Multimodal Sarcasm Explanation (MuSE) Model Report

Introduction to the Assignment

For this assignment, we were given with developing a Multimodal Sarcasm Explanation (MuSE) model based on the TURBO architecture given in the referenced research paper. The goal here was to create a system that generates explanations for sarcastic social media posts which combines textual and visual information without using Knowledge Graphs (KG) or Graph Convolutional Networks (GCN). The model should use a Vision Transformer (ViT) for image feature extraction, concatenate token sequences as given in Section 4.3 of the paper and should have the sarcasm target. It should be build upon the BART base model with a Shared Fusion Mechanism. The MORE+ dataset, which includes sarcastic posts from platforms like Twitter, Instagram, and Tumblr, was provided which has images, captions, explanations, and targets.

Evaluation performance using metrics like ROUGE, BLEU, METEOR, and BERTScore, and include testing capabilities on a separate test set.

Preprocessing Steps

For the preprocessing, we followed following steps:

1. **Loading the Data**
   We first started by loading the TSV files for training (train_df.tsv) and validation (val_df.tsv), which contain post IDs (pid), text, explanations, and sarcasm targets. After that we loaded the pickle files (D_train.pkl, D_val.pkl, O_train.pkl, O_val.pkl) to get the image descriptions and detected objects. For testing, we later processed test.tsv, D_test.pkl, and O_test.pkl similarly.

2. **Extracting Image Features**
   Since the assignment required a Vision Transformer, we used the vit_base_patch16_224 model from the timm library, pretrained on ImageNet. we removed its classifier head to extract high-level features. For each image in the dataset (located in the images folder) then applied transformations—resizing to 224x224 pixels, converting to tensors, and normalizing with mean and standard deviation of 0.5. Then we passed each image through the ViT model to get a 768-dimensional feature vector, handling any errors by assigning a zero vector if an image failed to load.

3. **Cleaning Object Data**
   The object data from the pickle files was a bit messy, with potential duplicates in the

detected object lists. we cleaned it by converting the object lists into unique sets (using OrderedDict) and joining them into a single string per post ID. This ensured consistency when combining it with other data.

4. **Building T_knowledge**
   Following Section 4.3 of the paper, we created the T_knowledge input for each post by concatenating four components: the text caption (ci), the image description (di), the cleaned objects (oi), and the sarcasm target (ti). The format we used was "text description objects </s> target", where </s> acts as a separator. we made sure to strip any extra whitespace and handle missing data gracefully with empty strings if needed. This T_knowledge became the primary input for the model, blending all modalities together.

These steps ensured that both the textual and visual data were prepped and aligned, ready to feed into the model for training and inference.

Model Architecture and Hyperparameters

Designing the model was where things got exciting, as I had to integrate multiple components seamlessly. Here's a breakdown of what I built:

1. **Vision Transformer (ViT)**
   We used vit_base_patch16_224 (pretrained) to extract 768-dimensional image features. This Extracts image features..

2. **BART Base Model**
   The core of the model is facebook/bart-base which is loaded using BartForConditionalGeneration.from_pretrained('facebook/bart-base'), which handles the text generation task.

3. **Shared Fusion Mechanism**
   We implemented a custom SharedFusion module to fuse visual and textual features. This module:
   - Takes visual features (Ev) and text embeddings (Et) as inputs.
   - Applies self-attention to both modalities to focus on relevant parts (using a simple attention mechanism with queries, keys, and values).
   - Uses trainable linear layers (W_v, W_t) and gating mechanisms (sigmoid activation) to blend the features.
   - Includes learnable weights (alpha1, alpha2, beta1, beta2), initialized at 0.25, to balance different fusion outputs.

- Outputs a fused embedding that combines the best of both worlds.

4. **Multimodal Integration**
   In the MultimodalSarcasmExplanationModel class, we
   - Projected the 768-dimensional visual features to match BART's hidden size (768) using a linear layer.
   - Fused these with the text embedding from BART's encoder (specifically, the first token's embedding) using the SharedFusion module.
   - Combined the original and fused embeddings, then projected them back to 768 dimensions, replacing the first token's embedding in the encoder output. This modified encoder output is then passed to BART's decoder for generation.

