Oscar Eduardo Nieto Espitia - A0105090

Performance analysis for Garbage Classification

The problem involves categorizing waste items into different classes, such as cardboard, glass, metal, paper, plastic, and trash. Accurate classification aids in effective recycling and waste reduction strategies.

The task often involves leveraging machine learning techniques, particularly deep learning, to develop models capable of automatically identifying and sorting various types of waste.

Dataset: Garbage Classification Data

The Garbage Classification Dataset is a collection of images of various types of waste items, which have been categorized into six different classes. Each class represents a specific type of garbage, and the dataset provides images for each class.

- Cardboard: 393 images
- Glass: 491 images
- Metal: 400 images
- Paper: 584 images
- Plastic: 472 images
- Trash: 127 images

Splitting the dataset

Using separate sets for training, testing and validation is a fundamental practice for the following reasons:

Model Performance Evaluation: One of the primary goals of machine learning is to build models that generalize well to unseen data. By splitting your dataset into three subsets, we can assess your model's performance accurately.

Identify Overfitting: The test set serves as a benchmark to check for overfitting. If a model performs well on the training data but poorly on the test data, it's a sign of overfitting.

Hyperparameter Tuning: To optimize the model's hyperparameters, we need a validation set. You can train multiple models with different hyperparameters and select the one that performs best on the validation set

Data Augmentation

Data augmentation involves applying various transformations to the existing dataset to create new, diverse training examples. In the context of garbage classification, these transformations typically include rotations, flips, shifts, zooms, and changes in brightness. The goal is to artificially expand the training dataset, providing the model with more varied examples to learn from.

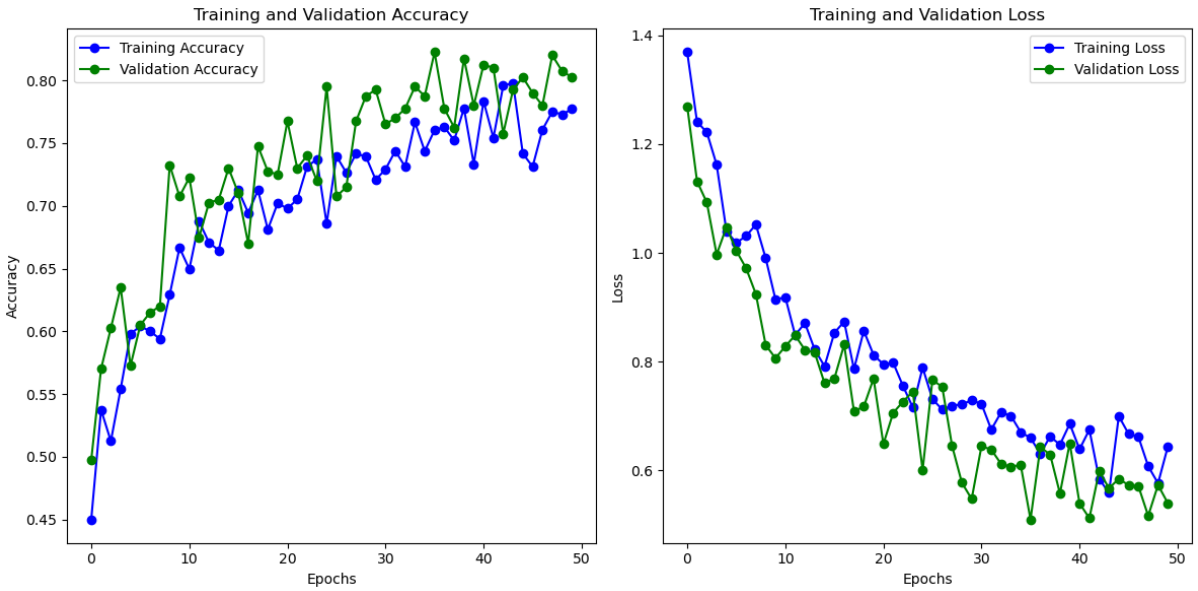
- **Training Dataset Only**: Data augmentation is typically applied to the training dataset. This ensures that the model learns from augmented examples during training but is evaluated on unaltered data during validation and testing.
- **Evaluation on Unaltered Data**: The validation and test datasets remain untouched by data augmentation. This separation is crucial for assessing how well the model generalizes to real-world, unaltered scenarios.

```
train_datagenerator = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
    validation_split=0.1,
    brightness_range=[0.7, 1.3],
    rotation_range=15
)

# Parameters for Data augmentation for the rest of the datasets (test
and validation)
test_datagenerator = ImageDataGenerator(
    rescale=1./255
)
```

