CSE4261: Neural Network and Deep Learning

Lecture: 24.06.2025

Sangeeta Biswas, Ph.D.
Associate Professor,
University of Rajshahi, Rajshahi-6205, Bangladesh

Gradient Descent Algorithm for Parameter Updating

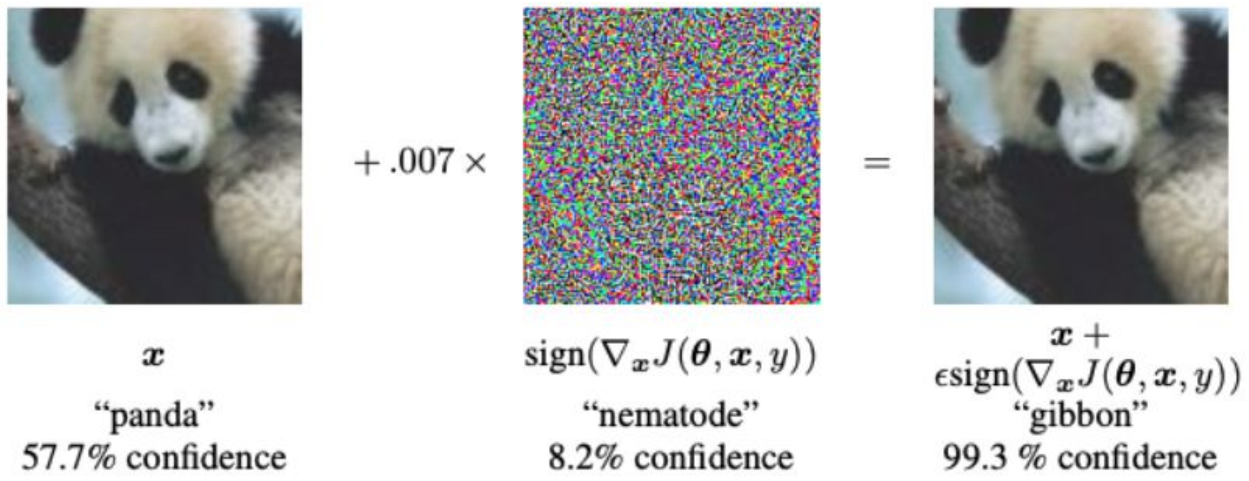
Gradient is calculated for cost function with respect to weights and biases for updating their values.

