The Impact of CLIP Encoders and CNN Architectures on Model Efficiency: A Case Study on Hate Speech Detection

Christof Kälin[1,2,*], Ellen Rushe[1,*], Stefan Kull[2] and Andrew McCarren[1]

[1]School of Computing, Dublin City University, Dublin 9, Ireland
[2]Lucerne School of Computer Science and Information Technology, Lucerne University of Applied Sciences and Arts, Rotkreuz, Switzerland

Abstract

In this paper, the impact of model architecture, model size and overall efficiency of deep vision-language models is explored, focusing on the trade-off between performance and resource utilisation. Specifically, the effectiveness of Contrastive Language-Image Pre-training (CLIP) encoders combined with Convolutional Neural Networks (CNNs) are evaluated for this task. To this end, a study is performed using the Facebook Hateful Memes Dataset (HMD) to train and evaluate the *Hate-CLIPper* architecture and its variations, alongside smaller scale extentions using convolutional layers. Though the evaluation demonstrates that *Hate-CLIPper* shows the strongest performance, the reduced versions of *Hate-CLIPper*'s cross-fusion mechanism see a large decrease in parameters without a comparably large decrease in performance. This suggests that the addition of parameters to these models may ultimately lead to diminishing returns in terms of performance. We also show that downscaling a smaller version of *HateCLIPper* leads to a larger reduction in performance than its larger-scale counterparts, suggesting a non-linear relationship between the number of parameters and performance gains. The question of overparameterisation in *Hate-CLIPper* is therefore raised, highlighting the importance of balancing model complexity with training efficiency. Through this case study, this research contributes to the development of more efficient methods for automatic hateful meme detection which, in extension, can improve content moderation practices and reduce the spread of online hate speech.

Keywords

Memes and Hate speech detection, Multimodal Data, Contrastive Learning, Convolutional Neural Networks, Intermediate fusion, Facebook Hateful Memes Dataset

1. Introduction

Large-scale deep models have become ubiquitous in vision and language technologies, with models typically containing hundreds of millions – if not billions – of parameters. These models tend to be extremely computationally expensive, impacting both their portability and usability when computational resources are comparatively scarce. The performance gains achieved by novel architectures tend to be valued highly in the literature while, the growing size of models come at a cost. These models have significant environmental impact due to the enormous amount of computational resources and energy needed to train them [1, 2]. Additionally, the increasing trend towards large-scale model development [1] limits the the number of users to those with the most resources, contributing to the de-democratisation of machine learning development [3]. It is critical to determine the actual gains provided by these increasingly large models in order to perform a cost-benefit analysis on their use. To this end, in this paper, we present a case study on a task where large-scale models form the current state-of-the-art: multi-modal hate speech detection. More specifically, we aim to analyse a large-scale CLIP-based vision-language model to determine the impact of different size architectures on the training and inference time, memory requirements and performance for a common hate-speech detection benchmark. We perform an evaluation across increasingly lightweight architectures on a fixed computational set-up and report the effects on performance.

AICS'24: 32nd Irish Conference on Artificial Intelligence and Cognitive Science, December 09–10, 2024, Dublin, Ireland
*Corresponding author.
christof.kalin2@mail.dcu.ie (C. Kälin); ellen.rushe@dcu.ie (E. Rushe); stefan.kull@hslu.ch (S. Kull); andrew.mccarren@dcu.ie (A. McCarren)

The remainder of the paper is organised as follows: Section 2 will outline the task of Hate Speech Detection and existing approaches to address it. Next, we outline the approach we take to evaluating the effect of model size and architecture on a competitive model applied to this task in Section 3. We describe the specific details of our experimental procedure in Section 4. Our findings are outlined in Section 5 and we finish with a discussion on our findings Section 6 .

2. Hateful Meme Detection

We first begin by providing an introduction to hate speech detection and then follow this discussion with a description of recent approaches to this task. We also motivate the architecture we choose to replicate and study in our experiments.

2.1. Task Overview

The traditional internet meme typically features a multimodal format: an image with an overlay of text. This combination often conveys a humorous or positive message, but it can also spread hateful content [4]. Such hateful messages can be disseminated widely via the internet and social media, leading to social issues [5]. Manual moderation of this content is not only slow and costly but has also been reported to be linked to post traumatic stress disorder-like symptoms among content moderators [6, 5]. Therefore, automating the process of content moderation has been adopted and commercialised with companies offering API-based solutions supporting various modalities [7, 8]. The complete automation of content moderation has been criticised, however, with a hybrid approach using automated systems for the majority of cases while also implementing measures to support human moderators, being preferred [9]. Nonetheless, these automated systems currently play a central role in content moderation due to the vast scale of material uploaded to internet platforms each day.

