CS7643 Project Report: Detecting Hateful Memes

Aditya Bhushan (abhushan30@gatech.edu) Xiaojun Bao (xbao41@gatech.edu)
Alessandro Bersia (a.bersia@gatech.edu) Yangwei Yang (yyang886@gatech.edu)
Georgia Institute of Technology

Abstract

In today's world where everything is moving online, hate speech is becoming an area of rising concern. Unfortunately, it is really easy to express extreme emotions in the form of an ever increasing social media trend of creating and sharing "memes". Memes use both text and image to convey strong messages which could hurt sentiments of people and promote discrimination. It has become very important to come up with innovative ways to identify and tackle this problem. Our paper focuses on exploring both unimodal and multimodal frameworks in deep learning research as a means to allow for easy identification of hateful memes. We looked into ways of improving the overall methodology beyond just fine tuning of vision and language models. Under unimodal approach, we experimented with models namely Roberta, Albert and DistilBert for language and ResNet, ViT, Wide Resnet and ResNext for vision; in contrast to multimodal approach, where we experimented with VL-BERT, LXMERT, UNITER and OSCAR models. In the end, we showcase the best approach for our "winning model" with accuracy of 65.80% and AUROC score of 72.07% and discuss the reasons for our observations and future scope of work.

1. Introduction

Detecting hate speech using deep learning is of great interest to social media platforms with the massive scale of the internet content. Attacks are not only in text form but may also be multimodal, in many cases the content from soly the image or the text can appear to be normal, but become hateful when combined together. This multimodal problem requires the model to understand image and language contents holistically. In order to develop models to tackle multimodal hateful content, Facebook created a Hateful Meme dataset, and crowdsourced solutions via a DrivenData competition [1]. In this project, we aim to develop models to perform binary classification on the provided dataset, with the goal to reach or beat the benchmark performance, and to analyze the success and failure of our models.

As of today, social media posts still rely heavily on human eyes to catch inappropriate contents. Facebook hires content moderators who have to go through lengthy training, earns below-median income, and bears high risk of mental health from their job [2]. Current SOTA models have classification accuracy of around 75%, still significantly lower than human accuracy of 84.7% on the same dataset. If one can develop a model which can reach or beat human performance, it will liberate human labors from the negative impact of the task. It will also make the content moderation process much faster, as model inference takes no time, whereas human annotators spent an average time of 27 minutes per final meme in the dataset [1].

The Facebook Hateful Meme dataset used in this project contains 10k memes, constructed from Getty Images with text overlaid on the image. The texts associated with each image are also provided separately. Each meme has a binary label indicating whether the meme is hateful or not. The images come with different dimensions, and can be a collage of 2-3 images. The text positions on the image are variable, which can be at the top, middle or bottom of the image. Text length and font size are variable as well. In this project, 8500 memes are used as training data, 500 memes are used as validation data during training, and a holdout set of 1000 memes are used for testing the final performance. Classes are slightly unbalanced in training data, with 35.5% positive labels. Validation data and test data has balanced classes with 49% positive labels.

2. Approach

We took two general approaches based on the benchmark paper. One is unimodal pretraining approach: each input modality goes through separate transformation (image transform includes resizing, normalization, or augmentation transform detailed later; text transform includes tokenization), then enters CNN or transformer models separately (pretrained, fine-tuned, or trained from scratch). After that, both features are concatenated and enter an MLP head for the final classification task. The second is a multimodal pre-training approach: each input modality goes through transformation, and enters the same multimodal transformer model, followed by an MLP head for classification (Figure 1). Based on the benchmark paper, the best models of each approach perform on par with each other, indicating that both approaches should have an opportunity for good performance.

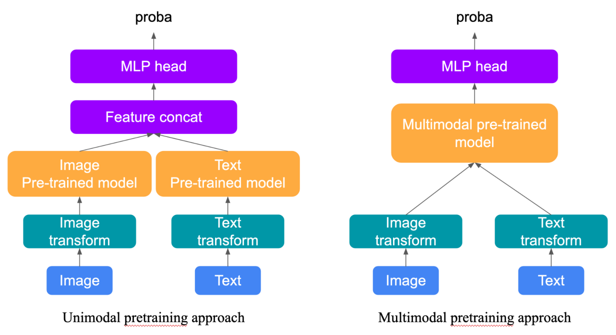


Figure 1. Unimodal vs. multimodal pretraining approach

2.1 Unimodal Pretraining Approach

For unimodal pretraining approach, the pipeline structure is set up based on the template provided by Driven Data [3].

2.1.1 Pretrained vision and language model with or without fine tuning. We started with a model closely resembling the concat BERT model in the benchmark paper. We used pretrained ResNet152 [4] model for image encoding, and pretrained RoBERTa [5] model for text encoding. RoBERTa is a retraining of BERT with 10x more data and dynamic masking during training. We tried freezing the RoBERTa weights vs. fine tuning RoBERTa, to test the semantic encoding capacity of RoBERTa (we always fine tune the visual model). We also experimented using the first vs. last token's last hidden state of the text encoding before fusing with visual features.

Next, we swapped the visual model with pretrained Vision Transformer (ViT) [6], a pure transformer architecture (as opposed to CNN) applied directly to sequence of image patches. In theory this should achieve the same attention requirement for interpreting image content in various locations.

Lastly, we experimented with the structure of the MLP classification layer. In particular, we tried MLP of 1 and 2 hidden layers, tweaking the number of cells in each layer, as well as adding a residual block to the MLP layer.

Among the models in this section, we used AdamW [7] optimizer and early stopping with 3 epoch patience to prevent overfitting.

2.1.2 Pretrained language model with improved vision model architectures. In this section, we will look at exploring the visual and language modalities. Here, we used different pre-trained language models whereas the vision models are built and trained from scratch. The advanced visual models deployed are 'Wide Resnet' and 'ResNext' built on top of resnet base architecture whereas language models used are Albert model and DistilBert which belong to the BERT family, but more efficient in their own respective ways.