$$w^{[l]} = w^{[l]} - \alpha \frac{\partial C}{\partial w^{[l]}}$$

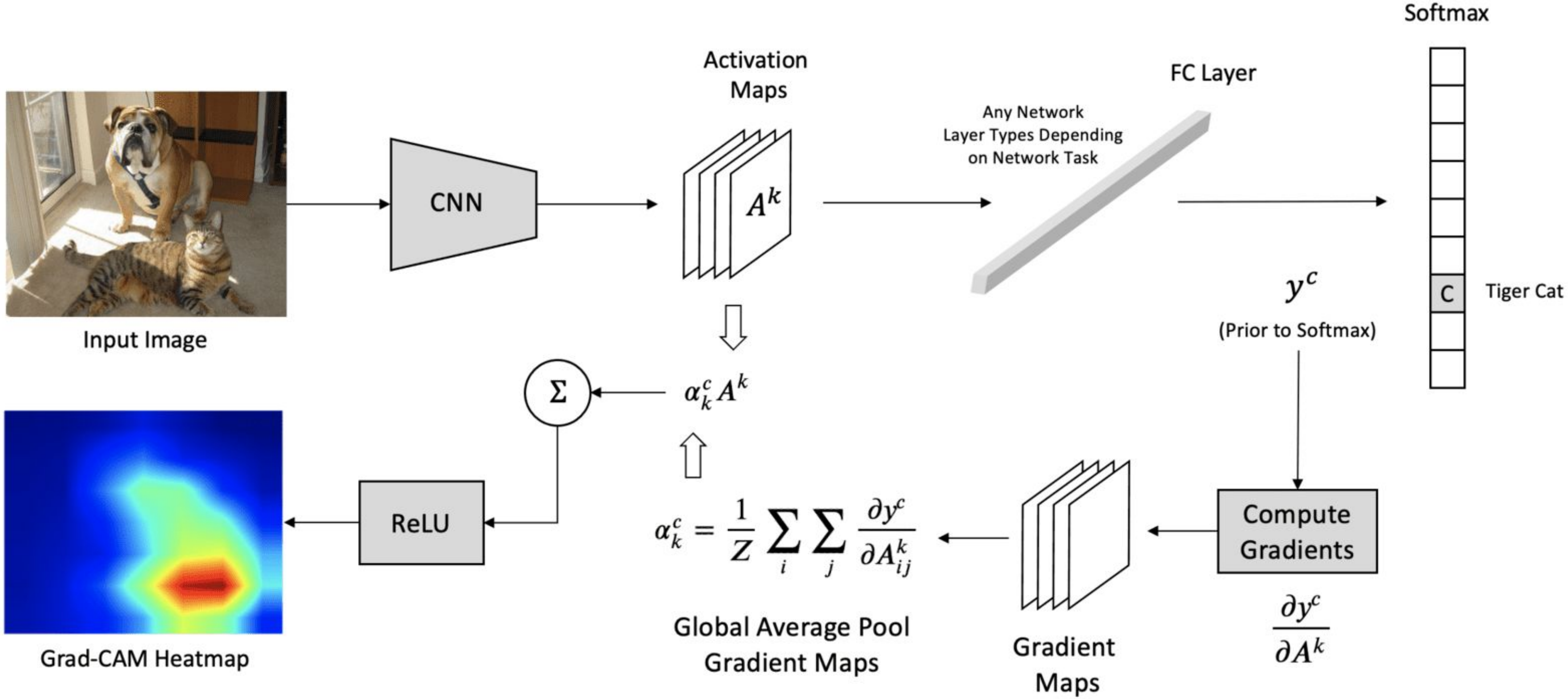
$$b^{[l]} = b^{[l]} - \alpha \frac{\partial C}{\partial b^{[l]}}$$

Adversarial Attack by Fast Gradient Signed Method (FGSM)

- FGSM uses the gradients of the loss with respect to the input image to create a new image that maximises the loss.
- Adversarial image, $adv\_x = x + \epsilon * \text{sign}(\nabla_x J(\theta, x, y))$

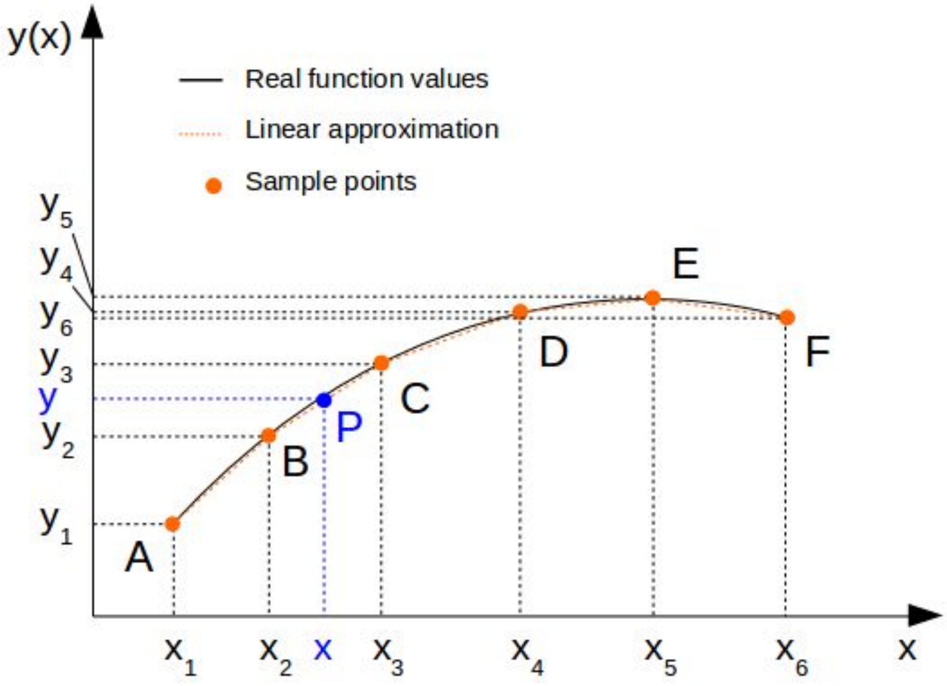


Grad-CAM (Gradient weighted Class Activation Mapping)