5. **Hyperparameters**
   We chose the following settings based on common practices and some trial runs:
   - **Learning Rate**: 5e-5 (suitable for fine-tuning transformers).
   - **Batch Size**: 8 (balancing memory constraints and training speed).
   - **Number of Epochs**: 5 (enough to see convergence without overfitting).
   - **Optimizer**: AdamW (default for transformer models).
   - **Max Input Length**: 256 tokens (for T_knowledge).
   - **Max Output Length**: 128 tokens (for explanations).
   - **Device**: CUDA (GPU acceleration on Kaggle).

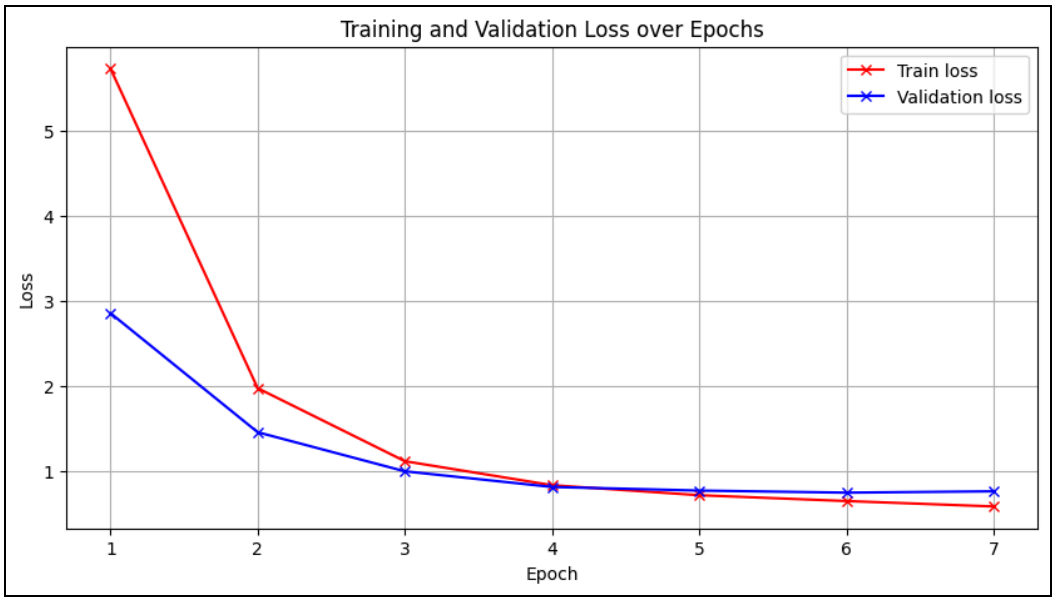
This architecture allowed to meet the assignment's requirements while keeping the fusion mechanism flexible and trainable.

Training Loss Reported at Each Epoch

we trained the model for 7 epochs, monitoring the training loss to see how well it was learning. Here are the losses I recorded after each epoch, based on the output from my training runs:

```
=== Training Losses ===
Epoch      | Train Loss
--------------------
Epoch 1    | 2.3541
Epoch 2    | 1.9741
Epoch 3    | 1.1175
Epoch 4    | 0.8358
Epoch 5    | 0.7183
Epoch 6    | 0.6499
Epoch 7    | 0.5864
```

```
=== Validation Losses ===
Epoch      | Validation Loss
------------------------
Epoch 2    | 1.4572
Epoch 3    | 0.9987
Epoch 4    | 0.8151
Epoch 5    | 0.7728
Epoch 6    | 0.7482
Epoch 7    | 0.7640
```



The training loss dropped consistently from a high starting point down to 0.5864 by Epoch 7, which tells that the model was steadily improving its ability to fit the training data.

