**Abstract**

This report provides a comprehensive analysis of the performance evaluation of multiple deep learning architectures for vehicle classification. The study involves data preprocessing, experimentation with model architectures, and comparison using two loss functions: Categorical Crossentropy and Focal Loss. Key findings include comparative metrics for accuracy, precision, recall, and F1-score, alongside the identification of optimal configurations. Future directions for improvement are also proposed.

**Data Analysis and Cleaning**

Here we have used **CleanVision** – a library for image data curation to clean the dataset and then use it.(Possible issues – odd\_size, odd\_aspect rario, blurry etc.)

The dataset consisted of vehicle images segregated into distinct categories. Initial exploration revealed imbalances across classes and inconsistencies in image resolutions. Data cleaning involved resizing images to a uniform dimension of 224x224 pixels and rescaling pixel values to a range of [0, 1]. Duplicate entries and corrupted images were removed to ensure data integrity. Additionally, the dataset was augmented to address class imbalance using techniques like rotation, flipping, and zoom.

**Data Preprocessing**

Key preprocessing steps included:

1. **Rescaling**: Pixel values were normalized to improve model convergence.
2. **One-Hot Encoding**: Labels were encoded to match the categorical format required by the models.
3. **Augmentation**: Applied transformations like horizontal flips, rotations, and zooms to enhance generalization.
4. **Batch Generation**: Data generators were used to streamline feeding batches during training.

Most of these is taken care by the datagen and the datagen,from\_directory

**Model Architecture**

The architectures evaluated include ResNet50, EfficientNetB0, MobileNetV2, VGG16, VGG19, InceptionV3, and DenseNet121. These models were selected for their proven efficacy in image classification tasks:

1. **ResNet50**: Handles vanishing gradients using residual connections.
2. **EfficientNetB0**: Optimizes accuracy and computational cost with compound scaling.
3. **MobileNetV2**: Efficient for mobile and embedded systems with depthwise separable convolutions.
4. **VGG16/VGG19**: Simple yet powerful models with sequential layers.
5. **InceptionV3**: Employs inception modules for multi-scale feature extraction.
6. **DenseNet121**: Promotes feature reuse through dense connections.

These architectures were augmented with custom classification heads involving global average pooling, dense layers, and a softmax output layer.

| **Model** | **Loss Function** | **Train Accuracy** | **Val Accuracy** | **Precision** | **Recall** | **F1-Score** |
| --- | --- | --- | --- | --- | --- | --- |
| ResNet50 | Categorical Crossentropy | 0.532135 | 0.537940 | 0.614734 | 0.512195 | 0.487126 |
| EfficientNetB0 | Categorical Crossentropy | 0.174828 | 0.210027 | 0.044111 | 0.210027 | 0.072910 |
| MobileNetV2 | Categorical Crossentropy | 1.000000 | 0.818428 | 0.817346 | 0.815718 | 0.814665 |
| VGG16 | Categorical Crossentropy | 0.872226 | 0.738482 | 0.733086 | 0.720867 | 0.720248 |
| VGG19 | Categorical Crossentropy | 0.816669 | 0.738482 | 0.763888 | 0.735772 | 0.732222 |
| InceptionV3 | Categorical Crossentropy | 1.000000 | 0.800813 | 0.795298 | 0.796748 | 0.794775 |
| DenseNet121 | Categorical Crossentropy | 1.000000 | 0.825203 | 0.815718 | 0.815718 | 0.813872 |
| ResNet50 | Focal Loss | 0.541316 | 0.559621 | 0.609153 | 0.559621 | 0.545787 |
| EfficientNetB0 | Focal Loss | 0.176358 | 0.210027 | 0.044111 | 0.210027 | 0.072910 |
| MobileNetV2 | Focal Loss | 1.000000 | 0.815718 | 0.785044 | 0.784084 | 0.778879 |
| VGG16 | Focal Loss | 0.862280 | 0.726287 | 0.742578 | 0.693291 | 0.671559 |
| VGG19 | Focal Loss | 0.817521 | 0.724932 | 0.805202 | 0.798103 | 0.796308 |
| InceptionV3 | Focal Loss | 1.000000 | 0.799458 | 0.783276 | 0.788168 | 0.782504 |
| DenseNet121 | Focal Loss | 1.000000 | 0.826558 | 0.814728 | 0.821183 | 0.820504 |

**Training and Experimentation**

Training was conducted over 10 epochs for each architecture and loss function, utilizing the Adam optimizer. Key settings included:

1. **Loss Functions**: Compared Categorical Crossentropy and Focal Loss- best for imbalance dataset.
2. **Hyperparameters**: Learning rate of 0.001, batch size of 32.
3. **Data Splitting**: 80-20 split for training and validation.
4. **Regularization**: Dropout layers to prevent overfitting.
5. **Evaluation**: Metrics tracked include accuracy, precision, recall, and F1-score.

