VDK CNNet: A Variable-Density Convolutional Kernel for Improved Input Attribution in Classification Tasks

M Shifat Hossain
Department of Electrical and Computer Engineering
University of Central Florida, Orlando, FL 32816, USA

Abstract

Classification problem is one of the fundamental problems in the realm of machine learning. Many different machine learning methods and models are established for many different data sources with a good prediction accuracy, which make the classification task seem like a trivial one. But, different adversarial attack methods can challenge even the most complicated models with a simple dataset. This brings back the question of model safety and correctness of machine learning models. In this study, aa variable-density kernel convolutional layer is proposed, which can generalize larger features from an image input. The usage of variable-density kernel convolutional layers are shown to improve the classification accuracy, better input attribution for classes, and robustness to attacks with a low computational burden.

Problem Statement

Convolutional neural network or CNN is a type of widely used neural network that is used mainly for image or video datasets. The CNN works by the extraction of features from an input image or video frame from neighboring pixels of the image. Then these features are used in latter CNN layer to further extract features. This cascading nature can give higher generalization accuracy, lower memory and computational resources, and lower of parameters in CNNs compared to Feed-Forward NN, Multi-Layer Perceptrons, or Deep-Dense Networks.

The CNN works by extracting features using a convolutional layer. This layer uses a fixed-sized kernel, which iterates over the whole input image/video frame to train the weights and biases associated with each kernel. The kernel used in these layers is a square matrix, and the kernel size is a hyperparameter in CNN models.

Usually the kernel sizes used in CNN models are of 3x3, 5x5, 7x7, etc. These kernel sizes are usually small compared to the input image. For IMAGENET dataset the input image size is 255x255. But there are some tradeoffs of using smaller kernel. The advantages of using smaller kernels are -

- Lower memory and memory bandwidth requirement
- Lower number of multiplications per operation

And the disadvantages of this smaller kernel approach can be summarized as -

- Inability to extract larger image features [1].
- Poor performance when using transformed images [1].

A) Inability to extract larger image features:

When using smaller kernel sizes (e.g., 3x3) for classification task in a neural network, the kernel iterates over the whole input image and learns the weight and bias values of the kernel. One kernel will extract one feature from the input image and the feature size should be the same as the kernel size. In the case of a small kernel, the kernel only looks at the pixels very neighboring the center pixel for one position at the image.

This is a limitation, specially if the image size is too large. For example, considering CIFAR10 dataset, the input image sizes are 32x32. Hence the largest hypothetical image feature can be of 32x32 pixels. So, using 3x3 kernel size can cover about 0.8% of the feature area (7x7 kernel can cover about 5% of the feature).

On the other hand, using IMAGENET dataset, the input image dimension is 255x255. The largest hypothetical image feature can be of 255x255. And now using the 3x3 kernel size can cover only 0.01% of the feature area, which is too low.

B) Poor performance for transformed images:

On the other hand, as using smaller kernel can only extract features from near space in the input image, the kernels have very narrow scope to learn about image transformation functions. As a result, simple transformation of the input image can result misclassification in classification networks.

These lack of distant neighbor information and lack of transformation knowledge for using smaller kernel is currently approached by using a deeper model constructed by a large number of convolutional layers. Which requires large memory and computational resources.

In this project, a variable-density kernel (VDK) for convolutional layer is proposed to solve these above-mentioned issues of smaller kernel. The following sections describe the theory and method of application of this variable-density kernel.

Methodology

A) Introduction to variable-density kernels:

A variable density kernel is defined by scaling up a conventional 3x3 kernel by adding non-trainable pixels in the kernel. Figure 1 explains the variable-density kernel shapes.

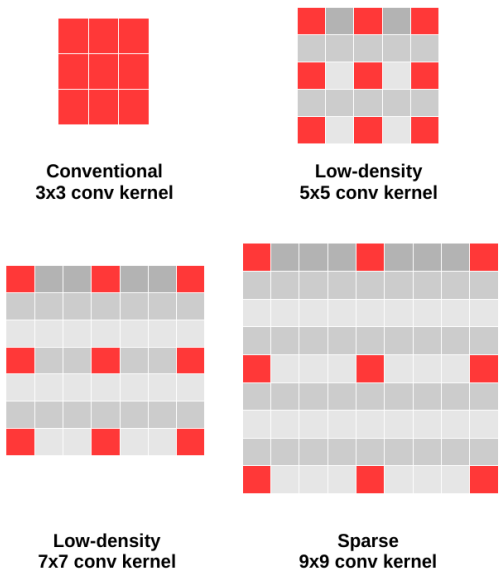


Figure 1: Variable-density kernels.

