Milestone 3 Report: Image Classification of Natural Scenes

Link to Project Code: Google Collab

1. Machine Learning Problem Definition

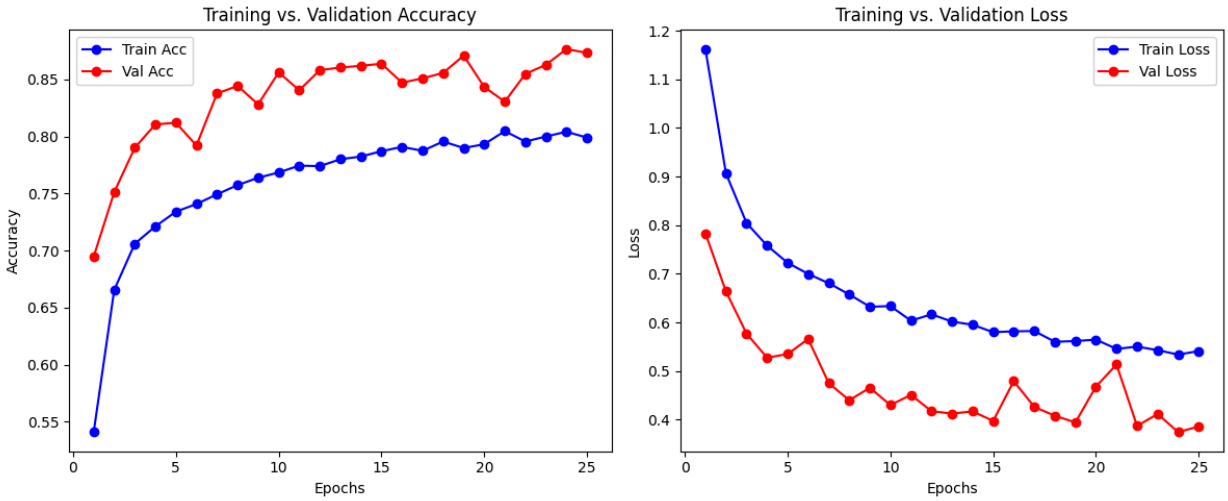
- **Classification**
  - **Problem:** The primary machine learning problem addressed in this project is the accurate classification of natural scene images into one of six distinct categories: buildings, forest, glacier, mountain, sea, and street. This is a multi-class classification problem.
  - **Input:** The input features for the classification task are the pixel data of the images. Each image is an RGB image, and the pixel values represent the color information at each spatial location within the image.
  - **Target Variable:** The target variable is the scene category, a categorical variable with six possible values corresponding to the six classes. The goal of the machine learning models is to predict the correct category for each input image.
- **Regression**
  - **Problem:** Estimate the environmental impact score of a given scene. A higher score should be assigned to scenes with more human-made structures (buildings, streets) and a lower score to natural scenes (forests, glaciers).
  - **Input:** The input is the class probabilities of each image from the classification model.
  - **Target Variable:** The target variable is a numerical score representing the estimated environmental impact.

2. Code and Experiments with Algorithms

- **2.1 Convolutional Neural Network (CNN)**
  - CNNs are the de facto standard for image classification due to their ability to capture local spatial hierarchies through convolutional filters. Their parameter-sharing and local connectivity make them efficient and effective for visual tasks, especially when datasets have spatial features like textures and edges.
  - **Implementation:** A CNN was implemented using PyTorch. The architecture consists of three convolutional layers with ReLU activations for feature extraction, dropout for regularization, followed by two fully connected layers for classification.
  - **Hyperparameters:**
    - `Number of Convolutional Layers`: 3 (conv1, conv2, conv3)
    - `Kernel Sizes`: 3x3 for all convolutional layers
    - `Number of Filters`: 32 (conv1), 64 (conv2), 128 (conv3)
    - `Activation Function`: ReLU
    - `Pooling Layers`: Max Pooling with kernel size 2 and stride 2
    - `Number of Fully Connected Layers`: 2 (fc1, fc2)
    - `Dropout Rate`: 0.25
    - `Optimizer`: Adam
    - `Learning Rate`: 0.001
    - `Loss Function`: Cross-Entropy Loss
    - `Number of Epochs`: 25
    - `Patience`: 5
    - `Batch Size`: 32
  - **Hyperparameter Selection:**
    - The architecture and initial hyperparameters were based on common CNN designs for image classification.
    - The learning rate was tuned by monitoring the validation loss. If the loss plateaued, the rate was reduced.
    - The number of layers and filters was adjusted to balance model complexity and performance, avoiding overfitting (high variance) or underfitting (high bias).
    - The kernel size of 3x3 was chosen to capture local spatial relationships in the images.
    - The number of filters was increased in each convolutional layer to allow the network to learn progressively more complex features.

