Occlusion as Augmentation: Overview

Occlusion as augmentation is a data augmentation technique that involves deliberately occluding parts of an image (e.g., by adding patches, masks, or noise) during training to improve the robustness of a deep learning model. By training the model on occluded images, the model learns to focus on globally relevant features rather than overfitting to specific local details.

Why Use Occlusion for Augmentation?

1. **Robustness to Missing Data**: Models become more resilient to missing or incomplete data, such as occlusions caused by real-world objects (e.g., hands covering a face).
2. **Encouraging Global Understanding**: It forces the model to rely on spatially distributed features rather than over-relying on any single region.
3. **Regularization**: Occlusion acts as a form of regularization, similar to dropout, helping prevent overfitting.

Techniques for Occlusion Augmentation

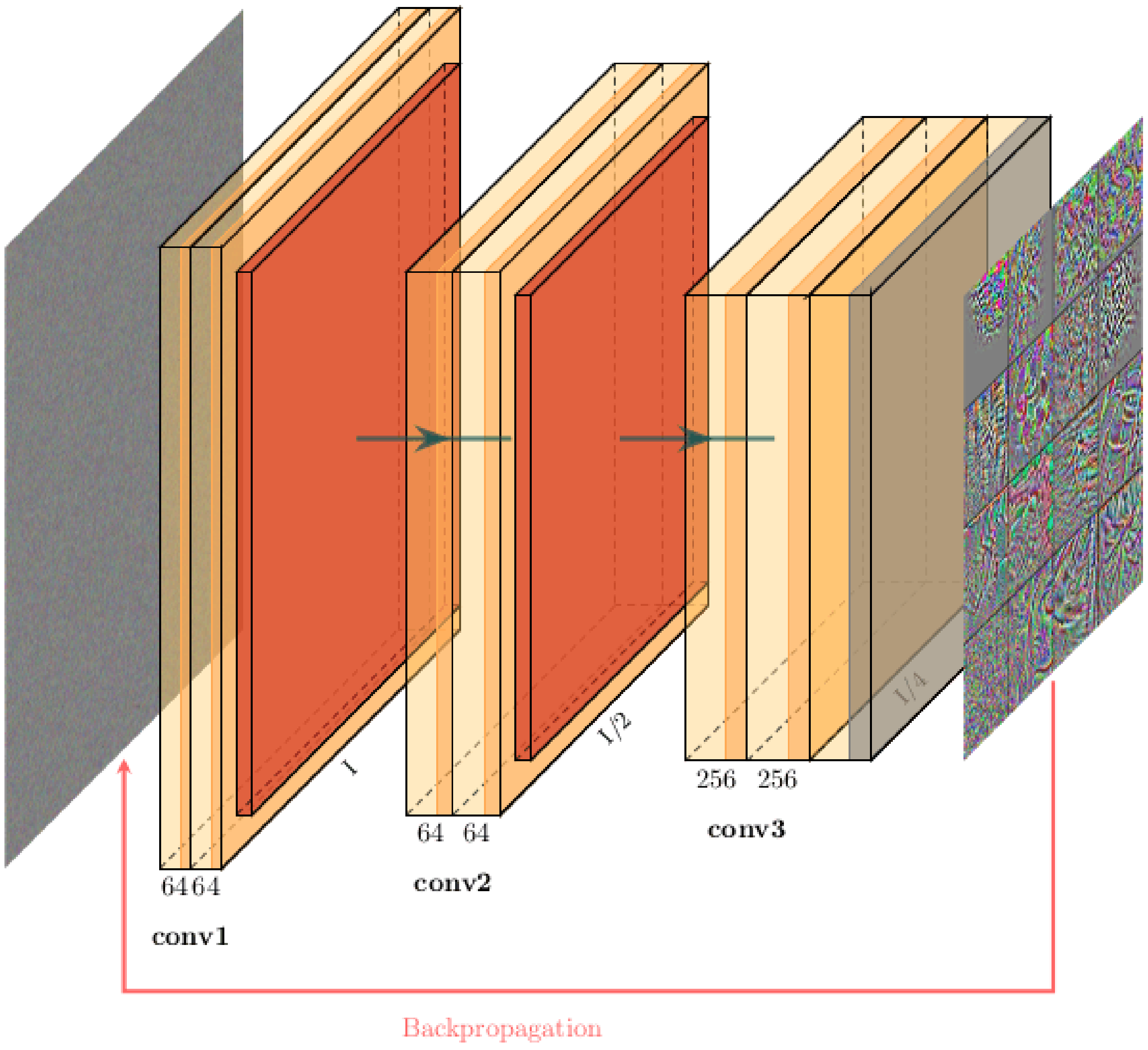
1. **Random Patch Occlusion (CutOut)**: Introduced as a regularization technique, this method involves replacing random patches of the image with constant values (e.g., zeros or a gray patch).
2. **Hide-and-Seek**: Occludes random regions in a grid pattern during training, with some regions visible and others hidden.
3. **MixUp/Masking**: Blends two images or replaces regions with segments from other images, creating a hybrid augmentation effect.
4. **Random Erasing**: Erases rectangular regions of an image with random sizes and positions.