In fig. 1, the low-density 5x5, 7x7, and sparse 9x9 kernels are examples of the variable-density kernels. Here, the red pixels are trainable parameters, and the gray pixels are non-trainable zero valued parameters.

These kernels by default takes larger area in the input image, but the computational and memory requirements are still the same as 3x3 kernels. Since, these kernels can cover a larger area in the input image, these kernels should solve the large feature extraction problem as well as the input transformation problem.

B) Application of VDK Conv Layer in Pytorch:

Though, these kernels theoretically sound interesting and plausible for solving the above-mentioned problems, these layers are very hard to implement in the deep-learrning frameworks, like Pytorch and Tensorflow. Because implementing these kernels and making some pixels non-trainable will require the framework to store the full data matrix and discard some pixels' data. This will be a waste of computational resources.

In this study, the VDK kernels are implemented in a different method to overcome these limitations of the deep-learning frameworks. The solution is to use downscaled input images for the convolutional layers. If an input image is downsampled using a pooling function, the convolution operation on that image is equivalent to perform convolution operation with low-density kernel on the full scale image. Figure 2 explains the method of variable-density kernel using downsampled images.

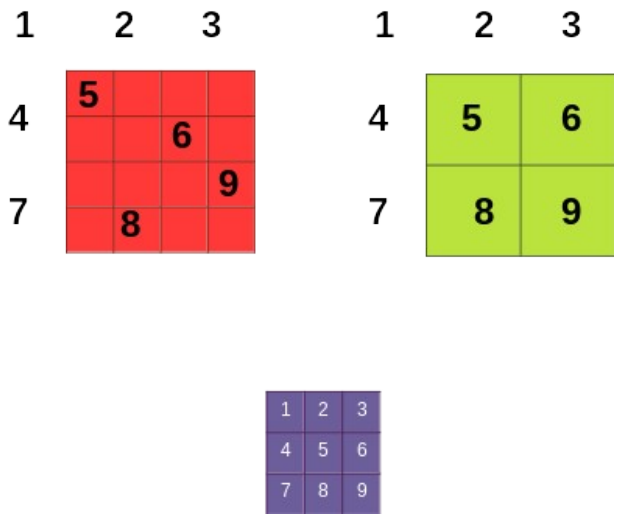


Figure 2: implementation of variable-density kernel using downsampled input images.

Here, in this image, the red image is 4x4 input image, the green is the 2x2 downsampled image using 2x2 maxpool layer of stride 2. The purple 3x3 matrix is the kernel. Now, if the kernel is used over the downsampled image (green image), this operation is equivalent to perform a low-density convolution operation (low-density 7x7) on the full scale image (red image). The numbers in the pixels in the image indicates the kernel pixel position over the input image.

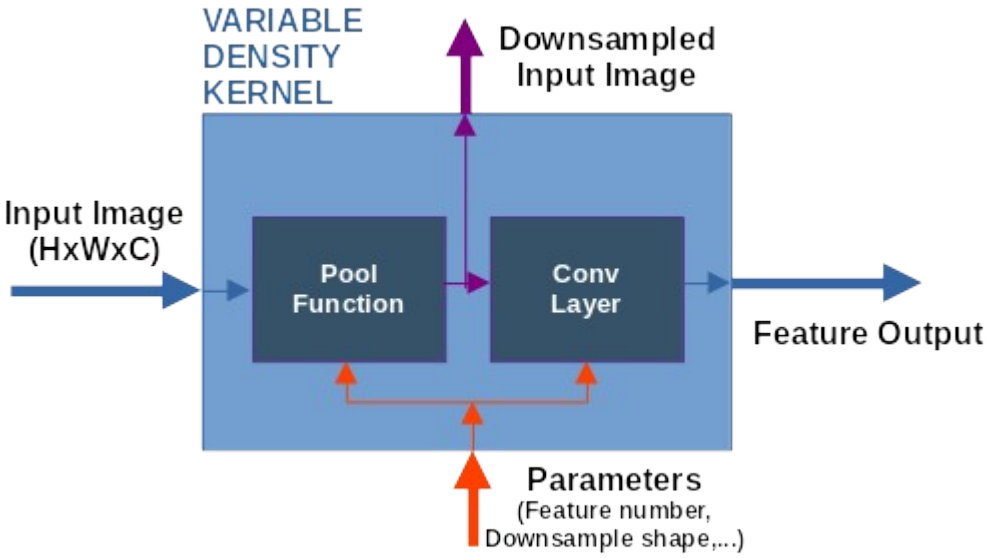


Figure 3: Block diagram of variable-density kernel.