- Max pooling with a 2x2 kernel was used to reduce the spatial dimensions and computational load.
- ReLU was chosen as the activation function due to its efficiency and ability to mitigate the vanishing gradient problem.
- The dropout rate of 0.25 was included to prevent overfitting.
- The fully connected layer sizes were chosen to gradually reduce the dimensionality of the feature representation before the final classification layer.
- The Adam optimizer was chosen for its adaptive learning rate capabilities, which often leads to faster convergence than standard SGD.
- A learning rate of 0.001 was found to be effective in minimizing the loss without causing instability during training.
- Cross-Entropy Loss is the standard loss function for multi-class classification problems.
- The model was trained for 25 epochs, with early stopping implemented using a patience of 5 epochs to prevent overfitting.
- A batch size of 32 was used as a trade-off between training speed and memory usage.

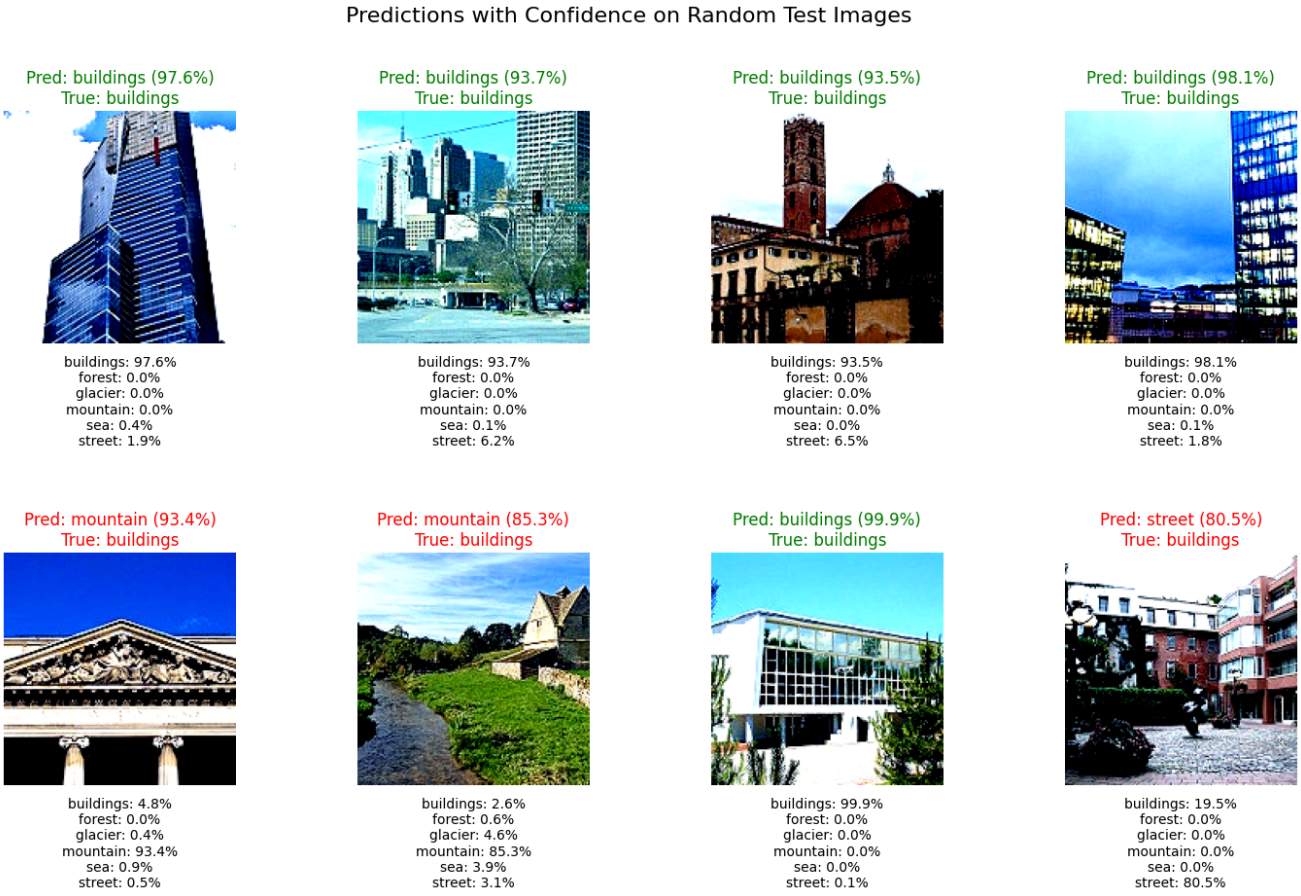
- **Training Curves:**



- **Performance Metrics**
  - Accuracy: 87.37%
  - Precision: 87.51%
  - Recall: 87.62%
  - F1 Score: 87.50%

