Image classification understanding with Model Inspector tool

Flávio A. O. Santos[1][0000−0003−2378−5376], Maynara Donato de Souza[1][0009−0004−0438−3229], Pedro Oliveira *[3],[0000−0001−7143−5413], Leonardo Nogueira Matos[2][0000−0002−6302−3299], Paulo Novais[3][0000−0002−3549−0754], and Cleber Zanchettin[1][0000−0001−6421−9747]

[1] Centro de Informática, Universidade Federal de Pernambuco, Recife, Pernambuco, Brazil {faos, mds3, cz}@cin.ufpe.br
[2] Departamento de computação, Universidade Federal de Sergipe, Aracaju, Sergipe, Brazil leonardo@dcomp.ufs.br
[3] University of Minho, Braga, Portugal
pedro.jose.oliveira@algoritmi.uminho.pt, pjon@di.uminho.pt

Abstract. This paper proposes a novel method called U Analysis for interpreting the behavior of image classification models. The method allows the evaluation of the interdependence between patches of information in an image and their impact on the model's classification performance. In addition, the paper introduces the Model Inspector tool that allows users to manipulate various visual features of an input image to understand better the model's robustness to different types of information. This work aims to provide a more comprehensive framework for model interpretation and help researchers and practitioners better understand the strengths and weaknesses of deep learning models in image classification. We perform experiments with CIFAR-10 and STL-10 datasets using the ResNet architecture. The findings show that ResNet model trained with CIFAR-10 and STL-10 presents counter-intuitive feature interdependence, which is seen as a weakness. This work can contribute to developing even more advanced tools for analyzing and understanding deep learning models.

Keywords: Interpretability · Trustworthy models · Image classification.

1 Introduction

As deep learning models become more popular, they become more present in our lives in different applications. This is no different for image classification models; they are important for applications of several domains, for example, classifying medical images such as MRI [15] and classifying images in e-commerce to automatically tag products [11]. Despite their success, image classification

* Supported by the doctoral Grant PRT/BD/154311/2022 financed by the Portuguese Foundation for Science and Technology (FCT), and with funds from European Union, under MIT Portugal Program.

models also have failures. For example, they are vulnerable to artificial and natural Adversarial attacks [20], biased to background information [21], and sometimes makes wrong prediction when we rotate the object in the scene [3]. This list of failures shows that we still need tools to harnessing and interpret the image classification model's behavior to understand its decision.

With the increase in model complexity and the resulting lack of transparency in the decision-making process, model interpretability methods have become increasingly important. The model transparency and interpretability usually are associated with the degree to which a human can understand the cause of a decision. When making predictions with a neural network, the data input is fed through many layers of multiplication with the learned weights and through non-linear transformations. A single prediction can involve millions of mathematical operations, thus being difficult for us humans to follow the exact mapping from data input to prediction. We would have to consider millions of weights that interact in a complex way to understand a prediction by a neural network. We need specific interpretability methods to interpret the behavior and predictions of neural networks.

In recent years, several methods have been proposed to interpret deep learning model outputs [17, 16, 19, 18]. Given an input x, model f, and target category y, these interpretability methods build an attribution map a with the same size as x, where a_i means how much important the feature x_i for $f(x)_y$. There are some libraries developed with Python that we can use to instantiate these methods and interpret models developed in Pytorch or TensorFlow, for example Captum [4], Innvestigate [2], and TensorFlow Interpretability [5]. These libraries have an easy-to-use interface where we can instantiate the interpretability methods to produce the attribution maps to our inputs. Still, we need to codify all input interactions that we want to infer the impact of feature changes in model output or attribution maps, thus being a challenge to beginner or even intermediate users to debug its models.

In this paper, we propose a new method called U Analysis, which allows us to evaluate the importance of different patches in an image and understand the impact of removing them on the model's classification performance. We also introduce a Model Inspector tool that allows users to manipulate various visual features of an input image to understand better the model's sensitivity to different types of information. Our goal is to provide a more comprehensive framework for model interpretation and help researchers and practitioners better understand the strengths and weaknesses of deep learning models in image classification. This work is structured as follows. Section 2 discusses the proposed U Analysis, and section 3 presents the Model inspector tool. The section 4 presents and discuss the experiments made with U Analysis, and we conclude the work in section 5.