Figure 3 shows the block diagram of the variable density kernel. This implementation contains one pool function and one conv layer (3x3) with an activation function. The module takes output feature number, downsample shape as parameters. This module gives the downsamples image as output to further use the downsampled image as an input to another VDK convolution layer. This module gives feature output from the convolution layer. The shape of the output features depend on the downsampled shape.

It is important to understand that, the VDK Conv layers are not replacement of the conventional dense kernel Conv2d layers. These VDK Conv layers are meant to be used in parallel to the conventional Conv2d layers in a multimodal-model setup (Fig. 4).

Results

A) Evaluation setup:

Two toy models were implemented and tested for classification accuracy, attribution analysis, and robustness. One of the model codenamed “VDK CNNet” uses the VDK conv2d layer (Fig. 4), and the other model uses simple conv2d layers, codenamed “Baseline CNN” (Fig.5) to classify the images. The dataset used is the CIFAR10 dataset.

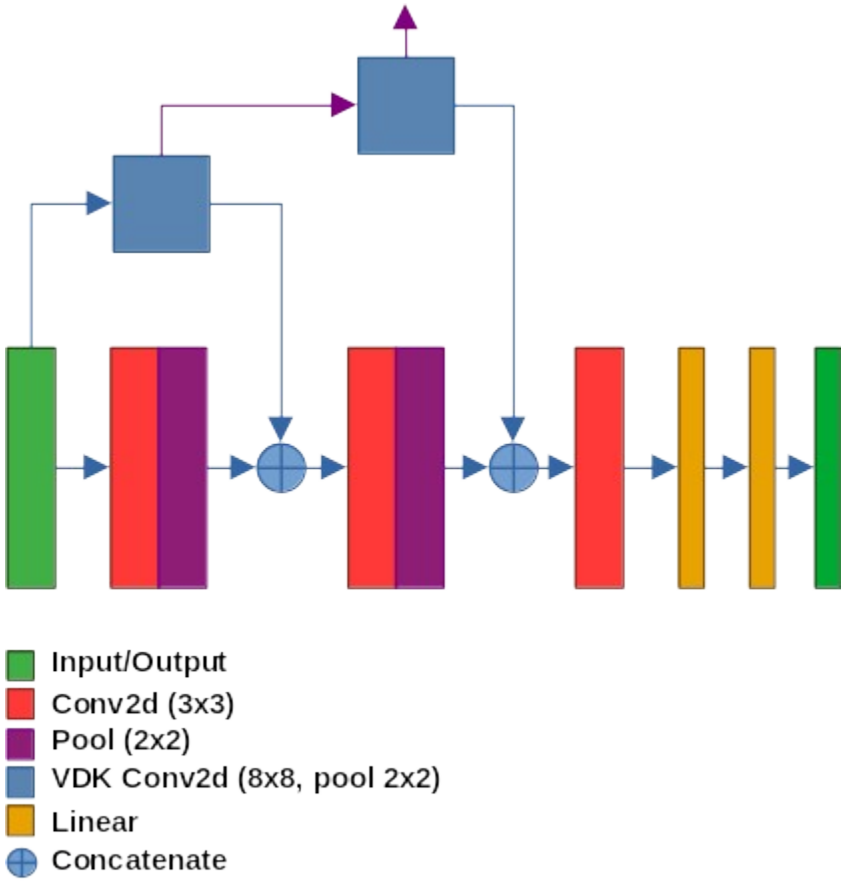


Figure 4: VDK CNNet architecture.