Exploring Textual Modality using BERT & friends: We chose a pre-trained Albert Model - a lite Bert version by Google Research because of certain advantages [8]. Albert-xxlarge architecture differs such that the number of hidden layers (4096) are higher and embedding size of 128 is much lower than Bert-base (768), this reduces memory consumption and enables a much faster processing [22]. Another is 'Parameter Sharing' wherein the same parameters are used by all encoder layers in Albert Model as opposed to unique parameters reqd. for each layer in Bert; reducing the number of parameters required by Albert. Another model that we studied in the lectures is the DistilBert Model wherein the biggest advantage is 'knowledge distillation' along with a 40% reduction in size, comparatively [9].

Exploring Visual Modality using advanced resnet architectures: We implemented the ResNext or the 'Aggregated Residual Network' which works on the idea that the existing resnet architectures can be improved by repeating a building block that aggregates a set of transformations with the same topology [10]. It introduces a new dimension called 'cardinality' which is a grouping of convolutions that represent the size of the set of transformations. As depicted in the Appendix 1.1, the contrast between the architectures of Resnet50 and ResNext50 with residual block of 32x4d is provided. We implemented two versions from scratch – ResNext 50 (32x4d) and ResNext 101 (32x8d) model where 50/101 denotes number of layers; 4d/8d denotes the bottleneck width and 32 denotes cardinality. The model performance comparison is shown in the Results Table in the end.

A single Bottleneck layer is a variation of the residual block with 1x1 convolutions that helps in representing inputs with reduced dimensionality. A typical bottleneck (in Appendix 1.2) is a stack of 1x1, 3x3, 1x1 layers respectively. This design improves efficiency as the 1x1 layers reduce the depth from 256 to 64, enabling the 3x3 layer to operate on a less dense feature vector and the final 1x1 layer restores to 256d. Using bottlenecks, we reduce the number of parameters and matrix calculations. This idea shared in the paper is to make residual blocks "thin" allowing us to have fewer parameters and increase the depth of the network without compromising on computing power [4]. This paper shows empirical evidence that such residual networks are easier to optimize and gain accuracy with increasing depth.

Another advanced architecture that makes use of the bottleneck layers that we have used in our experiment is the Wide Resnet 50-2, 101-2-bottleneck (WRN) [11]. WRN-L-k denotes the widening factor as k and number of convolution layers as L.

We have 3 types of blocks as shown in Figure 2:

Basic: We have 2 consecutive 3x3 convolutions with batch normalization and activation function preceding convolution

Bottleneck: We have a 3x3 convolution sandwiched between 2 consecutive 1x1 convolution for purpose of reducing dimensionality: conv1x1-conv3x3-conv1x1

Wide-dropout: We have a dropout layer introduced between two 3x3 layers for the purpose of regularization: conv3x3-dropout-conv3x3

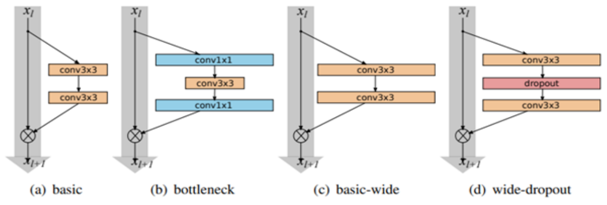


Figure 2. Advanced vision architectures

As we can see that a WRN is similar to Resnet except for the bottleneck number of channels which is twice as large in every block. The number of channels in outer 1x1 convolutions is the same, e.g. the last block in a ResNet-50 has 2048-512-2048 channels, and in Wide ResNet-50-2 we are having 2048-1024-2048 channels. In our experiments, we used WRN50-2 and WRN101-2 indicating that the widening is "twice" as large as that for the corresponding Resnet model.

Early Fusion Logic: Once we extracted the image and the text features, we prepared our fusion layer. We first deployed a batch normalization layer to normalize these features individually and then concatenated the two feature sets. The fused output is passed through a fully connected layer, followed by batch normalization and then through leaky_relu activation; we used dropout of 0.06 before being fed to the linear MLP classifier layer with output as the number of classes i.e. hateful(1) or peaceful(0) (Appendix 2).

2.2 Multimodal Pre Training Approach

At present, the multimodal model primarily adopts the transformer structure, and its training data is based on the image-text pair of the image annotation data set, where the text is the natural language description of the corresponding image. The main difference across multi-models lies in two areas: data stream processing methods and pre-training tasks.

Single-stream architecture such as OSCAR and UNITER treats image and text data in the same stream and data are directly fused and passed into the same transformer together. On the other hand, dual-stream architecture such as LXMERT treats image and text as two separated data streams which are passed into two different transformers, and they are fused through a cross interactive module. For pre-training tasks, the models can be trained through tasks such as token prediction, sentence order prediction and logical relationship prediction.

As many combinations are possible, we picked VisualBERT, OSCAR, UNITER and LXMERT models to support our explorations on meme data.

Anderson et al. [12] proposed a hard-attention architecture to extract salient regions from images, by implementing a "bottom up attention" approach on Faster R-CNN. The bottom up attention associates a feature vector to specific image regions. We found this approach particularly interesting for the hateful meme project, because it allows us to extract an arbitrary number of features from the images. The implementation was possible by leveraging the pyTorch library Decatron [13].

The best tuned model setup for meme dataset is pre-trained Resnet with depth 101 and 1600 classes for ROI heads, as well as 400 attributes for box heads. Processed images with samples are shown below with 36 boxes in the middle and 72 boxes on the right (Figure 3). The box features are packed with original OCR text and saved in tsv file format for downstream model consumption.

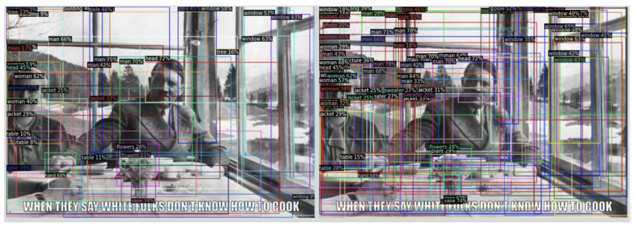

WHEN THEY SAY WHITE FOLKS DON'T KNOW HOW TO COOK
WHEN THEY SAY WHITE FOLKS DON'T KNOW HOW TO COOK

Figure 3. Sample image with 36 boxes (left) vs. 72 boxes (right)

We used the bottom up attention approach to extract visual features for all multi-models. The choice of the number of features, hence the number of boxes to extract from the images, is a hyperparameter to be tuned. In general, we have found out that for the hateful meme dataset, no further performance gains can be achieved once the number of boxes is above 50.

