CS 7643 Project Report
Hateful Memes Classification
Team JYY

Yuan Yang, Yiran Cui, Jinghuang Chen
Georgia Institute of Technology
{yyang998,ycui324,jchen3198}@gatech.edu

Abstract

This report presents a study on hate speech detection in memes, aiming to combat harmful and discriminatory content on online platforms. The project focuses on developing a multimodal framework using state-of-the-art models like ERNIE-VIL and UNITER, combining vision and language processing techniques for comprehensive hate speech comprehension. Data augmentation with the Memotion dataset and data cleaning enhance the training data's quality and diversity. Despite time constraints limiting hyperparameter tuning and ensemble implementation, individual models outperform baseline results, demonstrating their potential for combating harmful content.

1. Introduction/Background/Motivation

The project aims to address the problem of classifying hateful memes arising from the detrimental consequences they can have on both individuals and society at large. Hateful memes frequently reinforce stereotypes, propagate discriminatory ideas, and encourage hostility or even violence directed toward groups based on characteristics such as race, religion, gender, or sexual orientation. The detection and classification of such memes are essential to enable social media platforms and content moderators to respond appropriately, thereby minimizing the harmful impact they can have. Taking actions such as removal or flagging for review can help mitigate the negative effects associated with the dissemination of these memes on social media platforms.

Presently, hateful meme classification relies on both human moderation and AI-driven automated tools utilizing machine learning and natural language processing. However, current practices face limitations, including the challenge of grasping context, analyzing visual content effectively, adapting to evolving language, addressing bias and fairness concerns, managing false positives and negatives, preserving user privacy, countering adversarial attacks, and handling multilingual complexities. In addition, most existing work on hate speech detection has largely relied on text-based features with comparatively little work in the vision or multimodal domain [4]. To address these limits, ongoing research focuses on improving AI models with multimodal techniques. Other focus areas include exploring new techniques for visual content analysis, mitigating biases, and developing multilingual classifiers. This paper mainly focuses on how we can deploy a multimodal framework to improve the detection of hateful memes. Works from the Hateful Memes Challenge are the main source of reference in this paper, we intend to propose a new framework based on the existing state-of-art multimodal and target to achieve a better AUROC than the hateful memes 2020 contest baseline.

Advancing the technique for hateful memes classification will have a profound impact. It will foster a safer online environment, minimizing harmful content and protecting users from discrimination. Social media platforms will benefit from more efficient content moderation, while users will become more mindful of responsible content sharing. Furthermore, it sets a positive precedent for the role of AI in addressing societal challenges and encourages further advancements for the greater good.

The main source of data is the Hateful Memes dataset, which contains images with embedded captions and separate text extracted by OCR in Jsonl files. There are 12,140 memes from the HM dataset. It's split into train, development, and test sets. Among the 8,500 training memes, 36% are hateful and 64% are non-hateful [4]. As part of the data augmentation, we also added 311 memes into the training set from the Memotion dataset. The summary of our dataset is in Table 1.

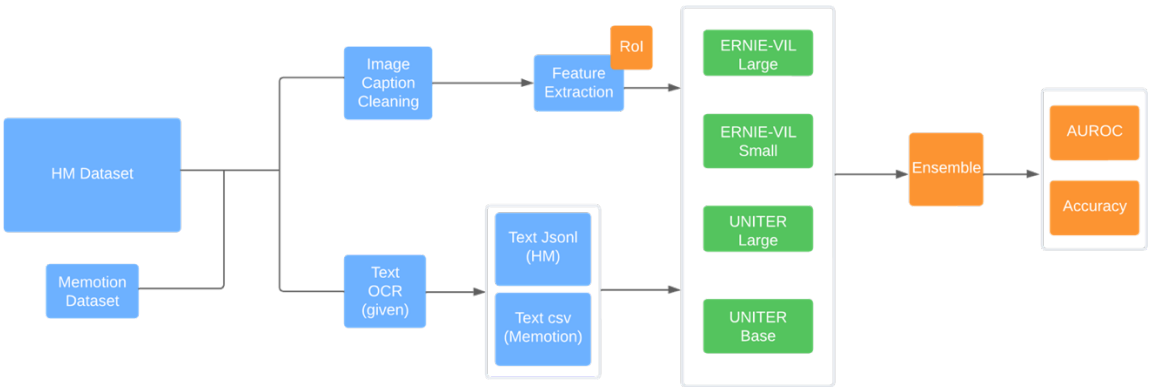

Figure 1. Team JYY Proposed Hate Speech Detection Pipeline. Refer to Section 2.1 for a detailed explanation of the approaches.

