Herald College, Kathmandu

Artificial Intelligence and Machine Learning.

6CS012

Assignment - 1

`An Image Classification with Convolutional Neural Network.`

Apr 1, 2025

Contents

1 Assignment Details and Submission Guidelines

1.1 Assignment Details:

Due	Marks	Submission
Week - 12 - Tutorial	40	A Report and Code Notebook.

1.2 Plagiarism and AI Generated Content

Plagiarism of more than 20% and any AI-generated content found in the report will be reported for academic misconduct. Thus, we highly encourage you to submit your original work.

1.3 Submission Guidelines:

- Working In Group:
  - You are allowed to work on common dataset in group of 3 members (for more than that, please take a approval from your instructor), but remember members will be assessed individually.
  - Make a presentation and present in group after the final submission.
- Individual:
  - All the members must write individual report, code independently and be able to answer the question and explain the code.
- What to Submit?
  - You are expected to submit a report based on the task, and associated code file.
  - For Code:
    1. All solutions - Code must be written in the Jupyter notebook.
    2. All codes must be pushed to `GitHub` before the deadlines.
  - For report:
    1. Please follow the APA format; for a sample, see the attached documents.
  - Where to submit?
    Designated portal opened on Canvas.
- After Submission:
  - You are expected defend your work after the submission.Please consult with your respected instructor for date and time of the viva.

The Final Date for submission is: `Tutorial of week - 12.`

1.3.1 Naming Conventions:

You are supposed to strictly follow the naming conventions, and any file that does not follow the naming conventions will be marked as "0".
`File Name: WLVID_FullName(firstname+last).ipynb`,
`File Name: WLVID_FullName(firstname+last).pdf`

2 Assignment Overview

Image Classification with Convolutional Neural Network.

2.1 About Assignment:

In this assignment, you will undertake a comprehensive end-to-end deep learning project for image classification. The objective is to deepen your understanding of the entire deep learning pipeline, from data preprocessing to model building and evaluation. This assignment integrates knowledge from Weeks 1 to 6, challenging you to apply these concepts to real-world image classification tasks. This assignment has two parts:

- In part A: You will train and experiment with baseline CNN model from scratch.
- In part B: Fine tuned a pre - trained model just as you do in real world application.

2.2 Cautions!!!

In this assignment, you will perform a series of task (explained in section 3) for Image Classification with a approprite dataset and provide a rigorous rationale for your solutions. We will determine scores by judging both the soundness and cleanliness of your **code**, the quality of the **write-up(report)** and your ability to answer the question during **viva**. Here are examples of aspects that may lead to **point deductions:**

- Use of misleading, unnecessary, or unmotivated graphic elements.
- Unreadable code.
- Missing or incomplete design rationale in write-up.
- Ineffective encoding for your stated goal (e.g., distracting colors, improper data transformation).

Tools and Python Package which can be used for this assignments (listed but not limited to):

1. **Numpy library(np)**
2. **Matplotlib library(plt)**
3. **sickit Learn(sklearn)**
4. **Deep Learning Framework** - {**Recommended** - **Keras**}.

2.3 Learning Outcomes:

Learning outcomes can be following but not limited to:

1. Use numpy, keras as the primary tool to process image data in Python,
2. Use matplotlib library to produce various plots for visualization,
3. Perform some basic image manipulation
4. Understand the architecture and various layer of Convolutional Neural Network.
5. Be able to design, build and train CNN Model, with modern deep learning library (Keras) for image classification task.

2.4 Dataset Selection:

1. Please feel free to pick any datasets in image format that matches the task requirements. But please take pre-approval from your respected instructor and Module leader.
2. If you are not sure about which dataset to pick, select one from the options provided.
3. Note: No two groups in one Section can have same datasets.

The best source to find datasets are but not limited to:

1. Kaggle Datasets:
   Kaggle provides a high-quality dataset in various formats that we can easily find and download.
   https://www.kaggle.com/datasets
2. UCI Machine Learning Repository:
   This repository contains databases, domain theories, and data generators that are widely used by the machine learning community for the analysis of ML algorithms.
   https://archive.ics.uci.edu/ml/index.php.
3. Google's Dataset Search Engine:
   Google dataset search engine is a search engine launched by Google on September 5, 2018. This source helps researchers to get online datasets that are freely available for use.
   https://toolbox.google.com/datasetsearch.