For multi-model implementation we leveraged the Vilio framework from Muennighoff [14] which is like Huggingface transformers model zoo that provides a universal interface for state-of-art visual-linguistic models. We also learnt from hateful meme contest experiences that additional data sources such as region of interest can help improve model performance. Therefore, features extractor of py-bottom-up-attention is introduced to generate bounding boxes as our model input.

LXMERT architecture is shown in Appendix 3. ROI and POS features from image as well as word embedding and position embedding features are summed together. Then layers are normalized and enter the two transformer stages respectively. Between the two transformers there exists a cross module whose inputs are the encoder's previous hidden layer plus all input information from the other image or text source side.

For model setup, we selected visual layer dimension 2048 for image features along with dimension 4 is selected for position features, along with 1600 object classes and

400 attributes dimension to be consistent with the output from the previous object detector. The encoder layer employed pre-trained Bert with 9 layers for language features and 5 layers for both image features and cross modality encoders that contain image and text together. This number of layers is enough to capture data complexity as evidenced by the learning curve mentioned later. Inspired by the GPT model from Open AI, GELU is applied for activation function because it provides better defined gradients in the negative regime to stop neurons dying which leads to more efficient training and better performance.

The UNITER model developed by Microsoft is easy to understand from Appendix 4. It only uses one transformer with input from a simple concatenation of ROI sequence and text token sequence. ROI is still feature extraction plus location information, but the location information also includes the height, width and area that forms a 7-dimension position vector. The processing of the token follows the normal BERT setting. For the same Transformer to handle two modalities, the representations of the two modalities need to be projected into the same representation space. Therefore, a linear layer is added on top of the original ROI and Token representations.The model setup is the same as LXMERT with visual features dimension 2048, 1600 object classes and 400 attributes, including GELU activation layer.

VisualBERT [15] model also appears to be a proper candidate for this particular task. Li et al. demonstrated how VisualBERT was able to achieve the state-of-the-art on vision and languages task by combining BERT [16], a transformer based language model, with an object detection model such as Faster-RCNN [17]. The objective of VisualBERT is to leverage self-attention mechanisms to discover alignments between images and text. Just like BERT, VisualBERT can be fine-tuned and the authors showed that it was able to achieve state-of-the-art performances on multimodal tasks (Appendix 5).

The OSCAR model structure still has only one Transformer. When compared with UNITER, the main contribution of OSCAR is that its input is no longer an image-text pair, but a text-label-image triplet where the label sequence is the classification label sequence corresponding to the ROI sequence. This is shown as the object tags section from Appendix 6.

We believe that introducing additional information that connects the two modalities to the input can provide a more direct inductive bias for the model. However, the ROI tag is generated by an upstream object detection model which still contains noise. So even though the OSCAR model in our opinion is the most elegant multi-model that fully utilizes the information, it does not solve the noise problem.

3. Experiments and Results

Same as the Driven Data competition, we focus our metrics on AUROC and accuracy on test data. We consider success to be reaching the performance as the benchmark, with the stretch goal of beating the benchmark.

3.1 Experiments with unimodal pretraining

3.1.1 Pretrained vision and language model with or without fine tuning. For the ResNet+RoBERTa model, we first experimented with freezing RoBERTa weights vs. fine-tuning it. When freezing RoBERTa weights, the model performed at chance level (test acc 52.9%, AUROC 51.87%), indicating that the text embeddings encoded by RoBERTa did not contain hateful sentiment information. While fine-tuning RoBERTa, the model was able to reach 60.3% accuracy after 4 epochs (Table 1). This indicates that RoBERTa is powerful at transfer learning, with a little fine tuning on a target task it was able to reach the same performance as Concat BERT in the benchmark. Keep training the fine-tuning model hurt the performance, with 10 epoch accuracy resulting in 53.8% and 20 epoch accuracy resulting in 51.6%. The learning curve of validation loss appears to be U-shape, indicating overfitting of the model.

We then experimented with the output of RoBERTa encoder. The last hidden state contains embedding of each token in the sentence, at this stage each token embedding should contain a representation of the sentence. We tried using the last hidden state of first token vs. the last token as input for feature fusion, and found that the resulting test accuracy are the same (60.3%), and last token has slightly higher AUROC compared to first token (64.04% vs. 62.04%).

Swapping in ViT for ResNet showed similar learning speed, it was able to reach the same AUROC (64.86%) than ResNet, however the accuracy was lower. From our experiment visual transformer and CNN structure seem to perform similarly in this sentiment classification task.

Lastly, we experimented with different MLP layer for final classification. We tried single layer with 1024 cells, double layer, and single layer with residual connection. Doubling the MLP layer did not elicit higher performance compared to single layer MLP (around 64% AUROC), but adding a residual connection resulted in a higher AUROC of 66.17%.

3.1.2 Pretrained language model with improved vision model architectures.

Improved Architectures. As detailed in the approach section, we experimented with 50 layers and 101 layers for both ResNext and Wide Resent architectures and found that deeper(101 layers) and wider(8d) design had better performance. Albert Model with ResNext 101(32x8d) and

DistilBERT with Wide Resnet101 performed the best as in the Results Table. The only downside was that it required higher number of epochs and much longer training time.

Vanishing and Exploding Gradients. We used rectified linear activation to avoid vanishing gradient issue but we had to avoid exploding gradients as well which can make our network unable to learn from training data, so we clipped the norm of the gradients to 1.0 (max) for text model parameters. This changes (rescales) the error derivative before propagating backwards through the network. We used pytorch's clip_grad_torch function and immediately observed improvement in overall loss.

