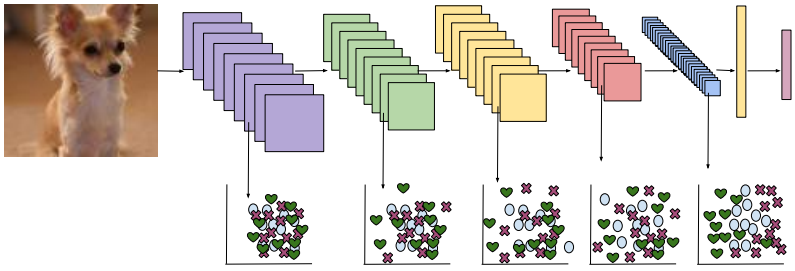

Figure 1: We assert that due to the complexity or corruption of the images or continuous improvement of discriminating features throughout the network, the network utilizes a higher number of layers in making a correct prediction. The bottom part of the figure depicts the proposed intuition validated through the extensive experiments as well. Three classes denoted in the shape of 'heart', 'oval', and 'cross' are used. In the early layers, a huge overlap between the classes can be observed leading to poor classification and as the layers increases, the overlap decreases accuracy improves.

computer vision Baldock et al. (2021); Huang et al. (2018) and natural language processing Liu et al. (2020); Xin et al. (2020) requires less computational overhead due to their early prediction. However, they do not formalize the processing of the data (specially modified examples) in the model or do not characterize the property on what basis an example can be termed as difficult. These studies also do not take into account the processing of high-resolution images and perturbed images. Our proposed research overcomes the limitations of the existing studies by presenting a detailed understanding covering both clean and corrupted samples through the deep model. It also helps in understanding several open issues such as example difficulty and the impact of hidden layer embeddings on the image classification.

In this research, we take a step back to effectively understand the network behavior from the perspective of common noise corruption samples. The behavior of the network is studied from both the statistical and learning view of the examples. To understand this, we propose a novel index termed as **corruption depth**, which captures the average highest depth in the network with the query image misclassified. To calculate the corruption depth, feature embedding obtained from the hidden layers of deep networks is utilized. On top of the embedding, several machine learning classifiers are attached for image classification. We assert that such understanding can help in mitigating the impact of corruption by either applying the pruning only in the desired area of the network or performing the weighted ensemble of feature representations from intermediate layers or weak classifiers.

1.1 Corruption Depth

We observe that as we move from the input stage to the prediction stage of the network, the classification decision of the images whether clean or modified changes significantly. As deep neural networks are often defined as a stack of layers, our proposed research studies the properties of the hidden representation computed during the processing of any example in an attempt to define the corruption depth. The corruption depth is defined as the highest depth layer needed to correctly classify an input. In other words, corruption depth is the notion used to refer to the highest network layer (number of layers) till the image in the network remains misclassified.

- The corruption depth is referred to as layer L before which the image in the network remains misclassified. In other words, if the image after layer L, i.e., $L+1$ is correctly classified and its decision match with the final decision of the network, then the layer L is the corruption depth of that image;
- To perform the image classification at the intermediate layers of the network and learn the corruption depth, we have used several machine learning classifiers including Neural Network (NNet), logistic regression (LR), and support vector machine (SVM).

Figure 1 shows the intuition and motivation behind the proposed corruption depth. It is to be noted that the corruption depth can be related to both clean and corrupted images. In the case of the clean images, the misclassification might happen due to natural adversaries or data having different data distribution than training Agarwal et al. (2020a); Carlini et al. (2019); Hendrycks et al. (2021).

Therefore, understanding their corruption depth is also crucial to improve the network performance even if there is no attacker tampering with input images. Whereas, in the case of corrupted images, for an attacker, it is essential to understand, how much difficulty needs to be added to the images to make them remain misclassified throughout the network. For a defender, this information is important in a way to understanding, how much and where in the network the pruning is needed so that the future layers receive the true information and the network can correctly classify even the corrupted images. Not only the pruning but also the boosting through the ensemble of weak classifiers based on their effectiveness at a particular layer can be considered for better robustness and accuracy.

1.2 CONTRIBUTIONS

❶ We measure a notion of DNN robustness to input data quality termed as **corruption depth** which tells when the images in the network remain misclassified. ❷ We showcase that even if the images are clean and of good quality, the corruption depth can be higher due to the complexity of the images. In other words, as the images pass through the network, the confidence of the network increases in its prediction. The finding is significantly contradictory to the work Baldock et al. (2021) which showcases that good quality images (visually) are generally classified early in the network and evaluated their understanding of low-resolution images only. ❸ Our study reveals the interesting behavior of multiple machine learning classifiers and the importance of intermediate feature representations regarding their robustness in handling corruptions. We have also demonstrated how the analysis produced in this research can help improve recognition accuracy (Section 4.1).

2 RELATED WORK

The existing works on understanding the vulnerability or the processing of complex, adversarial, and corrupted examples Bengio et al. (2009); Ren et al. (2018) think the issue from the side of the data manifold or global architecture of the network. For instance, it is believed the existence of manipulated data lies in low probability reasons of the input Gu & Rigazio (2014); Szegedy et al. (2013), most of the data lies close to the decision boundary Tanay & Griffin (2016), distribution of manipulated data is different Ghosh et al. (2019); Lee et al. (2017); Song et al. (2017), deep learning possesses linear hypothesis space when learning a decision boundary Goodfellow et al. (2014); Tabacof & Valle (2016); Tramèr et al. (2017), and the gradient vanishes during training make the network vulnerable Rozsa et al. (2016). Based on these understanding, several defense algorithms are developed which either enhance or generate different distribution of the images Agarwal et al. (2021d;c); Pérez et al. (2021); Rebuffi et al. (2021); Salman et al. (2020), retouch the entire network based on its component manipulation or retraining it Abusnaina et al. (2021); Andriushchenko & Flammarion (2020); Cui et al. (2021); Goswami et al. (2019; 2018); Jordao & Pedrini (2021), or learn a binary classifier based on training a entirely new end-to-end binary classification network Abusnaina et al. (2021); Agarwal et al. (2021a;b); Carrara et al. (2018); Yang et al. (2020).

