XAI Techniques Cheatsheet

Yvo Keller

February 5, 2025

1 Tabular Methods

1.1 Shapley Values

1.1.1 Overview

Shapley values provide a way to fairly distribute the prediction among features by considering all possible feature combinations.

1.1.2 Key Formula

The Shapley value for feature i is:

$$\phi_i(x) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|! \times (|N| - |S| - 1)!}{|N|!} \left(f_\theta(S \cup \{i\}) - f_\theta(S) \right) \tag{1}$$

where:

- N is the set of all features
- S is a subset of features excluding feature i
- f_θ is the model prediction
- $|S|$ is the size of subset S
- $|N|$ is the total number of features

1.1.3 Calculation Process

1. Select an instance to explain
2. For each feature:
   (a) Generate all possible feature coalitions excluding the target feature
   (b) For each coalition:
      i. Calculate model prediction with and without target feature
      ii. Compute marginal contribution
      iii. Weight contribution by coalition size
   (c) Sum weighted contributions
3. Average contributions over all permutations

1.1.4 Properties

- **Efficiency**: Sum of Shapley values equals model output minus baseline
- **Symmetry**: Equal contribution features receive equal Shapley values
- **Dummy**: Features with no marginal contribution get zero Shapley value
- **Additivity**: Values can be computed independently and summed

1.1.5 Intuitive Example: Ice Cream Shop

Let's understand Shapley values through a practical example of predicting ice cream sales.

Setup Consider a model predicting daily ice cream sales with features:

- x_1 = Day of the week
- x_2 = Number of flights arriving
- x_3 = Temperature
- x_4 = Total opening hours

Calculation Process To calculate the Shapley value for temperature (x_3):

1. **Select a sample**: Choose a specific day's data point
2. **Choose baseline**: Select a reference point (usually average values)
3. **Generate permutation**: e.g., (x_4, x_1, x_3, x_2)
4. **Calculate marginal contributions**:
   - Start with baseline prediction: f_{base}
   - Add features one by one:

$$\begin{aligned}
&f(x_4)\\
&f(x_4, x_1)\\
&f(x_4, x_1, x_3) \leftarrow \text{Temperature added here}\\
&f(x_4, x_1, x_3, x_2)
\end{aligned}$$

   - Temperature's contribution $= f(x_4, x_1, x_3) - f(x_4, x_1)$
5. **Repeat**: Do this for multiple permutations
6. **Average**: The Shapley value is the average contribution across permutations

Interpretation The final Shapley value for temperature tells us:

- Positive value: Higher temperatures increase ice cream sales
- Negative value: Higher temperatures decrease sales
- Magnitude: Size of temperature's impact on the prediction

1.1.6 Monte Carlo Approximation

For large feature sets, exact computation becomes infeasible. Monte Carlo approximation:

1. Sample random feature permutations
2. Calculate marginal contributions for each permutation
3. Average results over all samples

