Big Data and Machine Learning(CSC-345) : Assignment Report

Prajwal Bharadwaj - 2337862

Contents

1 Introduction.

Utilizing the CIFAR-100 Dataset to understand the methodology of implementing various optimization techniques to converge to a known solution. In this present report, we are utilizing CNN (Convolutional Neural Network) Moreover, the implementation of CNNs demanded meticulous experimentation with different methodologies. Our rigorous efforts involved two distinct approaches, each yielding varied accuracies.
While our initial method for Coarse Data Classes attained a **50.590%** accuracy rate on super-classes, indicating a satisfactory level of recognition within broader categories, the accuracy dropped to **32.940%** when applied to Finer subclasses. This variance in performance underscores the capturing process intricacies involved in fine-grained object recognition and highlights the need for further refinement and optimization of algorithms to enhance accuracy across all levels of categorization within the dataset.

2 Exploring Methodology-I

2.1 Training the Coarse model

We began by exploring the Convolutional neural network (CNN) which is neither Supervised nor Unsupervised rather a feature Extractor evolved with the algorithm to identity classes. Instead of usual matrix math, CNN uses operations like convolution across layers, mainly for image processing and recognition. Our coding journey started with importing essential Python libraries and dataset files provided for the task. We used a specific method via Google Drive to upload and access these files easily and also utilised Google Colab's **T4 GPU** functionality to utilise for faster computation.
First, we loaded and organized the training set by rearranging it using the NumPy library's `transpose` function. We picked an index to view a sample image using `imshow` To categorize the super-class, we created a dictionary. For Keras' model compatibility, we formatted our data using the `to_categorical` function and then divided it into training and validation sets with `train_test_split` We made sure to normalize pixel values to a maximum of 255 units for standardization.

Next, we built our model by importing libraries, incorporating convolutional 2D layers, Max pooling layers, and two dense layers for the 20 super-classes. We used the `Relu` activation function. The model was compiled using the Stochastic Gradient Descent optimizer and fitted with training and validation datasets.

While developing the model, we created a function to visualize training accuracy, validation accuracy, training loss, and validation loss. This model achieved an accuracy of 50.24%. Using the `sklearn` confusion matrix library, we visually examined the accuracy of our predictions across various classes.

We also looked into the classes frequently chosen and neglected during prediction. In this super-class classification model, every class was predicted. We then predicted a sample set of images, noticing that our model accurately identified most superclasses. However, there were some discrepancies, such as predicting an aquatic mammal (like a whale) as a vehicle 2, possibly due to its color. Nonetheless, the model correctly identified other categories like trees, large carnivores, large man-made outdoor things, and fruits/vegetables (even though the crab/spider in the first image seemed confusing).

From the above Confusion Matrix, and the Accuracy computed, the values are nearly converging to the Benchmark values provided as a comparison check. Higher the accuracy, defines the lack of capturing feature ability due to less training data (Coarse Labels), hence the wrong predicament of higher accuracy.

2.2 Training the Fine model

We worked with subsets, each containing 500 training images and 100 testing images among a hundred subclasses. Adapting the model for subclass classification, we adjusted the dense layer's output to 100. Afterward, we visualized historical data and conducted predictions using a sample dataset.

The subclass-trained model showed a significant accuracy drop **32.940%** compared to the superclass model. Many subclass categories remained unpredicted. To enhance performance, we applied SVM techniques in Method 2.

From the above Confusion Matrix, and the Accuracy computed, the values are nearly converging to the Benchmark values provided as a comparison check. Lower the accuracy, defines the ability to capture features due to abundant training data (Fine Labels), hence the right predicament of lower accuracy.

3 Exploring Methodology-II

In this method, we'll be utilizing Support Vector Machine (SVM) Technique, a method which is linearly classifying between the features. Now, the method is classified with an accuracy of **31.40%** , with just lower computational cost dataset, indicating the best ability to capture more features with lower consumption of Training Data set. Hence, this explains the ability of SVM to classify better at hyperplane.[1]

4 Conclusion

1.The plotted graph below illustrates how the accuracy values obtained are reasonably close to the benchmark. It's noteworthy that the finer dataset exhibits lower accuracy, suggesting its enhanced capability to capture intricate features. This lower accuracy is indicative of the finer dataset's ability to discern and represent more nuanced characteristics within the data.

2.The confusion matrix depicted below for the Support Vector Machine (SVM) reinforces the accuracy value comparable to the proposed benchmark. Importantly, SVM achieves this with fewer computational resources when compared to Convolutional Neural Networks (CNN) employed with the finer datasets. Despite CNN's utilization of finer data, its ability to capture features doesn't significantly surpass SVM. This indicates that SVM excels in extracting pertinent features from the data in a more efficient manner.

5 References

[1] Christopher Bishop *Pattern Recognition and Machine Learning* Microsoft Research Ltd, 2007

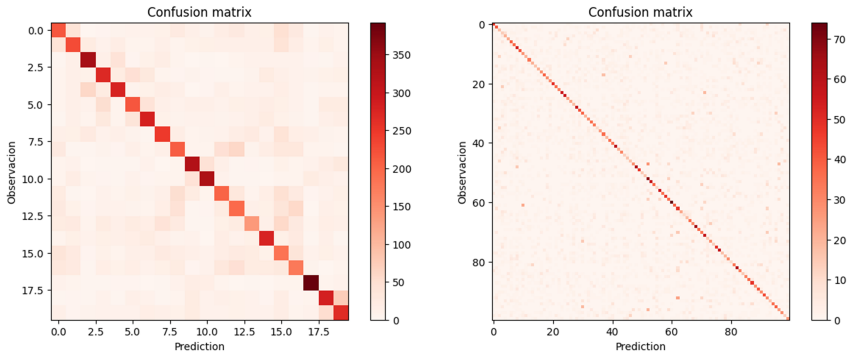

Figure 1: CNN - Confusion Matrix for Coarser and Finer Data Sets

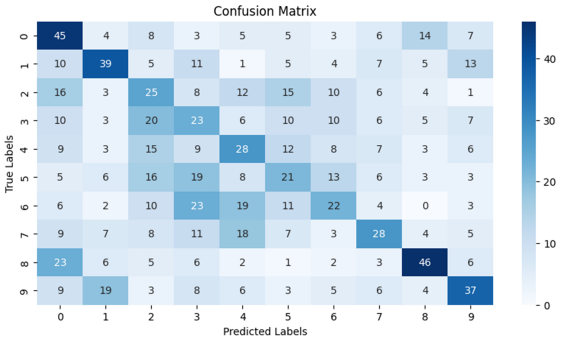

Figure 2: SVM - Confusion Matrix for Lower computational cost Data sets