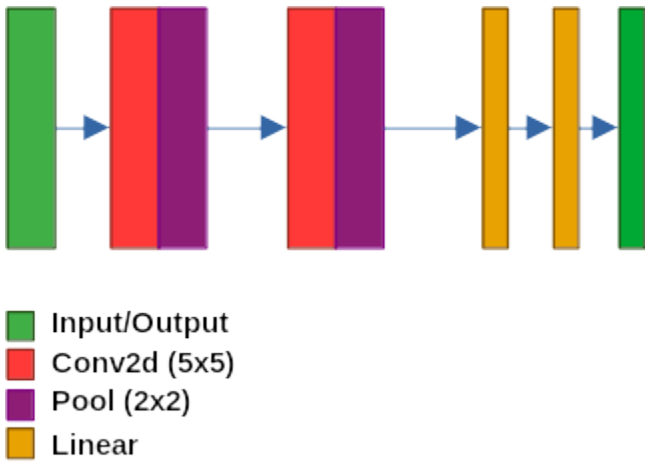
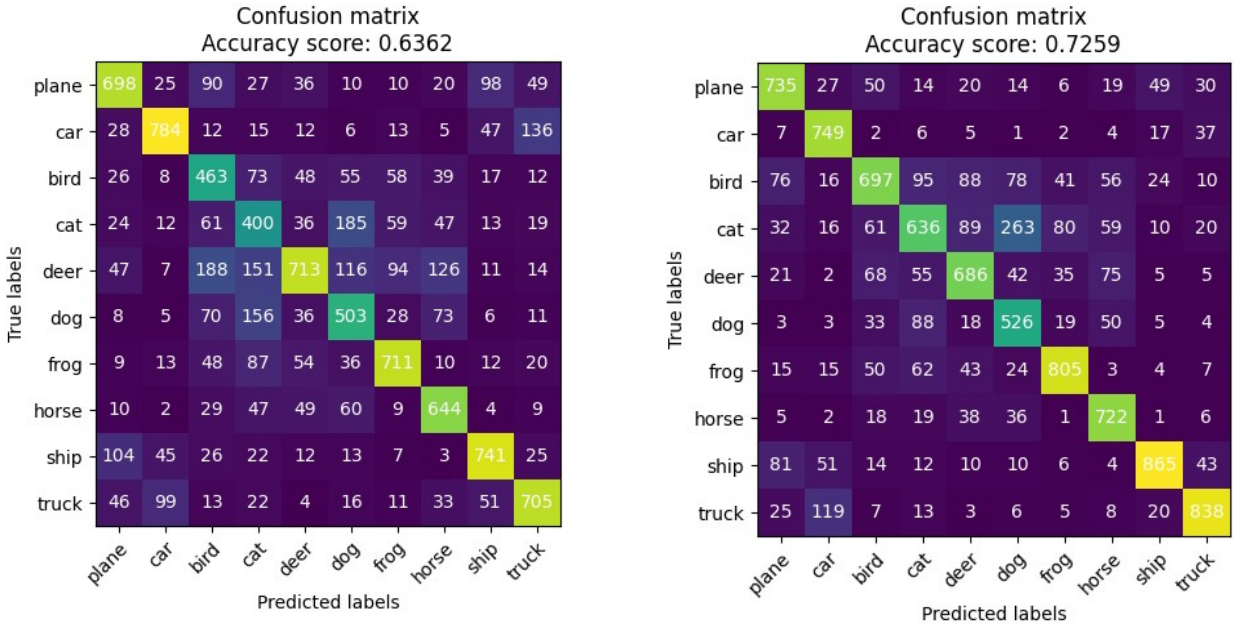


Figure 5: Baseline CNN architecture.

B) Classification Accuracy:

Comparing these two toy models, we got the classification accuracy of 72.5% and 63.6% for the VDK CNNet and Baseline CNN models, respectively. Figure 6 shows the confusion matrix.



(a) (b)

Figure 6: Confusion matrix for classification for, (a) Baseline CNN and (b) VDK CNNet models.

C) Attribution Analysis:

Comparing the input attribution of the models can give clearer idea of how the models are classifying one input image. For this reason, the VDK CNNet and Baseline CNN models are used to generate input attribution for a class (“car” in this example). Then the input attributions are visually inspected to understand if the features used to classify the object are meaningful to the object or not. Four different methods of input attribution are used – integrated gradients, gradient magnitudes, integrated gradients with SmoothGrad squared, and DeepLift.