1.1.7 SHAP Summary Plot (Beeswarm)

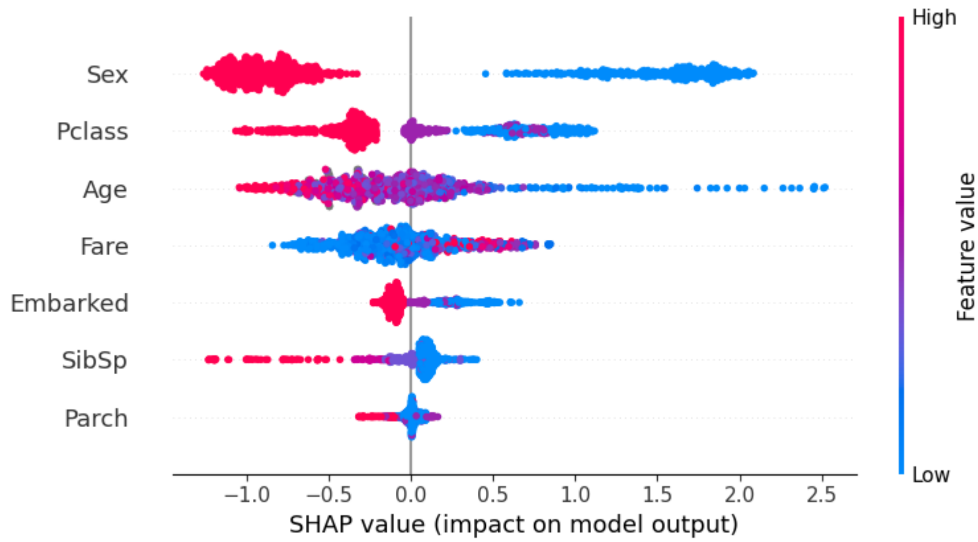

Figure 1: SHAP Summary Plot showing feature importance and impact distribution

How to Read the Plot

- **Target**: The model predicts passenger survival probability (0 = died, 1 = survived)
- **Feature Ranking**: Features are ordered by their absolute impact on survival prediction (most important at top)
- **Impact**: The x-axis shows SHAP values (impact on survival probability)
  - Positive values (right) increase survival probability
  - Negative values (left) decrease survival probability
  - The magnitude shows how strongly it affects the prediction
- **Distribution**: Each dot is one passenger in the dataset
- **Color Coding**:
  - For categorical features (like Sex): Blue/Red represent different categories
  - For numerical features: Red = high values, Blue = low values
  - Look at the pattern of colors and their position to understand the relationship
- **Spread**: Horizontal spread shows range of impact across different passengers

Example Interpretation Looking at the Titanic survival predictions:

- **Sex**: Strongest predictor of survival
  - Being female (blue) strongly increased survival probability (dots on right)
  - Being male (red) strongly decreased survival probability (dots on left)
  - Clear separation shows this was the most decisive factor
- **Pclass**: Second most important feature
  - First class (blue = low class number) increased survival probability
  - Third class (red = high class number) decreased survival probability
  - Shows clear socioeconomic divide in survival chances
- **Age**: Complex relationship
  - Younger passengers (blue) show mixed effects but often positive
  - Older passengers (red) tend toward negative impact on survival
  - Wide spread suggests age interacted with other factors
- **Fare**: Shows price paid for ticket
  - Higher fares (red) tend toward positive impact
  - Lower fares (blue) tend toward negative impact
  - Considerable overlap in the middle ranges
- **Embarked**: Port where passenger boarded
  - Different ports (shown by different colors) had varying impacts
  - Moderate but clear influence on survival chances
- **SibSp & Parch**: Number of family members aboard
  - More family members (red) shows mixed effects
  - Fewer family members (blue) also shows varied impact
  - Wide spread suggests complex relationship with survival

1.2 Partial Dependence Plots (PDP)

1.2.1 Overview

PDPs show how a feature affects predictions on average, while marginalizing over all other features.

1.2.2 Intuitive Example

Consider our ice cream sales model:

- To create a PDP for temperature:
  1. Pick a temperature value (e.g., 25°C)
  2. For every data point, set temperature to 25°C
  3. Get model predictions for all these modified points
  4. Average these predictions
  5. Repeat for different temperature values
  6. Plot temperature vs. average predictions