3 Tasks - To - Do:

Please Complete all the Tasks as instructed.

3.1 Part A: Implementing Convolutional Neural Network from scratch. [30].

This section focuses on designing, training, and evaluating convolutional neural networks (CNNs) from scratch. You will first build a baseline model with a simple architecture, then extend it into a deeper model with regularization to analyze performance improvements. A thorough experimentation section will guide comparative analysis of different architectures and optimizers.

3.1.1 Data Understanding, Analysis, Visualization and Cleaning [3]:

Before building models, analyze and understand the dataset by answering the following questions:

- What does the dataset represent? Provide a brief description.
- How many total images are in the dataset?
- What is the distribution of images across different classes?
- How is the dataset split into training and validation sets? Justify your choice.
- What preprocessing techniques (e.g., resizing, normalization) were applied?
- What data generators, if any, were used for preprocessing and augmentation?
- If data augmentation was applied, provide visualizations of sample augmented images.

3.1.2 Design, Train, and Evaluate a Baseline Model [7]:

1. **Model Architecture: [2]:**
   - Build a baseline CNN model with the following structure:
     - Three Convolutional layers, each followed by pooling layers.
     - Three fully connected layers (FCN)
     - An output layer suitable for classification.
   - Use an appropriate activation function and kernel size for each layer.
   - Print and analyze the model summary.
2. **Model Training [2]:**
   - Train the baseline model for an appropriate number of epochs.
   - Plot the training vs validation loss curves over epochs.
   - Note: We recommend using Google Colab with GPU or TPU acceleration for training.

3. **Model Evaluation [3]:**
   - Evaluate the model using appropriate metrics (e.g. accuracy, precision, recall and F-1 score).
   - Perform inference on sample images and plot the result.
   - Discuss key observations about model performance.

3.1.3 Design, Train , and Evaluate Deeper Architecture with Regularization Layer [7]:

1. **Model Architecture [2]:**
   - Extend the baseline model by modifying the number of filters and layers to build deeper architecture{at least double the layer in comparison to baseline model}.
   - Extend the layers introducing the following regularization techniques.
   - Print and analyze the new model summary.
2. **Model Training [2]:**
   - Train the deeper model for an appropriate number of epochs.
   - Compare its training time and loss curves against the baseline model.
3. **Model Evaluation [3]:**
   - Evaluate the deeper model on the same performance metrics as the baseline model.
   - Discuss improvements (or lack thereof) compared to the baseline model.

3.1.4 Experimentation and Comparative Analysis [13]:

1. **Baseline vs. Deeper Model Performance [3]:**
   - Compare classification accuracy, loss, and evaluation metrics.
   - Discuss whether adding more layers and filters improved performance.
2. **Computational Efficiency [2]:**
   - Compare training time and computational cost of the two models.
   - Discuss the trade - offs between model complexity and efficiency.
3. **Optimizer Analysis: SGD vs Adam [5]:**
   - Train the deeper model with SGD and then Adam.
   - Compare the impact of these optimizers on convergence speed and final performance.
4. **Challenges and Observations [3]:**
   - Discuss any difficulties faced, such as overfitting and underfitting.
   - Report the total training time for deeper architecture.
   - Mention if hardware acceleration (e.g. Google Colab, with GPU/TPU was used.

3.2 Part B: Fine-Tuning a Pre-Trained Model (Transfer Learning). [10].

In many real-world deep learning applications, we leverage pre-trained model weights from models trained on large-scale datasets such as ImageNet. These pre-trained models (e.g., Inception, ResNet, VGG, MobileNet) are highly effective because they have already learned rich feature representations. Transfer learning allows us to adapt these models for our specific image classification task, reducing the need for extensive labeled data and computational resources.