Figure 1: An example of a meme intended to insult the reader (Image sourced from [10, 11]).

Identifying implicit offensive information in memes presents a unique challenge, as text or image that comprise the meme may not appear hateful when each is presented in isolation [12]. Figure 1 illustrates the subtlety of this type of material. Additionally, offensive content can come in a wide variety of forms, and memes can be deemed offensive based on multiple factors, such as personal attacks, racial abuse, or attacks on minorities [4]. The following section will describe the different approaches that have been developed for this complex task.

2.2. Approaches to Hateful Meme Detection

Due to the multi-modal nature of the data and the complex interactions between features of each modality, approaches have primarily comprised of multiple model components or subtasks. For instance,

Hermida & Santos [4] argued that the text and the image of a meme alone would not suffice as an input and that data augmentation is necessary, either through object detectors, caption generators or extracting regions from images to create new ones. Furthermore, they argue that, as architectures are evolving, advanced techniques, such as ensembles of Transformer-based models, are expected to yield the best results.

These premises have been evidenced in the winning architectures of the Facebook Hateful Memes Challenge [1], conducted in 2020 using the competition's Hateful Meme Dataset (HMD) [11], with the two highest-ranked methods combining external data and subtasks. The first approach [13] enriches the memes with Google Vision's Web Entity Detection to capture the images' context. It includes race and gender labels created from the *FairFace* Dataset [14]. After this, it combined *VL-BERT* [15], *UNITER* [16], *VILLA* [17] and *ERNIE-ViL* [18] in one ensemble architecture, achieving an Area Under the Receiver Operating Characteristic (AUROC) of 84.5 [13]. The second-placed approach combined *ERNIE-Vil*, *UNITER*, *OSCAR* [19] and *VisualBert* [20], resulting in an AUROC of 82.52 [21].

Further research was conducted after the competition without ensemble-approaches, such as two-way feature extraction by including images alone as inputs. The extraction is followed by captioning the images with an encoder-decoder model and then feeding this along with Optical character Recognition (OCR)-extracted text into a sentiment analysis model. However, this resulted in a lower AUROC of 56.83 and reported accuracy of 64% [22]. In comparison, employing an approach using the *VinVl* [23] object detection model with *OSCAR+* and combined with a Random Forest (RF) classifier led to an AUROC of 76.8 and an accuracy of 68.4% [24].

Kumar and Nandakumar [25] took a different approach, applying CLIP's [26] pre-trained image and text encoders on the input. Specifically, encoded inputs are sent through projection layers and fine-tuned for the task of hateful meme detection. Projection layers are followed by intermediate fusion with either *cross fusion* or *align fusion* . Elements are then flattened and trained on a shallow neural network (so-called *pre-output layers*), before using the softmax function to retrieve predictions. The authors reported an AUROC of 85.8 on the HMD [25]. A disadvantage of the Hate-CLIPper architecture, however, is that the embeddings of memes have a high similarity despite opposite meanings. In order to address this, *Retrieval-Guided Contrastive Learning* (RGCL) was proposed by Mei et al. [27], who investigated semantically similar memes that are then separated depending on whether both of them are in the same class (so-called *pseudo-gold positive examples*) or opposite class (*hard negative examples*). These pairs are identified primarily by examining misclassifications in the dataset or instances in which confounders are present (an example of such confounders can be seen in Figure 2). RGCL raised the performance to an AUROC of 87.0.

For detecting offensive content, CNNs have also been used as both auto-feature extraction-based methods and as image and text classifiers [4]. In a unimodal context, Suryawanshi et al. [28] used the pre-trained CNN *VGG-16* [29] to classify memes based on the image only. However, a text classifier, *CNNText*, was also trained separately using three convolutional blocks. In a multi-modal context, the researchers used early fusion. For both image and text feature extraction, long short-term memory networks (LSTMs), *VGG-16* and their *CNNText* were employed [28]. Another approach proposes *VGCN-BERT* [30] to extract textual features and three different CNNs to extract image features: *ResNet-50* [31], *ResNet-152* and *VGG-16*. This was followed by creating multimodal models performing early fusion. Ultimately, *VGCN-BERT + ResNet50* yielded the best results with an AUROC of 81.69 on a dataset consisting of memes related to Italian political affairs [32].