Image Augmentation. At the transforming image stage, we experimented with Flipping (horizontal and vertical) and Cropping (random and centre) during training. In terms of results, we did not see much difference in performance and that may be attributed to the limited amount of data we had for training and validation.

Optimizers. In the lectures, we studied different optimizers which we put to the test. We picked pytorch's AdamW, SGD and SGD+momentum with CE loss function. We chose AdamW optimizer over simple Adam optimizer because AdamW decouples the weight decay from the optimization step. If we change the learning rate, it does not change the optimal weight decay and vice-versa. In terms of results, at a lower number of epochs, AdamW optimizer (lr=2e-5, eps=1e-8) performed better than SGD. SGD alone seemed to be unstable, so we combined it with momentum (0.9) and observed that with longer training time and higher number of epochs(20+), SGD+momentum outperformed AdamW, so we chose that. Further, using the learning rate StepLR scheduler with step size(4) and gamma(0.1) also helped.

3.2 Experiments with multimodal pretraining models

Initial Hyperparameters. Log SoftMax with cross entropy is calculated as the classifier's loss function for all multi-models. A smaller value of batch size of 8 is selected due to memory restriction and the calculated loss is scaled with this size as well. Dropout rate 0.1 is picked for regularization and models were trained consistently for 20 epochs with the best value identified in the middle before over-fitting happens. The weights from the epoch with the best validation score are taken and used for prediction on the test set. Learn rate values range from 1e-5 to 1e-3 are also tried for training and 1e-5 learning rate is optimal with gradual performance improvement across epochs. Larger learning rate caused performance score volatile with no improvement trend can be seen.

Detected object features. For model inputs both 36 and 72 bounding boxes with number of position feature 6 were tested and accuracy scores are identical for the same model. Adding more ROI features is not able to boost performance. We think that with 36 boxes the object detector has already captured full information from the images in the meme data set. This can be evidenced by the sample image shown before with 72 boxes. Compared with the sample image with 36 boxes, the 36 extra recognized tags are just repeated information, and they don't add much value.

Model capacity. All multi-models can reach a 99% accuracy score for the train set after 20 epochs. It indicates that with deep enough layers model capacity is sufficient to learn the classification task for meme dataset. In addition, the regularization did work to prevent the vanishing gradient problem caused by deep networks. However, the overfitting poses the issue that more input data is needed to help the model generalize better.

Less is more. VL-BERT, UNITER and OSCAR models achieve 69% level AUC-ROC score versus 65% level from LXMART, which indicates single stream architecture delivers superior performance than dual stream architecture. We believe that it's because of two reasons. First the only one transformer from single stream to process input data made model structure simple and according to Occam's razor this helps the model generalize better. Second, the only one transformer from single stream architecture already has its self-attention mechanism to learn the relationship between image and text and this is equivalent to justify the cross modality from dual stream architecture.

More is not always better. The 69% scores are close among VL-BERT, UNITER and OSCAR models, even though OSCAR includes extra entity label information, the benefit is not reflected in its score, demonstrating that common label information may not be sufficient to help the model solve the classification task for hateful memes. Instead, other labels such as race or gender may be able to help but unfortunately it is not included in the training dataset.

More details about our best model VisualBert. Li et al. demonstrates that VisualBERT was able to achieve state-of-the-art performances when using fine-tuning. For this implementation, we leverage the pretrained weights for two language models: (1) BERT trained on 16 GB of English text corpora; (2) RoBERTa, pretrained on over 160GB of English text corpora.

The masked language version we used the "transformer" library made available by HuggingFace [18] Given the superiority of the RoBERTa model, we expected to achieve slightly better performances when loading the pre-trained configuration of RoBERTa . The VisualBERT model was first implemented with "bert-base-uncased" pre-trained configuration (available on HuggingFace.co), however "roberta-base" pre-trained configuration achieved slightly better accuracy, confirming the initial hypothesis. Therefore, for the VisualBERT implementation, the pretrained weights were from RoBERTa.

Based on the experiments, it appears that the greatest impact on performances is due to the model architecture rather than the hyperparameter tuning.

However, some marginal increment to the AUC and accuracy was possible by tuning the learning rate at 0.00005. Multisample dropout [19] was implemented with probability of 0.4. This stabilized the learning curve and prevented the model from rapidly overfitting the data. A small batch size of 4 was chosen to offer some additional regularization effect [20].

The training metrics show that the models tend to overfit the data after 10 epochs, hence the maximum number of epochs set for the VisualBERT model was 10.

3.3 Results and Classified Images

Our results are captured in Table 1: Model Performance Comparisons. The snapshots in Appendix 7 are the sample images classified as hateful or peaceful with classification threshold set to probability 0.5. It also shows the corresponding sample confusion matrices for one of our training experiments.

4. Discussion

Where our unimodal models fail is the fact that they can not gather adequate information to perform better classification whereas more information can be aggregated by the multimodal models from the extracted features by virtue of their design. This is shown by our experiments wherein even after improving the architecture in case of unimodal, the performance is not at par with the latter, given the same set of training data. Furthermore, unlike unimodels, multimodels encode image and text features from different modalities into a common hidden space and then map the hidden representations of the inputs, much more effectively.

In recent years, deep multimodal learning has proven to be an effective strategy for deep learning architectures to leverage data obtained from different sources. At an intuitive level, the multimodal approach can achieve higher performances because it can leverage pre-trained transformer models on both images and language, whereas the unimodal approach is not optimized for the pairing of modalities. More specifically, deep multimodal learning can provide several architectural advantages [21], including implicit dimensionality reduction and little preprocessing of input data required. There is consensus in the literature that deep multimodal learning achieves better performances for a wide range of problems, and we were able to quantify the improvements in our experiments for the hateful meme project.

It is worth mentioning that for the proposed multimodal approach, traditional data augmentation techniques, such as rotation and cropping, did not produce quantifiable results. This is probably due to the implementation of a bottom-up attention mechanism on the images, that is not only effective in extracting relevant features from the images, but it also assists downstream models pay attention to interested regions, thus reducing the need for additional training samples. However, it is likely that the performances could further improve if more memes were collected and added to the training dataset. It would be interesting to further explore some advanced data augmentation techniques with Generative Adversarial Networks, albeit this was not possible in the time frame of this project.