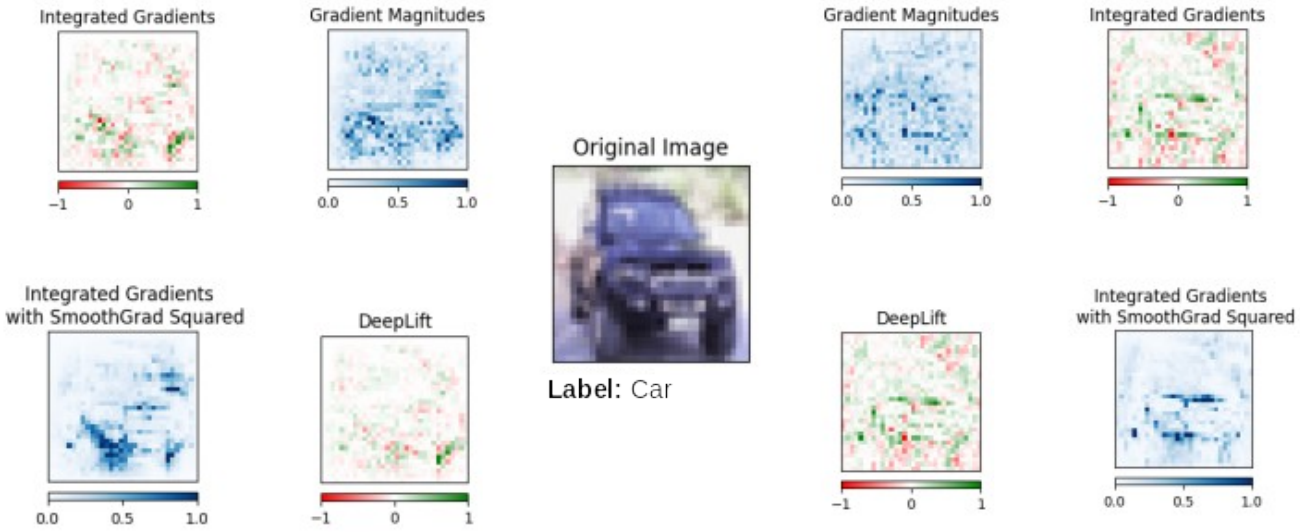


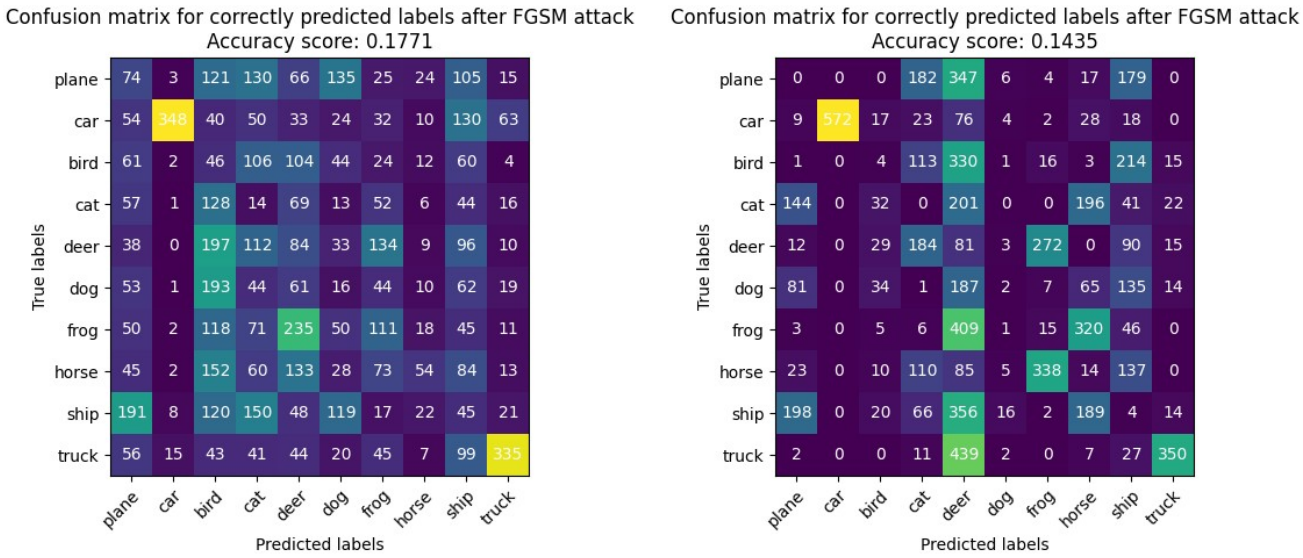
Figure 7: input attribution analysis for a sample image of a class “car”.

From this image (fig. 7) it can be seen that, the important features seen from the integrated gradient with SmoothGrad squared attribution are the car wheel shadows for the Baseline CNN model. But for the VDK CNNet, the important feature for distinguishing the car was the front grills and headlight outlines, which is a natural distinguishing factor for classifying a car.

D) Robustness analysis:

I. Fast Gradient Signed Method (FGSM) Attack:

The FGSM attack was performed to observe the robustness of these two models. The FGSM method works by modifying the input image by adding noise generated in the dradient of the cost function. In this way, the attack can make some sample inputs which will be misclassified by the classifier model. Figure 8 depicts the confusion metrics and accuracy scores of these two models. None of the model had significant performance in this test.