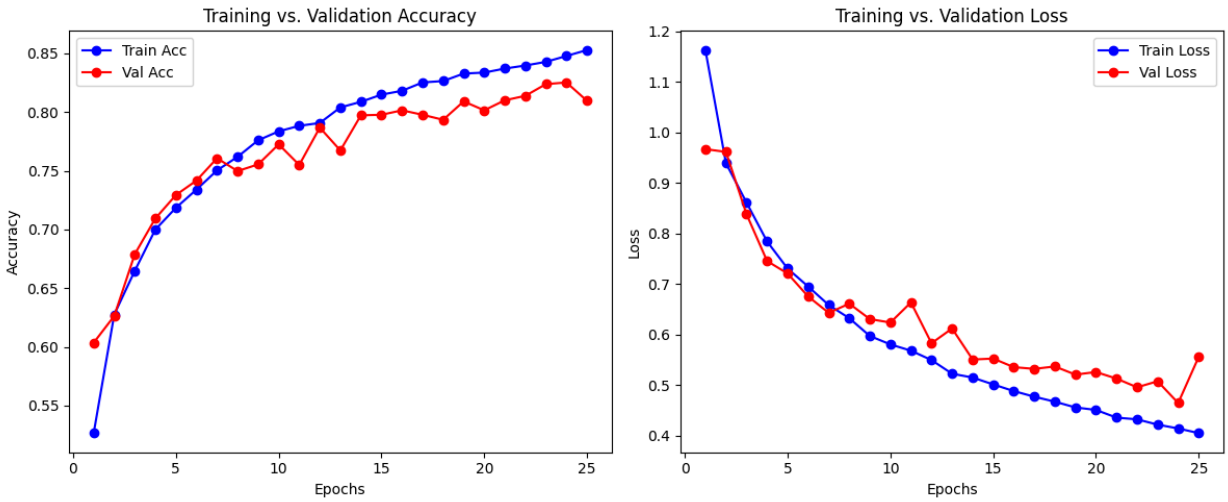
- **Confusion Matrix**

- **Prediction on Test Sample**

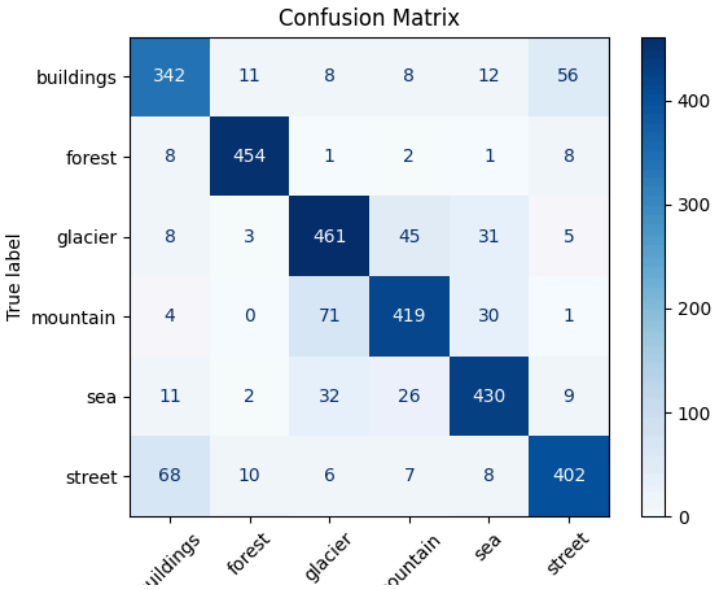


- **2.2 Vision Transformer (ViT)**
  - Transformers have revolutionized NLP, and their adaptation to computer vision (ViT) enables capturing long-range dependencies and global context via self-attention. This model was chosen to evaluate transformer-based performance on spatial data and to compare it with convolutional methods.
  - **Implementation:** A Vision Transformer model was implemented using PyTorch. ViT treats image patches as "words" and uses a Transformer encoder to learn relationships between them. The input image is split into 16×16 patches, and each patch is embedded and passed through multiple transformer encoder layers. Class token is used for final classification
  - **Hyperparameters:**
    - **Patch Embedding:**
      - `Image Size`: 224
      - `Patch Size`: 16
      - `Input Channels`: 3
      - `Embedding Dimension`: 256
    - **Attention Mechanism:**
      - `Number of Attention Heads`: 8
      - `QKV Bias`: True
      - `Attention Dropout`: 0.1
      - `Projection Dropout`: 0.1
    - **Transformer Block:**
      - `MLP Ratio`: 4.0
    - **Transformer Model:**
      - `Number of Blocks (Depth)`: 6
      - `Dropout Rate`: 0.1
      - `Number of Classes`: 6
    - **Training:**
      - `Optimizer`: Adam
      - `Learning Rate`: 0.0001
      - `Loss Function`: Cross-Entropy Loss
      - `Batch Size`: 32
      - `Number of Epochs`: 25
      - `Patience`: 5

- **Hyperparameter Selection:**
  - Base values were chosen based on the standard ViT models.
  - Patch size influences how much local information is preserved. Smaller patches can capture finer details but increase the sequence length.
  - Other hyperparameters were tuned to optimize performance and prevent overfitting, similar to the CNN approach.
  - Image size was set to 224 to balance resolution and computational cost.
  - A patch size of 16 was chosen to create a reasonable sequence length for the transformer.
  - An embedding dimension of 256 was used as a trade-off between model capacity and efficiency.
  - 8 attention heads were used to allow the model to capture different aspects of the input.
  - GELU activation function was chosen because it often works better than ReLU in Transformers.
  - 6 Transformer blocks were stacked to enable the model to learn hierarchical representations.
  - Adam optimizer with a learning rate of 0.0001 was used for efficient training.
  - Dropout rates of 0.1 were applied to prevent overfitting.
  - A batch size of 32 was used based on available GPU memory.
  - Early stopping with a patience of 5 was employed to avoid training for unnecessary epochs.
- **Training Curves:**



- **Performance Metrics**
  - Accuracy: 83.60%
  - Precision: 83.66%
  - Recall: 83.63%
  - F1 Score: 83.63%
- **Confusion Matrix**