Source	Type	Model	Val Acc.	Val AUROC	Test Acc.	Test AUROC
Benchmark		Human			84.7	
Benchmark	Unimodal Pretraining	MMBT-Region	64.75	72.62	67.66	73.82
Benchmark		Visual BERT	65.01	74.14	66.67	74.42
Our code		Resnet152+RoBERTa	57.2	63.74	60.3	64.03
Our code		ViT+RoBERTa	56	63.57	56.4	66.17
Our code		ALBERT+ResNext101-32x8d	58.5	60.11	59	61.35
Our code		ALBERT+ResNext50-32x4d	55.08	59.28	57.08	60.7
Our code		DistilBERT+WRN-101-2-bottleneck	60.11	63	59.33	62.55
Our code		DistilBERT+WRN-50-2-bottleneck	57	57.89	58.8	60.32
Benchmark	Multimodal Pretraining	ViLBERT CC	66.1	73.02	65.9	74.52
Benchmark		Visual BERT COCO	65.93	74.14	69.47	75.44
Our code		LXMERT	63.4	69.62	62.4	65.79
Our code		UNITER	65.93	77.93	65.2	69.03
Our code		OSCAR	61.6	68.92	61.2	68.08
Our code (best)		VL-BERT	65.04	71.05	65.8	72.07

Table 1. Model Performance Comparisons

5. Work Division

Team member	Contributed Aspects	Details
Aditya Bhushan	Unimodal methodology, implementation and experiments; abstract; discussion & results	Started with understanding and implementing a bunch of pre-trained models for exploring text modalities Distilbert, Albert, Deberta,mobileBert, etc. Based on research papers, implemented bottleneck architecture with wide resnet50,101 and resNext32,50,101 models for image features. Implemented early fusion logic with improved flowchart; addressed vanishing/exploding gradient problem,image augmentation and optimizer experiments apart from HP tuning with learning rate scheduler. Prepared appendix with confusion matrix,sample classified images.
Alessandro Bersia	Multimodal implementation, experiments, hyperparameter tuning and discussion; Literature research, particularly on multimodal approaches	Set up a VM instance on gcp to work with multimodal models on gpu. After setting up the benchmark with baseline, I have focused on VisualBERT improvements with a bottom-up attention approach and pre-trained language models. Tuned hyperparameters. Wrote multimodal experiments, findings from literature and conclusions.

Xiaojun Bao	Unimodal implementation and experiments; report writing and formatting	Set up unimodal training workflow in colab;
		Experimented with ResNet+RoBERTa, ViT+RoBERTa frameworks;
		Experimented with MLP classification head structure;
		Tuned hyperparameters (learning rate, dropout rate, epochs, early stopping patience)

Yangwei Yang	Data transformation, object detection and multi-model implementation	Customized vilio framework to fit for meme dataset Tuned object detector with bounding boxes and visualization. Generated tsv files for downstream model consumption. Implemented and tuned LXMERT, UNITER and OSCAR Wrote report for multi-models sections

Table 2. Contributions of team members.

6. Reference

[1] Kiela, Douwe, et al. "The hateful memes challenge: Detecting hate speech in multimodal memes." arXiv preprint arXiv:2005.04790 (2020).

[2] Newton, C. "The secret lives of Facebook moderators in America". The Verge (2019)

[3] https://www.drivendata.co/blog/hateful-memes-benchmark/

[4] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[5] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).

[6] Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

[7] Loshchilov, Ilya, and Frank Hutter. "Decoupled weight decay regularization." arXiv preprint arXiv:1711.05101 (2017).

[8] Lan, Zhenzhong, et al. "Albert: A lite bert for self-supervised learning of language representations." *arXiv preprint arXiv:1909.11942* (2019).

[9] Sanh, Victor, et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." *arXiv preprint arXiv:1910.01108* (2019).

[10] Xie, Saining, et al. "Aggregated residual transformations for deep neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[11] Zagoruyko, Sergey, and Nikos Komodakis. "Wide residual networks." *arXiv preprint arXiv:1605.07146* (2016).

[12] Anderson, Peter, et al. "Bottom-up and top-down attention for image captioning and visual question answering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[13] https://github.com/airsplay/py-bottom-up-attention

[14] https://github.com/Muennighoff/vilio

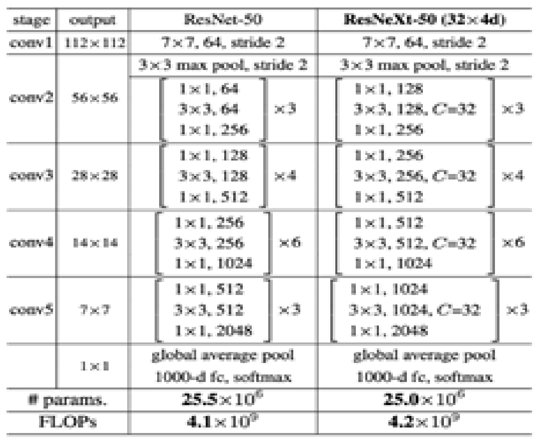
[15] Li, Liunian Harold, et al. "Visualbert: A simple and performant baseline for vision and language." arXiv preprint arXiv:1908.03557 (2019).

