CIFAR10 Image Classification

Introduction

This report entails the methodologies and findings on CIFAR10 image classification using TensorFlow. The challenge consists of building a baseline image classification model and optimizing it by transfer learning using a pre-trained model of choice. The report will document about the exploratory data analysis of the data, the preprocessing techniques, model architecture, the training and fine-tuning processes as well as the performance metrics and finally, model deployment.

1. Problem Understanding

The challenge focuses on developing image classification models for the CIFAR10 data. Two primary tasks are defined:

a) Building the Baseline Model: Using TensorFlow's Keras API, build a baseline image classification model. The task requires the training and the evaluation of the model capable of classifying the CIFAR10 image data.

b) Fine-tuning a pre-trained model for transfer learning: Selecting a pre-trained model and fine- tuning it using the dataset to classify the images and then comparing its performance with the baseline model..

2. *Exploratory Data Analysis and Pre-processing*

● EDA: Data exploration steps conducted include previewing the images to understand their characteristics and observing the class distribution of the images. Checking for class distribution is a necessary step in classification tasks to assess class balance and identifying any possible biases in the dataset because the model tends to be biased towards the majority class. In this case, the classes were equally distributed.

● Data pre-processing: The main pre-processing steps done on the data were normalization and encoding the labels. The normalization step ensures the pixel values range between 0 and 1which is suitable for input to the neural network because it contributes to model convergence and stability during training. The labels were encoded because it is easier for processing during training. The data was then saved to CSV files because of ease of use, storage and sharing.

3. Modeling

<u>Baseline Model</u>

Model Architecture: The baseline model architecture consists of the input layer, convolutional layers and fully connected layers. The input layer dimensions correspond to the pixel size of the input data.

- The first convolutional layer is useful in detecting textures and edges. It has 32 filters and a three by three kernel as well as the ReLu activation. The ReLu introduces non-linearity which helps in capturing intricate relationships in the data.
- Batch normalization is then applied because it helps in handling stability of model training as well as handling speed y normalizing the activation function.
- To correct and regulate overfitting, max pooling reduces complexity computation by minimizing the spatial dimensions of the feature maps
- Droput is then applied to prevent overfitting. It achieves this by dropping units randomly during training.

- The feature maps are the flattened by fully connected layers and passed through dense layers to do the last classification.
- The model is compiled with the Adam optimizer and categorical cross-entropy loss, which are standard choices for multi-class classification problems. The Adam optimizer is robust to noisy gradients and does bias correction, which increases efficiency. Cross-entropy loss adjusts incorrect predictions making it a good choice.

Model Training: The model was trained for 10 epochs to ensure the model learnt and adjusted its weights through the several steps. The samples processed was 64, as determined by the batch size parameter ensuring there is both stability and efficiency in computation. To monitor overfitting, a validation set was used to evaluate its performance and identify how well it can generalize to unseen data.

Model Performance: The metrics used include accuracy because it is straightforward, to measures the proportion of correctly classified instances out of the total instances. Loss was also used to measure the error between the predicted probabilities and the actual labels. Precision was used to help in understanding the model's ability to make accurate positive predictions while recall was used to help in understanding the model's ability to capture all relevant positive instances. The baseline model had an accuracy of 83.66% meaning that the model correctly classified 83.66% of the training images, and 81.99% of the validation images. It had minimal loss which kept decreasing, indicating better model performance.

Pre-trained Model (Vgg16 Model)

The pre-trained model chosen for this task is Vgg16 because of its deep architecture that contributes to its excellent performance. The architecture also contributes to its ability to capture details and patterns in the data gaining vast knowledge, minimizing the need for extensive training.

The first step of fine-tuning the model included freezing the base layers to maintain the learned features by retaining the weights that have already been pre-trained. To learn the features of this specific task, custom layers were added. The model was evaluated on validation and test sets but showed signs of overfitting, with an accuracy of 64% after training and a test accuracy of 60%. To improve the model performance, the top four layers were unfrozen and with a lower learning rate. This way, the model was able to learn more specific and relevant features. The validation accuracy increased and the loss decreased to 0.8002 implying better model generalization. The optimized model performed better with an accuracy of 0.92, than the baseline model which had an accuracy of 0.83. Therefore, it can generalize better on unseen data than the base model.

4. Model Deployment

The best model, which is the fine-tuned pre-trained model, was saved to a binary format because it is easier to share, ready for deployment using Flask. Flask was used because of its flexibility and it is a lightweight framework. A virtual environment was initially set up locally to manage the dependencies. The model was loaded in the deployment file and then setting up the prediction endpoint, and then running it locally to ensure it works as expected.

Conclusion

In this report, a thorough modeling was conducted to do image classification on the CIFAR10 dataset and investigated the predictive capabilities of these models and evaluated their performance on both validation and test datasets. Key findings were observed:

1. Model Performance: The fine-tuned pre-trained model outdid the baseline model. That is because of its deep architecture to capture intricate patterns.

2. Model Optimization: The pre-trained model improved predictive performance further, achieving an accuracy of 0.92 as well as a decreasing loss value. This shows the effectiveness of fine-tuning.

3. Model Selection: Based on a thorough analysis and evaluation of the image classification models, pre-trained models are the recommended choice for such tasks. Their excellent architecture and predictive performance, and efficiency make it ideal for real-world applications in image classification.

Recommendations

1. Pre-trained models implementation: I recommend using pre-trained models as the primary regression algorithm for image classification tasks. Its superior predictive performance and efficiency make it an excellent candidate for these prediction tasks.

2. Continuous Model Evaluation: Continuously monitor and assess the predictive models' performance in real-world scenarios. To ensure that the models remain accurate and relevant, they must be validated and refined on a regular basis based on new data and changing requirements.

3. Fine-tuning: Explore additional fine-tuning techniques and domain-specific insights to improve the models' predictive performance. Adjusting model parameters such as learning rates improve model performance.

4. Diverse Dataset Utilization: Additional step to improve model performance is data augmentation. This is because of diverse data and increase in robustness to various changes.

In conclusion, using pre-trained models as the primary image classification algorithm, combined with fine-tuning techniques and continuous model evaluation, can significantly improve predictive accuracy and reliability, resulting in more informed decision-making. These recommendations aim to drive informed decision-making and improve real world applications.