- **Prediction on Test Sample**

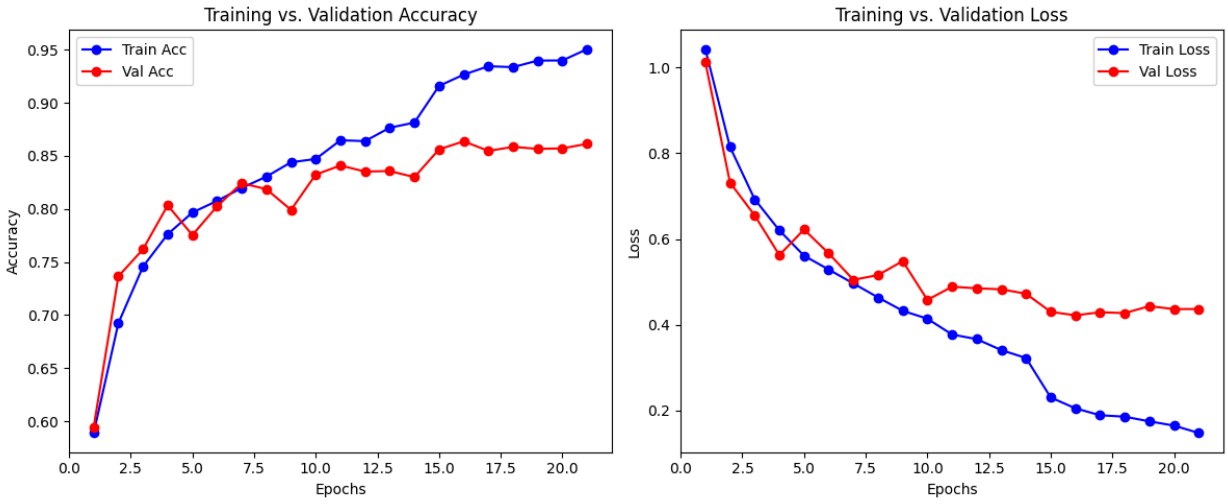
Predictions with Confidence on Random Test Images

- **2.3 MLP-Mixer**

    - The MLP-Mixer challenges the dominance of CNNs and Transformers by relying solely on fully connected layers. It processes spatial patches and channel information using multilayer perceptrons. It was selected to explore a novel architecture and compare performance without convolution or attention.
    - **Implementation:** An MLP-Mixer model was implemented using PyTorch. The input image is divided into patches. This architecture relies entirely on MLPs, alternating between mixing features across spatial locations and feature channels. Token-mixing is done across patches and channel-mixing within each patch. Final classification is done through the MLP head
    - **Hyperparameters:**
        - `image_size`: 224
        - `patch_size`: 16
        - `num_blocks`: 8
        - `hidden_dim`: 512
        - `token_dim`: 256.
        - `channel_dim`: 2048
        - `batch_size`: 32
        - `lr`: 1e-4
        - `weight_decay`: 1e-5
        - `Loss Function`: Cross-Entropy Loss
        - `optimizer`: AdamW
        - `num_epochs`: 25
        - `patience`: 5
    - **Hyperparameter Selection:**
        - Hyperparameter choices are once more common practices for MLP-Mixers and are adapted through experimentation.
        - The dimensions of the MLPs were crucial for the model's capacity to learn complex relationships.
        - The image size of 224 was chosen to balance image resolution and computational efficiency.
        - A patch size of 16 allows the model to capture local features while reducing the sequence length.
        - 8 blocks were chosen to provide sufficient model capacity to learn complex patterns in the data and balance excessive computational cost.

