DSAI4101 App Deep Learn & Neural Net-4

Ashraf Salem – 60104564

Term Project

To: Dr. Omer Koksal

1. Introduction

The project mainly focuses on how we can develop a deep learning model with PyTorch for multi-class classification tasks, As the dataset used contains class-labeled images with four categories: daisy, roses, sunflowers, and tulips. The goal was to classify them into categories using Deep Neural Network (DNN) and evaluate the performance using various metrics and visualizing misclassified examples.

1. Explanation of the architecture

The project was done by implementing a fully connected deep neural network (DNN) as it was designed to make the inputs of picture size to 224x244 as the architecture had an input layer to process the flattened images, Hidden layers with a gradually decreasing node size to be able to capture any complex patterns in the data and at the end an output layer. Relu activation function was used also in the hidden layers for non-linearity with SoftMax function also applied for probabilistic interpretation.

1. Loss function

The Choice of CrossEntropyLoss was because it’s a standard and effective choice for multi-class classification tasks as it combines the Softmax and NLLLoss functions, making it numerically stable and effective for training classification models. Moreover, while training the loss for training and test sets were plotted to be easier to analyze the model learning progress. In the graph provided the training loss was decreasing over the epochs which indicated that the model is effectively learning from the data. Moreover, the test loss decreases also after a few epochs as it suggests that while the model is learning well on the training data, its generalization on unseen data could be inconsistent.

A graph with blue and orange lines

Description automatically generated

*Fig .1. Train / test loss over epochs*

1. Hyperparameters

The Deep Neural Network Architecture Hyperparameters was tuned to optimize the performance and to ensure an efficient learning environment, The input size was indicating the dimensions of the flattened input images with an architecture consisting of 5 hidden layers with one output layer, In addition, the number of nodes was decreasing till the final layer that have equal nodes to targeted classes. Learning rate of a value 0.001 was selected to balance the speed and stability during convergence. Mini-batch of 64 was chosen also for its efficiency and stability while Relu was employed as the activation function to prevent vanishing gradients and ensure the effective learning environment. Moreover, to ensure adaptive learning rates and momentum capabilities Adam optimizer was used for further enhancements. The model was trained for 20 epochs, providing sufficient time to learn complex patterns without overfitting or excessive computational cost.

1. Code explanation

The model in the class “Classify\_Flower” was used to classify the flowers in categories also by using the PyTorch nn.sequential module to stack layers in a structured manner where the layers also include a fully connected layer with Relu activation function to enhance the model learning complex patterns. The arc

Training process was by a function that uses forward and backward for epochs as during each forward pass the model process the input data to generate predictions while the CrossEntropyLoss calculates the error between the predicted and true labels, On the other hand, backward pass used to compute gradients which is used to update model weights with Adam optimizer, at the end the training function save and record the loss after each epoch

Evaluation function was used to calculate performance metrics for the accuracy and test loss, additionally function includes tools to analyze the model Performances as it shows the loss curves for both training and test sets to help identify trends and prevent potential overfitting. Moreover, another function was used to simply show and highlight misclassified examples

1. Performance model

The model was evaluated with accuracy, F1-score and confusion matrix as the results achieved were an accuracy of 60% and F1-score of 59 %that both indicates a fair classification performance, Errors was primarily from misclassified classes which led to class confusion.

A black background with white text

Description automatically generated

*Fig .2. Confusion Matrix*

The confusion matrix highlighted the model’s difficulty in learning some features from certain categories

To improve the model performance techniques like regularization dropout and hyperparameter tuning could be applied. The model also does not show sever overfitting but underfitting is slightly there, ways like limiting epochs or using simpler architecture might help in avoiding overfitting

1. Error analysis and Misclassified examples

A close-up of flowers

Description automatically generated

*Fig .3. Misclassified examples*

Error analysis revealed misclassification examples from the train dataset which highlighted parts where the model struggles, as shown in the Figure 3 the model frequently misclassifies flowers within similar categories as this suggest that model finds some challenges to identify and distinguish flowers with some overlapping features such as colors

These error indicates that the model might be underfitting some classes to extract sufficient features, to solve techniques like using more complex architecture (CNNs) or fine-tuning hyperparameters could be employed

1. Discussion and conclusion

The model was quite good enough to classify flowers with sort off a high accuracy as it demonstrates that DNNs is powerful enough to capture complex relationships in data, However performance could be improved by dropout regularization, tuning hyperparameters or using convolutional neural networks (CNNs), all of these strategies with larger datasets could enhance our accuracy. The project showcased the effectiveness of DNNS for classification tasks and also a foundations for future enhancements and advancements