Multimodal Hateful Memes Detection

May 4, 2025

1 Project Overview

This individual project, weighted at 15% of the course grade, requires you to develop a multimodal machine learning system to classify memes as hateful or non-hateful using the Hateful Memes dataset from Meta AI. You will use PyTorch to implement a custom dataset class, preprocess and visualize data, compare text (LSTM, BERT) and image (CNN, ResNet) models, and develop late and early fusion strategies. The project emphasizes data augmentation for class imbalance, TensorBoard for evaluation, and a novel contribution. The report (5–8 pages,) must include visualizations of late and early fusion diagrams, model and strategy comparisons, and evaluation results. The deadline is 15 May 2025.

2 Learning Objectives

- Implement a custom PyTorch dataset class and dataloader with augmentation for class imbalance.
- Preprocess and visualize image and text data, highlighting important text words.
- Compare text processing (LSTM, BERT) and image processing (CNN, ResNet).
- Develop and compare late and early fusion strategies for multimodal classification.
- Evaluate models using TensorBoard and justify results in a comprehensive report.

3 Project Requirements

You will work individually using PyTorch and the Hateful Memes dataset (provided with access instructions). If unavailable, an alternative multimodal dataset will be supplied. The project is divided into two deliverables: code (80%) and report (20%).

3.1 Tasks

1. **Preprocessing, Dataset Class, and Visualization** (15%):
   - Implement a custom PyTorch `Dataset` class to load images and text, using `torchvision.transforms` for images and `transformers` tokenizer for BERT.
   - Create a `DataLoader` with batching (e.g., batch size 32, 64, or 128) and shuffling.
   - Preprocess images: resize (224x224), normalize (ImageNet stats).
   - Preprocess text: tokenize, remove stopwords, prepare for LSTM (GloVe embeddings) and BERT.
   - Address class imbalance with augmentation: image (rotation, flipping via `torchvision.transforms`), text (synonym replacement).
   - Visualize data: display sample memes, class distribution histogram, word cloud of important text words (high TF-IDF terms) using `matplotlib` and `wordcloud`.
2. **Late Fusion Model** (25%):
   - Implement separate text and image models:
     - **LSTM**:nn.embeddings, bidirectional LSTM, linear classifier.
     - **BERT**: Fine-tune `bert-base-uncased` using `transformers`.
     - **CNN**: n-layer convolutional network (e.g., 32/64/128 filters), max-pooling, fully connected layers.
     - **ResNet**: Fine-tune `torchvision.models.resnet50(pretrained=True)`.
   - Combine outputs (logits or probabilities) by concatenation and pass through a linear layer for final prediction.
   - Train using `torch.optim.Adam`, apply dropout for regularization.
   - Compare LSTM vs. BERT and CNN vs. ResNet within late fusion (e.g., AUROC, training time).
3. **Early Fusion Model** (25%):
   - Extract features: LSTM hidden states or BERT `[CLS]` embeddings for text, CNN/ResNet penultimate layer for images.
   - Concatenate features and feed to a multi-layer perceptron or attention-based classifier.
   - Fine-tune BERT and ResNet using transfer learning.
   - Optionally, implement cross-modal attention to weigh text/image contributions.
   - Train and compare with late fusion (e.g., AUROC, computational efficiency).

4. **Evaluation and TensorBoard** (15%):
    - Compute AUROC, precision, recall, and F1-score for late and early fusion models.
    - Use `torch.utils.tensorboard` to log training/validation loss, AUROC, and sample predictions.
    - Visualize results: confusion matrices, ROC curves via `matplotlib`.
    - Analyze correctly/incorrectly classified memes, justifying results (e.g., why BERT outperforms LSTM, why early fusion improves AUROC) and limitations (e.g., class imbalance, dataset biases).

3.2 Deliverables

- **Code Repository** (80% of project grade):
    - Code repository with custom `Dataset` class, `DataLoader`, preprocessing, visualization, LSTM, BERT, CNN, ResNet, late/early fusion models, TensorBoard logs, and evaluation scripts.
    - Well commendted and clear reproduction instructions.
- **Report** (20% of project grade):
    - 5–8 pages.
    - Sections: Introduction, Methodology (dataset class, preprocessing, fusion strategies), Results (TensorBoard metrics, model/strategy comparisons), Discussion.
    - Visualizations: Diagrams of late and early fusion (e.g., TikZ or external tool), evaluation plots (confusion matrices, ROC curves).

4 Timeline

Deadline: 15th May 2025

5 Grading Rubric

The project is worth 15% of the course grade, with the following breakdown:

5.1 Code Repository (80%)

5.2 Report (20%)

Criterion	**Description**	**Points**
Preprocessing & Dataset (15%)	Functional `Dataset` class, `DataLoader`, augmentation for imbalance, visualizations	15
Late Fusion (25%)	Implements LSTM, BERT, CNN, ResNet, combines outputs, optimizes	25
Early Fusion (25%)	Extracts features, concatenates early, optimizes, compares with late fusion	25
Evaluation & TensorBoard (15%)	Logs metrics, visualizes results, analyzes predictions	15

Table 1: Code Repository Rubric

Criterion	**Description**	**Points**
Clarity & Structure	Well-written, logical flow, IEEE format, 5–8 pages	5
Technical Depth	Details dataset class, preprocessing, fusion strategies, comparisons	5
Visualizations	Includes late/early fusion diagrams, evaluation plots	5
Analysis & Novelty	Justifies results, discusses limitations, describes novel contribution	5

Table 2: Report Rubric