[4] https://captum.ai/
[5] https://tf-explain.readthedocs.io/en/latest/

2 Related works

In the recent years, the scientific community has proposed several interpretability methods [17, 16, 19, 18]. These methods tries to decompose the model output strength into the input space, attributing a importance weight for each input pixel. Although these methods represents a path towards black-box model understanding, there is still concerns about its usefulness to infer how each input region (grid of features) impacts the model decisions [1]. Instead of attribute a degree of importance for each input feature, other methods proposes to infer the input feature importance by manipulating the input image and verify its impact on model output. [6] proposes the deletion and preservation games, where the first is to search for the smallest deletion mask (M_d) that when applied to the input image the model will change its prediction, on the other hand, the preservation game search for the smallest preservation mask (M_p) that when represents the sufficient input region to the model classify the input image correctly. [22] proposes the Occlusion approach to verify if some input image patch is important to model prediction by occluding it with a constant signal, such as grey patch.

The related works discussed so for focus on interpretability or model understanding methods that aim to attribute importance weights to individual input pixels, or infer the input region importance for model prediction. On the other hand, the method proposed in this work, called U Analysis (UA), addresses the co-dependence of input image patches for accurate model prediction. UA focuses on identifying groups of patches that need to co-exist to the model make accurate predictions. It systematically verifies the significance of these co-dependent patches by systematically manipulating the input image and observing the impact on the model's output. This method differs from the related works by explicitly finding the co-dependence among image patches rather than assigning importance weights to individual features or patches, instead it uses these weights as input to perform the analysis.

3 U Analysis

Different methods were proposed to visualize features and concepts learned by the neural network models, which have a performance that is less 'interpretative' and usually qualitatively evaluated. These methods compute how much each input feature contributes to the model output/prediction, but they do not explain the input features' interdependence nor the order of importance. In this work, we propose the U Analysis (UA), a systematic method to verify the co-dependence of input image patches for model prediction, a group of patches that must co-exist for the model to predict accurately.

The Algorithm 1 presents the U Analysis steps with a Python-based syntax. Given an input image x, model f, and attribution map I, the UA method first computes the importance of each x's patch of dimension $W \times W$ by summing up all attribution weight of each respective feature. Next, it sorts the x's patches

Algorithm 1: U Analysis.

Input: Given a trained deep learning model f, input image x, target, interpretability vector I, window size w, order type $order$, and noise type n

$region\_weights \leftarrow get\_region\_importance(I, w)$;
$region\_sorted \leftarrow sort\_region(region\_weights, order)$;
$gen\_batch \leftarrow remove_regions(x, region\_sorted, noise)$;
$pred\_batch \leftarrow f(gen\_batch)$;
$y\_batch\_pred \leftarrow pred\_batch.argmax(1)$;
$pos\_pred\_correct \leftarrow where(y\_batch\_pred == target)$;
$u\_triples \leftarrow []$;
for $i = 0$ **to** $len(y\_pred\_correct) - 1$ **do**
    $idx\_left = pos\_pred\_correct[i]$;
    $idx\_right = pos\_pred\_correct[i + 1]$;
    if $(idx\_right - idx\_left)$ *is larger than one* **then**
        $middle\_idx \leftarrow choice\_between(idx\_left, idx\_right)$;
        $u\_triples.append((idx\_left, idx\_middle, idx\_right))$;
return $u\_triples$

by importance. It cumulatively replaces each one of the input images by noise, creating new x_i images, where the x_i images are equal to x, except that does not have the information about the first i patches (i.e. gen_batch variable). Thus, assuming that the input image has N patches, the x_N image does not have any information (i.e., only noise such as zeros, ones, or Gaussian noise). Figure 1 shows a sample of the UA processing.

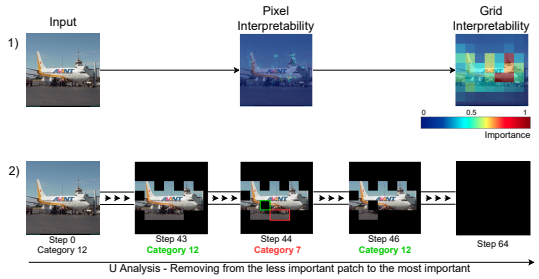

Fig. 1. U Analysis pipeline. The UA has two main steps, 1) given a input image it compute the model inference and interpretability, then it process the grid level interpretability. 2) Given the grid level interpretability, it sorts each image patch according to its importance and removes patch-by-patch from the input image from the less important to the most one.