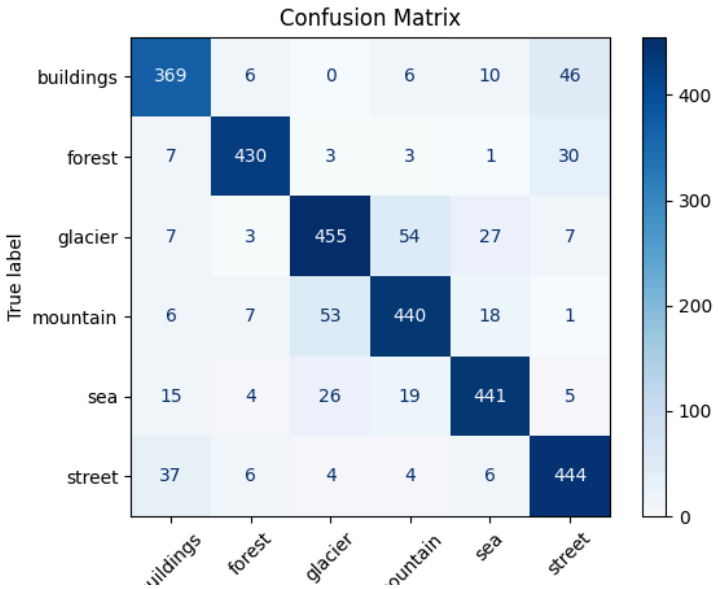
- The dimension of the patch embeddings and hidden representation of 512 was chosen to provide a reasonable trade-off between model capacity and computational load.
- A token dimension of 256 was chosen to allow for sufficient interaction between patches.
- A channel dimension of 2048 allows the model to capture more complex feature relationships.
- A common batch size of 32 was used again.
- A relatively small learning rate of 1e-4 was chosen to ensure stable training and prevent oscillations.
- A weight decay of 1e-5 was used with the AdamW optimizer. This L2 regularization term helps to prevent overfitting by penalizing large weights
- Cross-entropy loss is standard for multi-class classification problems.
- AdamW optimizer decouples weight decay from the gradient updates, leading to better generalization performance.
- An initial number of epochs was set to 25 with an early stopping mechanism.
- A patience value of 5 was used for early stopping to provide a reasonable buffer for minor fluctuations in validation loss