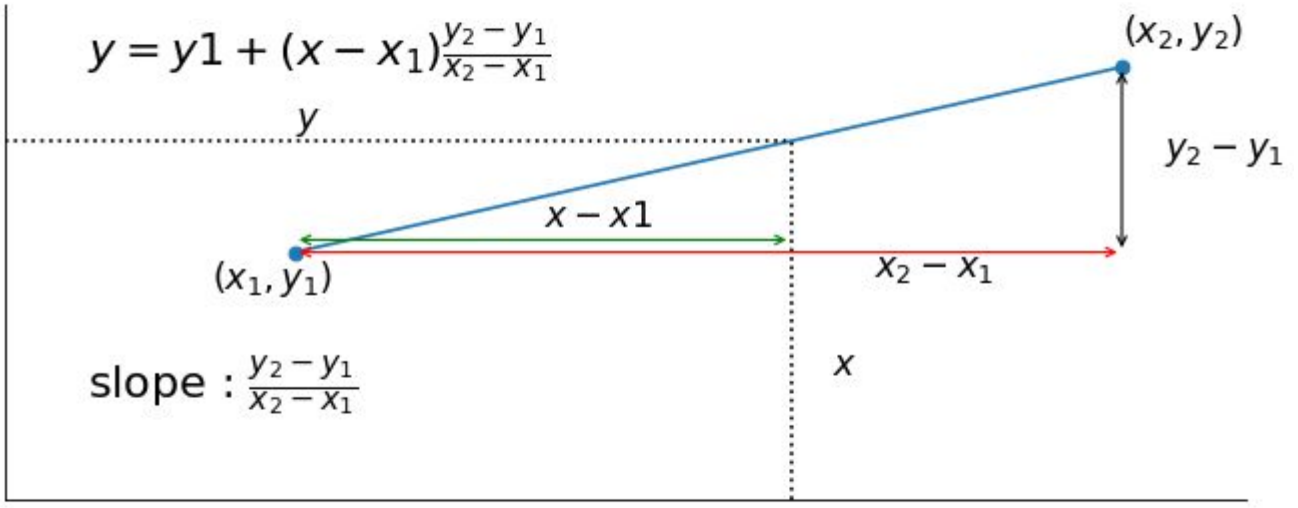
Interpolation

- It is a method to estimate the value of a function at a point within the range of a known set of data points.
- Given a set of data points (x1, y1), ..., (x_n, y_n), interpolation finds the value of y for a given x that lies within the range of x_1 and x_n



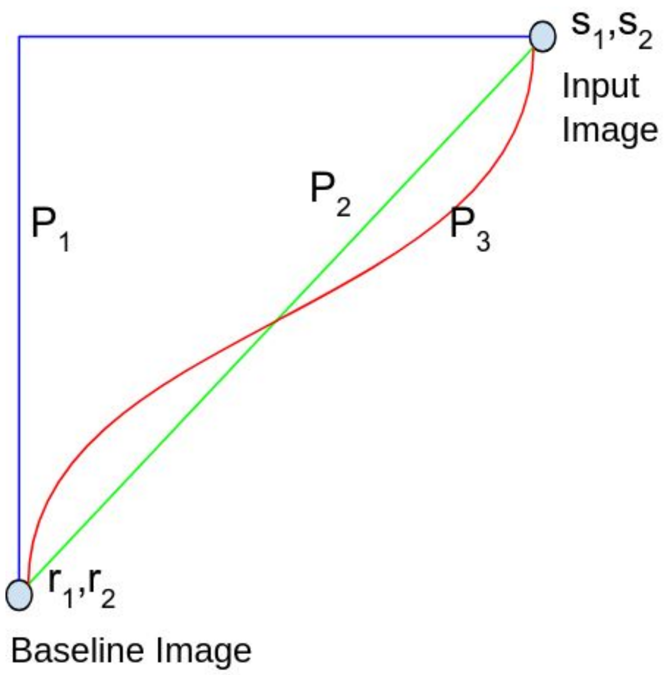
Linear Interpolation

- It finds an intermediate value between two known values by assuming a linear relationship between them.