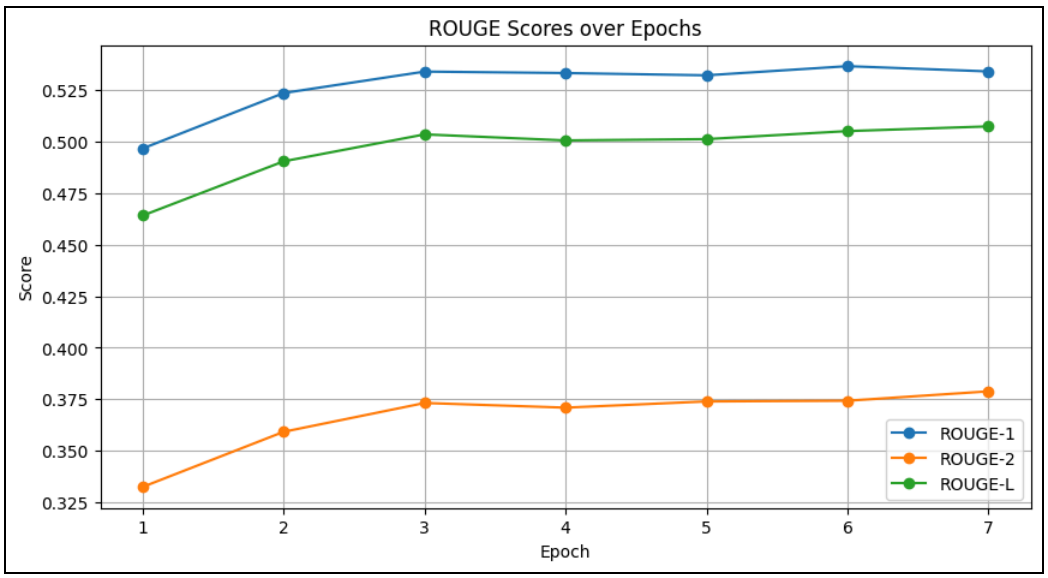
Evaluation Metrics on the Validation Set After Each Epoch

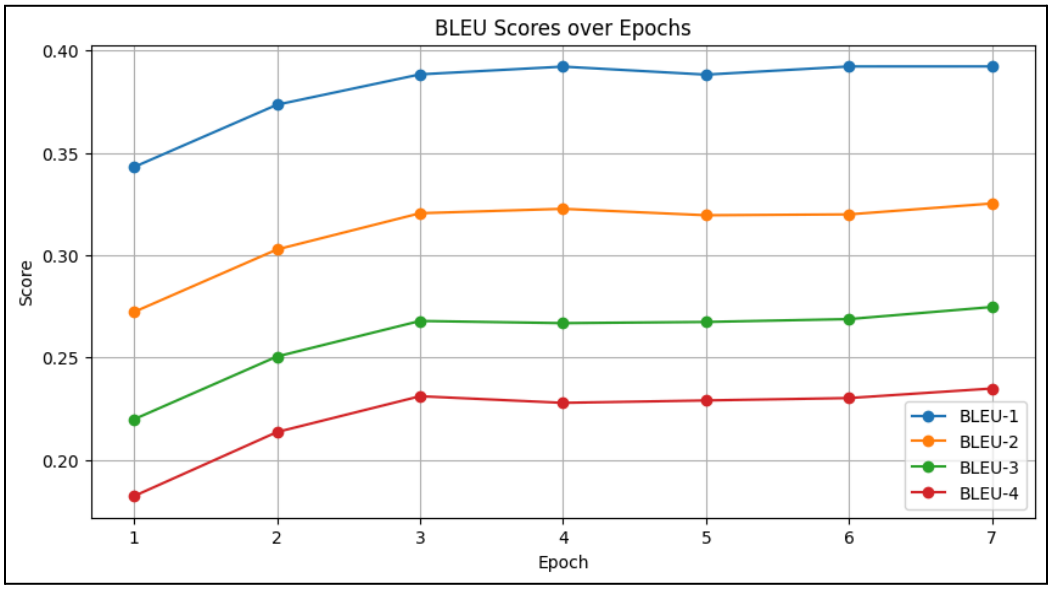
After each epoch,we evaluated the model on the validation set