The existing defense algorithms do not take into account the processing of images through the network, and hence either perform the modification at the entire network or on the whole dataset. For instance, Goswami et al. Goswami et al. (2019; 2018) have performed the selective dropout at the entire network or used the intermediate embedding of each layer to build a defense against common facial corruptions. Similarly, Jordo and Pedrini Jordao & Pedrini (2021) have applied the filter or layer pruning to the entire network without worrying about whether the images are misclassified into that layer or not. Similar analysis can be seen for the defense which denoise every sample or trains the network end-to-end to increase its robustness Salman et al. (2020); Xie et al. (2019). The significant drawback of lacking such understanding is that in the process of increasing the robustness of the network, we lose the accuracy of the clean or nonperturbed samples/examples.

Apart from the defense work, several works utilize hidden layer embedding to understand the behavior of deep neural networks. Cohen et al. Cohen et al. (2018) utilize machine learning classifiers to study the issue of generalization and memorization. However, the authors have not tackled the issue of how the individual data points get processed in the network and how easy and difficult they are to classify at a particular depth. Alain and Bengio Alain & Bengio (2016) claim the increase of linear separability with the increasing depth in the network embedding. Baldock et al. Baldock et al. (2021) have introduced the notion of example difficulty and claim that the good quality images are getting classified earlier in the network and challenging images require more depth. However, the notion of

Table 1: Parameters used for the different severity levels of each common corruption.

Severity	Gaussian Noise (GN)	Uniform Noise (UN)	Salt & Pepper Noise (SPN)	Shot Noise (SN)	Impulse Noise (IN)	Speckle Noise (SPKN)
S1	0.12	0.1	0.3	25	0.06	0.20
S2	0.18	0.3	0.5	12	0.09	0.35

example difficulty is defined based on visual appearance only which is highly preserved in minute corruption cases. The experiments are performed with low resolution, a single classifier namely K-nearest neighbor (KNN), and does not take into account the corruptions.

The proposed study can help in overcoming the limitations of these two different schools of study (defense vs. example difficulty) by learning the processing of images (clean and corrupted) and how long they remain misclassified into the network. Such study can help in understanding not only the processing of images based on their complexity but also can help in building fast and effective defense solutions. For instance, if we are aware that in a particular layer, the network misclassifies the majority of the images (due to its complexity), the pruning can be applied at that layer only.

3 Experimental Setup

In this research, we have considered several critical factors to reach any robust conclusion: (i) high-resolution images; (ii) multiple CNNs, (iii) several machine learning classifiers, and (iv) numerous corruptions with varying severity. Further, we describe each ingredient used to perform the research such as dataset, corruptions, CNNs, and machine learning probes for classification.

Dataset: We have used the subset of ImageNet Deng et al. (2009) namely ImageNette ima consisting of images of 10 different classes. We have used the pre-defined subsets which contains more than 9000 images for training and $3,000$ images for testing.

Common Corruptions: We have used six common corruptions and generated noisy images with varying levels of severity of the corruptions. The scale parameters of each corruption: Gaussian noise (GN), Uniform noise (UN), Salt & Pepper noise (SPN), Shot noise (SN), Impulse noise (IN), and Speckle noise (SPKN) are detailed in Table 1.

CNNs: The experiments are performed using VGG Simonyan & Zisserman (2014), MobileNet Howard et al. (2017), and Xception Chollet (2017). These networks are heavily used as a backbone architecture in several computer vision tasks and hence understanding these architectures in terms of example processing can put a significant step toward robustness. The networks are trained using Adam optimizer Kingma & Ba (2014) where the batch size is set to 32 and initial learning is set to $1e^{-3}$. The networks are trained for 200 epochs or until converged.

Machine Learning Probes: To resolve the shortcomings of the existing studies in the understanding of the example difficulty, we have used four different machine learning classifiers namely support vector machine (SVM), logistic regression (LR), K-nearest neighbor (KNN), and neural network (NNet). In this research, we have used the C-SVM which aims to regularize the classifier to avoid the overfitting issue. We have used a 50 neighbor to compute the similarity in the KNN algorithm. However, the findings are consistent with a wide range of values of K. We have also built a deep neural network architecture of five layers to perform the image classification on the hidden layer embeddings. The ReLU non-linearity has been used in the hidden layers of the network. The vast variation among the classifiers ensures the robustness of the study. The scikit-learn Pedregosa et al. (2011) library along with the default parameters of each classifier are used for image classification.

4 Experimental Results and Analysis

An extensive experimental evaluation has been performed to study the corruption depth both in terms of several critical factors including multiple CNNs and machine learning probes. Therefore, the analysis of the proposed research can be done based on the following terms: ❶ impact on different CNNs, ❷ corruption depth of individual corruptions, ❸ impact of severity of corruptions, ❹ machine learning probes, and ❺ role of hidden embedding of the deep CNNs.

Before studying the impact of corruption depth and associated factors, let us first understand, how successful three CNNs are for the image classification. The VGG network yields an accuracy (final softmax prediction) of 90.40% on the clean test set of the dataset. The classification performance

Table 2: Corruption depth is obtained using the hidden embeddings of the VGG architecture coupled with several machine learning classifiers namely KNN, NNet, LR, and SVM. '–' represents the average performance across each type of data including real and multiple corruptions. The performances are reported in terms of image classification accuracy at a particular layer/depth of the network.

Type	**L3**				**L9**				**L15**			
	NNet	**KNN**	**SVM**	**LR**	**NNet**	**KNN**	**SVM**	**LR**	**NNet**	**KNN**	**SVM**	**LR**
Real	51.80	39.06	54.73	50.40	78.50	60.80	79.10	78.66	89.50	87.86	90.10	89.43
GN_S1	28.76	27.50	23.56	25.10	38.40	27.96	42.50	39.40	70.46	56.96	64.36	65.53
GN_S2	21.53	22.26	15.63	15.76	24.93	24.23	29.40	26.70	46.46	28.73	37.43	39.53
IN_S1	24.73	25.13	18.66	19.83	31.16	26.43	33.26	29.56	57.16	39.90	47.56	49.26
IN_S2	21.50	21.93	15.13	15.23	25.60	24.06	27.30	25.16	42.00	24.43	31.13	32.20
SN_S1	30.20	29.36	24.40	26.40	39.56	32.06	43.36	41.06	73.50	58.46	69.43	69.50
SN_S2	24.16	24.03	17.90	18.33	28.26	27.40	31.66	29.73	52.76	34.00	43.90	45.16
SPN_S1	12.96	16.23	12.23	13.13	10.16	18.63	2.60	19.83	17.86	13.20	0.46	13.13
SPN_S2	13.20	14.63	4.00	13.13	9.96	13.00	12.56	18.40	15.26	12.76	0.00	13.13
UN_S1	47.06	38.43	52.80	49.10	73.33	54.80	75.73	74.30	86.43	84.93	87.20	86.43
UN_S2	34.20	28.29	33.96	34.93	51.50	36.06	53.66	50.66	78.96	71.26	76.56	76.73
SPKN_S1	37.36	34.23	35.19	39.13	55.56	38.56	56.80	57.90	82.43	75.06	81.69	80.80
SPKN_S2	27.43	26.40	21.50	22.20	34.73	30.43	37.40	37.73	66.50	48.80	60.80	61.53
–	28.84	26.73	25.36	26.36	38.59	31.88	40.41	40.70	59.94	48.95	53.12	55.57

Table 3: Corruption depth is obtained using the hidden embeddings of the MobileNet architecture coupled with several machine learning classifiers namely KNN, NNet, LR, and SVM. '–' represents the average performance across each type of data including real and multiple corruptions. The performances are reported in terms of image classification accuracy.

