Data & AI 5/Artificial Intelligence: Deep Learning Project

Alexander Michielsen

Overview

In this project, you will work in groups of three. Your goal is to fine-tune multiple pretrained vision models on a real-world image classification task, explain how it works using Grad-CAM or similar, and document your findings in a short report. In addition, you are also going to compare pretrained models with a simple CNN built from scratch.
Imporant: This project is not just about building a model that works. Critical understanding, meaningful experiments, and clear documentation are key to a successful project.

Objectives

- Apply transfer learning to a custom image classification task.
- Conduct meaningful experiments and comparisons with different models.
- Train and optimize the model, adjusting hyperparameters as necessary.
- Explain the model's decisions using Grad-CAM.
- Evaluate the model's performance using appropriate metrics and analysis techniques.
- Document the process and findings in a comprehensive report (woven within your notebook).

Project Requirements

Note: These are the minimum requirements for a passing grade. You are encouraged to go beyond these requirements to demonstrate your understanding and creativity which will be reflected in your grade. Have a look at the section on possible extensions for inspiration. Please also take into account that part of your final grade will be based on the oral examination of the project.

0. Research and Planning

- Research pretrained vision models suitable for transfer learning.
- Research data preprocessing techniques for CNNS.
- Write a brief plan outlining your approach, including dataset choice, preprocessing, model selection, and evaluation metrics.
- Justify your choices based on the research you have done.
- Include this plan as a separate section in your notebook.

1. Data Preprocessing and Augmentation

- Choose one of the provided datasets on Canvas.
- Apply the necessary exploration and preprocessing steps to prepare the data for training, following the ML lifecycle seen during the course.

2. Model Training and Optimization

- Fine-tune a pretrained vision model (e.g., ResNet, VGG, EfficientNet) on your preprocessed dataset. Make a **justified choice** of model and transfer learning strategy (i.e., don't choose blindly).
- First train a baseline with default parameters, then, tune at least **one** hyperparameter (e.g., learning rate, batch size). To do this, use 3 well-chosen values for this hyperparameter and observe the effects.
- Build a simple CNN from scratch (similar to ones seen during class). Train it for an equal number of epochs as the pretrained models.
- Log training progress and save checkpoints **for every run**. Use an appropriate logging tool such as Tensorboard for this. Think about which metrics you want to log **before the start of your experiments!**
- Aim for 5-15 epochs for each run depending on dataset size.

3. Model Explanation

- Use Grad-CAM to visualise the baseline TL model's decision-making process.
- Explain how the model arrives at its predictions and what features it focuses on.
- Include at least 5 example visualizations
- Include error analysis: show and discuss at least 3 misclassified samples (can be part of the 5 visualisations from the previous step). Discuss predicted vs. true class, what the Grad-CAM reveals, and what feature(s) the model seems to over-rely on.

4. Evaluation and Analysis

- Assess each model's performance using appropriate metrics and visualisations.
- Compare the performance of the pretrained model with the CNN built from scratch.
- Compare the performance of the default pretrained model and the tuned variants
- Also include the training progress of all models (loss and accuracy curves).
- 1–2 paragraphs for each analysis is enough. Avoid overly vague, generic, or AI-generated content (i.e., no AI slop).
- Provide recommendations for future work.

5. Reporting

- Combine code, explanations, and results (including curves, analysis, . . .) in **one** clear, well-structured Jupyter notebook.
- Use the following structure:
  - Introduction and plan
  - Data exploration and preprocessing
  - Baseline model (with performance analysis)
  - Hyperparameter experiments (with performance analysis)
  - Scratch CNN (with performance analysis)
  - Grad-CAM and error analysis
  - Final comparison between models
  - Conclusions and future work

Deliverables

One ZIP file containing **ONLY**

- Your Jupyter notebook (.ipynb file) with all cells run and error-free.
- A folder (*saved_models*) containing a folder for each final model, each of which should contain:
  - Checkpoint file of the final model
  - Logs (Tensorboard event files)

Do **NOT** include:

- Separate PDF/Word/Markdown,. . . report
- Raw datasets
- Multiple notebooks
- A README file

Additional Notes

- Each team member needs to have a complete understanding of all the work that has been delivered.
- You can go as far as you want with this project, however, keep your time management in check. Prioritize if necessary.
- Consider using cloud-based platforms like Google Colab for GPU support.
- Training does not need to converge fully, aim for meaningful comparisons rather than fully optimized models

Possible Extensions

Only look into these once you have completed all aforementioned steps! These are bonus steps for additional marks! You cannot use these to compensate for any of the core requirements. Only when they are added on top of the core requirements do you get bonus points. If you have any other ideas, feel free to implement them as well.

- **Model Exploration and Advanced Architectures**
  - Try another transfer learning strategy from the one implemented. Compare performance.
  - Fine-tune a second pretrained model and compare performance (e.g., ResNet vs EfficientNet)
- **Data Strategy and Robustness Testing**
  - Combine datasets from multiple sources to expand or diversify training data. Analyze how this affects model performance.
  - Test on out-of-distribution (OOD) samples or edge-case inputs. Describe the effects on model performance.
  - Compare model performance when trained on different fractions of the dataset (e.g., 20%, 50%, 100%)
  - **Warning: Advanced, only try if you have time!**: Employ Generative Adversarial Networks (GANs) to create synthetic images, expanding the dataset.
- **Explainability**
  - Apply another XAI technique (e.g., Integrated Gradients, occlusion) including visualizations and discussion
- **Deployment**
  - Build a simple web interface that accepts image uploads and returns predictions (possibly with grad-CAM visualisations).

Submission Guidelines

- **Deadline**: *28 December 2025, 23:59*.
- **Submission Method**: Upload your ZIP file to Canvas

Good Luck!