197 Final Research Project

Angelina Jordan

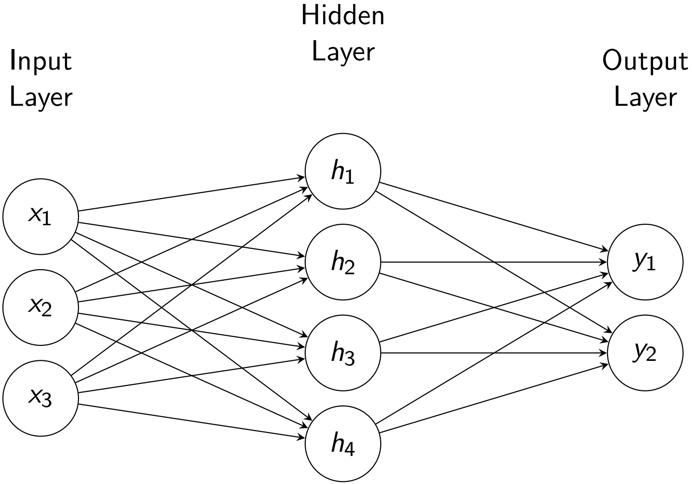
June 2025

What Is a Neural Network?

In Simple Terms:

- A neural network is a system that learns patterns in data.
- It takes input (like numbers or images) and passes it through layers to make predictions.
- Each layer transforms the data step by step.

Simple Feedforward Neural Network



More about Neural Netwroks

- It consists of layers of connected units (neurons), typically represented as:

$$f(x) = (f_L \circ f_{L-1} \circ \cdots \circ f_2 \circ f_1)(x)$$

  where each f_l is a layer function.
- In a simple MLP (Multilayer Perceptron), each neuron computes:

$$z_j = \sum_{i=1}^{M} w_{ji} x_i + b_j, \quad y_j = \sigma(z_j)$$

  where σ is a nonlinear activation function (e.g., ReLU).
- The network is trained end-to-end to approximate a function $y = f(x; \theta)$.

Why Are Neural Networks Important?

- Neural networks are the core building blocks of deep learning.
- Deep learning uses large neural networks with many layers to learn complex patterns in data.
- These networks can automatically extract useful features from raw input — such as images, audio, or text.
- This ability has led to breakthroughs in areas like computer vision, natural language processing, and robotics.

Attribution Methods in Neural Networks

What Are Attribution Methods?

- Explain which input features contribute most to a model's prediction.
- Help build trust, transparency, and diagnose models.
- We'll test Saliency, Gradient $\times$ Input, Integrated Gradients, and Shapley Values.

Sensitivity Analysis

$$R_i^c(x) = \left| \frac{\partial S_c(x)}{\partial x_i} \right| \tag{1}$$

Description:
Measures how much the output changes with an infinitesimal change in input feature x_i.
Useful for assessing local sensitivity but can be noisy.

Gradient × Input

$$R_i^c(x) = \frac{\partial S_c(x)}{\partial x_i} \cdot x_i \tag{2}$$

Description:
Scales the gradient by the input feature value.
Reflects the feature's actual contribution to the output.

Integrated Gradients

$$R_i^c(x) = (x_i - \bar{x}_i) \cdot \int_{\alpha=0}^{1} \frac{\partial S_c(\tilde{x})}{\partial \tilde{x}_i}\Big|_{\tilde{x}=\bar{x}+\alpha(x-\bar{x})} d\alpha \tag{3}$$

Description:
Computes the average gradient as the input changes from a baseline $\bar{x}$ to the input x.

Baseline: A reference input (e.g., all zeros) used as the starting point to compare the actual input in Integrated Gradients.

Shapley Values

$$R_i = \sum_{S \subseteq P \setminus \{i\}} \frac{|S|!(|P| - |S| - 1)!}{|P|!} \left[\hat{f}(S \cup \{i\}) - \hat{f}(S)\right] \qquad (4)$$

Description:
Provides a theoretically fair distribution of the model output across features.
Requires exponential computation in number of features.

Deep Approximate Shapley Propagation (DASP)

$$\mathbb{E}[R_i^c] = \frac{1}{N} \sum_{k=0}^{N-1} \mathbb{E}_k[R_{i,k}^c] \tag{5}$$

$$\mathbb{E}_k[R_{i,k}^c] = \mathbb{E}_{S \subseteq P \setminus \{i\}, |S|=k}[f_c(x_{S \cup \{i\}})] - \mathbb{E}_{S \subseteq P \setminus \{i\}, |S|=k}[f_c(x_S)] \tag{6}$$

Description:
Approximates Shapley values efficiently using random coalitions. Designed for deep neural networks.

Key Idea: Estimate Shapley values efficiently by propagating uncertainties through the network rather than sampling all subsets.

Saliency Maps

Formula:

$$R_i^c(x) = \left| \frac{\partial S_c(x)}{\partial x_i} \right|$$

Implementation Steps:

- Set model to evaluation mode.
- Enable gradient tracking on the input.
- Perform a forward pass to get output score $S_c(x)$.
- Compute the gradient of that score with respect to each input x_i.
- Take the absolute value of the gradient as the attribution score.

Gradient × Input

Formula:

$$R_i^c(x) = \frac{\partial S_c(x)}{\partial x_i} \cdot x_i$$

Implementation Steps:

- Compute the gradient of the model's output with respect to input.
- Multiply each gradient value by its corresponding input value.
- The result reflects each input's contribution to the output.

Integrated Gradients

Formula:

$$R_i^c(x) = (x_i - \bar{x}_i) \cdot \int_0^1 \frac{\partial S_c(\bar{x} + \alpha(x - \bar{x}))}{\partial x_i} \, d\alpha$$

Implementation Steps:

- Choose a baseline input $\bar{x}$ (e.g., all zeros).
- Interpolate inputs between baseline and actual input.
- At each step, compute gradients of output w.r.t. input.
- Average the gradients and multiply by $(x - \bar{x})$.

Shapley Values (Sampling)

Formula:

$$R_i = \sum_{S \subseteq P \setminus \{i\}} \frac{|S|!(|P| - |S| - 1)!}{|P|!} \left[\hat{f}(S \cup \{i\}) - \hat{f}(S) \right]$$

Implementation Steps:

- Define a baseline input (e.g., zeros).
- Sample many subsets S of features without i.
- For each subset, compute model output with and without feature i.
- Compute the difference and weight it based on subset size.
- Average the results to estimate the contribution of feature i.

Deep Approximate Shapley Propagation

Approximate Shapley Value Formula:

$$R_i \approx \mathbb{E}_{S \subseteq P \setminus \{i\}} \left[\hat{f}(S \cup \{i\}) - \hat{f}(S) \right]$$

where expectations are approximated via probabilistic propagation through layers.

Implementation Steps:

- Represent input features as probabilistic distributions conditioned on presence/absence.
- Propagate these distributions forward through each neural network layer using uncertainty propagation techniques.
- Approximate the marginal contributions of each feature without enumerating all subsets.
- Aggregate propagated contributions to compute an efficient estimate of Shapley values.

Example Output

Model Input

$$\texttt{input} = [0.5, -0.5]$$

Attribution Results

Method	Feature 1	Feature 2
Saliency	0.15	0.02
Gradient $\times$ Input	0.12	-0.03
Integrated Gradients	0.10	-0.01
Shapley Values	0.08	0.00