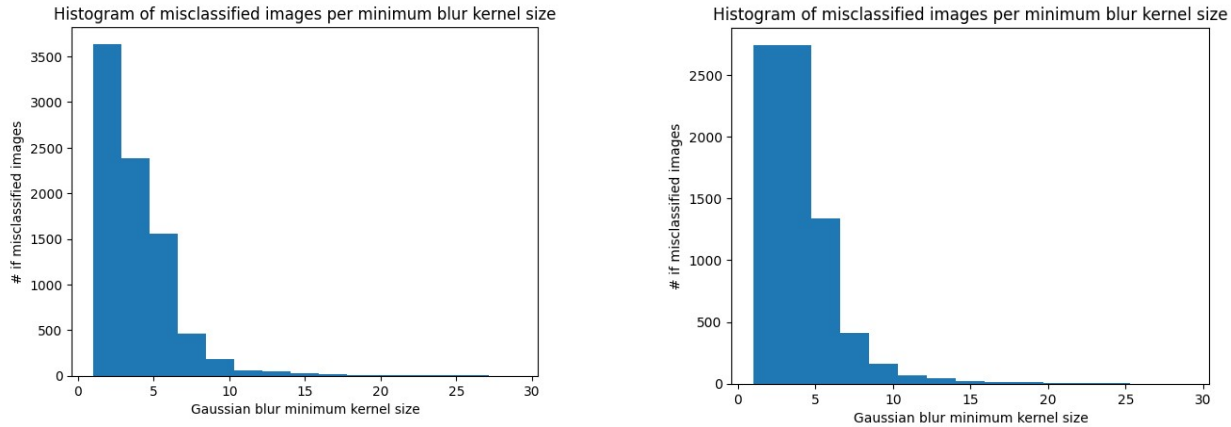


(a) (b)

Figure 8: Confusion metrics after FGSM attack for (a) Baseline CNN and (b) VDK CNNet models.

II. Gaussian Blur Attack:

Finally the models' robustness was checked using Gaussian blur attack to check the minimum number of Gaussian blur kernels required to misclassify an image, which is called minimal perturbation using Gaussian blur attack. Figure 9 depicts the histogram of the minimal perturbation (Gaussian blur). There were some samples, which were immune to the perturbation attacks. The immune samples are given in Table 1. We can see significant improvement of VDK CNNet compared to the Baseline CNN model in this test.



(a) (b)

Figure 9: Histogram of the minimal perturbation (Gaussian blur) for (a) Baseline CNN and (b) VDK CNNet models.

Table 1: Table of immune and affected samples after minimal perturbation.

	Baseline CNN	**VDK CNNet**
Immune Samples	15.91%	**24.29%**
Affected Samples	**84.09%**	75.71%

Possible Issues

There are possible few drawbacks in this approach of simulating large convolution kernel. Since this approach is simulating large kernel by using low-density or sparse matrix and then multiply it with a large input area, this loses many pixel in between the trainable pixels (those are set zero and non-trainable). This loss of data might cause issues extracting features which are distributed near the active pixels.

These kernels may also fail to generalize in some cases, since the active pixels in the kernel might not be looking into pixels of a same image feature. The distant neighbor pixels might be looking into other image features.

And finally, too sparse or too large VDK Conv layer might cause problems in feature extraction. This is because of excessive loss of data and losing coherence in the input space.

Future Prospects

Though there are some possible drawbacks of this approach, we have seen in this test that the VDK CNNet model had higher classification accuracy and more robust against attacks. So, based on these results this process of integrating variable-density kernels in a model should improve the accuracy, input attribution, and make the model more robust against attacks.

Moreover, based on the claims of [1] this method of variable-density kernel should be more efficient and accurate in object segmentation tasks as well.

Conclusion

CNN is a major model in the realm of image processing and image recognition tasks. A new kernel for convolutional operation increasing the classification accuracy, input attributions, and robustness can really improve the quality of the CNN and the derived models. In this study, tests were performed using two toy models and seen that the model with VDK Conv layer performed better in most of the tests and can improve model performance.

References

[1] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, “Large Kernel Matters -- Improve Semantic Segmentation by Global Convolutional Network,” presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4353–4361. Accessed: Dec. 05, 2022. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Peng_Large_Kernel_Matters_CVPR_2017_paper.html