Type	**L3**				**L9**				**L15**			
	NNet	**KNN**	**SVM**	**LR**	**NNet**	**KNN**	**SVM**	**LR**	**NNet**	**KNN**	**SVM**	**LR**
Real	30.03	30.03	36.53	28.26	84.43	72.06	85.50	85.10	91.80	91.56	93.30	93.10
GN_S1	30.43	30.76	36.03	28.33	34.23	24.00	32.76	37.43	71.36	65.73	72.76	74.90
GN_S2	30.26	30.43	34.53	28.00	15.33	7.33	16.20	16.93	46.76	32.50	48.36	52.70
IN_S1	30.40	30.83	35.76	27.80	24.36	24.73	28.76	39.20	63.60	50.00	63.86	67.60
IN_S2	30.76	31.03	33.30	27.83	15.20	10.73	15.06	19.23	45.73	31.26	46.66	51.46
SN_S1	30.36	31.20	35.93	28.59	38.36	27.20	34.46	40.96	69.6	62.43	71.00	74.03
SN_S2	30.36	30.93	32.30	28.26	20.36	10.33	20.56	21.56	50.16	34.30	49.80	54.56
SPN_S1	28.10	27.73	18.46	25.23	7.83	0.03	0.16	1.33	15.86	10.20	15.43	16.36
SPN_S2	24.56	25.06	15.23	21.16	10.83	0.00	0.00	0.30	14.26	10.03	11.73	14.53
UN_S1	27.43	29.79	35.50	27.76	78.53	70.06	78.56	81.69	90.96	89.93	92.46	92.13
UN_S2	21.23	21.33	28.06	24.03	51.96	39.16	48.60	52.73	80.30	78.80	83.33	83.76
SPKN_S1	30.33	31.46	36.36	28.23	59.19	48.19	56.20	63.00	79.86	78.30	82.53	83.23
SPKN_S2	30.46	30.40	32.40	28.10	34.80	26.13	29.26	35.90	60.83	50.43	62.76	66.86
–	28.82	29.31	31.57	27.04	36.57	27.69	34.31	38.10	60.08	52.73	61.08	63.48

improves further when MobileNet and Xception architectures are used and they yield an accuracy of 92.80% and 93.90%, respectively. A significant variation in the accuracy and architecture tells that understanding corruption depth can give a broad view of it.

The corruption depth analysis on VGG is shown in Table 2. The network is divided into multiple equal spacing parts; however, for simplicity in the paper, the results are reported on the three parts only referred to as L3, L9, and L15. On the clean/real images, across the depth of the network and classifiers, monotonically increasing behavior in the accuracy is observed. Interestingly, towards, the end of the network, the KNN classifier shows a sharp jump which is more than double the jumps noticed against other classifiers. The other classifiers show this trend when they used the embedding of the center part of the network as compared to the initial part. In terms of corruption analysis, on the GN corruption, towards the center of the network, not a significant increment has been noticed. However, when the embedding at the final layer is used, the accuracy shows significant improvement. For example, from the initial layer to the center layer, the improvement notice is 9.64% which increases to 32.04% when moved from center to end of the network. The SPN corrupted images do not follow monotonically increasing behavior, and on NNet and SVM the network shows a 'U' shape trend, where the accuracy at the center is lower than the initial and end part of the network. In terms of classifier, LR follows the reverse trend as NNet and it shows upside down 'U' shape curve ('$\cap$'), where the accuracy at the center part increases from the initial part and again decreases at the end

Table 4: Corruption depth is obtained using the hidden embeddings of the Xception architecture coupled with several machine learning classifiers namely KNN, NNet, LR, and SVM. '–' represents the average performance across each type of data including real and multiple corruptions. The performances are reported in terms of image classification accuracy.

Type	**L3**				**L9**				**L15**			
	NNet	**KNN**	**SVM**	**LR**	**NNet**	**KNN**	**SVM**	**LR**	**NNet**	**KNN**	**SVM**	**LR**
Real	72.63	58.59	75.00	75.00	88.33	79.53	88.70	88.96	89.90	86.23	89.96	89.76
GN_S1	23.30	17.20	15.10	16.96	67.46	50.73	56.30	65.03	75.16	63.56	71.83	69.26
GN_S2	17.33	15.36	8.03	13.36	49.8	31.56	39.80	45.96	59.09	45.00	54.56	50.36
IN_S1	21.00	15.03	16.60	14.76	58.19	38.2	46.33	56.10	68.00	55.90	64.03	60.93
IN_S2	18.16	14.49	11.83	13.26	48.83	29.66	35.86	44.70	57.80	45.13	54.53	50.60
SN_S1	25.33	19.76	17.13	18.36	67.46	49.33	57.69	67.30	74.26	64.00	70.73	66.73
SN_S2	19.06	17.83	12.40	15.36	49.60	31.83	39.83	46.63	58.56	45.26	54.50	49.40
SPN_S1	13.13	13.70	11.76	13.13	20.53	0.33	7.30	15.29	26.66	16.20	18.50	20.06
SPN_S2	13.13	12.80	11.76	13.13	12.80	0.00	0.03	10.43	12.96	13.23	13.06	13.93
UN_S1	60.13	47.26	61.33	59.46	85.83	78.13	85.26	85.90	87.73	82.89	87.16	87.53
UN_S2	31.10	22.96	21.60	29.93	77.16	64.50	70.16	75.63	79.86	71.93	78.40	76.26
SPKN_S1	34.86	24.33	22.93	33.26	77.9	63.13	73.33	78.86	81.66	74.23	79.86	77.53
SPKN_S2	21.83	20.06	15.90	17.06	56.46	40.30	48.73	58.53	65.40	53.73	61.86	57.80
–	28.54	23.03	23.18	25.61	58.49	42.86	49.95	56.87	64.39	55.18	61.46	59.24