	Train	Train Added	Dev Seen
HM Dataset	8,500	411	500
	Test Seen	**Dev Unseen**	**Test Unseen**
	1,000	540	2,000

Table 1. Summary of Dataset. The Memotion dataset is added to the train set.

2. Approach

The main idea was to propose a new pipeline to detect multimodal hate speech based on the methodologies of the top 5 teams from the 2020 Hateful Memes competition. This included carefully analyzing their techniques, strategies, and algorithms to gain insights into the most successful approaches. Some of the best features from the top 3 teams' methods were merged into our pipeline.

Here is a summary of our work:

- The Memotion dataset was added to the original datasets to serve as an augmentation strategy for enhancing the model's performance with multimodal data (text and images).
- Data cleaning was then performed on the merged datasets to find and remove text from memes pictures to improve the quality of both feature extraction and model training.
- Applied feature extraction on caption-cleaned memes to further speed up the training process.
- Merged the text and label files in Jsonl and CVS formats from HM and Memotion.
- Fed the extracted vision features and merged language data into ERNIE and UNITER models.
- Ensembled the model output and analyzed it with AUROC and accuracy measurement.

The final output of our work is a full-size multimodal hate speech detection framework, incorporating several vision and language processing techniques (*e.g.*, cross-modal vision-languages transformers). The goal was to develop a powerful and adaptable hate speech detection system capable of identifying hateful content in diverse contexts.

2.1 Data pre-processing

2.1.1 Augmentation

One key effort to improve the hate detection result is to get more accurately labeled data or extract more information from current data. Ron Zhu from the 1st ranked team found that race and gender information can be extracted with Web Entity Detection and FairFace classifier [1]. Riza and Jewgeni from the 3rd ranked team found that the Memotion dataset has 14K human-labeled memes with sentiment (positive, negative, neutral) and type of emotion (sarcastic, funny, offensive, motivational) information. This is similar to HM memes and can be used as a supplement to the training dataset [3]. However, they found the label quality is not very good and their detection score was not improved in the initial test. They then hand-picked 328 memes with corrected labels and fed them into the training set.

The approach used by Ron is effective but rather complicated. Due to the time and computing resource limit, we decided to use the Memotion approach from Riza and Jewgeni. We solved the file name compatibility issue between Memotion memes and the ERNIE-VIL model by renaming the Memotion memes. Also, we manually excluded 17 pictures with very large resolution sizes which caused our text cleaning code to run out of GPU RAM. We also included in the training set 100 additional memes set that are in dev seen but not in dev unseen as found by Riza and Jewgeni. This increased our final training set from 8,500 to 8,911 memes.

Figure 2. Example of Image Text Cleaning (16849.png). From left to right: 1) original meme image; 2) text detected using EasyOCR with annotation of text displayed; 3) text mask area generated; 4) image cleaned using MMEDIT for inpainting.

2.1.2 Image Text Cleaning

For this part of the work, several packages and models are used to erase the text from meme images in the datasets. The work here can be divided into two major steps.

The first step is to detect the text in the image and generate a mask area of the text from annotation. The OCR (Optical character recognition) strategy was used here to detect the English text and provide text annotations, and EasyOCR [8] based on Pytorch was utilized here. After applying EasyOCR to an image, the annotations of text in this image are obtained. The annotations were then used to generate segmentation masks. The mask generation strategy can be referred to in the methods illustrated in this repo [12]. The basic idea is to get text boxes using previously obtained annotations and then create segmentation masks for the text. The text boxes were obtained by calculating the four corners of the text area based on the annotations (four coordinates among all the annotations - most bottom left and right, most top left and right).

The second major step is to use the original image and the generated segmentation masks to get a clean image through inpainting. DeepFillv2 [13] from MMEditing in Pytorch was used for inpainting. This whole process is illustrated in Figure 2 using an example meme image. The time consumed for image cleaning of the entire dataset (12140 + 311 images) is about 3hrs on a Titan RTX 24GB GPU. As shown in Figure 2, the cleaning performance is acceptable. After processing all the figures, it was found that the cleaning performance on the original hateful meme data outperforms the performance on the Memotion datasets added during augmentation. This observation might be caused by the poor image quality of the Memotion datasets compared to the original datasets.