After constructing the sequence of new images, it is possible to see that all the information from image x_i is present in $x_{j<i}$. Thus, if we have a case where

$left < middle < right$ and $f(x_{left}) = y$, $f(x_{middle})! = y$, and $f(x_{right}) = y$, it means that the information contained in x_{left} and x_{right} is sufficient for the model to infer correctly. However, the patches in x_{middle} but not in x_{right} create negative strength for y when they coexist with the other patches. We call this counter-intuitive case as U-occurrence.

4 Model Inspector

Image comprises different types of visual information, such as shape, color, texture, patterns, and objects. Each type of information may impact the model classification decision in different ways, for example, a model may be biased to texture, color, or shape [14, 3, 9, 23]. Thus, tweaking these types of information in the input image and evaluating the model with the newly image can help assess the classification model's robustness regarding different versions of the same signal, thus producing a local analysis of the model. Interpretability methods also can be used to debug image classification models. They produce how important is each input image pixel for model decision and attribution map to visualize. Thus the pixel importance can be used to compute metrics such as Top-K erasing and RFS [13]

Beyond these types of visual information, an image can be composed of two main spatial regions: foreground and background. The foreground is the image's main focus, which includes the subjects or objects of interest. On the other hand, the background is all the information that is not in the foreground. Usually, in image classification tasks, we want to classify the information that is in foreground, thus, it can be considered the signal while the background is the context. Background robustness is the ability of an image classification model to classify a signal even when it is on a different background. [21] showed that image classification models may be biased to background information and make a wrong prediction even when the foreground is present in the image but have a not common background. Thus, it is important to evaluate if the signal information is enough for the model to classify the image accurately or if the model is biased to background information.

The discussion presented so far shows we need a pipeline to evaluate the image classification model weakness. Therefore, we propose the image classification Model Inspector tool [6] whose goal is to allow users to evaluate image classification robustness against different types of transformations. It is composed of three different modules which the user can evaluate the image classification against different image processing functions, interpretability maps, and signal vs. noise sensitivity. The Model inspector is developed as a Web App which the user can load models from Pytorch or Timm[7] library and interact with its input-output results. In the following, we will present each Model inspector's module in detail.

[6] https://github.com/faos/image-classifier-model-inspector
[7] https://timm.fast.ai/

4.1 Image processing

The image processing module comprises several image transformation functions that add noise to the input image. The user can apply these transformations in the input image and visualize the model output difference with the original image, thus getting insights about the model robustness related to the transformation. The module implements three types of noise: Gaussian noise, Shot noise, and Impulse noise. Although we may add them intentionally, they can be naturally caused by phenomena such as random variations in light, sensor noise in the camera, interference in the transmission process, or bit errors. These noise functions can also be found in [8], where the authors created ImageNet [5] variations with them to evaluate the robustness of several image classification architectures on the ImageNet [5] dataset.

Gaussian, Shot, and Impulse noise are transformations that change the image color of texture. However, in addition to them, the image processing module also has spatial transformations (e.g., Patch Shuffle, Horizontal Shuffle, Vertical Shuffle) that deform the shape of the objects in the image, but it keeps the texture and color information, so it is helpful to infer whether the object's shape is important to the model predictions. Patch shuffle transformation splits the input image in disjoint squared patches with size $W \times W$, shuffles them, and creates a new image. On the other hand, horizontal shuffle creates disjoint horizontal patches with height H and size equal to the input image, shuffle them, and create a new image. The vertical shuffle is similar to the horizontal, the difference is that the patches are vertical, so their size is equal to the image height, and the width is W. Figure 2 presents the pipeline of this module and an example of each analysis function.

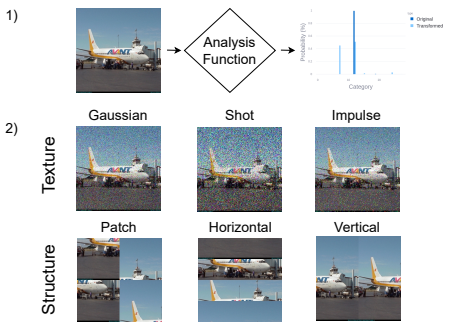

Fig. 2. Image processing pipeline. The part 1) show the main pipeline of the image processing; Given a input image, the user should select which function it will use to process the image and compare the model inference with the original image inference. The part 2) shows the functions available in model inspector, they are grouped by two types, texture and structure.