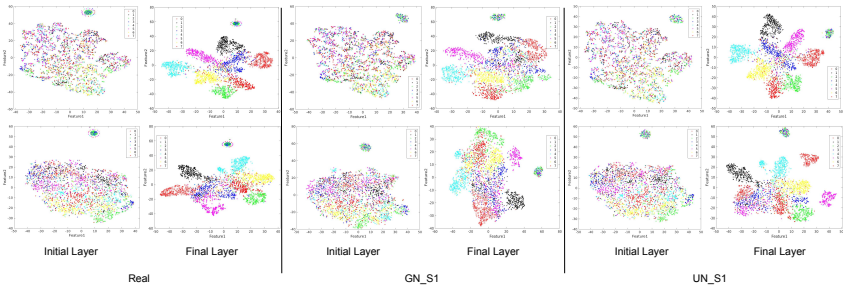

Figure 2: t-SNE Van der Maaten & Hinton (2008) representation of the hidden layers embedding of the VGG and MobileNet on real and two types of corruptions (GN and UN). Initial layer embedding is highly cluttered in both CNNs and across data types and shows lower classification accuracy. Final layer embedding decreases this clutter and hence leads to improved classification performance.

of the network. Except for SPN, image classification accuracy on other corruptions shows upward trends only with the increase in the depth of the network.

Similar to VGG as shown in Table 3, MobileNet reflects a multifold jump in the classification accuracy at the center of the network. For instance, the NNet and SVM, and LR yield more than 50% better performance at the center than an initial layer of the network. However, later the accuracy converges and shows a slight improvement. In contrast to VGG, on a majority of the corrupted images and classifiers, the 'U' shape accuracy trend is observed except for UN and SPKN. Interestingly, classification accuracy on SPN images never surpasses the accuracy obtained at initial layers even the depth of the network increases tremendously. In a unique observation, the initial layers of MobileNet architecture are found less sensitive against most of the corruptions of low severity (S1) as compared to the VGG. Even in the majority of the cases, the performance on corrupted images is slightly higher than the clean images.

The analysis of Xception is shown in Table 4. In contrast to previous architectures, on clean images, Xception shows significantly higher performance at the beginning of the network itself; hence, although the accuracy increases with the depth of the network its nature is not as sharp as is for other networks. Again, the embedding for image classification at the center of the network is found extremely effective and yields multiple-fold improvement using any classifier. For instance, the LR classifier on GN corruption shows at least 3 times increment at the center. The performance is in the sink with the use of any classifier and the majority of corruption types. We assert this understanding and consistency can help in network pruning and only storing partial network information to reduce the computational load on mobile devices without sacrificing the accuracy drastically. Whereas,

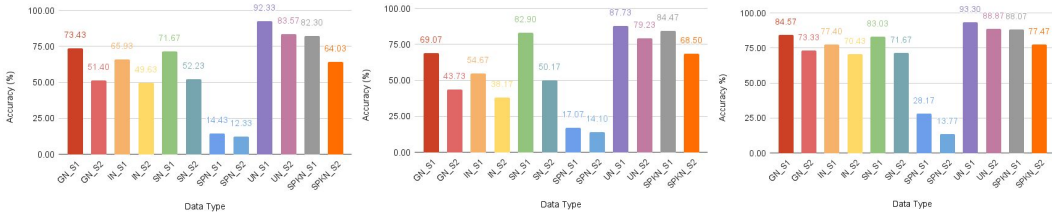

Figure 3: Softmax accuracies of multiple CNNs on various corruptions. The accuracies of VGG (left), MobileNet (center), and Xception (right) networks on the clean test images are 90.40%, 92.80%, and 93.90%, respectively.

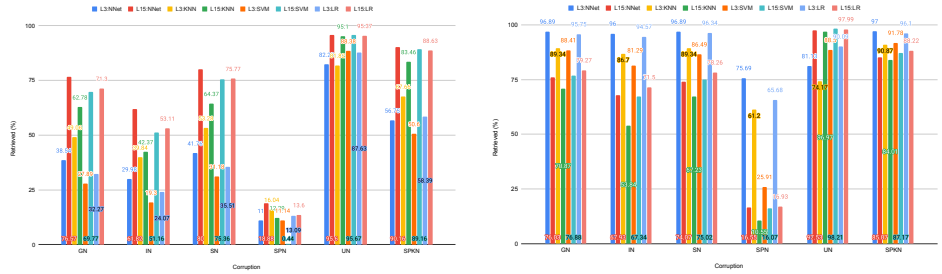

Figure 4: Probability of the prediction of true class of the examples when they are not perturbed and when they are corrupted using the common corruptions. The findings are reported on VGG (left) and MobileNet (right).

in terms of corruption depth, the center of the network is critical and the majority of the clean images find the center of the network as their corruption depth. In a unique analysis of Xception, on low severity SPN corruption ($S1$), the NNet and LR show monotonically increasing behavior and performance even increase from the value obtained at the initial layer embedding of the network. However, with increased severity trends similar to previous networks are observed.

Figure 2 shows how the separability of both clean and corrupted examples as they progress in the network. Figure 3 shows the softmax accuracies of the different CNNs across a wide range of corruption data types. We have observed that the final classification (i.e., softmax accuracies) is sensitive toward corruptions and yield lower performance than hidden embeddings. The probable reason might be that the minute noise in the image gets accumulated in the network and leads to misclassification at the end of the network Amirian et al. (2018); Goswami et al. (2019; 2018); Li & Li (2017). Interestingly, only in the case of VGG, on the UN corruptions, the performance increases and surpasses the best performance obtained using the hidden layer embeddings. For instance, on the clean images, the VGG network yields 90.40% accuracy, which improves to 92.33% when UN corruption of low severity (S1) is applied to the images. It shows the existing statistical and learning view based on the visual appearance of clean examples is not sufficient to understand the example difficulty as claimed in the literature Agarwal et al. (2020b); Baldock et al. (2021); Hooker et al. (2019).