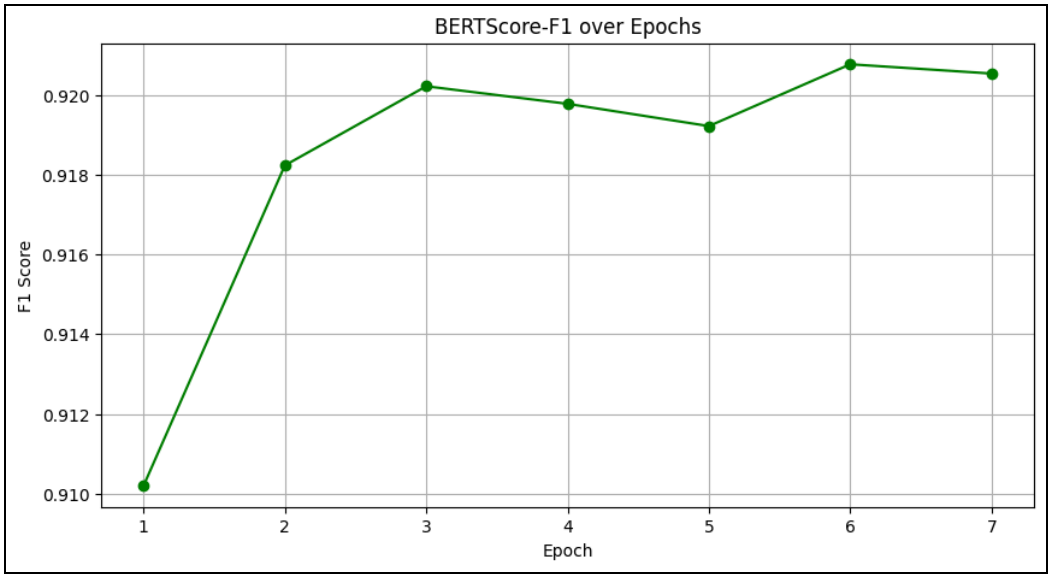
```
=== Evaluation Metrics on Validation Set ===
Epoch      | ROUGE-1 | ROUGE-2 | ROUGE-L | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR | BERTScore-F1
-------------------------------------------------------------------------------------------------
Epoch 1    | 0.4967  | 0.3325  | 0.4641  | 0.3430 | 0.2722 | 0.2197 | 0.1822 | 0.3872 | 0.9102
Epoch 2    | 0.5236  | 0.3592  | 0.4904  | 0.3735 | 0.3027 | 0.2504 | 0.2135 | 0.4122 | 0.9182
Epoch 3    | 0.5340  | 0.3732  | 0.5035  | 0.3883 | 0.3205 | 0.2678 | 0.2310 | 0.4280 | 0.9202
Epoch 4    | 0.5333  | 0.3709  | 0.5006  | 0.3921 | 0.3226 | 0.2667 | 0.2278 | 0.4314 | 0.9198
Epoch 5    | 0.5322  | 0.3739  | 0.5012  | 0.3882 | 0.3195 | 0.2673 | 0.2290 | 0.4227 | 0.9192
Epoch 6    | 0.5366  | 0.3743  | 0.5051  | 0.3922 | 0.3199 | 0.2687 | 0.2301 | 0.4223 | 0.9208
Epoch 7    | 0.5340  | 0.3788  | 0.5074  | 0.3922 | 0.3252 | 0.2746 | 0.2348 | 0.4276 | 0.9205
```