4.2 Interpretability

The interpretability module is composed of two main components: (1) interpretability methods and (2) U Analysis. In the first component, we implement a wrapper for the Captum library so the user can select the interpretability method and visualize its outputs. The second component implements the U Analysis discussed before, where the user can choose the noise method and window size. Besides, if the analysis finds counter-intuitive samples, it will show them. It is important to highlight that the attribution map used in the U Analysis is the output of the first component.

4.3 Signal

The signal module allows the users to interact with the input by selecting which region of the input image they consider the signal. The users may select the signal with three formats: rectangle, circle, and polygon. After selecting the signal, the module computes the signal-to-noise and background texture analyses. The signal-to-noise analysis calculates the importance of the signal region and compares it with the context to verify which region is more important to the model. The background texture analysis allows the user to apply all the image transformations from the first module to the image background only, thus verifying if the model decision is impacted by background texture changes while keeping the signal information. The noise analysis proposed in [13] inspired the background texture analysis. Figure 3 presents the signal pipeline.

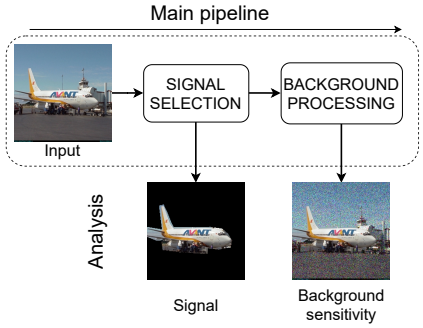

Fig. 3. Signal analysis pipeline. This module allows the user to select the signal of the input image using different formats, for example polygon, rectangle, and circle. After select the signal, it can add transformation to the image background to verify the model sensitivity to background changes while keeps the original signal. The main pipeline is composed of three steps, 1) select the input image, 2) select the signal information, and 3) apply the background processing functions. The signal selection and background processing has a output to compare the model inference using only each information.

5 Experiments and results

5.1 U Analysis

This section presents the experiments and results achieved with the U Analysis. First, we describe the datasets and architecture used and then present the percentage of U occurrence found. To perform the experiments, we used the CIFAR-10 [10] and Self-Taught Learning 10 (STL-10) [4] datasets. The CIFAR-10 dataset is composed of 60,000 32x32 color images grouped into ten classes and has 50,000 images for training and 10000 for testing. It is a well-balanced dataset. Thus, each class has 5,000 training and 1,000 testing images. On the other hand, STL-10 has the same classes as CIFAR-10 (i.e., airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck) but has 13,000 96x96 color images, where 5,000 are for training and 8,000 for testing. We train a ResNet-18 [7] instance for each dataset using Stochastic gradient descent (SGD) with a learning rate $1e-2$. Each network was trained by 50 epochs, and we chose the model with best accuracy on test set to perform the U Analysis.

The U analysis has several hyperparameters. For example, we can use different interpretability methods to compute the contribution of each input pixel to the model output, the order in which we sort the patch can be random, increasing, or decreasing, and we can replace the original patch information with several types of noise, and the patch's size (noise window size) itself is a parameter. To perform the U Analysis with the ResNet-18, we use 10 different interpretability methods, 3 sorting types, 6 noise types, and 5 different patch sizes, resulting in 900 runs for each dataset. Figure 4 presents the results achieved by the U analysis with all datasets and parameters. The results group the U occurrence for each hyperparameter value to infer which configuration is more susceptible to finding counter-intuitive behavior in image classification.

The findings show that all the attribution methods used have almost the same U occurrence. Thus they affect it in the same way. This conclusion is similar to the sorting method, in which all of them have almost the same U occurrence percentage, except the Increasing order in CIFAR-10, which is slightly higher than others. Although the attribution methods and noise window order have close U occurrence percentages, the type of noise has different values for each parameter value. The U occurrence was the lowest for both datasets when we used the image region mean and Gaussian noise. We argue that this behavior can be due to different reasons. For example, while the Gaussian noise does not represent information regarding the dataset, thus is easy for the model ignores it, the image region mean is a statistic of the patch that was removed, thus it still has information about the original patch.

