3D BugNIST Classification: Revealing Convolutional Complexities

Morten Møller Christensen (s204258), Jacob Schrøder Ipsen (s204440), Rasmus Munch Kielsgaard Nielsen (s204504), Jonas Ryssel (s184009)

DTU Compute, Technical University of Denmark

Introduction

2D image classification is one of the most common deep learning tasks. Is it an image of a dog, of a tree, or of something else? This is notoriously done using 2D convolutions. But what if we aren't classifying a standard RGB (or greyscale) 2D image? What if it's a volumetric object that describes some type of bug?
In this project we classify bugs based on their volumetric object, visualize the saliency maps to "reveal" its learning patterns. We also discover that a neural network don't necessarily learns from the things we expect it to learn from.

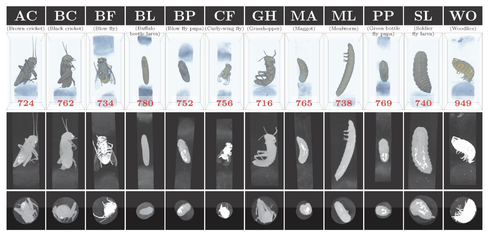


Figure: An overview of the BugNIST dataset [4]

Data

The BugNIST3D dataset is a three-dimensional collection that includes 12 distinct bug types. For most of the bugs there is around ~750 objects. However for the last bug type (woodlice) we have around 949 objects. All volumes are 128x64x64 voxels and contain one bug.[1] In the figure below one of each bug type can be seen.
(Note that for these visuals we have added a simple threshold to remove the cotton that is around the bugs. This have also removed some detail on the bugs such as the wings on the blow fly.)

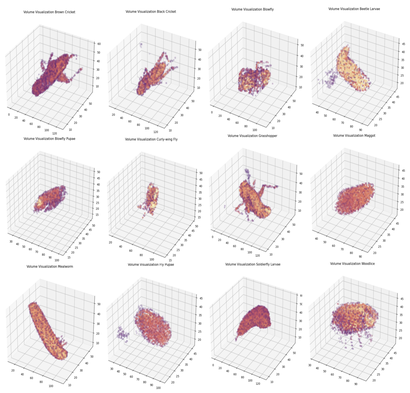


Figure: 3D visualization of the datasæt

Model

The most common convolution to be used is the 2D-convolution which is used in practically whenever images are involved however since this is a 3d volume 3D-convolutions will be used. The concept is basically the same as in 2D the main difference is that the kernel is a NxNxN cube rather than a NxN square. In the figure below a 3D kernel is shown:

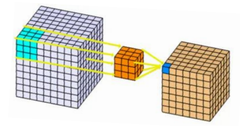


Figure illustrating a 3D-convolution [2]

Our network processes data using four convolutional layers and pooling layers, then channels it through two feed-forward layers, producing a 12x1 vector. This vector indicates the probability of belonging to each class. The network can be seen below:

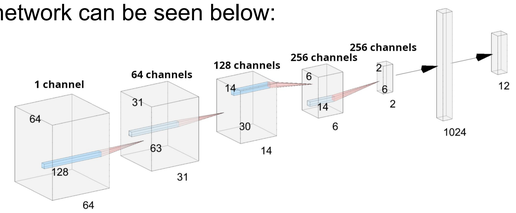


Figure: 3D Convolutional model used for the classification task

Saliency Maps

One of the common problems with neural networks is to understand the reasoning behind the output of the network. For classification problems it makes sense to look at how much each pixel/voxel contributes to the final classification. This can be done using a saliency map. Different approaches exist to generate a saliency map. We have decided to use SmoothGrad [3] to visualize the saliency maps.

$$\hat{M}_c(x) = \frac{1}{n}\sum_{1}^{n} M_c(x + \mathcal{N}(0, \sigma^2))$$

Compact way to write the SmoothGrad saliency map [3]

SmoothGrad works through 3 simple steps:

1. Add random noise to the image n times. This helps reduce the noise in the gradients.
2. For each new image the gradient of the model's output with respect to the input image is calculated.
3. After all the gradients have been calculated take the average of them to get the SmoothGrad saliency map.