2.1.3 Feature extraction

The cleaned images are processed through the Detectron2 framework, which is capable of detecting objects and regions of interest (RoIs) in the images. Instead of using the entire images, only relevant features from the detected objects and RoIs are extracted. To boost performance, we use different models for feature extraction. These models are pre-trained on VisualGenome, which is a large dataset containing images with detailed annotations of objects and regions. Figure 3 shows an example of predicted RoI's.

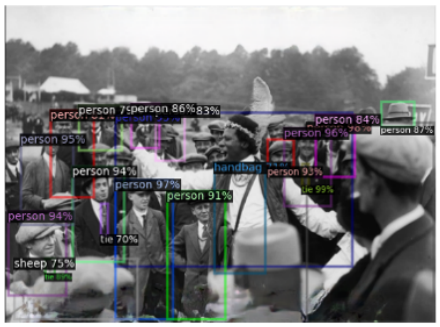

Figure 3: Predicted RoIs on a cleaned image from Hateful Memes Dataset

2.2 Modeling

2.2.1 ERNIE-VIL

ERNIE-VIL framework [6] was used by Niklas Muennighoff from the 2nd ranked team and we adopted it into our modeling approach and finetuned it with our HM and Memotion dataset.

ERNIE-VIL is a knowledge-enhanced framework designed to learn joint representations of vision and language. It's developed by PaddlePaddle based on VilBERT and the ERNIE transformer. Its primary function is to establish semantic connections between objects in visual scenes. With pre-training on two image-text aligned datasets, Conceptual Captions, and SBU Captions, it can achieve state-of-the-art performance in downstream tasks. The model is optimized with a binary cross-entropy loss on soft target scores.

In our HM dataset, there are no ground truth boxes provided, so a second set of features extracted with 10-100 boxes is used as pseudo-ground truth as suggested by Niklas. In this self-supervised approach, all the other configs are compared with box 10-100 config. Also, specific to our task, a VCR (Visual Common-Sense Reasoning) pre-trained weight is also added to increase diversity. [2] This is the same approach as Niklas used in the contest and we adopted it into our pipeline.

2.2.2 UNITER

UNITER (UNiversal Image-TExt Representation) [5] used in this task is a multimodal transformer model, which is pre-trained on image-text tasks with 4 image-language

datasets (COCO, Visual Genome, Conceptual Captions, and SBU Captions). This architecture contains several self-attention layers and uses binary cross-entropy loss for optimization. It applies self-attention mechanisms to the inputs of both image and text and uses an image encoder and a text encoder to encode visual and text features.

UNITER is widely used due to its outstanding performance over multiple V+L tasks, and its metrics on Hateful Meme Dataset were used as an official benchmark for the original challenge. Since several data preprocessing procedures were added to this project, we believe using UNITER to train the preprocess data could achieve better outcomes compared to the training results on original data.

This part of the work referenced work performed by Muennighoff [2]. The major modification we made is to use both UNITER-base and UNITER-large models for training to better serve the scope of this project and these two experiments can be ensembled together to reach even better metrics.

The major difference between these two models is the size: UNITER-base has 12 layers and UNITER-large has 24 layers. It is reported that even the UNITER-base model can provide considerable performance improvement compared to some of the other models [5].

2.3 Approach Summary

The approach was believed to be successful for several reasons:

Learning from Top Performers: Studying the methodologies of the top 5 teams provided valuable insights into the strategies and techniques that led to their success. By combining the best elements from the top 3 teams, it was expected that the new pipeline would inherit the strengths and effectiveness of their approaches.

Multimodal Data: The combination of HM and the Memotion dataset, containing both text and image data, provided a solid data foundation for the approach. Leveraging more multimodal data allowed the model to capture more nuanced patterns and context, potentially improving the overall performance of hate speech detection.

Data Cleaning: By performing data cleaning, the approach ensured that the model was trained on high-quality and consistent data. This step aimed to reduce noise and prevent the model from learning from irrelevant or misleading information.

Full-Size Multimodal Framework: Combining both vision and language processing techniques enabled the model to comprehend hate speech in a more comprehensive manner, potentially capturing more semantic information and improving detection accuracy.

In summary, we proposed a new hate detection pipeline based on the best features of the methodologies of the top 3 teams. By studying the published papers from the top 5 teams, we found there are many commonalities in their workflow. However, there are some unique ideas from the top 3 teams that made their final performance excel the others. We solved some compatibility issues among their approaches and merged these methods into our new pipeline.