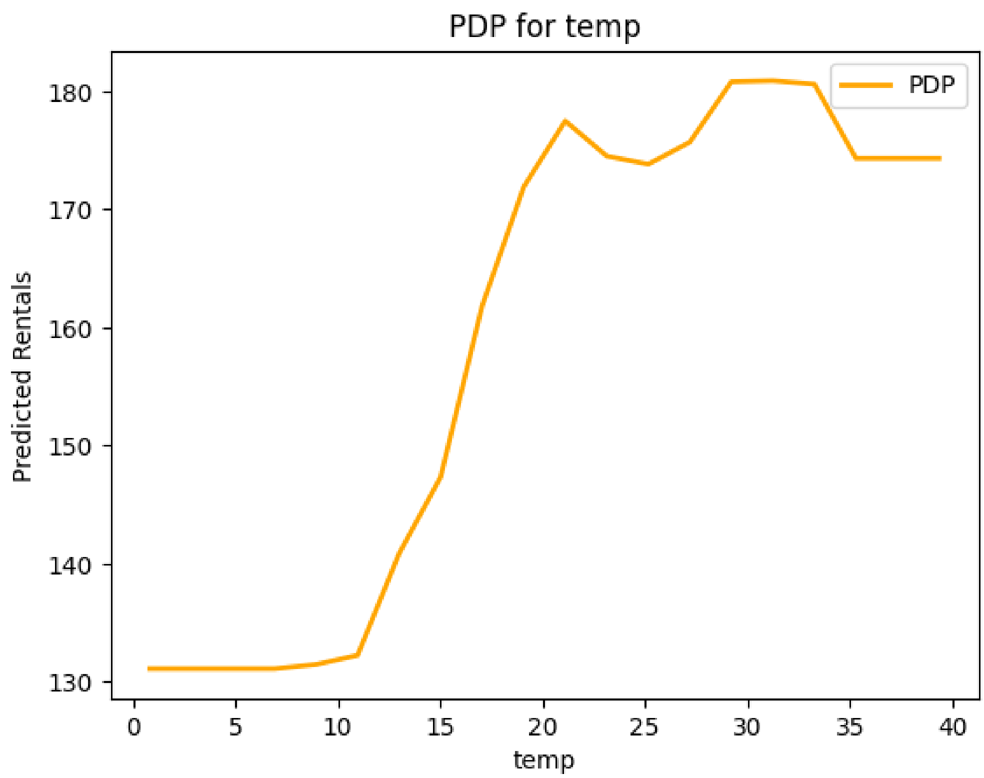

Figure 2: Example of a Partial Dependence Plot (PDP)

1.2.3 Interpretation

- Slope shows relationship strength
- Shape reveals non-linear effects
- Flat regions indicate no impact
- Limitations: Can miss feature interactions

1.3 Individual Conditional Expectation (ICE) Plots / ICPs

1.3.1 Overview

ICE plots (also known as Individual Conditional Expectation Plots or ICPs) extend PDPs by showing how predictions change for individual instances, revealing heterogeneous effects hidden by PDPs.

1.3.2 Intuitive Example

Using our ice cream model:

- For each individual day in our dataset:
  1. Keep all features fixed except temperature
  2. Vary temperature across its range
  3. Plot prediction line for this specific day
- Result: Multiple lines, one per instance
- PDP would be the average of these lines