- **Training Curves:**

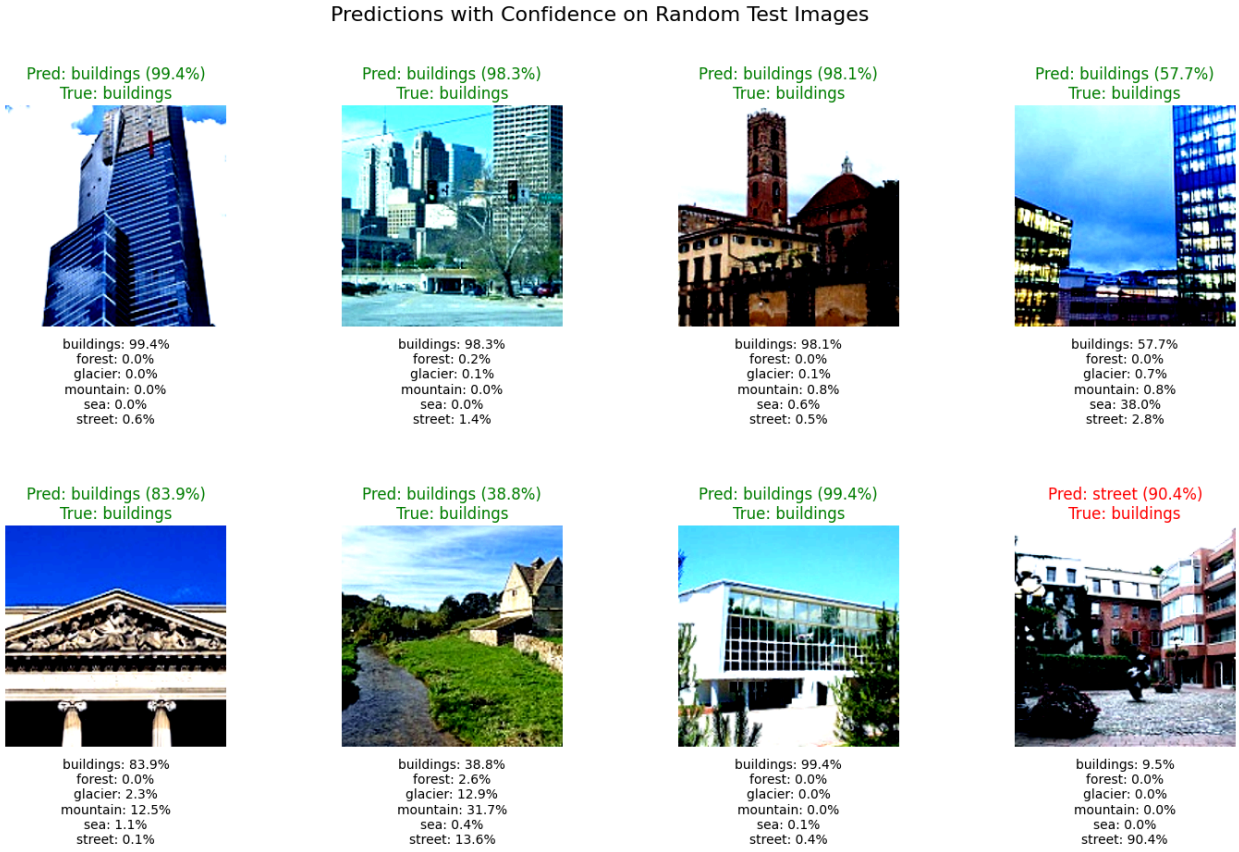


- **Performance Metrics**
  - Accuracy: 85.97%
  - Precision: 86.12%
  - Recall: 86.06%
  - F1 Score: 86.06%