The results show that the noise window size is the most important hyperparameter, with a patch size of 10% the parameter value with the most U occurrence in all scenarios. This result indicates that tiny patches instead of bigger ones may impact the ResNet-18, as the 33% presents a low U occurrence. In addition, the ResNet-18 may correlate the features of lower patches instead of bigger ones.

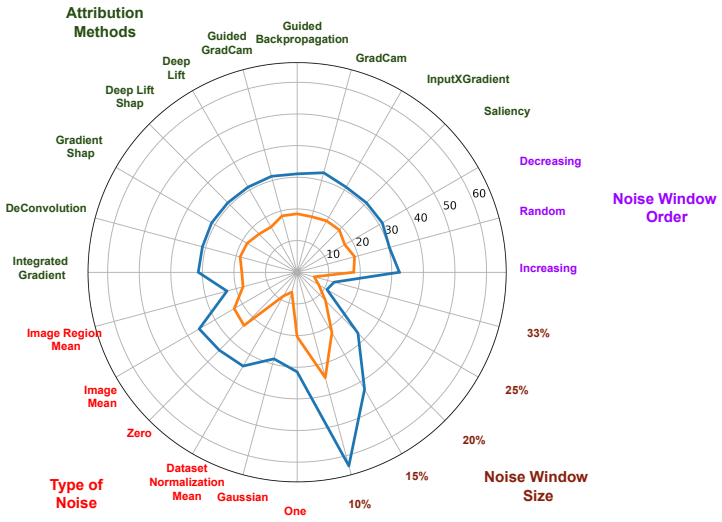

Fig. 4. U analysis results. The graph shows the results grouped by each parameter type (i.e. Attribution methods, type of noise, noise window size, and noise window order) and value. Besides, each curve represent the results for each dataset. The blue curve represents the results obtained from the CIFAR-10 dataset while the orange is with STL-10.

5.2 Model inspector demonstration

This section analyzes the model inspector tool and shows how the user can use it to infer insights about image classifier models. This analysis uses a ResNet-18 architecture trained with FGVC Aircraft [12] dataset to classify the aircraft manufacturer. Figure 5 presents the outputs obtained from the model inspector and has three crops extracted.

Part 1 shows the Image processing module applying the Gaussian transformation on the input image, while part 2 is the Patch shuffle transformation. Part 3 shows a sample of the signal module when we select the aircraft as the signal and apply the Gaussian noise into the background. All three parts have a barplot on the right to compare the ResNet output when we input the original image and the respective transformed image. Part 1 shows that when we insert Gaussian noise in the input image, the model changes its prediction, thus being sensitive to Gaussian noise. Part 2 also shows that when we destroy the spatial information with patch shuffle, the model also changes its prediction, which means that the signal structure is important for the prediction. Finally, part 3 shows that when we insert noise only in the background, the model does not change its prediction. Thus, joining these results with part 1, we can conclude that the model is sensitive to change in the signal only as it keeps its decision when we keep the original signal.

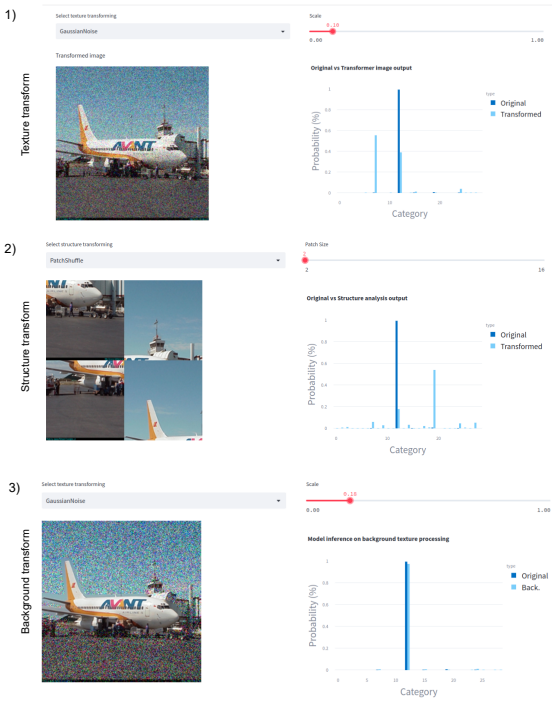

Fig. 5. Model inspector demonstration. Parts 1 and 2 show the outputs of the Image processing module for texture and structure transformation, respectively. While part 3 shows the result of the signal background texture transformation. On the left side, all parts have a selectbox so the user can select the transformation, and on the right side, there is a slider so the user can select the parameter value for the transformation.