In contrast to existing studies of understanding the '*statistical view*' of the example difficulty[1], we present an entirely different view referred to as predicting the probability that the corrupted sample can be correctly classified as it was classified in the absence of any corruption. This view is more realistic as compared to the assumption of its presence in the training set, as the model is expected to be generalized against the unseen test set. The findings of such a view are reported in Figure 4 using VGG and MobileNet. As expected with the increase in the depth, higher examples are correctly predicted even in the presence of any corruption. In other words, the correctly predicted images when they are perturbed and not perturbed increase with the increase of the depth of the network. However, this has an interesting caveat related to the accuracy of the network. For instance, as seen in Table 3, the accuracies on real and corrupted examples do not vary significantly on the initial layer embedding; therefore, it is observed that the probability of predicting the correct label of corrupted examples is

[1]is referred to as predicting the probability of correct label of an example if it not present in the training set Baldock et al. (2021); Jiang et al. (2021)

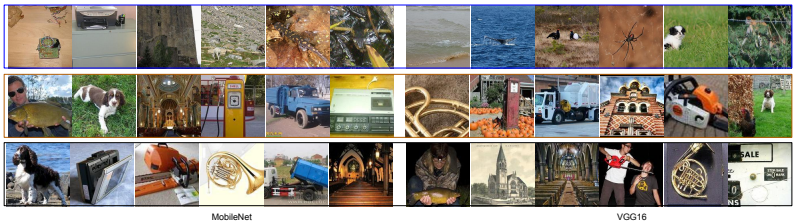

Figure 5: Samples represent the learning view of the example difficulty. The first row shows the clean test samples which are incorrectly classified due to their complexity such as low foreground region and cluttered background. The second row shows the clean and corrupted samples obtained using any corruption correctly classified by NNet only. The last row shows the samples either clean or corrupted correctly classified by each classifier (LR, SVM, KNN, and NNet).

high. We want to mention here the accuracy of the network on initial layer embedding is poor. As soon as the accuracy of the network increases at the later part of the network, the gap between the clean and corrupt examples' accuracy widens, and hence, the probability of predicting the true class on both clean and corrupted examples decreases. Moreover, the VGG and Xception show a significant difference in the accuracy from the beginning of the network; therefore, the probability keeps on increasing as the processing of the examples progresses through the network. The observation is found consistent across the machine learning probes used for image classification on the hidden layer embeddings of the CNNs.

Similar to the statistical view, in this research, we present a different angle to the '*learning view*'[2] of example difficulty presented in the literature Baldock et al. (2021); Toneva et al. (2018). In this research, at a broad level, we have three parameters: (i) corruption types, (ii) hidden embedding location, and (iii) machine learning probes. In the first angle of the learning view of example difficulty, we have to find out the samples (whether perturbed or clean) correctly classified by a machine learning classifier. For this, we have used initial layer and final layer embeddings and classification has been done using NNet. The middle part of Figure 5 shows such samples which are correctly identified by NNet where the images are corrupted using any corruption used in this research or are clean images. In the bottom part (last row), the samples are shown which are either clean or corrupted using one noise (say Gaussian) and the images are correctly classified by each machine learning probe used in this research. The top part of Figure 5 shows the complex clean images which get misclassified by each classifier and lead to higher corruption depth. The samples presented a wide variation in terms of contrast, texture, and illumination covering different object classes and raises the question of understanding existing research about example difficulty.

4.1 Discussion

Corruption depth: We assert that the corruption depth is an intuitive measure to understand the processing of images in the network. It is highly explanatory as compared to the notion of example difficulty which take into account only the visual characteristics of images. In this research, in terms of clean images, we found that the images which have cluttered objects and background, and limited foreground region as compared to the background have higher corruption depth (first row of Figure 5). In other words, images with these characteristics can be termed as difficult examples and need a large number of layers for processing and ground truth label prediction. Apart from these, corruption type and amount of corruption play an important role in the corruption depth computation. For instance, the speckle noise (SPKN) follows the uniformly distributed random noise and if the images are corrupted by this, they have significantly higher classification performance in the initial layer (lower corruption depth) as compared to other noises such as shot noise (SN) which follows the Poisson distribution. Another interesting noise called Uniform noise (UN) does not increase the corruption depth significantly on most of the networks if applied with decent severity (say S1). As soon as the severity increases or example difficulty increases, it leads to higher corruption depth. Salt&Pepper noise which is distributed as a Gaussian probability density function and is independent of the image intensity at a particular location is found to have the highest corruption depth where the corruption depth does not strictly follow monotonically increase behavior.

[2]earliest iteration after which the predicted label of the sample remains the same with the true label.

Table 5: Impact of network pruning and ensemble of robust classifies for improved accuracy (%).

Corruption	Type	Clean	GN	IN	SN	SPN	UN	SPKN
VGG	Original	90.40	73.43	65.93	71.67	14.43	92.33	82.30
	Purified	**93.57**	**75.62**	**66.89**	**74.56**	**20.52**	**93.15**	**84.69**
MobileNet	Original	92.80	43.73	38.17	50.17	14.10	79.23	68.50
	Purified	**95.10**	**55.42**	**53.16**	**57.89**	**27.13**	**85.75**	**70.10**

Pertinence to other topics in the field: Curriculum learning and Meta-learning Bengio et al. (2009); Graves et al. (2017); Ren et al. (2018) is a parallel field of machine learning which handles the difficult example (not corrupted) from the easy samples separately. Robustness to not only the difficult examples which might be due to distribution shift or environmental factors but also corruption examples is a critical component for building a fair machine learning system. In this research, we showcase several corruptions have different form of difficulty and different depth of the network treats them differently. Therefore, the proposed research is relevant to such research communities as well who are actively building fair, robust, and generalized algorithms such as selective dropout Goswami et al. (2019), guided dropout Keshari et al. (2019), and filter drop Nagpal et al. (2020).

Improving the accuracy: Through extensive experiments, we have observed that the different classifiers are effective at different depths of the network or can handle different corruption with varying efficacy. Can this understanding of how different data types and machine learning classifiers process the data help in building fair and robust models? The preliminary experiments performed using the prediction probability fusion showcase that it is feasible. For instance, on the Xception embedding of the initial layer, we have trained the four classifiers used in this research and fused their prediction probabilities. We have observed an improvement from 0.3% to 2.7% across different types of data including clean and corrupted. We assert that this hypothesis can be validated further where in place of combining classifiers trained on the same depth, we must combine classifiers trained on different depth embeddings. Therefore, we have further performed the aggregation of classifiers by dropping the neuron information from sensitive layers. The results reported in Table 5 show how the proposed study can be helpful to improve the accuracy of different types of images. Here purification refers to the weighted combination of layers (by pruning) and classifiers: $w_1 * L_3 + w_2 * L_9 + w_3 * L_{15}$. Here w_i represents the pruning percentage of the layer and lies in the range of [0, 1] and L_i represents the layer location. w_i is learned on the training set of the dataset.