Creating the model

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 9, 9, 512)         14714688

 flatten_8 (Flatten)         (None, 41472)             0

 dense_17 (Dense)            (None, 256)               10617088

 dense_18 (Dense)            (None, 6)                 1542

=================================================================
Total params: 25333318 (96.64 MB)
Trainable params: 10618630 (40.51 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

- **Transfer Learning with VGG16**: The use of the VGG16 pre-trained model with weights='imagenet' is an effective form of transfer learning. This allows your model to start with a strong foundation in understanding image features.
- **Flatten Layer:** The layers.Flatten() layer is used to flatten the output from the convolutional layers.
- **Dense Layers**: The first dense layer with 256 units serves as a feature extractor, learning high-level features from the flattened feature maps. The ReLU activation introduces non-linearity, which is crucial for capturing complex patterns in the data. The final dense layer with 6 units and a softmax activation is responsible for making predictions across the six classes.
- **Model Compilation**: The choice of categorical_crossentropy as the loss function is suitable for multi-class classification problems.
- **RMSprop:** The RMSprop algorithm is renowned for its ability to dynamically adapt the learning rate during training. It operates by calculating a weighted moving average of the squares of previous gradients and then divides the current gradient by the square root of this average. This aids in controlling the convergence speed and mitigating issues such as premature convergence or training slowdown.



Training Acc: 0.7312
Validation Acc : 0.8025

Both training and validation accuracies improve over time, indicating that the model is learning and generalizing from the data. The training accuracy is consistently higher than the validation accuracy, and the training loss is consistently lower than the validation loss. This suggests that the model is overfitting to the training data. An accuracy of around 80.25% on the validation set is a decent result, especially considering the model architecture and the amount of training data. However, there is still room for improvement.

Model Improvements

L2 Regularization

- The addition of L2 regularization in the new model architecture is aimed at preventing overfitting, which is a common issue when the model performs well on the training data but doesn't generalize well to unseen data (as indicated by a lower validation accuracy).
- L2 regularization introduces a penalty term in the loss function that discourages large weights. Large weights can lead to a model that fits the training data too closely, capturing noise and specific details of the training set that may not generalize well to new data. Adding dropout can be particularly beneficial when you observe an improvement in the validation accuracy compared to the training accuracy, as it suggests that the model is generalizing better.

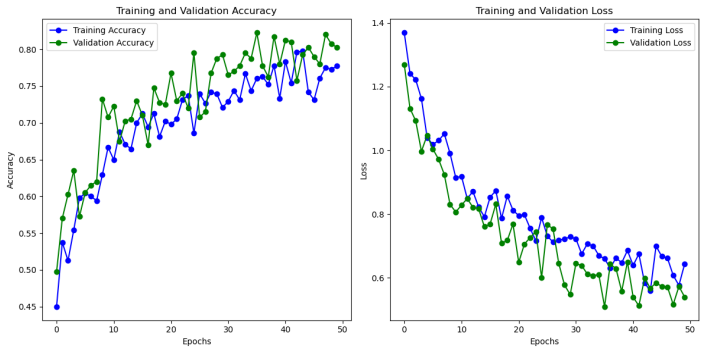
Dropout Layer

- The addition of a **Dropout layer** with a dropout rate of 0.5 is added after the first dense layer. Dropout randomly sets a fraction of input units to 0 at each update during

training. This helps in preventing the model from relying too much on specific features and encourages the network to learn more robust and general representations.

- L2 regularization penalizes large weights in the model, which can help prevent overfitting. Dropout, on the other hand, introduces redundancy by randomly dropping units, reducing the reliance on specific neurons. Combining both techniques provides complementary regularization effects.

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 9, 9, 512)         14714688

 flatten_10 (Flatten)        (None, 41472)             0

 dense_21 (Dense)            (None, 256)               10617088

 dropout_3 (Dropout)         (None, 256)               0

 dense_22 (Dense)            (None, 6)                 1542

=================================================================
Total params: 25333318 (96.64 MB)
Trainable params: 10618630 (40.51 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

Old Model	New Model
	

Training Acc: 0.7312 **Validation Acc** : 0.8025	**Training Acc**: 0.7688 **Validation Acc** : 0.8135

The absence of significant overfitting is evident in both models. The training and validation losses are decreasing over epochs, which is a positive sign. This implies that the models are learning meaningful representations without memorizing noise in the training data.

The dropout layer and L2 regularization in the new model have likely played a crucial role in preventing overfitting. The dropout layer introduces randomness during training, discouraging the model from relying too heavily on specific neurons, and L2 regularization penalizes large weights, promoting more robust learning.

The new model, with the incorporation of L2 regularization and dropout, demonstrates improved generalization to unseen data, with a slightly higher validation accuracy than training accuracy. The iterative improvements over epochs and the absence of significant overfitting highlight the effectiveness of the chosen regularization techniques in enhancing the model's performance.