**Auburn Waves- Project Report**

**Overview**

This report describes a machine learning model designed to classify images from the given dataset. The model uses a pre-trained MobileNetV2 architecture as a base and is fine-tuned for this specific image classification task.

**Data Extraction**

* **Training Data**: The training images are extracted from a ZIP file (train\_dataset.zip) and placed in a directory.
* **Labels**: The labels for the training images are stored in a CSV file (train.csv), which is loaded into a Pandas DataFrame. The labels are one-hot encoded.

**Image Preprocessing**

* Images are resized to 224x224 pixels and normalized to a range of [0, 1].
* A custom MarsLandDataset class is used to load and manage the images and labels.

**Data Augmentation**

* An ImageDataGenerator is used to perform data augmentation, which includes random rotations, shifts, shearing, zooming, and horizontal flips to improve model robustness.

**Model Architecture**

* **Base Model**: MobileNetV2 pre-trained on ImageNet, with its top layer removed (include\_top=False). The model's weights are initially frozen.
* **Custom Layers**:
  + A GlobalAveragePooling2D layer reduces the feature maps to a single vector.
  + A BatchNormalization layer standardizes inputs.
  + A Dense layer with 256 units and ReLU activation adds a fully connected layer.
  + A Dropout layer with a 0.5 dropout rate helps prevent overfitting.
  + A final Dense layer with softmax activation provides class predictions.

**Training Process**

* **Loss Function**: Categorical Crossentropy.
* **Optimizer used**: Adam.
* **Learning Rate Adjustment**: The learning rate is dynamically adjusted if the validation loss does not improve.
* **Early Stopping**: The training process includes early stopping to prevent overfitting. If the validation loss does not improve for three consecutive epochs, training stops.
* **Model Checkpointing**: The best model weights (based on validation loss) are saved to model.weights.h5.

**Evaluation**

* The model is evaluated on a separate validation dataset (acquired by splitting the training dataset), achieving a validation accuracy displayed during the training loop (approx. 93%).

**Testing**

* **Test Data**: Test images are extracted from another ZIP file (test\_dataset.zip).
* **Predictions**: The model predicts the labels for test images, and any invalid predictions raise an error.
* **Output**: Predicted labels for all the 2000 test images are printed.

**Performance**

* The model provided utilizes transfer learning to achieve reasonable performance on the provided dataset. The combination of data augmentation, a pre-trained model, and dynamic learning rate adjustment helps optimize the model for image classification tasks.

**Thought Process**

* Initially, we implemented a simple 2D CNN, achieving about 63.5% accuracy on the validation set, which was not ideal. After seeking advice from ChatGPT, we applied some effective data preprocessing techniques that enhanced the training process. We then tried using the VGG16 pre-trained network, which also didn’t fare very well. However, switching to the pre-trained MobileNetV2 neural network significantly improved both the performance and speed of the model compared to the 2D CNN and VGG16. Since our team is not very experienced in coding neural networks or advanced data preprocessing, we relied on ChatGPT for syntax guidance and additional suggestions.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_MartianMavericks\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_