SAMSUNG

AI Course

Capstone Project Idea Presentation

Together for Tomorrow!
Enabling People
Education for Future Generations

Course	AI Course
Team Name	Code HUNTS
Team Members	Hamza Waheed, Noman Abid
Project Title	Make Unification Simple in Image Classification

Goal

The ultimate aim of this research is mainly to propose the optimal methods for fine-tuning the pre-trained deep learning models towards image classification. The goal is to improve the models' precision and minimize the computational expenses using sequential fine-tuning strategies and determining task arithmetic operations.

Abstract

Transfer learning is a key method necessary for enhancing pre-trained models' results in deep learning models. However, model fine-tuning in the case of the multiple tasks at once leads to the accuracy decrease and general performance degradation. This proposal involves assessing if fine-tuning models on one task at a time produces better results than fine-tuning on more than one task at the same time as has been the norm. MNIST datasets will be employed in the experiments with the aim of fine-tuning models step by step on certain classes chosen with the assessment of the given approach's influence on the accuracy. Thus, through progressive validation of this hypothesis, our work proposes an enhanced procedure for better performance on the task of interest while overcoming the limitations of traditional fine-tuning approaches.

Proposed Method

Dataset Preparation

- The MNIST dataset comprises of 60, 000 samples for training and 10, 000 samples for testing and will be used in this research.
- 3, 4 and 7 classes are chosen to carry out fine tuning experiments, out of thousand samples for each class exclusively used for particular task.
- The rest of the data is stored separately for model calibration and for comparison with baseline results.

Model Design

- Input layer: 784 parts (28×28 pixel for ten images of MNIST)
- Hidden layers: 128 and 64 amps, respectively
- Output layer: 10 units for, the classification of digits 0- 9.
- To benchmark, the model will first be trained on the whole MNIST data to get a starting set of results.

Sequential Fine-Tuning Process

- For fine-tuning, the preprocessed model learnt will be fine-tuned successively for each chosen class of 3, 4, and 7.
- Here, it is proposed that fine-tuning will be carried out for 10 epochs in each of the classes while maintaining information learned from previous tasks.
- The last experiment will result from the combination of all three classes' datasets and the simultaneous updating of the model to contrast with the previous approach.

Evaluation and Metrics

- The models' performances will be measured using accuracy, precision, recall, and F1-score.
- That way, confusion matrices will be helpful driving understanding of classifier's general tendencies of error.
- Instead, actual loss curves and trends of accuracy will be charted in order to understand training dynamics and make sure that sequential fine-tuning works.

Implementation Environment

- To implement and test these theories, several experiments will be performed in the Python programming language; Libraries that will be utilized are; PyTorch, NumPy, Matplotlib, Seaborn.
- Otherwise, the use of Graphics Processing Unit for enabling faster model training and evaluation will be adopted wherever possible.
- The goal of this methodology is to prove the previously formulated hypothesis that sequentially fine-tuning a model on separate tasks leads to better model accuracy and better performance of the model on certain tasks compared to jointly fine-tuning on multiple tasks.

Data

Data Source:
This work's dataset is the MNIST (Modified National Institute of Standards and Technology) dataset which has become the benchmark data set for many machine learning and computer vision tasks. It consists of 60 grayscale images each of ten different digits starting with 0 followed by 9. The dataset is available for public use and will be imported using PyTorch torchvision where the data has been normalized for easier access.

Dataset Details:

- **Training Set:** 60,000 images (28×28 pixels).
- **Test Set:** 10,000 images (28×28 pixels).
- **Classes:** A total of 10 different classes of objects which include digits from zero up to nine.

Initial Pre-Training:
All the training dataset will then be employed to pre-train a Simple Neural Network, (SimpleNN) in order to get a measure of how well other models will perform.

Task-Specific Fine-Tuning:

- For the classes 3, 4 and 7, data samples of 1000 instances each will be extracted.
- These subsets will be stored as individual datasets and used for sequential fine tuning analysis in terms of the specific task or tasks of interest.

Combined Dataset Experiment:

A new merged dataset with the extracted classes (3, 4, and 7) will be used to assess the performance of joint fine-tuning, with the results compared to those of the method of sequential fine-tuning.

Pre-Processing:

- The raw pixel data of images will be scaled again to to the range [-1, 1] in order to preserve the input range of the neural network.
- PyTorch's torchvision.transforms will be used for any transformations

required as it properly loads data to the right model format.

Expected Outcome

The process starts with the evaluation of the task vector, which is the vector between a pretrained model and a corresponding finetuned model. This is done by simple subtraction of the parameters (or embeddings) of the pretrained model from that of the finetuned model. Since the current study is based on the pretrained model, the parameters of that model will used to analyze the dataset.=Task Vector=Finetuning's parameters − Pretraining's parameters difference between a pretrained model and its finetuned counterpart. This is achieved by subtracting the parameters (or embeddings) of the pretrained model from those of the finetuned model. Mathematically, this can be expressed as:

Task Vector = Parameters of Finetuned Model − Parameters of Pretrained Model

This task vector incorporates the detail of the certain changes and adaptations made to the pretrained model during the finetuning to the specific task.

Subsequently, we use this task vector to fine-tune such a pretrained model for the same task as the one used previously. This is achieved by incorporating the task vector to the parameters of the new pretrained model of the project. In other words, it can adjust the newly defined target model with the help of new pretrained model without going through full training process. Since the finetuning process is based on BERT model, some of the parameters of the new model includes the following;

Parameters of the Pretrained Model newPars of New Finetuned Model= Pars of New Pretrained Model + Task Vector between a pretrained model and its finetuned counterpart.

This is achieved by subtracting the parameters (or embeddings) of the pretrained model from those of the finetuned model. Mathematically, this can be expressed as:

Task Vector=Parameters of Finetuned Model−Parameters of Pretrained Model

Next, we leverage this task vector to adapt a similar pretrained model to perform the same task. This is accomplished by adding the task vector to the parameters of the new pretrained model. Essentially, this operation fine-tunes the new pretrained model without requiring a full training process. Mathematically, the new model parameters can be expressed as:

Parameters of Finetuned Model (New)=Parameters of Pretrained Model (New)+Task Vector

It allows the transferral of learned transformations (as represented by the task vector) in subsequent models, thereby saving a great deal of computation while offering model-specific accuracy.

Using this technique, we effectively transform task-specific information from one pretrained model to the other making it the best approach in situations whereby the amount of computational resources is a limiting factor or the need for quick adaptability arises.

Role by Member

Exploratory Data Analysis (EDA) of the dataset will be conducted by Nouman Abid. This is about providing a detailed review of the data and reasons behind collecting the data set, data cleaning analysis and summarization, and variable analysis and determination of association. Nouman will also be in charge of cleaning and pre-processing the data as and when required for other stages within the project. Moreover, Nouman will make documentation of the project including the specification of the dataset, the detailed explanation of the analysis activities, and lessons learned on the EDA phase. The result described in this documentation may act as the reference material to interpret the information and the nature of the data it contains.

Ramadhan Awadh will work mainly on the model training part of the project whereby he will fashion a machine learning or deep learning model suited for the project. He will also make sure that the output for each model is as accurate and performs well than the other. Also, Hamza will calculate the task vector that looks at the variation of a pretrained model with its finetuned version to apply in the same models optimization. In addition to his technical responsibilities on the project, Hamza will write down the manner in which the model was trained, how the various vectors of the 'task' feature were computed, and anything that occurred in the process that needed to be addressed or decided. It will support Nouman's documentation, which will be the record of the complete workflow and methods of the project.

Comment & Assessment	