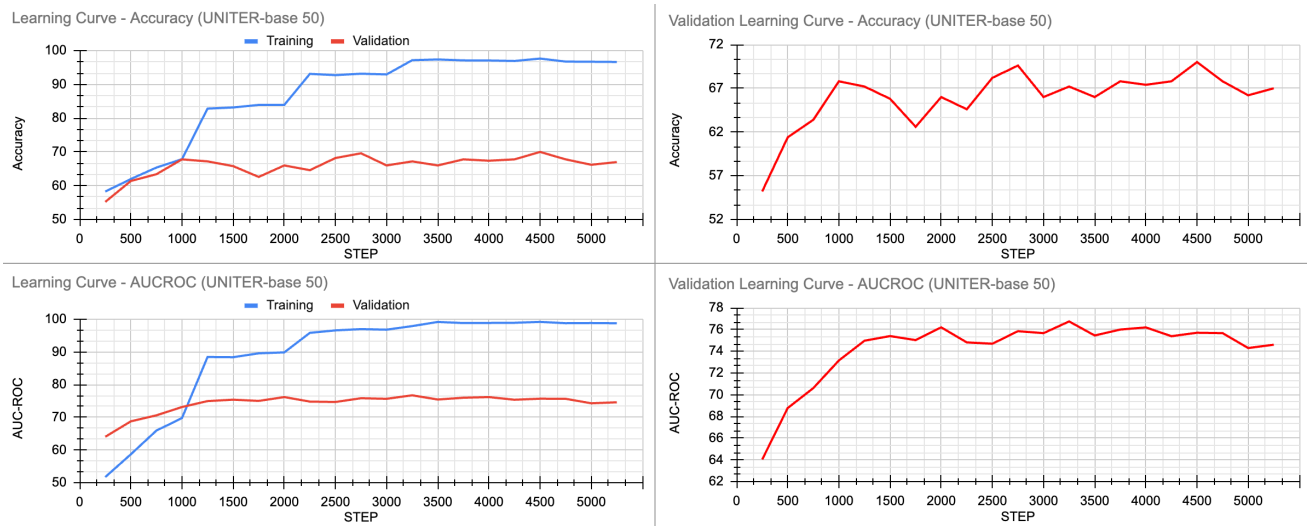

Figure 4. Learning curves of UNITER-base on 50-50 features. Both accuracy and AUC-ROC are in percentage (%). Upper left: Accuracy vs. Step for both Training and Validation; Lower left: AUC-ROC vs. Step for both Training and Validation; Upper right: Accuracy vs. Step for Validation itself; Upper left: AUC-ROC vs. Step for Validation itself.

3. Experiments and Results

3.1 UNITER Setting and Results

For the UNITER [14] training and testing part, the features extracted with 50-50 boxes and 36-36 boxes were used for both base and large models. Other hyper-parameters are the same across all the experiments: batch size is 8 and learning rate is 1e-5. For the testing, both test_seen and test_unseen datasets were tested. In total, 4 experiments were run: UNITER-large on features extracted with 50-50 and 36-36, UNITER-base on features extracted with 50-50 and 36-36. And then the results from experiments with 50-50 and 36-36 features were merged. Figure 4 illustrated the learning curves of UNITER-base on 50-50 features. The learning curves obtained from other experiments demonstrate similar trends (refer to the project Google Drive Folder for the detailed results [15]).

As shown in Figure 4, the upper left and lower left figures are the comparisons of the learning curves between training and validation. Both accuracy and AUC-ROC curves for training reached around 0.98, while validation curves converge around 0.68 for accuracy and 0.75 for AUC-ROC. This observation indicates that there is an overfitting effect during training. Since the UNITER model has many parameters, it is very likely that the model could tend to memorize features in the training data, causing the training metrics to be exceptionally high. Other than the overfitting, the validation accuracy and AUC-ROC is slightly higher than the baselines in the original paper (accuracy ~0.60, AUC-ROC ~0.70) [7].

The upper right and lower right figures are the validation curve by itself for accuracy and AUC-ROC. A better convergence pattern can be observed in these two figures.

Regarding the generalization performance, the testing results are also higher than the baselines (accuracy ~0.65, AUC-ROC ~0.73). For UNITER-base on 50-50 features, the accuracy is 0.6940 and the AUC-ROC is 0.7878 on the test-seen dataset, and the accuracy is 0.7270 and the AUC-ROC is 0.7854 on the test unseen dataset.