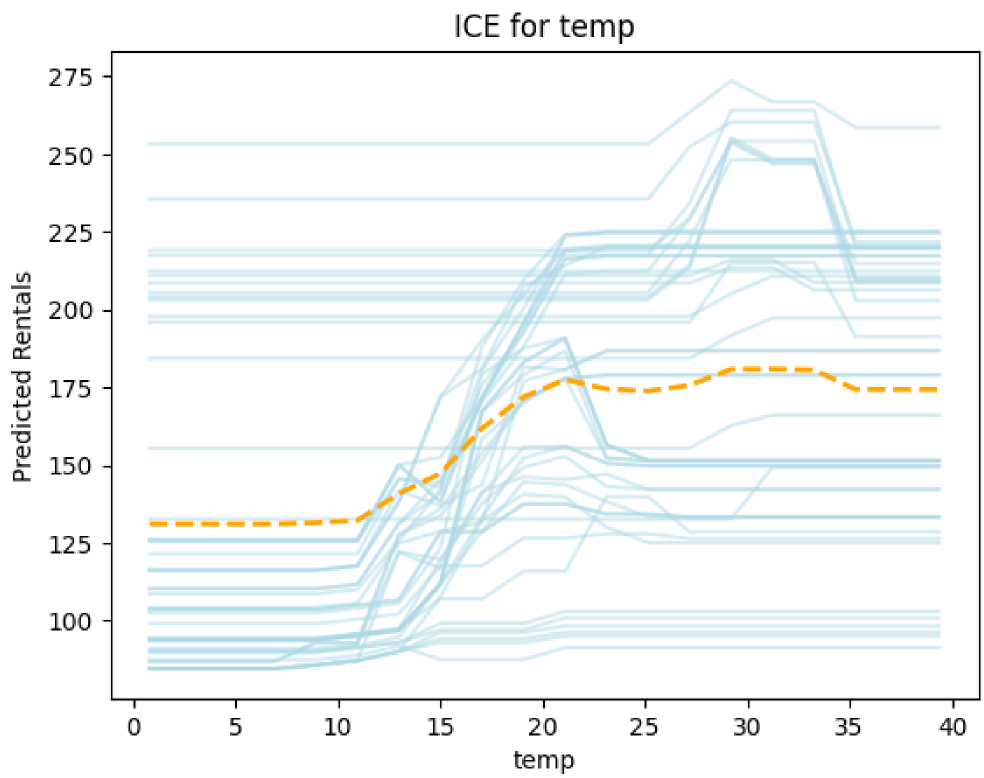

Figure 3: Example of Individual Conditional Expectation (ICE) plots

1.3.3 Key Insights

- Diverging lines suggest feature interactions
- Parallel lines indicate consistent effects
- Crossing lines show complex relationships
- More informative than PDP alone

1.3.4 When to Use

- Feature interaction analysis
- Detecting heterogeneous effects
- Model behavior validation
- Identifying outlier instances

2 Text Data Methods

2.1 LIME (Local Interpretable Model-agnostic Explanations)

2.1.1 Overview

LIME explains individual text predictions by approximating the complex model locally with an interpretable one.

2.1.2 How it Works

1. **Input**: Take a text instance to explain
2. **Perturbation**:
   - Create variations of the input by removing words
   - Keep track of which words are present/absent
3. **Prediction**: Get model predictions for all variations
4. **Local Model**:
   - Fit an interpretable model (e.g., linear) around the instance
   - Weight samples by proximity to original text
5. **Explanation**: Show which words contributed positively/negatively

2.1.3 Example Interpretation

For a sentiment analysis model:

- Positive words might be highlighted in green
- Negative words might be highlighted in red
- Weight of highlighting shows importance
- Example: "The movie was great but the ending was disappointing"

2.2 Gradient × Input

2.2.1 Overview

Gradient × Input identifies important words by computing how much a small change in each input token would affect the prediction.

2.2.2 Calculation

1. Calculate gradient of output with respect to input embeddings
2. Multiply gradient by the actual input embeddings
3. Aggregate for each token (usually L2 norm across embedding dimensions)

2.2.3 Properties

- **Advantages**:
  - Computationally efficient
  - Exact gradient computation
  - Works with any differentiable model
- **Limitations**:
  - Can be noisy
  - May not capture non-linear interactions well
  - Requires model gradients

2.3 Attention Matrices

2.3.1 Overview

Attention matrices visualize how different parts of the input text relate to each other in transformer-based models.