Computational cost: The incorporation of several machine learning classifiers in the hidden embedding of the CNNs does not incur any significant computational cost. For instance, the training of all four machine learning classifiers on the final layer embeddings of the Xception took 95 seconds on the NVIDIA GeForce RTX 2080 GPU machines with the CUDA v11+.

Limitations and Societal impact: Through preliminary experiments, we have demonstrated that the merger of weak classifiers and intermediate feature maps can significantly boost image classification performance. Robustness through the filtering or dropping of corrupted information from the network can also be seen as one potential advancement.

5 Conclusion

In this research, we have proposed a notion of corruption depth based on the processing of different types of images in the network. The corruption depth reveals for how long a particular image in the network might remain misclassified based on its distribution shift or incorporation of any corruption. Through the extensive use of multiple machine learning classifiers, corruptions, and a variety of CNNs, several insightful observations have been noticed. It is also found that simple and effective machine learning classifiers such as logistic regression and K-nearest neighbors can handle the complexity and difficulty of the examples developed due to corruption in the initial layers more effectively. We have also shown that a few corruptions such as uniform noise have lower corruption depth in a particular network in comparison to the other corruptions. On top of that, each network has a different level of robustness in handling corruption, some have a high impact of corruption such as MobileNet while others can handle it effectively such as XceptionNet. The behavior of such broad corruptions throughout the network and the use of several machine learning probes can better help in building generalized and robust machine learning models. Several interesting shreds of evidence are also provided in the appendix.

REFERENCES

Imagenette. https://github.com/fastai/imagenette.

Ahmed Abusnaina, Yuhang Wu, Sunpreet Arora, Yizhen Wang, Fei Wang, Hao Yang, and David Mohaisen. Adversarial example detection using latent neighborhood graph. In *IEEE/CVF International Conference on Computer Vision*, pp. 7687–7696, 2021.

Akshay Agarwal, Mayank Vatsa, Richa Singh, and Nalini K Ratha. Noise is inside me! generating adversarial perturbations with noise derived from natural filters. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 774–775, 2020a.

Akshay Agarwal, Gaurav Goswami, Mayank Vatsa, Richa Singh, and Nalini K. Ratha. DAMAD: Database, attack, and model agnostic adversarial perturbation detector. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2021a. doi: 10.1109/TNNLS.2021.3051529.

Akshay Agarwal, Richa Singh, Mayank Vatsa, and Nalini Ratha. Image transformation-based defense against adversarial perturbation on deep learning models. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2106–2121, 2021b. doi: 10.1109/TDSC.2020.3027183.

Akshay Agarwal, Mayank Vatsa, Richa Singh, and Nalini Ratha. Intelligent and adaptive mixup technique for adversarial robustness. In *IEEE International Conference on Image Processing*, pp. 824–828, 2021c.

Akshay Agarwal, Mayank Vatsa, Richa Singh, and Nalini Ratha. Cognitive data augmentation for adversarial defense via pixel masking. *Pattern Recognition Letters*, 146:244–251, 2021d.

Chirag Agarwal, Daniel D'souza, and Sara Hooker. Estimating example difficulty using variance of gradients. *arXiv preprint arXiv:2008.11600*, 2020b.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Mohammadreza Amirian, Friedhelm Schwenker, and Thilo Stadelmann. Trace and detect adversarial attacks on cnns using feature response maps. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pp. 346–358, 2018.

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems*, 33:16048–16059, 2020.

Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 34, 2021.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, pp. 41–48, 2009.

Nicholas Carlini, Ulfar Erlingsson, and Nicolas Papernot. Distribution density, tails, and outliers in machine learning: Metrics and applications. *arXiv preprint arXiv:1910.13427*, 2019.

Fabio Carrara, Rudy Becarelli, Roberto Caldelli, Fabrizio Falchi, and Giuseppe Amato. Adversarial examples detection in features distance spaces. In *European Conference on Computer Vision Workshops*, pp. 0–0, 2018.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258, 2017.

Gilad Cohen, Guillermo Sapiro, and Raja Giryes. Dnn or k-nn: That is the generalize vs. memorize question. *arXiv preprint arXiv:1805.06822*, 2018.

Jiequan Cui, Shu Liu, Liwei Wang, and Jiaya Jia. Learnable boundary guided adversarial training. In *IEEE/CVF International Conference on Computer Vision*, pp. 15721–15730, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pp. 2232–2241, 2019.

Partha Ghosh, Arpan Losalka, and Michael J Black. Resisting adversarial attacks using gaussian mixture variational autoencoders. In *AAAI Conference on Artificial Intelligence*, volume 33, pp. 541–548, 2019.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Gaurav Goswami, Nalini Ratha, Akshay Agarwal, Richa Singh, and Mayank Vatsa. Unravelling robustness of deep learning based face recognition against adversarial attacks. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Gaurav Goswami, Akshay Agarwal, Nalini Ratha, Richa Singh, and Mayank Vatsa. Detecting and mitigating adversarial perturbations for robust face recognition. *International Journal of Computer Vision*, 127(6):719–742, 2019.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *International Conference on Machine Learning*, pp. 1311–1320, 2017.

Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15262–15271, 2021.

Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248*, 2019.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2018.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C Mozer. Characterizing structural regularities of labeled data in overparameterized models. *International Conference on Machine Learning*, 2021.

Artur Jordao and Hélio Pedrini. On the effect of pruning on adversarial robustness. In *IEEE/CVF International Conference on Computer Vision*, pp. 1–11, 2021.

Rohit Keshari, Mayank Vatsa, Richa Singh, and Afzel Noore. Learning structure and strength of cnn filters for small sample size training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Rohit Keshari, Richa Singh, and Mayank Vatsa. Guided dropout. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4065–4072, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Hyeungill Lee, Sungyeob Han, and Jungwoo Lee. Generative adversarial trainer: Defense to adversarial perturbations with gan. *arXiv preprint arXiv:1705.03387*, 2017.

Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *IEEE International Conference on Computer Vision*, pp. 5764–5772, 2017.

Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. *Advances in neural information processing systems*, 33:20331–20342, 2020.

Shruti Nagpal, Maneet Singh, Richa Singh, and Mayank Vatsa. Attribute aware filter-drop for bias invariant classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 32–33, 2020.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine Learning research*, 12:2825–2830, 2011.