3.2 ERNIE-VIL Setting and Results

For the ERNIE-Vil [11] model, we use 500 warmup steps and 2000 training steps with a learning rate of 1e-5. Muennighoff from the 2nd ranked team used 5,000 steps in total, of which 2,500 were for validation and 2,500 for training and testing. Due to the training time of the ERNIE model, we chose to run the second half of the process only. The feature size of the model is 2048. The batch size is also 8, the same as UNITER. The weight decay ratio is 0.1.

The major difference between the two large and small versions of the ERNIE models is the config. For ERNIE-Villarge, the hidden size is 1024. There are 16 attention heads and 24 hidden layers. For ERNIE-Vil small, the hidden size is 768, There are 8 attention heads and 12 hidden layers. The main difference between ERNIE and UNITER settings is that UNITER runs 5 epochs and ERNIE only runs 1.

The AUROC result of ERNIE is shown in Table 2. It's better than the HM baseline and is comparable with the top 5 teams' results. Since there is only one epoch in ERNIE result, we only output the final averaged AUROC and accuracy results from all 5 configs. As mentioned above, validation output running is saved for future work.

3.3 Measure of Success

We had originally planned to measure success through a comparison of our results with the 1st prize winner in the Hateful Memes Challenge whose model achieves 0.845 AUROC. Despite facing time constraints, we managed to develop four individual models that surpassed the baseline performance. This encouraging result, as demonstrated in Table 2, indicates that our approach was effective.

Although we were not able to perform the ensemble step due to the time limitation, we are aware that this technique has the potential to further improve the overall performance of our models. Nonetheless, the fact that all four of our individual models outperformed the baseline shows that our methods and strategies were sound and effective.

Notably, the UNITER model stood out as the best-performing one during testing. This outcome suggests that the capabilities of the UNITER model make it suitable for generalization to other similar datasets and tasks, thereby increasing the potential for broader applications.

For the scope of this project and considering all the constraints, we think the goal has been achieved successfully. However, we also recognize that there is still plenty of room for future improvement. In subsequent iterations, we plan to explore other pre-trained models used by the top 3 teams and invest time and effort into implementing the ensemble step. By doing so, we aim to elevate our performance even further and gain more valuable insights into solving similar challenges in the future.

Table 2. Model Performance

Source	Model	Validation		Test	
		Acc.	**AUROC**	**Acc.**	**AUROC**
Hateful Memes Baseline	Human	-	-	84.7	82.65
	ViLBERT	62.2	71.13	62.3	70.45
	Visual BERT	62.1	70.6	63.2	71.33
	ViLBERT CC	61.4	70.07	61.1	70.03
	Visual BERT COCO	65.06	73.97	64.73	71.41
Our model	UNITER Base	67.80	75.21	72.68	78.37
	UNITER Large	69.20	77.64	72.95	78.57
	ERNIE-ViL Small	-	-	76.15	71.49
	ERNIE-ViL Large	-	-	77.85	76.41
	Ensemble	-	-	-	-

4. Challenges and Lessons Learned

The project faced several anticipated and unexpected challenges during its execution, including the computation resource and environment limit, incomplete information from the author, and compatibilities among different approaches.

GCP Resource Limit: Initially, one of the main anticipated problems was the limited availability of GPUs in the Google Cloud Platform (GCP) for training the deep learning models. As deep learning tasks often require substantial computational power, the scarcity of GPUs could potentially hinder the progress of the project. Our group was not able to always obtain an available GPU on the GCP, therefore we decided to use Colab as our main platform for training the model.

Colab Environment: In the Colab and Google Drive environment, we faced new challenges such as Google Drive's delay in syncing large datasets, Colab's session termination in 30 minutes, and automatic disconnection after 4 hrs. We addressed these issues by optimizing data processing workflow, upgrading to pro subscription, task scheduling, and careful Colab environment setup.

Memotion Dataset: Unlike the images from the HM dataset having uniform resolution and image format, Memotion Dataset contains raw images from the internet with varying resolutions and formats. This caused difficulty for our code to detect all the images. Some images had such a large resolution that it caused the text-cleaning code to run out of the 24G GPU RAM. Memotion also has a file naming convention starting with "Image_" which caused an error in the ERNIE model which requires a log10 treatment of the picture name. We ended with manually deleting oversized memes, unifying the image format, and renaming the selected memes.