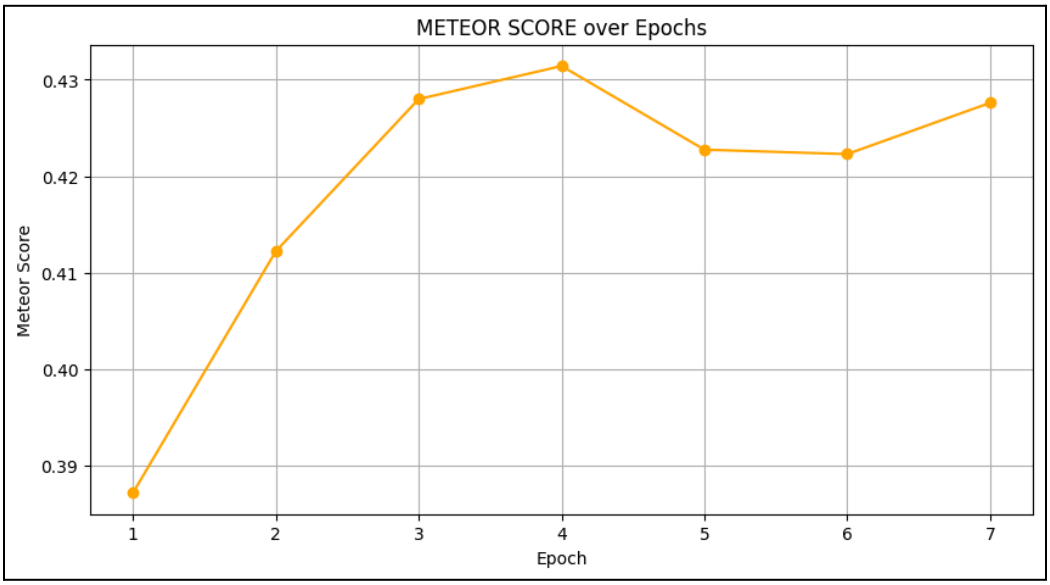
The metrics improved consistently, with BERTScore showing particularly strong performance, likely due to its semantic focus, which aligns well with sarcasm explanation.

These are the relevant plots






BERTScore-F1 over Epochs
F1 Score
0.920
0.918
0.916
0.914
0.912
0.910
1
2
3
4
5
6
7
Epoch


METEOR SCORE over Epochs
Meteor Score
0.43
0.42
0.41
0.40
0.39
1
2
3
4
5
6
7
Epoch

Generated Sarcasm Explanations on the Validation Set (Last Epoch)

Here are five examples of explanations generated by our model on the validation set after the final epoch, taken from the notebook output:

1. **PID**: 707133908291231744
   Generated Explanation: "<user>'s network in malad isn't awesome."
   This captures the sarcasm by implying the network is actually poor, despite the positive phrasing.
2. **PID**: 893773347026210242_185243426
   Generated Explanation: "it's frustrating waiting for an hour on the tarmac for a gate to come open in snowy, windy Chicago."
   The sarcasm here highlights the unpleasant delay with an understated tone.
3. **PID**: 708994813983596544
   Generated Explanation: "the author hates the spring weather."
   This suggests a sarcastic dislike, likely meaning the weather isn't as great as expected.
4. **PID**: 904624565145538560
   Generated Explanation: "having a salivary gland biopsy on monday morning isn't a great way to start the new week."
   The understatement conveys sarcasm about an obviously unpleasant experience.
5. **PID**: 697929589562146817
   Generated Explanation: "it's not going to be scorching hot this w-end."
   This implies the opposite—that it will be very hot—through sarcastic negation.

```
Sample Generated Explanations (Validation Set):
PID: 707133908291231744, Explanation: <user>'s network in malad isn't awesome.
PID: 893773347026210242_185243426, Explanation: it's frustrating waiting for an hour on the tarmac for a gate to come open in snowy, windy Chicago.
PID: 708994813983596544, Explanation: the author hates the spring weather.
PID: 904624565145538560, Explanation: having a salivary gland biopsy on monday morning isn't a great way to start the new week.
PID: 697929589562146817, Explanation: it's not going to be scorching hot this w-end.
```

These outputs show that my model learned to generate explanations that reflect the sarcastic intent, often aligning with the target and context provided in T_knowledge.

Conclusion

Overall, Our MuSE model successfully integrates visual features from ViT and textual data via BART, with a custom Shared Fusion Mechanism that keeps things trainable and focused. The preprocessing we used made sure that all modalities were well-represented, and the evaluation metrics suggest the model improved over time. The generated explanations capture sarcasm reasonably well.