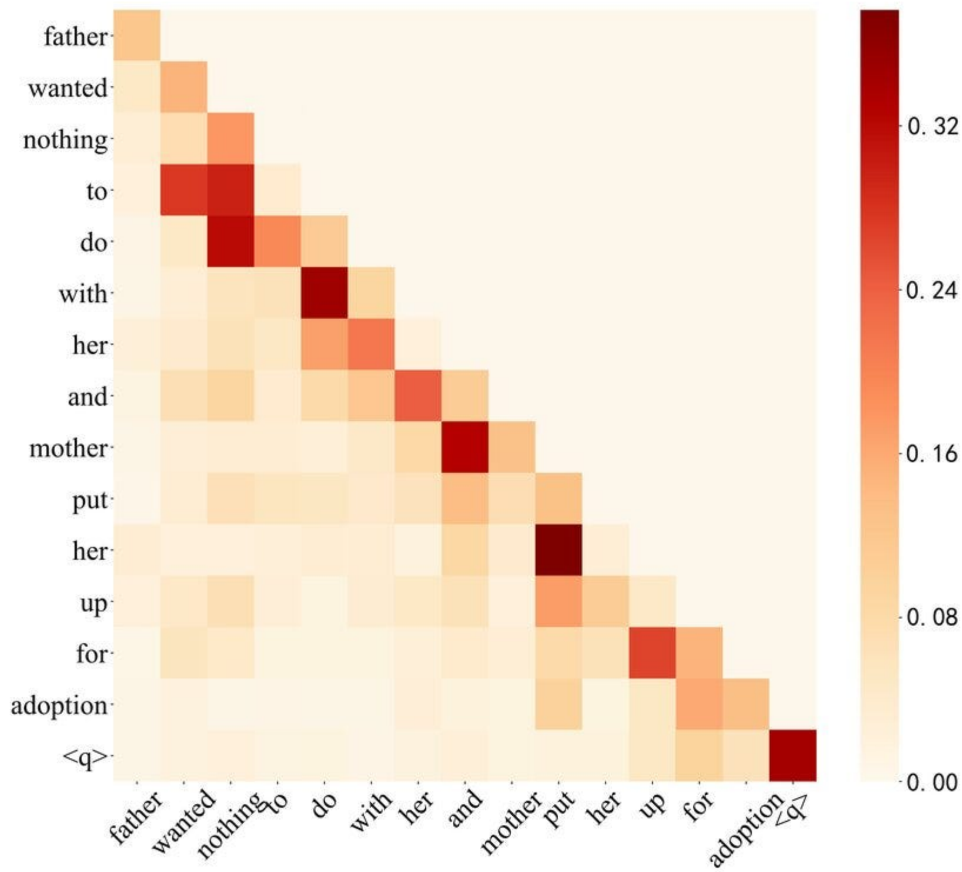

Figure 4: Example of an Attention Matrix showing token relationships in a transformer model

2.3.2 Components

- **Attention Heads**:
  - Each head learns different relationship patterns
  - Multiple heads capture different aspects of the text
- **Attention Weights**:
  - Show how much each token attends to other tokens
  - Higher weights indicate stronger relationships
- **Layers**:
  - Earlier layers often capture syntactic relationships
  - Later layers tend to capture semantic relationships

2.3.3 Visualization

- Usually shown as a heatmap matrix
- Rows: Source tokens
- Columns: Target tokens
- Color intensity: Attention weight
- Can be aggregated across heads or shown per head

2.3.4 Example Patterns

- **Diagonal**: Token attending to itself
- **Vertical stripes**: Important context words
- **Horizontal stripes**: Influential tokens
- **Blocks**: Related phrase chunks
- **Off-diagonal**: Long-range dependencies

2.3.5 Interpretation Tips

- Look for consistent patterns across multiple heads
- Compare patterns at different layers
- Consider linguistic relationships (e.g., subject-verb, coreference)
- Be cautious: attention $\neq$ causation
- Use in conjunction with other explanation methods

2.4 Embedding Visualization (T-SNE and PCA)

2.4.1 Overview

These techniques help visualize high-dimensional word/token embeddings in 2D/3D space, making it possible to understand relationships between words and how the model represents language.

2.4.2 Key Methods

- **PCA (Principal Component Analysis)**:
  - Linear dimensionality reduction
  - Preserves global structure and distances
  - Faster but may miss non-linear relationships
- **T-SNE (t-Distributed Stochastic Neighbor Embedding)**:
  - Non-linear dimensionality reduction
  - Better at preserving local clusters
  - Shows semantic relationships more clearly

2.4.3 Interactive Tool

The TensorFlow Embedding Projector (`projector.tensorflow.org`) allows:

- Interactive exploration of embeddings
- Switching between PCA and T-SNE views
- Finding nearest neighbors
- Visualizing word clusters and relationships

2.5 Language Interpretability Tool (LIT)

2.5.1 Overview

LIT is an interactive platform for analyzing NLP models, combining multiple interpretation techniques in one interface.