- **Confusion Matrix**

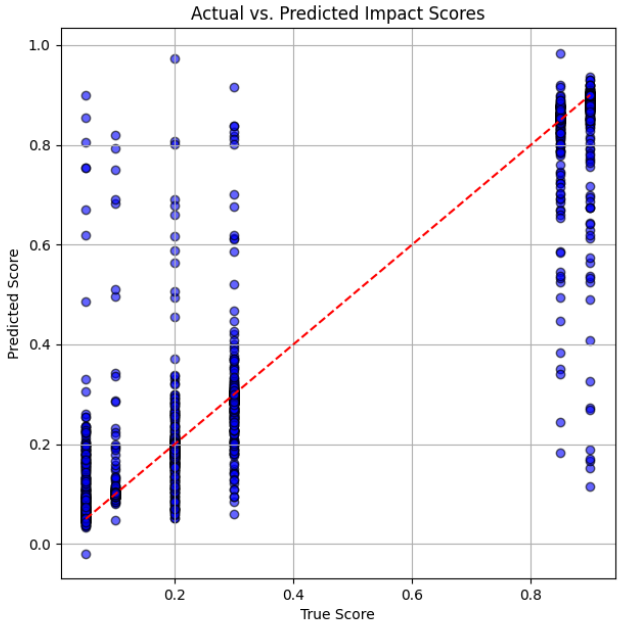


- **Prediction on Test Sample**

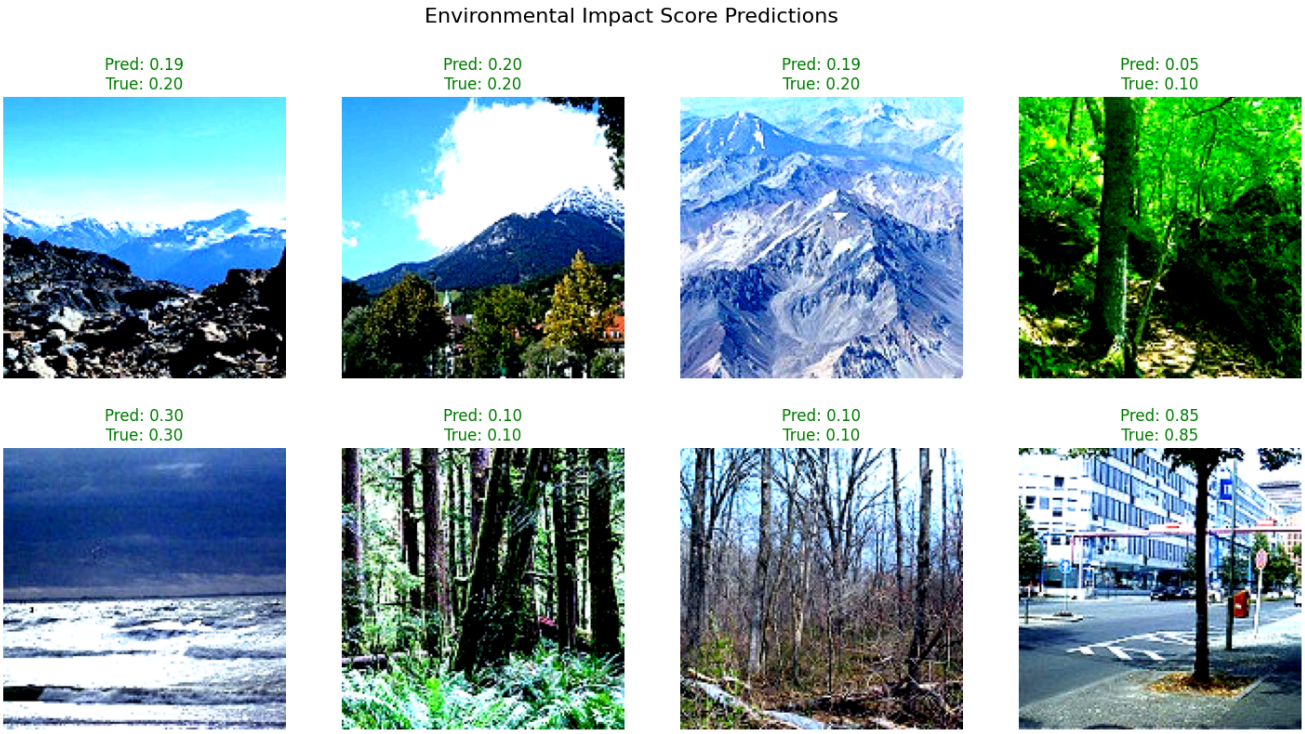


- **2.4 XGBoost (Regression)**
    - **Note:** While the primary task is classification, XGBoost was used for the *regression* problem of predicting an "environmental impact score." This is a separate experiment.
    - XGBoost is a powerful ensemble algorithm known for its speed and predictive accuracy on structured data. It was ideal for regression once we extracted deep features from images using a pretrained CNN. Unlike deep models, XGBoost can efficiently handle high-dimensional tabular data derived from image embeddings.
    - **Implementation:** XGBoost was used with scikit-learn's wrapper for ease of integration. Multiple trees are trained on the dataset, and their predictions are combined for the final output. XGBoost handles automatic feature scaling and tree pruning.
    - **Hyperparameters:**
        - `n_estimators`: 100
        - `learning_rate`: 0.1
        - `objective`: Mean Squared Error
    - **Hyperparameter Selection:**
        - 100 estimators were chosen to balance performance and overfitting risk
        - Learning rate of 0.1 is a commonly used default that balances learning speed and stability
        - MSE is the standard objective for regression problems
    - **Performance Metrics:**
        - RMSE: 0.0027
        - MAE : 0.0191
        - R^2 : 0.9769

- **Actual vs. Predicted Impact Scores Scatterplot:**



- **Prediction on Test Sample**



3. Summary of Observations

