Dependable AI

Assignment Report

Roll Number	B21AI055, B21CS092
Name	Drithi Davuluri, G Mukund
Assignment Number	02
Assignment Title	Math Server

Objective

Choose a classification problem, train a neural network, and use the tools available in iNNvestigate to explain the predictions. Analyze the interpretations and write a report.

Classification Problem

For this assignment, we chose two well-known image classification problems: MNIST handwritten digit recognition. These benchmark datasets are widely used for evaluating the performance of machine learning models, particularly in the field of computer vision.

- **MNIST Dataset**

  The MNIST dataset is a benchmark image dataset comprising 70,000 grayscale 28x28 pixel images of handwritten digits (0-9), divided into 60,000 training and 10,000 test samples. Widely used for developing and evaluating image classification models, MNIST provides a simple yet valuable starting point for exploring techniques like convolutional neural networks and model interpretability methods.

Neural Network Model

To tackle these classification tasks, we designed and trained convolutional neural networks (CNNs) using TensorFlow and Keras. The CNN architectures for both datasets were identical, consisting of the following layers:

1. A 2D convolutional layer (Conv2D) with 32 filters, a 3x3 kernel size, and ReLU activation function.

2. A 2D max-pooling layer (MaxPooling2D) with a 2x2 pool size for spatial downsampling.
3. Another Conv2D layer with 64 filters, a 3x3 kernel size, and ReLU activation.
4. Another MaxPooling2D layer with a 2x2 pool size.
5. A third Conv2D layer with 64 filters, a 3x3 kernel size, and ReLU activation.
6. A Flatten layer to convert the 2D feature maps into a 1D vector.
7. A dense fully connected layer with 64 units and ReLU activation.
8. The output dense layer with 10 units (for the 10 classes) and a softmax activation function.

We compiled the models using the Adam optimizer, categorical cross-entropy loss function, and accuracy as the evaluation metric. The models were trained for 5 epochs with a batch size of 64, using the provided training data and evaluating on the test data after each epoch.

iNNvestigate

iNNvestigate is a Python library that provides tools for interpreting and explaining the predictions made by neural networks. It offers various analysis techniques, such as gradient-based methods and pattern attribution, to compute relevance scores or heatmaps that highlight the input features most responsible for a model's output. This interpretability is crucial for building trust and transparency in neural network models.

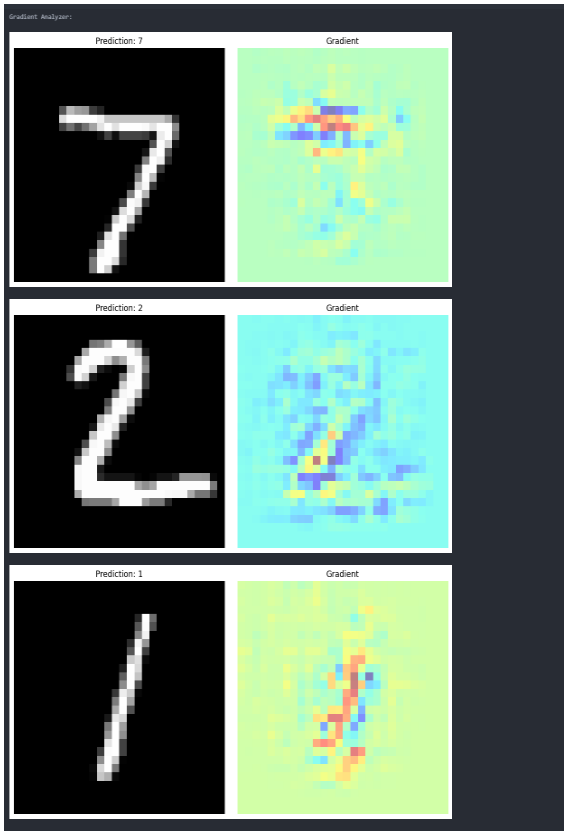
Usage of iNNvestigate

After training the models, we aimed to explore techniques for interpreting and explaining their predictions. To this end, we leveraged the iNNvestigate library, which provides various analyzers designed for interpreting neural network decisions.

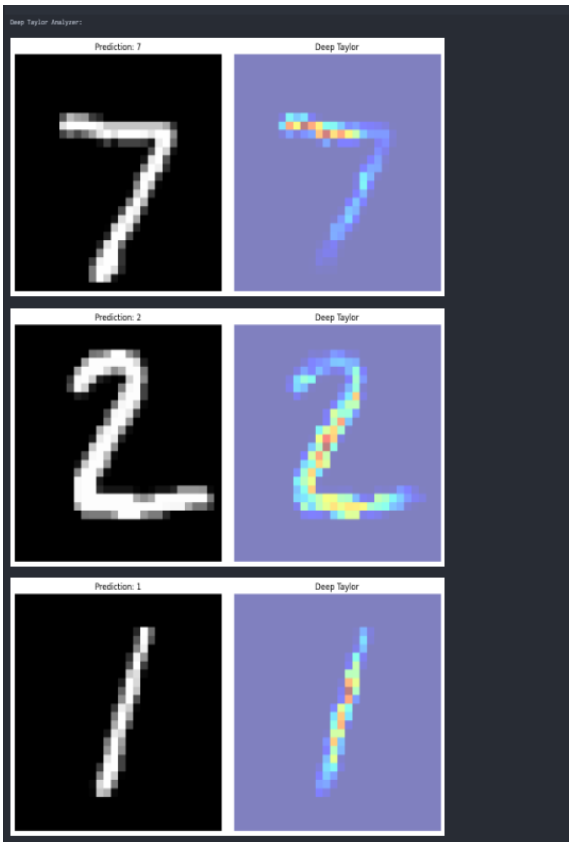
To prepare the models for analysis with iNNvestigate, we created a new model model_without_softmax by removing the final softmax layer. Additionally, we defined a custom ReLU activation function custom_relu and replaced the standard ReLU activations with this custom function, as mandated by certain iNNvestigate analysis techniques.