2.5.2 Key Features

- **Model Analysis**:
  - Attention visualization
  - Salience maps
  - Counterfactual generation
- **Dataset Exploration**:
  - Data point inspection
  - Slice analysis
  - Error analysis
- **Interactive Testing**:
  - Real-time predictions
  - Custom input testing
  - Model comparison

2.5.3 Use Cases

- Debug model predictions
- Identify dataset biases
- Compare model versions
- Explore model behavior systematically

3 Image Data Methods

3.1 Grad-CAM (Gradient-weighted Class Activation Mapping)

3.1.1 Core Concept

Grad-CAM answers the question: "Which regions of an image did the CNN focus on to make its prediction?" It does this by:

- Looking at the last convolutional layer's feature maps

- Weighting them based on their importance for the target class
- Creating a heatmap showing which image regions were most influential

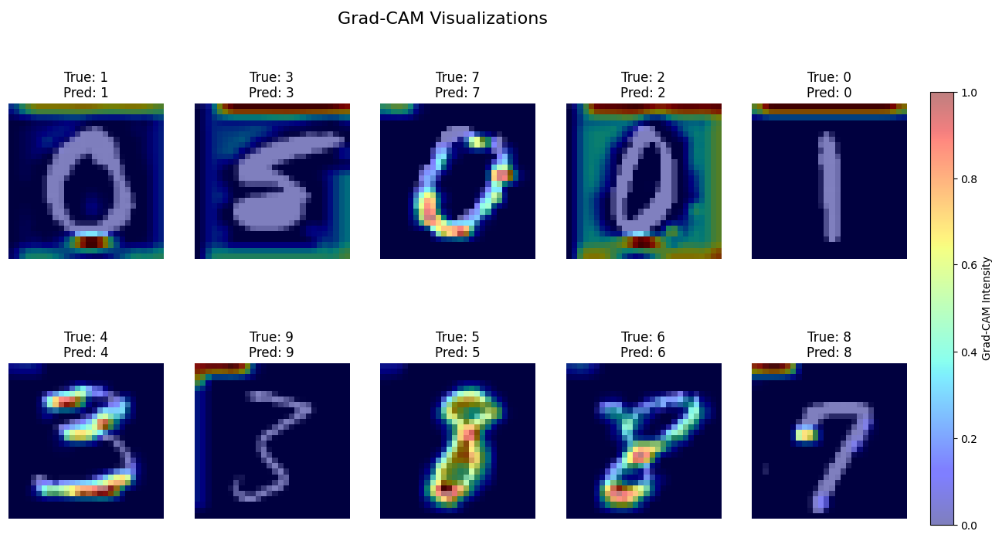

Figure 5: Example of Grad-CAM visualization showing regions of interest for classification

3.1.2 Why It Works

- Later conv layers capture high-level features
- Each feature map specializes in detecting different patterns
- Gradients tell us which patterns were important for the specific prediction
- Combining this information creates an interpretable visualization

3.1.3 The Process

1. Get feature maps from last conv layer for an input image
2. Calculate how important each feature map was for the prediction
3. Create weighted combination of feature maps
4. Upscale to original image size and overlay

3.1.4 Technical Implementation

Based on implementation in `notebooks/mini_challenge.ipynb`:

1. **Feature Extraction**:
   - Forward pass through CNN
   - Capture activations at chosen conv layer
2. **Importance Weights**:
   - Compute gradients for target class

- Average gradients for each feature map:

$$\alpha_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{2}$$

3. **Create Heatmap**:
    - Weight each feature map by its importance
    - Sum all weighted maps
    - Apply ReLU to highlight positive contributions:

$$L_{\text{Grad-CAM}}^c = \text{ReLU}\left(\sum_k \alpha_k A^k\right) \tag{3}$$

3.1.5 Reading the Output

- **Heatmap Colors**:
    - Red regions: Strongly support the prediction
    - Blue regions: Less important for the prediction
- **What to Look For**:
    - Does it focus on meaningful parts?
    - How focused/dispersed is the attention?
    - Does it match human intuition?