6 Conclusion

In this paper, we presented U Analysis, a novel method for visualizing and interpreting the behavior of image classification models. UA allows us to understand the importance of patches in an image and their interactions, which can be used to understand how models make inferences and identify their weaknesses. Furthermore, we proposed a tool, Model Inspector, that allows users to interact with the input image and analyze the robustness of image classification models by changing visual information, such as texture, color, and shape.

Our experiments with UA show that the U-occurrence phenomenon can occur in some cases thus showing the image classification models has counter-intuitive feature interaction. We also showed that Model Inspector can be used to evaluate the robustness of image classification models to different versions of an image, and to detect biases in the model's decision-making process.

In summary, UA and Model Inspector are powerful tools for understanding and interpreting image classification models and can be used to improve their

performance and identify their weaknesses. Our work may help further research in the field of model interpretability and lead to the development of even more advanced tools for analyzing and understanding deep learning models.

Acknowledgements This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

References

1. Adebayo, J., Muelly, M., Abelson, H., Kim, B.: Post hoc explanations may be ineffective for detecting unknown spurious correlation. In: International Conference on Learning Representations (2022)
2. Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K.T., Montavon, G., Samek, W., Müller, K.R., Dähne, S., Kindermans, P.J.: innvestigate neural networks! Journal of Machine Learning Research **20**(93), 1–8 (2019), http://jmlr.org/papers/v20/18-540.html
3. Alcorn, M.A., Li, Q., Gong, Z., Wang, C., Mai, L., Ku, W.S., Nguyen, A.: Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
4. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 215–223. JMLR Workshop and Conference Proceedings (2011)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
6. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: Proceedings of the IEEE international conference on computer vision. pp. 3429–3437 (2017)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
8. Hendrycks, D., Dietterich, T.G.: Benchmarking neural network robustness to common corruptions and perturbations. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), https://openreview.net/forum?id=HJz6tiCqYm
9. Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., Song, D.: Natural adversarial examples. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15262–15271 (2021)
10. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
11. Liu, S., Li, L., Song, J., Yang, Y., Zeng, X.: Multimodal pre-training with self-distillation for product understanding in e-commerce. In: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining. pp. 1039–1047 (2023)
12. Maji, S., Kannala, J., Rahtu, E., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. Tech. rep. (2013)

13. Moayeri, M., Pope, P., Balaji, Y., Feizi, S.: A comprehensive study of image classification model sensitivity to foregrounds, backgrounds, and visual attributes. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022. pp. 19065–19075. IEEE (2022). https://doi.org/10.1109/CVPR52688.2022.01850, https://doi.org/10.1109/CVPR52688.2022.01850
14. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 427–436 (2015). https://doi.org/10.1109/CVPR.2015.7298640
15. Saurav, S., Sharma, A., Saini, R., Singh, S.: An attention-guided convolutional neural network for automated classification of brain tumor from mri. Neural Computing and Applications **35**(3), 2541–2560 (2023)
16. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 618–626 (2017)
17. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings (2014), http://arxiv.org/abs/1312.6034
18. Sudhakar, M., Sattarzadeh, S., Plataniotis, K.N., Jang, J., Jeong, Y., Kim, H.: Ada-sise: Adaptive semantic input sampling for efficient explanation of convolutional neural networks. IEEE International Conference on Acoustics, Speech and Signal Processing (2021)
19. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 3319–3328. PMLR (2017), http://proceedings.mlr.press/v70/sundararajan17a.html
20. Wong, E., Rice, L., Kolter, J.Z.: Fast is better than free: Revisiting adversarial training. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net (2020), https://openreview.net/forum?id=BJx040EFvH
21. Xiao, K.Y., Engstrom, L., Ilyas, A., Madry, A.: Noise or signal: The role of image backgrounds in object recognition. In: International Conference on Learning Representations
22. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13. pp. 818–833. Springer (2014)
23. Zhang, T., Zhu, Z.: Interpreting adversarially trained convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 7502–7511. PMLR (2019), http://proceedings.mlr.press/v97/zhang19s.html