Creating instances of 3 different analyzers provided by iNNvestigate to compute relevance scores or gradients that explain the model's predictions:

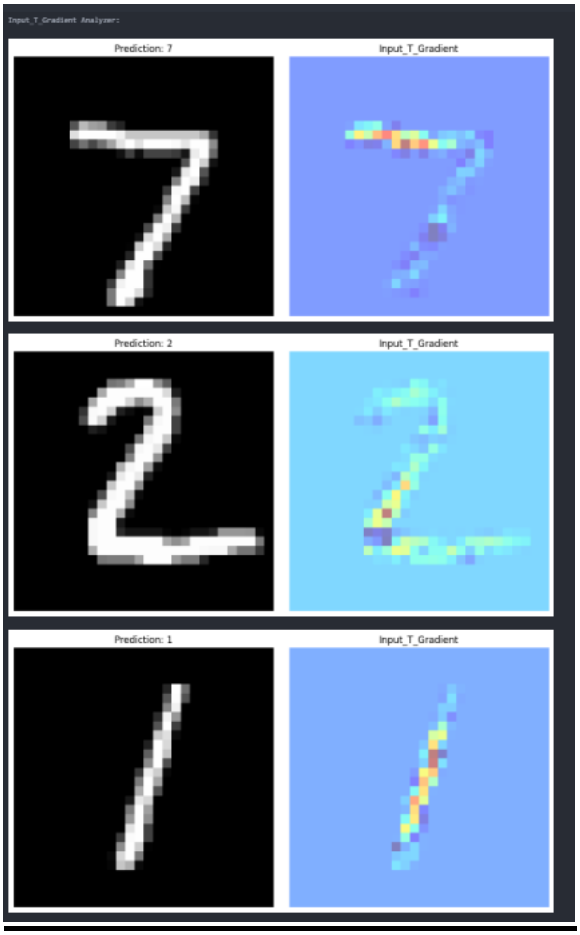
- The "deep_taylor" analyzer employs a Taylor decomposition of the neural network to compute relevance scores for each input feature, indicating their importance in the model's decision-making process.
- The "gradient" analyzer computes the gradients of the output with respect to the input, revealing how sensitive the prediction is to changes in each input feature.
- The "input_t_gradient" analyzer is a variant of the gradient method that considers the input values themselves when computing the gradients.



Gradient Analyzer



Deep Taylor Analyzer

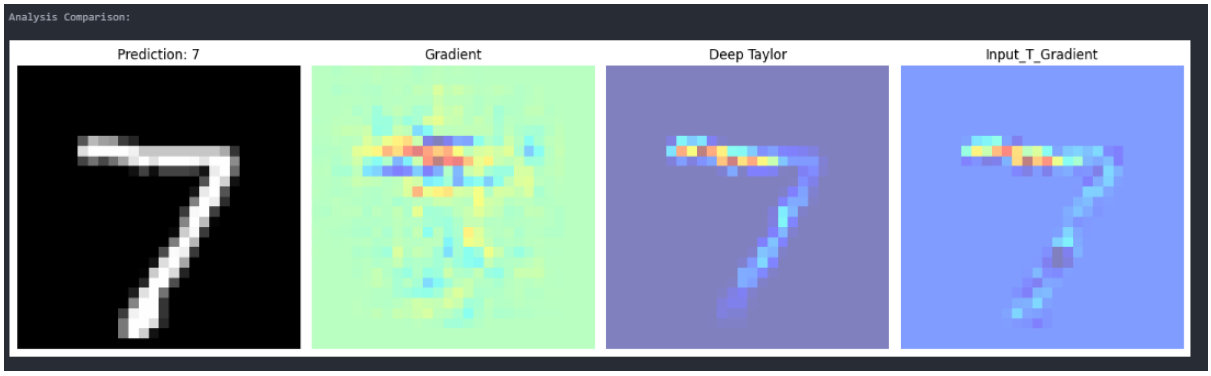


Input_T_Gradient Analyzer

Observation

To interpret these results, we visualized the analysis heatmaps by overlaying them on the original test images. Initially, for the MNIST dataset, we plotted 10 examples using only the "deep_taylor" analyzer. Subsequently, we visualized the analysis results for all three analyzers ("deep_taylor", "gradient", and "input_t_gradient") on both MNIST dataset, plotting 10 examples for each.

By examining these visualizations, we could interpret which input features (pixels in the case of images) contributed the most to the model's predictions. The heatmaps highlighted the relevant regions of the input that influenced the model's decision-making process, providing insights into the underlying patterns and features the model had learned.



Analysis Comparison

Conclusion

We overall explored the application of convolutional neural networks for image classification tasks and leverage the iNNvestigate library to interpret and explain the models' predictions. The visualization of analysis results facilitated a deeper understanding of the decision-making process and the features that influenced the models' decisions.