3.1.6 Key Strengths

- Simple yet effective visualization
- Works with any CNN architecture
- No need to modify or retrain the model
- Class-specific explanations

3.2 Integrated Gradients (IG)

3.2.1 Core Concept

Integrated Gradients answers: "How much did each input feature contribute to the prediction, compared to a baseline?" It does this by:

- Considering a path from a baseline (usually zeros) to the actual input
- Accumulating gradients along this path
- Showing which input features were most influential for the prediction

3.2.2 Why It Works

- Gradients at a single point might be noisy or saturated
- By integrating over a path, we capture the cumulative effect
- The baseline (e.g., black image) serves as a natural reference point
- Satisfies important theoretical properties (completeness, symmetry)

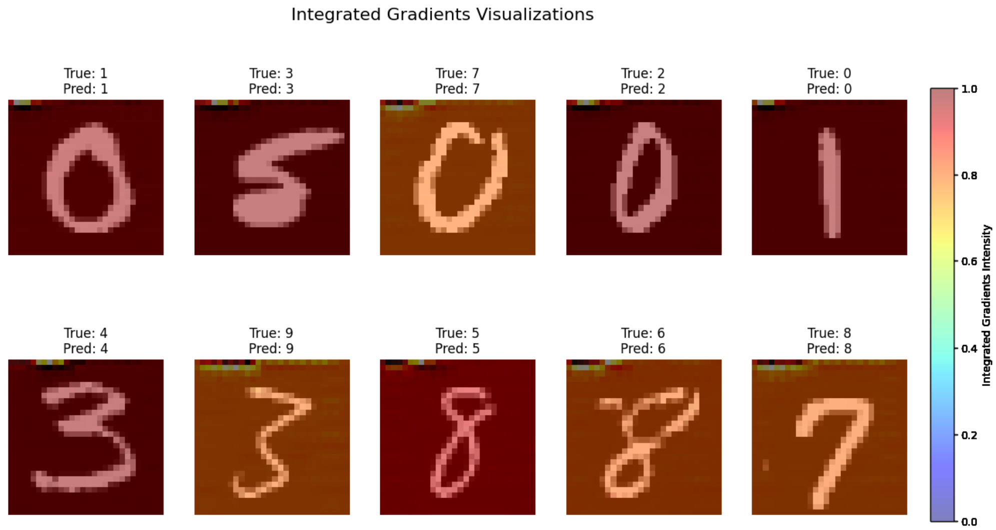

Figure 6: Example of Integrated Gradients showing pixel-wise contributions to the prediction

3.2.3 The Process

1. Define baseline (e.g., black image)
2. Create steps between baseline and input
3. Compute gradients at each step
4. Average the gradients
5. Multiply by (input - baseline)

3.2.4 Mathematical Foundation

The integrated gradients along the ith dimension is:

$$IG_i(x) = (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} d\alpha \tag{4}$$

where:

- x is the input
- x' is the baseline
- F is the model
- α is the interpolation parameter

3.2.5 Technical Implementation

Based on implementation in `notebooks/mini_challenge.ipynb`:

1. **Setup**:
   - Define baseline (e.g., zero tensor)
   - Choose number of steps (e.g., 50)
   - Identify target class (predicted or specified)

2. **Path Interpolation**:
   - Create scaled inputs between baseline and input:
   $$x_\alpha = x' + \alpha(x - x') \quad \text{for } \alpha \in [0, 1] \tag{5}$$
   - Discretize path into steps
3. **Gradient Accumulation**:
   - For each step along path:
     - Forward pass through model
     - Compute gradients w.r.t. input
     - Store gradients
   - Average all collected gradients
4. **Final Attribution**:
   - Multiply average gradients by (input - baseline)
   - Result shows per-feature contributions

3.2.6 Reading the Output

- **Values**:
  - Positive: Feature pushed prediction toward target class
  - Negative: Feature pushed prediction away from target class
  - Magnitude: Strength of contribution
- **What to Look For**:
  - Which features had strongest contributions
  - Whether contributions match domain knowledge
  - Balance between positive and negative contributions

3.2.7 Key Strengths

- Theoretically sound with axiomatic justification
- Works for any differentiable model
- Provides pixel-level attributions for images
- Computationally tractable