3.2.1 Loading and Adapting a Pre - Trained Model [2]:

- Select an appropriate pre - trained CNN model (e.g. VGG, ResNet, Incpetion) based on the characterstics of your dataset.
- Load the pre - trained model using keras.
- Modify the model architecture to fit your specific classification task.
  - Remove the original fully connected layers.
  - Add a new dense layers tailored to your dataset's number of classes.
- Choose whether to freeze certain layers or fine - tune the entire model based on your dataset size and computational constraints.

3.2.2 Model Training and Fine - Tuning [5]:

1. Training Strategies:
   - Feature Extraction:
     - Freeze the convolutional base of the pre - trained model to retain prviously learned feature representations.
     - Train only the newly added layers for classification.
   - Fine - Tuning:
     - Unfreeze some or all convolutional layers and allow them to adjust to the new dataset.
     - This approach requires a lower learning rate to avoid overfitting and catastrophic forgetting.
2. Handling Input Size Differences:
   - ImageNet models expect a specific input image size (e.g.224×224 pixels for VGG, ResNet).
   - Resize dataset images to match the required dimensions using appropriate Data Generator in Keras.
3. Handling Output Layer Differences:
   - ImageNet trained models have a 1000 - class softmax output layer.
   - Modify the final classification layer to match the number of classes in your dataset.

3.2.3 Model Evaluation and Prediction [3]:

- Evaluate the fine tuned model using the same performance metrics as in Part A.
- Make predictions on test data and compare results with the baseline and deeper models from Part A.
- Discuss whether transfer learning outperforms training from scratch for your dataset.

4 Reference and Guidelines for Final Report.

This section serves as a reference for structuring your project report.

1. Title

Provide a clear and descriptive title that represents your work effectively. The title should reflect the problem being solved or the key focus of your project.

2. Abstract

The abstract should be a concise summary (150–250 words) of your work, including:

- **Objective**: What problem are you solving?
- **Methods**: Briefly describe the models/techniques used.
- **Key Findings**: Highlight significant results.
- **Conclusion & Impact**: Summarize insights and recommendations.

3. Introduction

Clearly define the problem statement and its real-world significance. Provide background information on deep learning for image classification and discuss relevant prior work. Outline the goals and scope of the project.

4. Dataset

Describe the dataset(s) used:

- Source (e.g., CIFAR-10, custom dataset).
- Number of images, classes, and resolution.
- Any preprocessing applied (e.g., resizing, normalization).

If using a custom dataset, explain:

- Collection method (e.g., web scraping, manual labeling).
- Labeling approach (manual or automated).
- Challenges faced during data collection.

5. Methodology

Detail the deep learning models used, including architecture design:

- **Baseline Model**: 3 convolutional + pooling layers, 3 FCN layers, output layer.
- **Deeper Model**: Additional layers with Batch Normalization and Dropout.

- **Loss Function**: Cross-entropy, etc.
- **Optimizer**: SGD, Adam, or other choices.
- **Hyperparameters**: Learning rate, batch size, epochs.

6. Experiments and Results

Describe the experiments performed to compare models.

6.1 Baseline vs. Deeper Architecture

Compare accuracy, loss, and training time. Analyze whether adding layers/filters improved performance.

6.2 Computational Efficiency

Compare training time and memory requirements. Discuss hardware acceleration (e.g., GPU/TPU usage in Google Colab).

6.3 Training with Different Optimizers

Train the deeper model using both SGD and Adam. Compare convergence speed and final accuracy.

6.4 Challenges in Training

Discuss issues such as overfitting, underfitting, or unstable training. Report total training time and computational cost.

7. Fine-Tuning or Transfer Learning

Describe the pre-trained model chosen (e.g., VGG, ResNet, Inception). Explain whether you used feature extraction (freezing layers) or fine-tuning. Show and discuss performance improvements using transfer learning. Provide a comparison with models trained from scratch.

8. Conclusion and Future Work

Summarize key findings and insights from the project. Discuss trade-offs between different architectures and methods. Identify limitations or failure cases in your models. Suggest potential improvements (e.g., additional datasets, hyperparameter tuning). Mention future research directions if applicable.