Each model was selected to test a different paradigm in computer vision: CNNs for local pattern detection, ViTs for global attention-based reasoning, MLP-Mixers for fully connected non-convolutional processing, and finally XGBoost for structured regression on learned image features. All models performed strongly. The CNN model performed best, leveraging its ability to learn spatial hierarchies. It achieved a test accuracy of 87.37%. The Vision Transformer showed competitive results, demonstrating the effectiveness of the attention mechanism for image classification, and achieved an accuracy of 83.60%. The MLP-Mixer offered an interesting alternative and surprisingly performed better than Vision Transformer. Its accuracy was 85.97%. Both ViT and MLP-Mixer were competitive, validating the viability of newer architectures; however, they tended to be more computationally expensive than CNNs. XGBoost excelled at regression, yielding a high R^2 score of 0.9769 and a low error of RMSE 0.0027, suggesting our features effectively captured environmental context. This project provided valuable insights into model selection, training, evaluation, and how architecture influences performance in real-world image tasks.

4. Non-Standard/Novel Aspects

Combining image classification with a subjective regression task (environmental impact) is a less common application. It explores how image analysis can be linked to higher-level, potentially abstract concepts. This can open new opportunities for understanding complex relationships between visual features and numerical indicators. This integration can facilitate the development of tools for real-time environmental monitoring, enabling more informed decision-making.