Intuitive Neural Networks: How Software can Learn from Experience

SCM113

Purpose

- Neural networks (NNs) "learn" from examples like humans
- Trained NN can be effective at tedious, repetitive tasks
  - e.g. Detecting cancer cells in microscope slides
- By 2024 their market is expected to be $21 billion
- But we don't have much insight into how NNs learn
  - Hence nickname "black box"
- **Purpose:** Understand how object detecting NNs train themselves to detect objects

Background

- 3 years ago, I ran a biological experiment with cockroaches
  - Camera took picture of roaches & background objects once a minute
  - Image analysis of all ~24000 images took long time
- Trained NN to detect roaches for me
  - Showed it examples: positive & negative images
  - Positive images have roaches
  - Negative images don't have roaches

Positive Image

Negative Image

- Each image was subtracted from previous to remove background objects

Subtracted Positive Image

- NN is made up of 140 neurons
- During training NN adjusts each neuron's weights and biases until it finds optimum values that give highest accuracy score for the training images
- After training NN detected roaches in new images w/ accuracy of 95%

Research Question

- **Research question:** Do the weights & biases of this roach detecting NN translate into the detection of roach key features visible to humans, like shape or color?
- **Hypothesis:** The roach detecting NN mimics human vision using roach-like shape and roach-like color as key features

Materials

- Google Colab, a platform with free GPUs to run code faster
- Keras, a Python open-source NN library
- LeNet, a structure for convolutional (object detecting) NNs
- Grad-CAM, a software that generates heat maps which enable the programmer to see what NNs are detecting

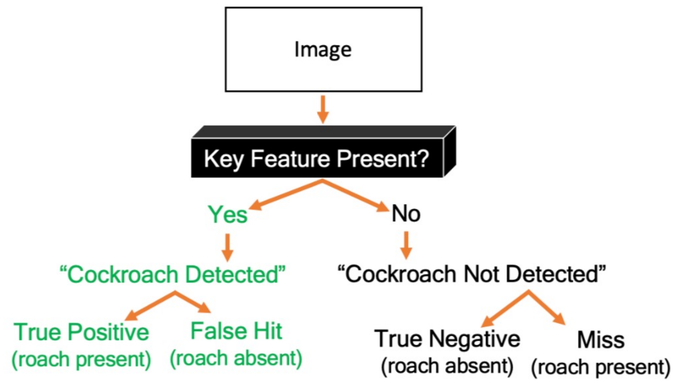
Photo Credits: All pictures were taken by me and all diagrams were made by me in PowerPoint

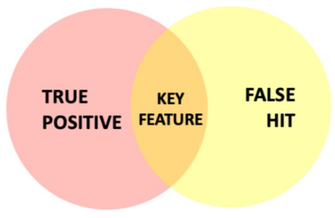
Procedure

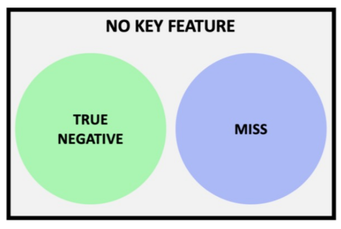
Finding the NN's key features:

- Using the NN's detection results there are 4 ways to classify an image
- The network can:
  1. Correctly detect a roach in an image **(true positive)**
  2. Incorrectly detect a roach in an image **(false hit)**
  3. Correctly detect no roaches in an image **(true negative)**
  4. Incorrectly detect no roaches in an image **(miss)**

- For roach to be detected in an image → key features must be present
- Roach **detected** in *true positives* & *false hits* → key features **present**
- Roach **NOT** detected in *true negatives* or *misses* → key features **absent**
- This means key features are present in every *true positive* and *false hit* but absent in every *true negative* and *miss*

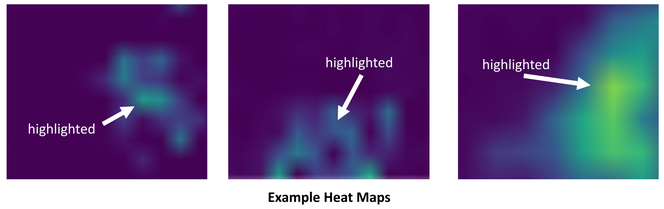






Confirming the NN's key features:

- When image is inputted, heat map of image is outputted
- Heat maps highlight features in input image acknowledged by NN
- For roach to be detected in an image → key features must be acknowledged/highlighted in image's heat map
- Roach **detected** in *true positives* & *false hits* → their heat maps **must** acknowledge/highlight key features
- Roaches **NOT** detected in any *true negatives* or *misses* → features acknowledged/highlighted in their heat maps **cannot** be *key* features