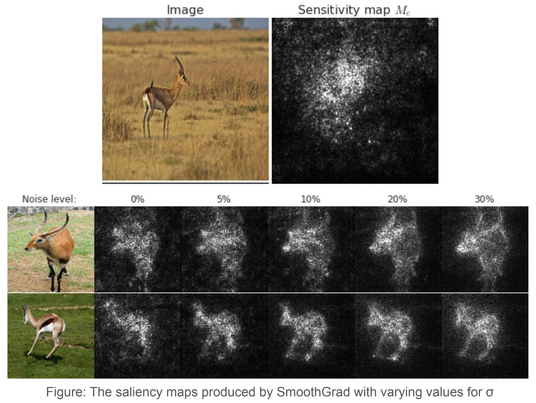
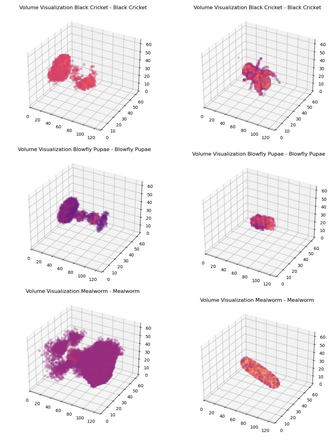


Figure: The saliency maps produced by SmoothGrad with varying values for σ

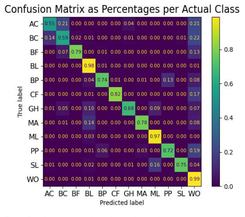
Results

Shortcut learning

Saliency maps **3D Bug**



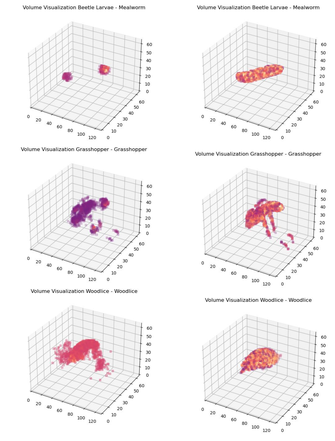
From the confusion matrix (calculated on a test set) one will see that this model performs well on basically all bugs. Amazing one might think, but talking a closer look at the saliency maps one would notice why it performs so well. The network has learned the wrong thing, as it focuses on the cotton in the ends of the samples to classify the bugs. By removing the cotton the shortcuts to classifying the bugs will hopefully be gone.



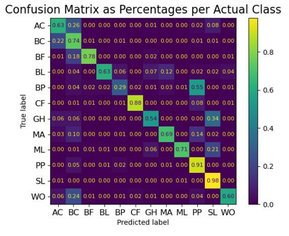
Confusion matrix showing the percentage per Actual Class for our shortcut model. Notice how well it now classifies "BL" (Blowfly) and "BP" (Blowfly Pupae) compared to the other model.

Removing the Shortcuts

Saliency maps **3D Bug**



We can see on the saliency maps, that after we remove the cotton, the model now actually learn to use the bugs themselves for classification. The downside of this is that now the network performs worse, as we can see from the accuracies in the confusion matrix.



Confusion matrix showing the percentage per Actual Class for our "non-shortcut" model. Notice how the network misclassifies the majority of the class "BP" (Blowfly Pupae) as "PP" (Fly Pupae).

References:

[1] Patrick Møller Jensen, Vedrana Andersen Dahl, Carsten Gundlach, Rebecca Engberg, Hans Martin Kjer, & Anders Bjorholm Dahl. (2024). BugNIST – a Large Volumetric Dataset for Object Detection under Domain Shift.

[2] Biplab Barman, "3D-Convolutions and its Applications", 2020 https://biplabbarman097.medium.com/3d-convolutions-and-its-applications-6dd2d0e9e63f

[3] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, Martin Wattenberg. 2017. SmoothGrad: removing noise by adding noise.

[4] Anders Bjorholm Dahl, Christian Kento Rasmussen, M Kjer, Patrick M. Jensen, Vedrana. (2024). BugNIST2024. Kaggle. https://kaggle.com/competitions/bugnist2024fgvc