[16] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint

arXiv:1810.04805 (2018).
[17] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems 28 (2015): 91-99.
[18] https://huggingface.co/transformers/
[19] Inoue, Hiroshi. "Multi-sample dropout for accelerated training and better generalization." arXiv preprint arXiv:1905.09788 (2019).
[20] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. PMLR, 2015.
[21] Ramachandram, Dhanesh, and Graham W. Taylor. "Deep multimodal learning: A survey on recent advances and trends." IEEE signal processing magazine 34.6 (2017): 96-108.
[22] https://github.com/google-research/albert

7. Appendix

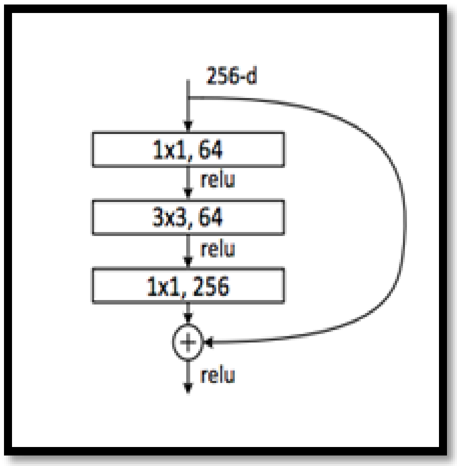
A1.1 ResNext Vs ResNet



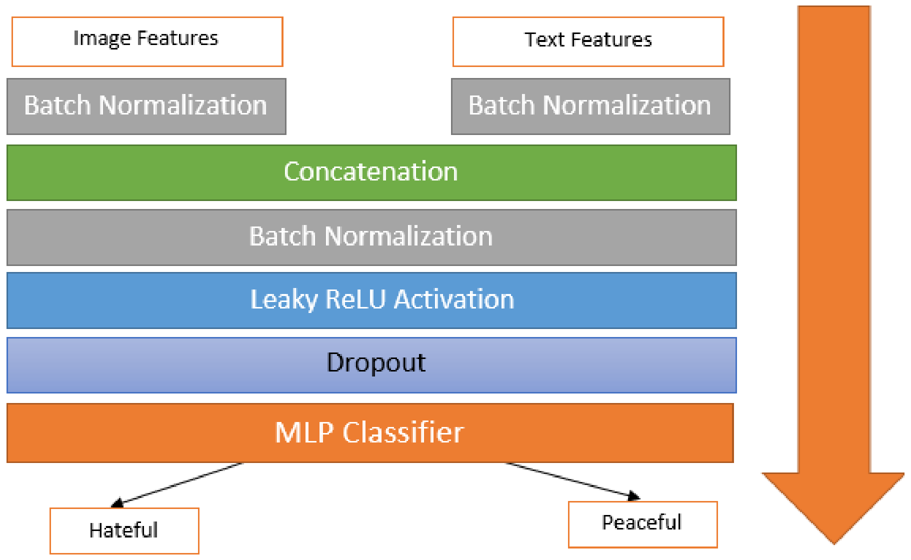
stage	output	ResNet-50	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 128\\3\times3, 128, C{=}32\\1\times1, 256\end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 256\\3\times3, 256, C{=}32\\1\times1, 512\end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 512\\3\times3, 512, C{=}32\\1\times1, 1024\end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 1024\\3\times3, 1024, C{=}32\\1\times1, 2048\end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**$\times10^6$	**25.0**$\times10^6$
FLOPs		**4.1**$\times10^9$	**4.2**$\times10^9$

Table 1. **(Left)** ResNet-50. **(Right)** ResNeXt-50 with a 32×4d template (using the reformulation in Fig. 3(c)). Inside the brackets are the shape of a residual block, and outside the brackets is the number of stacked blocks on a stage. "C=32" suggests grouped convolutions [24] with 32 groups. *The numbers of parameters and FLOPs are similar between these two models.*

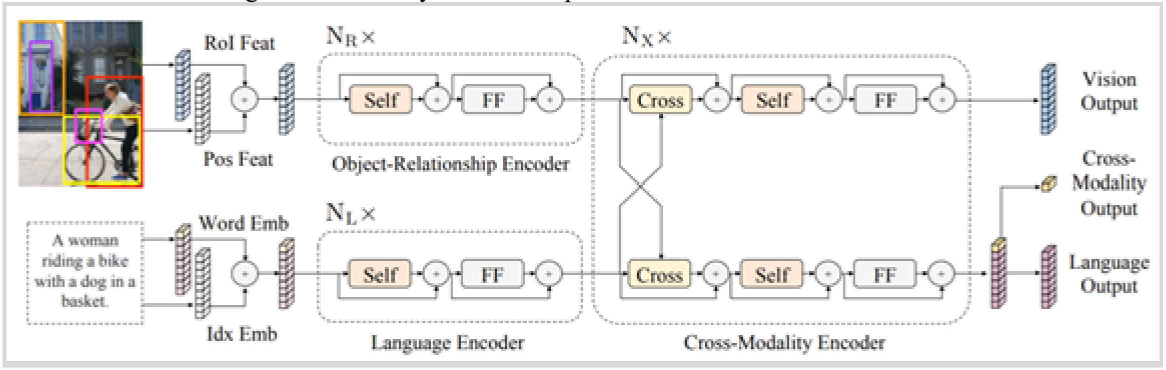
A1.2 Bottleneck Layer



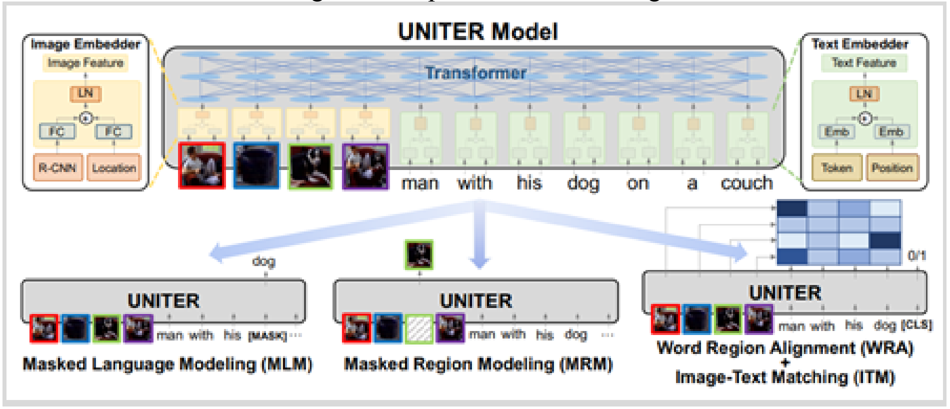
A2 Early fusion logic



A3 LXMERT: Learning Cross-Modality Encoder Representations from Transformers



A4 UNITER: UNiversal Image-Text Representation Learning


UNITER Model
Image Embedder
Image Feature
LN
FC
R-CNN
Location
Transformer
Text Embedder
Text Feature
Emb
Token
Position
man with his dog on a couch
dog
0/1
UNITER
Masked Language Modeling (MLM)
Masked Region Modeling (MRM)
Word Region Alignment (WRA)
+
Image-Text Matching (ITM)