Example Heat Maps

Results

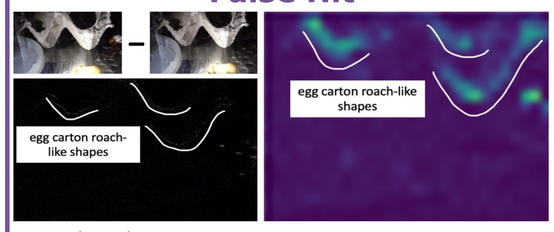
- Results show how presence of the roaches' 2 main features, shape and color, affect NN's detection
- An example subtracted input image & its heat map is given for all 4 image types

True Positive

Subtracted Input Image — Heat Map

- Subtracted input image has roach shape and color
- Heat map acknowledges roach shape and color
- NN detects roach

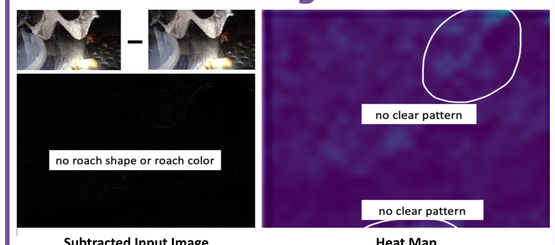
False Hit



Subtracted Input Image — Heat Map

- Input has roach-like shapes (egg carton), but no color
- Heat map acknowledges roach-likes shapes (not color)
- NN still detects roach in absence of color

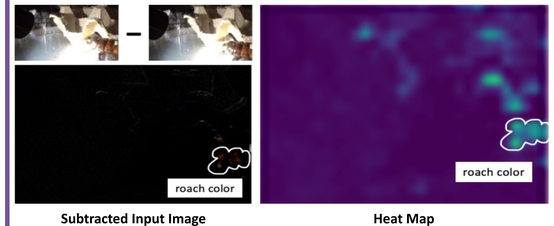
True Negative



Subtracted Input Image — Heat Map

- Input image doesn't have roach-like color or shape
- Heat map doesn't acknowledge any clear features
- NN doesn't detect roach

Miss



Subtracted Input Image — Heat Map

- Input image has roach color, but no shape
- Heat map acknowledges color (not shape)
- NN doesn't detect roach despite presence and acknowledgment of color

Results

4 Types of Input Images & Heat Maps

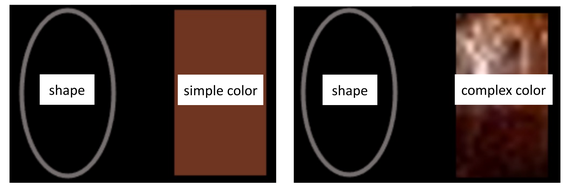
- All true positives & false hits have roach shape
- No true negative or miss have roach shape
- True positive & false hit heat map acknowledge roach shape
- True negative & miss heat map don't acknowledge roach shape

Therefore, roach shape is the key feature

Results

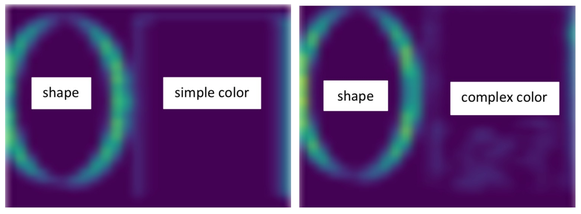
Verification Image

- To verify that roach shape is the key feature I made the 2 images below
- The verification images have figures representing 2 possible key features, roach shape & roach color



Verification Images

- Grey ovals represent only roach shape
- 1st rectangle represents only simple roach color
- 2nd rectangle represents only complex roach color
- Inputting verification images into NN made heat maps (examples below)



Heat Maps of Verification Images

- Oval highlighted brightly → roach shape is key feature
- Rectangle highlighted dimly → roach color isn't key feature
- NN isn't as flexible as human vision
  - Can't understand large spectrum of browns is a feature

Conclusion

- Object detecting NNs mimic human vision by learning & looking for visual key features
  - This NN uses roach shape as its key feature
- Analyzing and incorporating unused key features can make NNs more effective without extra training images or CPUs/GPUs
  - Less training images & CPUs/GPUs → more NN applications
  - Application examples: Cancer-detecting NN, obstacle detection for self-driving cars, facial recognition for security
- This 140 node roach-detecting NN can't use color as key feature
- Our primary visual cortexes in our brains use color, shape, & texture as key features & have ~140 million neurons
- If NN structures are built to mimic the structure of our visual cortex, they may find more visual key features that we use, like color and texture