To optimize performance, frozen base layers allowed efficient fine-tuning of the classification head. Learning curves were monitored to identify overfitting.

**Results and Key Findings**

**Overall Performance:**

The performance of different models was evaluated based on train accuracy, validation accuracy, precision, recall, and F1-Score using both Categorical Crossentropy and Focal Loss. The following are the key findings:

1. **Top-Performing Models:**
   * **Using Categorical Crossentropy, DenseNet121 achieved the best validation accuracy of 82.52%, with an F1-Score of 81.38%.**
   * **Using Focal Loss, DenseNet121 also stood out with the highest validation accuracy of 82.65%, accompanied by an F1-Score of 82.05%.**
2. Architectural Comparison:
   * MobileNetV2, InceptionV3, and DenseNet121 consistently performed better across all metrics when compared to models like EfficientNetB0 and ResNet50.
   * EfficientNetB0 performed poorly across both loss functions, indicating that this architecture might not be well-suited for the given dataset or task.
3. Loss Function Analysis:
   * Categorical Crossentropy slightly outperformed Focal Loss for models like DenseNet121 and MobileNetV2 in validation accuracy.
   * Focal Loss, designed to handle class imbalance, did not show significant improvements over Categorical Crossentropy, likely because the dataset was well-balanced.
4. Confusion Matrix Insights:
   * The confusion matrix revealed that models like DenseNet121 and MobileNetV2 classified most classes accurately, with minimal confusion.
   * Lower-performing models such as EfficientNetB0 and ResNet50 showed higher confusion rates, leading to reduced precision and recall.
5. Training Stability:
   * Models such as InceptionV3 and DenseNet121 achieved consistent results across epochs, demonstrating stable training behavior.
   * EfficientNetB0 displayed unstable learning curves, indicating issues with convergence.
6. Trade-Off Between Complexity and Performance:
   * MobileNetV2 achieved competitive results (validation accuracy of 81.84%) while being more computationally efficient compared to heavier models like DenseNet121 and InceptionV3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Future Work**

**Future Work**

1. **Unfreezing Pretrained Layers:**
   * Fine-tuning the pretrained layers of models like DenseNet121 and InceptionV3 **to** capture dataset-specific features for improved performance.
2. **Data Augmentation:**
   * Implementing advanced data augmentation techniques such as Random Erasing, Mixup, and Cutout to make the model more robust to variations in the dataset.
3. **Ensemble Models:**
   * Combining predictions from multiple high-performing models (e.g., DenseNet121 and MobileNetV2) to improve overall accuracy and robustness**.**
4. **Hyperparameter Tuning:**
   * Using Bayesian Optimization or Grid Search for optimizing hyperparameters like learning rate, batch size, and dropout rates for better convergence and performance.
5. **Optimization Techniques:**
   * Experimenting with more advanced optimizers like SGD with Momentum, AdamW, or Ranger to see if they lead to better results compared to the default Adam optimizer.
6. **Loss Function Exploration:**
   * Trying other loss functions like Label Smoothing Loss or Dice Loss, which could further improve performance on misclassified or ambiguous samples.
7. **Class Imbalance Handling:**
   * If class imbalance exists in unseen datasets, methods like Class Weighting, Synthetic Minority Oversampling (SMOTE), or Cost-Sensitive Learning can be incorporated.
8. **ONNX Model Deployment:**
   * Exploring lightweight deployment of ONNX models on edge devices or mobile platforms for real-world applications.
9. **Dataset Expansion:**
   * Gathering additional data or using techniques like Self-Supervised Learning or Semi-Supervised Learning to enhance the generalization capability of the models.
10. **Explainability and Interpretability:**
    * Incorporating tools like SHAP or Grad-CAM to visualize and interpret how models make predictions, which is essential for gaining insights and improving trust in the models.