Juan C Pérez, Motasem Alfarra, Guillaume Jeanneret, Laura Rueda, Ali Thabet, Bernard Ghanem, and Pablo Arbeláez. Enhancing adversarial robustness via test-time transformation ensembling. In *IEEE/CVF International Conference on Computer Vision*, pp. 81–91, 2021.

Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pp. 4334–4343, 2018.

Andras Rozsa, Manuel Gunther, and Terrance E Boult. Towards robust deep neural networks with bang. *arXiv preprint arXiv:1612.00138*, 2016.

Hadi Salman, Mingjie Sun, Greg Yang, Ashish Kapoor, and J Zico Kolter. Denoised smoothing: A provable defense for pretrained classifiers. *Advances in Neural Information Processing Systems*, 33:21945–21957, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Samuel L Smith, Benoit Dherin, David GT Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. *arXiv preprint arXiv:2101.12176*, 2021.

Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Pedro Tabacof and Eduardo Valle. Exploring the space of adversarial images. In *IEEE International Joint Conference on Neural Networks*, pp. 426–433, 2016.

Thomas Tanay and Lewis Griffin. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690*, 2016.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*, 2020.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 501–509, 2019.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*, 2020.

Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael Jordan. Ml-loo: Detecting adversarial examples with feature attribution. In *AAAI Conference on Artificial Intelligence*, volume 34, pp. 6639–6647, 2020.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *IEEE international conference on big data (Big data)*, pp. 581–590, 2020.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

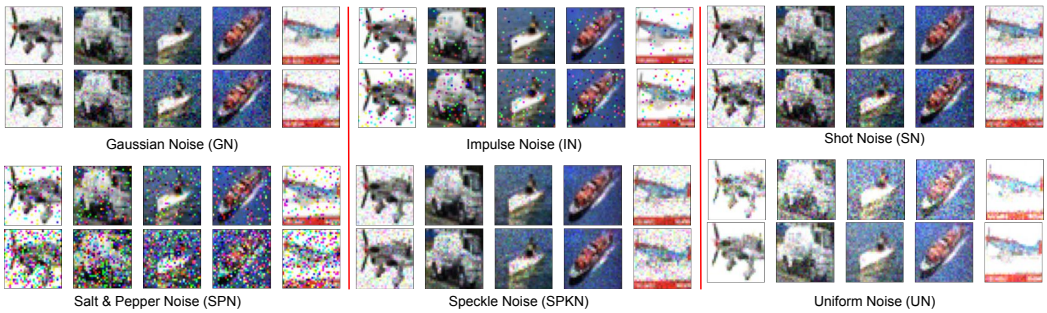

Figure 6: Images imbued with common corruptions with different severity levels which also changes their perceptibility from perceptible to quasi-imperceptible.

A Impact Statement

This research has a significant impact due to several factors which are still unexplored in the current literature: (i) first and foremost, the limitations of previous studies are well identified. We believe highlighting the limitations of current research can pave an effective way for future research. (ii) processing of different amount of complex images help in "identifying various phenomena that would be impossible to characterize without the proposed extensive experimentation". The understanding of how and why the images get classified correctly or incorrectly helps in building robust and trustworthy networks. (iii) Trends such as the "*U*" shaped drop in accuracy with prediction depth in Tables are interesting and still unknown even though the current deep network architecture has seen tremendous success. It might be a probable reason for the adversarial sensitivity of current deep networks.

We believe that the understanding of the corruption depth can **open a new dimension of model explainability**, where in place of just visualizing the attention map, the classification progress can be seen throughout the network. It can help in explaining the fact whether the images were correctly classified at any point of time in the network or remain misclassified throughout the network. Based on this understanding, feature visualization can be helpful to demonstrate why after a particular depth network classify/misclassify an image.

B Corruption and Complexity of Images

The corrupted images generated using varying severity levels are also presented in Figure 6. It can be seen due to the severity level, the added noise is in some cases highly perceptible; while in other cases it is either quasi-perceptible or imperceptible from the naked eye. For instance, row two of the salt&pepper corruption shows us that the added noise is highly perceptible and distorts the visual features of the images. Therefore, when these images pass through the network, at some point noise gets accumulated (for the majority of the cases, it is the center of the network) and shows a 'U' shaped curve. This observation is consistent across the networks such as VGG, Xception, and ResNet. The center part of the MobileNet shows the U-shaped curve for the majority of the corruption which is different from other models such as VGG, Xception, and DensNet. The probable reason might be that the MobileNet aims to tradeoff between accuracy and latency. Therefore, the model builds are highly customized and compressed due to the width multiplier factor and resolution multiplier.

C Impact on Other Forms of CNNs

We have conducted a study on the two CNNs namely DenseNet and ResNet which have different forms of connections and layers such as bottleneck blocks and residual connections. The experimental results reflect the similar observation of monotonically increasing behavior of accuracy with the depth of the network. The behavior of corruption depth analysis on these networks is reported in Table 6. Apart from that, we have also conducted a study where similar to previously used networks, we aim to improve the recognition accuracy of these two architectures. By applying the purification

Table 6: Corruption depth obtained using the intermediate embeddings of the ResNet and DenseNet architectures coupled with several machine learning classifiers namely KNN, NNet, LR, and SVM.

CNN	Type	L3	L3	L9	L9	L15	L15
		NNet	SVM	NNet	SVM	NNet	SVM
DenseNet	GN	15.56	13.65	25.86	23.25	43.89	45.78
	UN	32.86	31.53	56.20	57.95	75.47	73.90
ResNet	GN	15.20	14.36	24.89	25.63	42.10	44.25
	UN	33.68	32.30	54.29	55.42	77.20	75.26

knowledge obtained using the corruption depth layers, we have observed at least 3% improvement in accuracy on DenseNet and 3.7% improvement on ResNet.

D Corruption Depth Analysis Concerning Classifiers

Table 7: Corruption depth obtained using the intermediate embeddings of the VGG architecture coupled with several machine learning classifiers namely KNN, NNet, LR, and SVM.