A5 VisualBERT

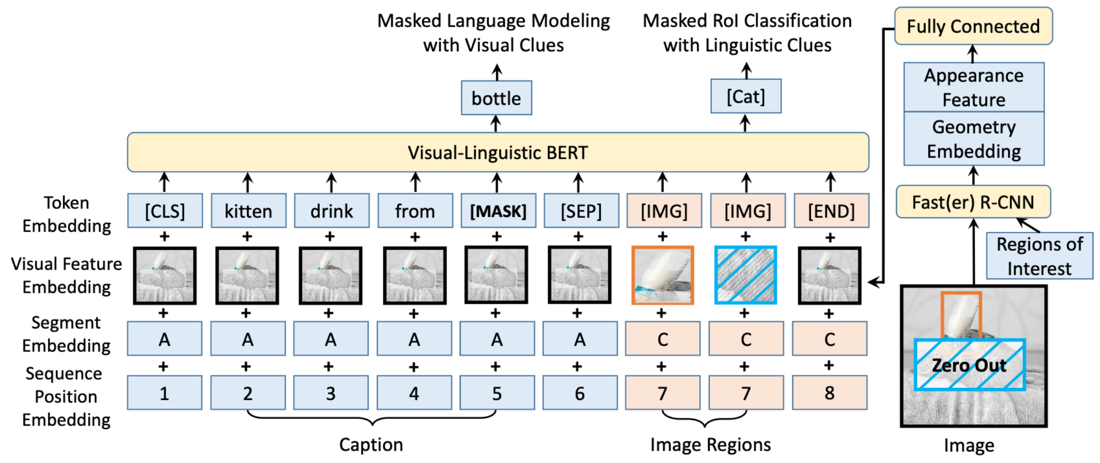
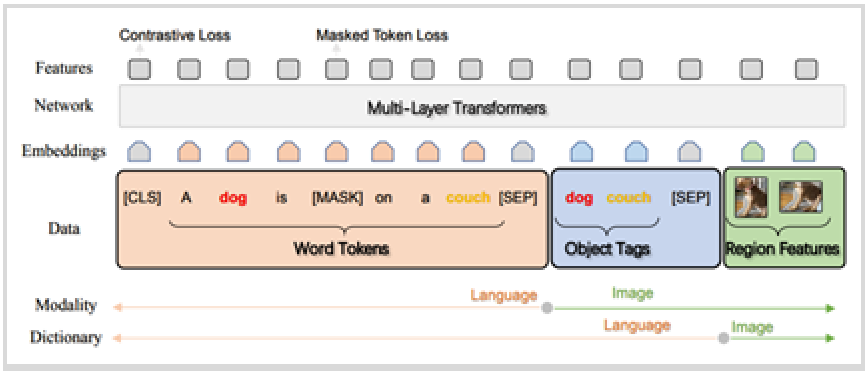


Figure 1. Architecture for Pre-training VL-BERT

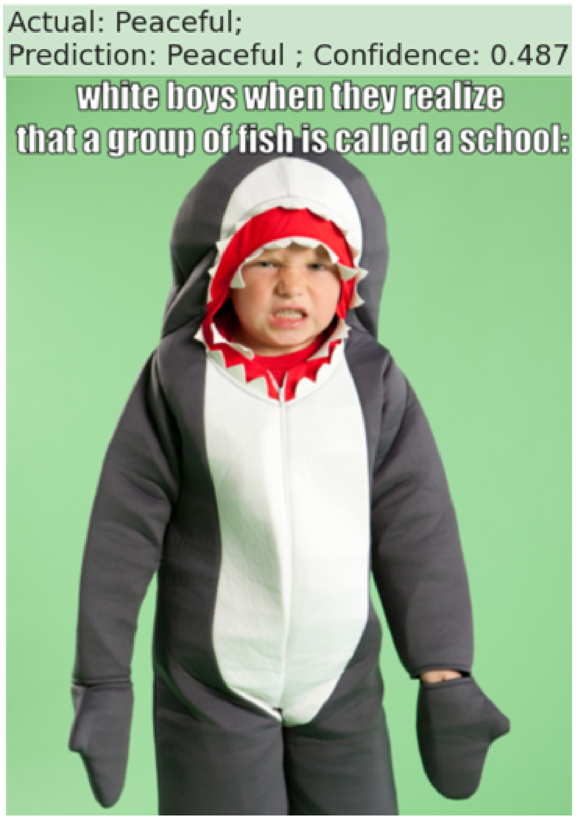
A6 Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks

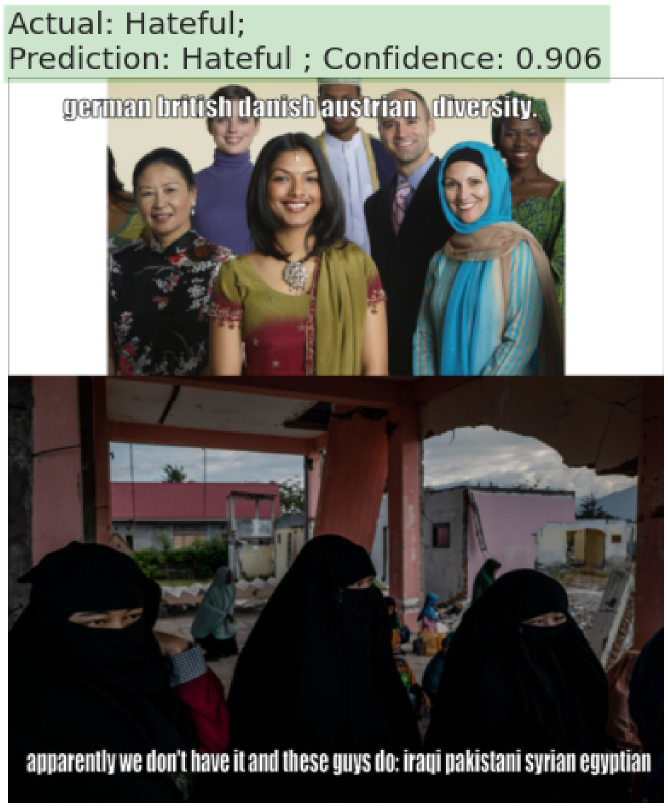


A7 Sample Classified Images

True Negative (TN)