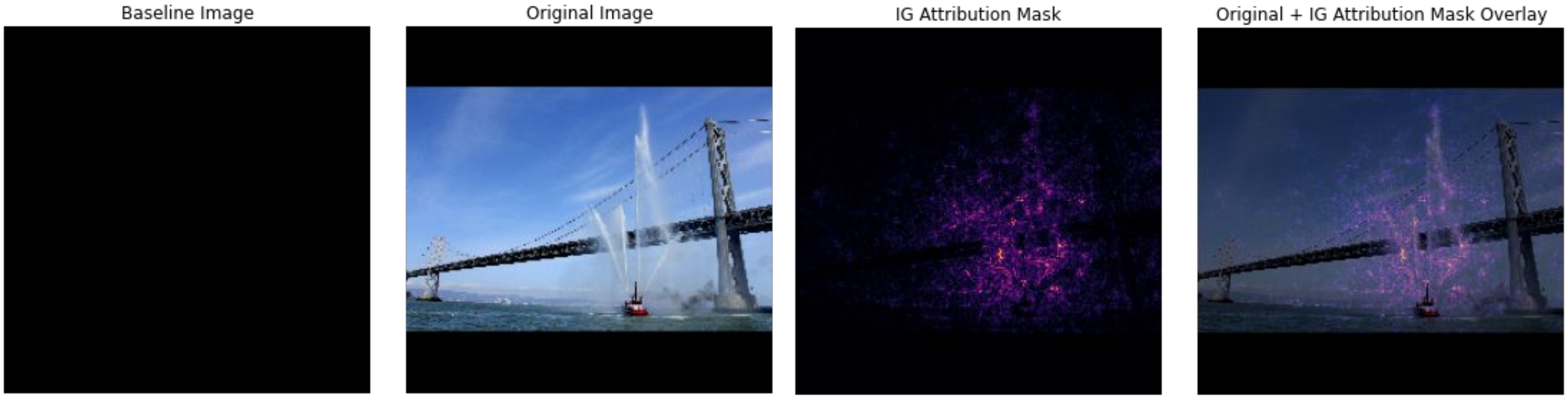


Integrated Gradients (IG) Method as an XAI Technique

- In IG, a baseline image is used with the input image.
- Linear interpolation is used to approximate the integral of gradients along a path between a baseline and the input instance (i.e., path P2).
- The integral is approximated by summing the gradients at multiple interpolated points along this line.



Integrated Gradients (IG) Method as an XAI Technique



- Tutorial: https://www.tensorflow.org/tutorials/interpretability/integrated_gradients

Example of Linear Interpolation

- Five-step interpolation between the baseline x’ and the input image x

Formula of IG

$$IntegratedGrads_i^{approx}(x) ::= (x_i - x'_i) \times \sum_{k=1}^{m} \frac{\partial F(x' + \frac{k}{m} \times (x - x'))}{\partial x_i} \times \frac{1}{m}$$

i = feature (individual pixel)

x = input (image tensor)

x’ = baseline (image tensor)

k = scaled feature perturbation constant

m = number of steps in the Riemann sum approximation of the integral

$(x_i\text{-}x'_i)$= a term for the difference from the baseline

F = model prediction function

Formula of IG

$$IntegratedGrads_i^{approx}(x) ::= (x_i - x_i') \times \sum_{k=1}^{m} \frac{\overbrace{\partial F(\text{interpolated images})}^{\text{compute gradients}}}{\partial x_i} \times \frac{1}{m}$$

$\frac{\partial F}{\partial x_i}$ = gradient

- The gradient tells us which pixels have the strongest effect on the model's predicted class probabilities.

Gradient Computation for IG by Tensorflow

```
def compute_gradients(images, target_class_idx):
    with tf.GradientTape() as tape:
        tape.watch(images)
        logits = model(images)
        probs = tf.nn.softmax(logits, axis = -1)[:, target_class_idx]
    return tape.gradient(probs, images)
```

In Grad-CAM:
model.layers[-1].activation = None # Remove last layer's softmax

Alternative Baseline

(a) Gaussian Baseline (b) Blur Baseline (c) Max Distance Baseline (d) Uniform Baseline