Hyperparameter Tuning: In the area of experimentation and tuning of hyperparameters, we also anticipated that the iterative nature of this process meant that the first attempt might not have yielded satisfactory results, requiring multiple iterations to achieve the desired performance. However, due to time constraints and our model's size, we could not conduct a hyperparameter sensitivity analysis. Although Niklas mentioned hyperparameter tuning will not affect the model result too much [2], we were still curious and wanted to try it out.

Lack of critical information: Some authors only mentioned they used Python 3 in their code. However, in their documentation and code, they didn't specify the Python 3 version, and most packages used have a certain level of dependency on the other and the Python version. We tried different versions of Python from 3.6 to 3.10 and finally used different versions in different modules in order to go through the whole pipeline.

Interpret model output: In the fine-tuning process with the pre-trained Ernie and Uniter model, we found it challenging to read the extremely long output logs, interpret the results, correlate them with our intuition, and debug the errors. However, this is also an important learning experience.

Lesson Learned: First, the combination of Colab and google drive is not an ideal computing environment for full-size multimodal deep-learning training. It's good for small model training and tends to discourage long-time running even with pro and pro+ subscriptions. Second, put enough resources and attention into data preprocessing. It's critical and don't underestimate the workload. Third, keeping good file structure and documentation helps with efficiency and reduces the risk of error. It also makes it easy for others to follow the workflow and duplicate the result. Finally, read the output from the models carefully. Catch anything that does not match intuition and dig deep into it. Don't run the model blindly.

5. Work Division

In Table 3, a summary of contributions is provided.

Student Name	Contributed Aspects	Details
Yuan Yang	Data Pre-processing, Implementation, Result Visualization and Analysis	Performed image cleaning by removing text from images in both original and newly added Memotion datasets using OCR and inpainting techniques; Built UNITER base and large modeling pipeline; Analyzed the results and metrics obtained from UNITER experiments and generated learning curves.
Yiran Cui	Data Pre-processing, Implementation, and Analysis	Built feature extraction and Ernie-vil base and large modeling pipeline, preprocessed Memotion dataset and merged into HM training data set
Jinghuang Chen	Analysis and Result Post-processing	Created RoIs image and compared/validated detectron2 model with pre-processed data, compared and analyzed model results using metrics including AUCROC and accuracy.

Table3. Work Division

6. References

[1] Ron Zhu, Enhance Multimodal Transformer With External Label And In-Domain Pretrain: Hateful Meme Challenge Winning Solution, arXiv:2012.08290v1
[2] Niklas Muennighoff, Vilio: State-of-the-art Visio-Linguistic Models applied to Hateful Memes, arXiv:2012.07788v1
[3] Riza Velioglu, Jewgeni Rose, Detecting Hate Speech in Memes Using Multimodal Deep Learning Approaches: Prize-winning solution to Hateful Memes Challenge, arXiv:2012.12975v1
[4] Phillip et al., A Multimodal Framework for the Detection of Hateful Memes, PMLR 2021
[5] Chen et al., UNITER: UNiversal Image-TExt Representation Learning, arXiv:1909.11740
[6] Yu et al., ERNIE-ViL: Knowledge Enhanced Vision-Language Representations through Scene Graphs, Proceedings of the AAAI Conference on Artificial Intelligence 2021
[7] Kiela, Douwe, et al., The hateful memes challenge: Detecting hate speech in multimodal memes, Advances in neural information processing systems 2020
[8] EasyOCR based on Pytorch for detecting text in images: https://github.com/JaidedAI/EasyOCR
[9] Vili: https://github.com/Muennighoff/vilio.git
[10] Bottom-up Attention with Detectron2: https://github.com/airsplay/py-bottom-up-attention.git
[11] Ernie-vil: https://github.com/PaddlePaddle/ERNIE/tree/repro/ernie-vil
[12] Enhance Multimodal Transformer With External Label And In-Domain Pretrain: https://github.com/HimariO/HatefulMemesChallenge
[13] DeepFillv2 from MMEditing based on Pytorch: https://github.com/HimariO/mmediting-meme/blob/master/demo/inpainting_demo.py
[14] UNITER: https://github.com/ChenRocks/UNITER
[15] Project Google Drive: https://drive.google.com/drive/folders/14zuClwq09CT1Dnea0D37E5iedrvpsbSS?usp=sharing