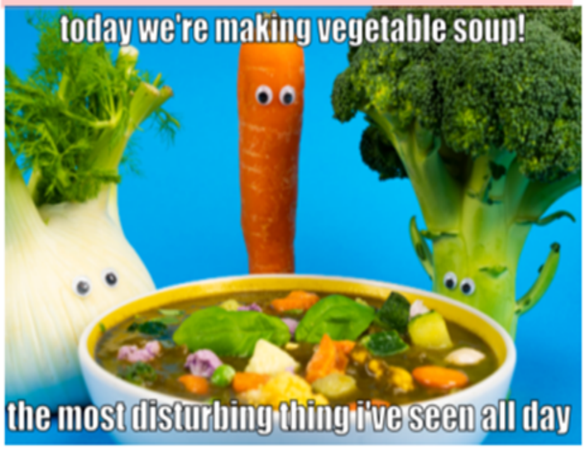
TruePositive (TP)


Actual: Peaceful;
Prediction: Peaceful ; Confidence: 0.487
white boys when they realize
that a group of fish is called a school:


Actual: Hateful;
Prediction: Hateful ; Confidence: 0.906
german british danish austrian diversity.
apparently we don't have it and these guys do: iraqi pakistani syrian egyptian

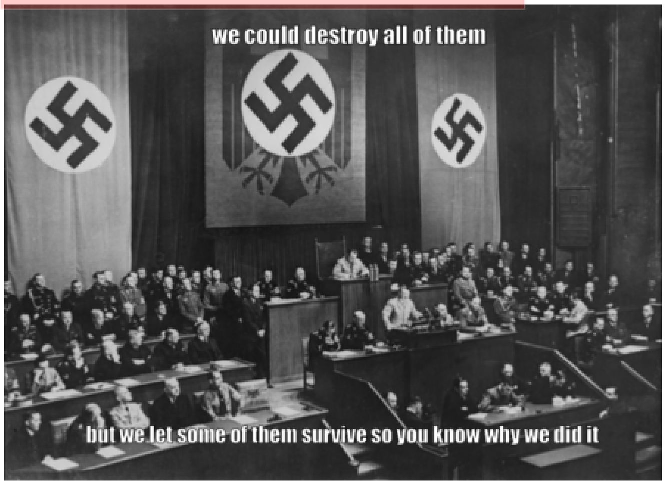
False Positive (FP)

Actual: Peaceful;
Prediction: Hateful ; Confidence: 0.726

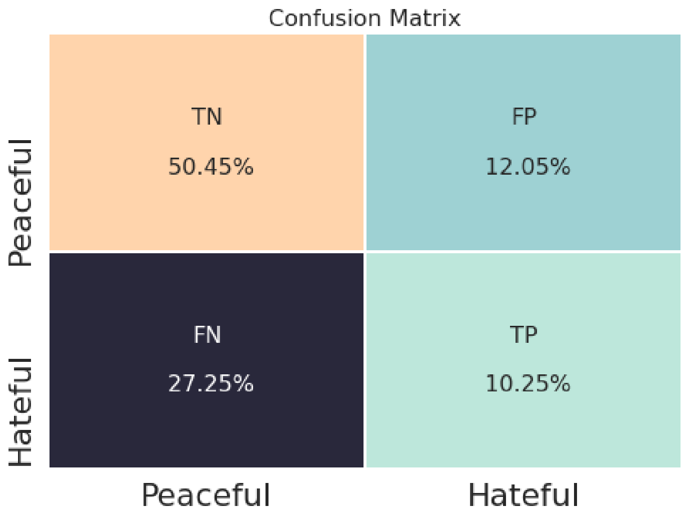

today we're making vegetable soup!
the most disturbing thing i've seen all day

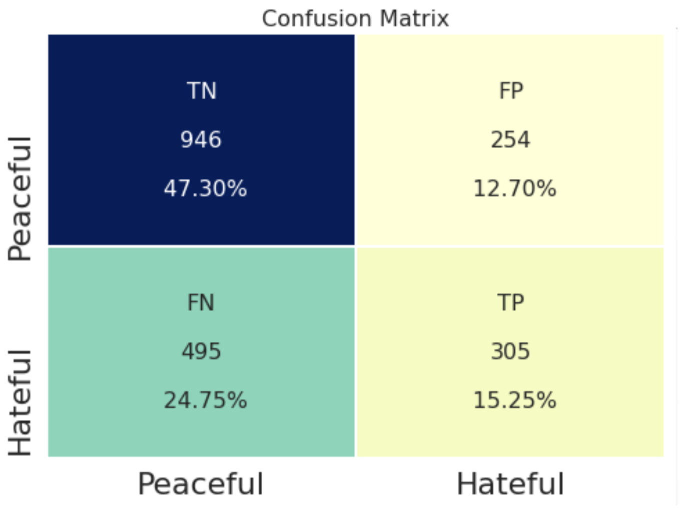
False Negative (FN)

Actual: Hateful;
Prediction: Peaceful ; Confidence: 0.45


we could destroy all of them
but we let some of them survive so you know why we did it

Sample CM:


Confusion Matrix
TN
FP
FN
TP
Peaceful
Hateful
Peaceful
Hateful


Confusion Matrix
TN
FP
FN
TP
Peaceful
Hateful
Peaceful
Hateful