Occlusion Sensitivity

The idea behind Occlusion Sensitivity is to **hide parts of the image and see the impact on the neural network's decision** for a specific class.

On the animation below, we run a blue patch over a cat image and extract the confidence at each step. When the patch goes over the cat, confidence drops, so we can identify the region behind the patch as hot. When the patch does not occlude the cat, the confidence stays even or eventually goes up.

This happens because we potentially hide elements that degrade the performance.



Heatmap generation process for class Cat

The heatmap generated carries the information "Does this part of the image helps to improve confidence". Here, the resolution is pretty poor. You can improve it by varying the patch size to capture influences from micro to macro zones of the image.

The process to generate the heatmap is decomposed simply :

- Create a batch of images with patches applied
- Run predictions
- Save confidence for the target class

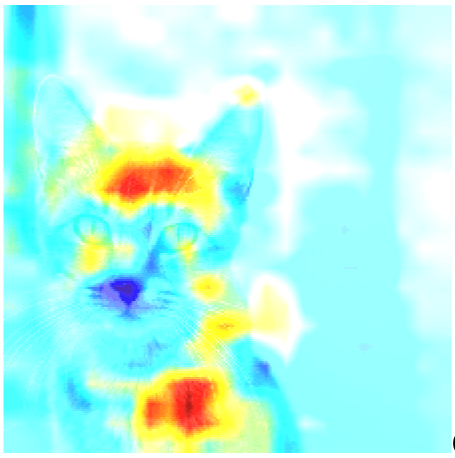
- Regroup confidences in the resulting map

```
import numpy as np


import tensorflow as tf

# Create function to apply a grey patch on an image
def apply_grey_patch(image, top_left_x, top_left_y, patch_size):
    patched_image = np.array(image, copy=True)
    patched_image[top_left_y:top_left_y + patch_size,
top_left_x:top_left_x + patch_size, :] = 127.5

    return patched_image

# Load image
IMAGE_PATH = './cat.jpg'
img = tf.keras.preprocessing.image.load_img(IMAGE_PATH,
target_size=(224, 224))
img = tf.keras.preprocessing.image.img_to_array(img)

# Instantiate model
model = tf.keras.applications.resnet50.ResNet50(weights='imagenet',
include_top=True)

CAT_CLASS_INDEX = 281  # Imagenet tabby cat class index
PATCH_SIZE = 40

sensitivity_map = np.zeros((img.shape[0], img.shape[1]))

# Iterate the patch over the image
for top_left_x in range(0, img.shape[0], PATCH_SIZE):
    for top_left_y in range(0, img.shape[1], PATCH_SIZE):
        patched_image = apply_grey_patch(img, top_left_x, top_left_y,
PATCH_SIZE)
        predicted_classes = model.predict(np.array([patched_image]))[0]
        confidence = predicted_classes[CAT_CLASS_INDEX]

        # Save confidence for this specific patched image in map
        sensitivity_map[
            top_left_y:top_left_y + PATCH_SIZE,
            top_left_x:top_left_x + PATCH_SIZE,
        ] = confidence
```

view rawocclusion_sensitivity.py hosted with ❤ by GitHub

Occlusion Sensitivity Implementation

Note: This code translates the algorithm logic, but should be optimized by first generating all the patched images and then running the predictions in batches.


Occlusion Sensitivity Map