Type	NNet			KNN			SVM			LR		
	L3	L6	L9	L3	L6	L9	L3	L6	L9	L3	L6	L9
Real	51.80	78.50	89.50	39.06	60.80	87.86	54.73	79.10	90.10	50.40	78.66	89.43
GN_S1	28.76	38.40	70.46	27.50	27.96	56.96	23.56	42.50	64.36	25.10	39.40	65.53
GN_S2	21.53	24.93	46.46	22.26	24.23	28.73	15.63	29.40	37.43	15.76	26.70	39.53
IN_S1	24.73	31.16	57.16	25.13	26.43	39.9	18.66	33.26	47.56	19.83	29.56	49.26
IN_S2	21.50	25.60	42.00	21.93	24.06	24.43	15.13	27.3	31.13	15.23	25.16	32.20
SN_S1	30.20	39.56	73.50	29.36	32.06	58.46	24.40	43.36	69.43	26.40	41.06	69.50
IN_S2	24.16	28.26	52.76	24.03	27.40	34.00	17.90	31.66	43.90	18.33	29.73	45.16
SPN_S1	12.96	10.16	17.86	16.23	18.63	13.20	12.23	2.60	0.46	13.13	19.83	13.13
SPN_S2	13.20	9.96	15.26	14.63	13.00	12.76	4.00	12.56	0.00	13.13	18.40	13.13
UN_S1	34.20	51.50	78.96	28.29	36.06	71.26	33.96	53.66	76.56	34.93	50.66	76.73
UN_S2	47.06	73.33	86.43	38.43	54.80	84.93	52.80	75.73	87.20	49.10	74.30	86.43
SPKN_S1	37.36	55.56	82.43	34.23	38.56	75.06	35.19	56.80	81.69	39.13	57.90	80.80
SPKN_S2	27.43	34.73	66.50	26.40	30.43	48.80	21.50	37.40	60.8	22.20	37.73	61.53

Table 8: Corruption depth obtained using the intermediate embeddings of the Xception architecture coupled with several machine learning classifiers namely KNN, NNet, LR, and SVM.

Type	NNet			KNN			SVM			LR		
	L3	L6	L9	L3	L6	L9	L3	L6	L9	L3	L6	L9
Real	72.63	88.33	89.90	58.59	79.53	86.23	75.00	88.70	89.96	75.00	88.96	89.76
GN_S1	23.30	67.46	75.16	17.20	50.73	63.56	15.10	56.30	71.83	16.96	65.03	69.26
GN_S2	17.33	49.80	59.09	15.36	31.56	45.00	8.03	39.80	54.56	13.36	45.96	50.36
IN_S1	21.00	58.19	68.00	15.03	38.20	55.90	16.60	46.33	64.03	14.76	56.10	60.93
IN_S2	18.16	48.83	57.80	14.49	29.66	45.13	11.83	35.86	54.53	13.26	44.70	50.60
SN_S1	25.33	67.46	74.26	19.76	49.33	64.00	17.13	57.69	70.73	18.36	67.30	66.73
SN_S2	19.06	49.60	58.56	17.83	31.83	45.26	12.40	39.83	54.50	15.36	46.63	49.40
SPN_S1	13.13	20.53	26.66	13.70	0.33	16.20	11.76	7.30	18.50	13.13	15.29	20.06
SPN_S2	13.13	12.80	12.96	12.80	0.00	13.23	11.76	0.03	13.06	13.13	10.43	13.93
UN_S1	60.13	85.83	87.73	47.26	78.13	82.89	61.33	85.26	87.16	59.46	85.90	87.53
UN_S2	31.10	77.16	79.86	22.96	64.50	71.93	21.60	70.16	78.40	29.93	75.63	76.26
SPKN_S1	34.86	77.90	81.66	24.33	63.13	74.23	22.93	73.33	79.86	33.26	78.86	77.53
SPKN_S2	21.83	56.46	65.40	20.06	40.30	53.73	15.90	48.73	61.86	17.06	58.53	57.80

In this research, we have selected the layers which are the best representative of the results. In other words, the selected layers represent the initial (L3), center (L9), and final part (L15) of the networks. We have also performed the experiments using the layers which lie between these layers as well. For example, in the Tables 7, 8, and 9, the results are reported using one of the layers between L3 and L9. We selected L6 at the center of L3 and L9 to avoid any possible bias. As seen earlier, with the increase of the depth of the network, the accuracy of the clean image monotonically increases. We want to highlight that the accuracy on L6 embedding is always lower than L9, therefore, the

Table 9: Corruption depth obtained using the intermediate embeddings of the MobileNet architecture coupled with several machine learning classifiers namely KNN, NNet, LR, and SVM.

Type	NNet			KNN			SVM			LR		
	L3	L6	L9	L3	L6	L9	L3	L6	L9	L3	L6	L9
Real	30.03	84.43	91.80	30.03	72.06	91.56	36.53	85.50	93.30	28.26	85.10	93.10
GN_S1	30.43	34.23	71.36	30.76	24.00	65.73	36.03	32.76	72.76	28.33	37.43	74.90
GN_S2	30.26	15.33	46.76	30.43	7.33	32.50	34.53	16.20	48.36	28.00	16.93	52.70
IN_S1	30.40	24.36	63.60	30.83	24.73	50.00	35.76	28.76	63.86	27.80	39.20	67.60
IN_S2	30.76	15.20	45.73	31.03	10.73	31.26	33.30	15.06	46.66	27.83	19.23	51.46
SN_S1	30.36	38.36	69.6	31.20	27.20	62.43	35.93	34.46	71.00	28.59	40.96	74.03
SN_S2	30.36	20.36	50.16	30.93	10.33	34.30	32.30	20.56	49.80	28.26	21.56	54.56
SPN_S1	28.10	7.83	15.86	27.73	0.03	10.20	18.46	0.16	15.43	25.23	1.33	16.36
SPN_S2	24.56	10.83	14.26	25.06	0.00	10.03	15.23	0.00	11.73	21.16	0.30	14.53
UN_S1	27.43	78.53	90.96	29.79	70.06	89.93	35.50	78.56	92.46	27.76	81.69	92.13
UN_S2	21.23	51.96	80.30	21.33	39.16	78.80	28.06	48.60	83.33	24.03	52.73	83.76
SPKN_S1	30.33	59.19	79.86	31.46	48.19	78.30	36.36	56.20	82.53	28.23	63.00	83.23
SPKN_S2	30.46	34.80	60.83	30.40	26.13	50.43	32.40	29.26	62.76	28.10	35.90	66.86

utilization of that layer for accuracy improvement does not make sense. Further, we want to highlight that the analysis is agnostic to CNNs as shown in multiple Tables.

We have performed experiments where we utilize these intermediate layers in the purification process; however, no improvement in the accuracy is observed. The prime reason can be seen from the fact that these intermediate layers yield an accuracy lower than the accuracy followed. Further, the inclusion of more layers will increase the computational cost of the system.