For the baseline model in the *Memotion* [33] dataset, GloVe [34] word embeddings were used as an input to a CNN with 64 filters of size 1x5 to extract textual features [35]. In order to maximize F1 scores on the *Memotion* dataset, Alzu'bi et al. [36] used *ResNet-152* and *ResNet-50* to encode the images and then used pooling to produce distinct image embeddings.

By prompting the large language model *RoBERTa* [37] to classify memes, an experimental of AUROC of 90.96 (and an accuracy of 84.47%) has been reported. However, the evaluation is based on different datasets, in which the HMD is not included. This approach indirectly incorporates CLIP encoders [38]

[1] https://hatefulmemeschallenge.com/

by using *ClipCap* [39] to generate captions for the images [40].

An instruction tuning framework was proposed by Hu et al. [41] to enhance the performance of visual reasoning tasks in vision-language models (VLMs), achieving an AUROC of 89.2 and an accuracy of 80.8% on HMD. The proposed framework works in two steps: It generates a program that solves the query, resulting in a chain-of-thought reasoning process. This process is then fed into fine-tuned VLMs (e.g., PaLI-X-VPD [42]) along with the visual input and the textual query. In addition to its performance, another advantage is the capability of explaining why a meme is considered hateful.

Given the complex set of features and interactions necessary for models to extract from hateful memes, the above architectures have proven to be multi-facited and often large - in particular those using ensembles. In this paper, we seek to determine the extent to which the size of architectures affects the performance of the resulting models. To this end, of the approaches discussed in this section, we have chosen the HateCLIPper architecture as our base model, due to its high performance (with a reported AUROC of 85.8 [25] for their best performing model), open source codebase [2], and end-to-end architecture.

3. Methodology

In this section, we will describe the base architecture that our proposed approaches build on, and the specific techniques used to reduce the parameters of the chosen architecture.

3.1. Base model: Hate-CLIPer

Hate-CLIPper [25] is based on CLIP [26], a zero-shot classifier published by OpenAI based on a pre-training task to predict the captions belonging to images. In order to accomplish this task, the authors trained image and text encoders on 400 million image-text pairs. CLIP matches the performance of *ResNet-50* on the ImageNet challenge despite not using any of the original training samples. Pre-training an image encoder and a text encoder enables the prediction of the paired text in the dataset. These are then used to create a zero-shot classifier while the classes are converted into captions for CLIP to predict output class.

HateCLIPper builds on CLIP by extracting text and image encodings from CLIP's encoders and applying *intermediate fusion* where features are fused in the intermediate layers of the network [25]. Kumar and Nandakumar [25] argue that this approach is the most appropriate as this allows interaction between the features of each modality, which is important given that intermediate features learned from text are likely to be related to those learned for images. *HateCLIPper* implements intermediate fusion by taking the features output from CLIP's encoder and modelling the interactions between the resulting image and text features using a *feature interaction matrix* (FIM). The authors add this component to measure the associations between image and text features directly, motivated by the idea that the similarity between the features of text and image pairs that were associated with each other in CLIP may show different associations in hateful memes. Specifically, *HateCLIPper* projects CLIP features using a set of trainable layers to two vectors p_i and p_t, each of the same dimension n (the underlying CLIP encoders are not finetuned). The feature interaction matrix consists of the outer product of these two vectors $p_i \otimes p_t$ leading to an $n \times n$ matrix. The use of the outer product operation here is referred to as *cross-fusion*. The FIM is then flattened and followed by a number of fully connected layers to make the final classification, i.e. hateful or non-hateful.

One drawback of the FIM is that it increases the size of the model substantially. In terms of computational requirements, powerful hardware is required to train this model, with [25] reporting that *Hate-CLIPper* was originally trained on an NVIDIA Tesla A100 GPU with 40 GB GPU RAM, with training taking approximately 30 minutes. Access to this level of hardware is both expensive and scarce. In more accessible environments, such as Google Colab, these types of resources are not guaranteed [43]. Due to the increased parameters introduced by the use of cross-fusion, the authors create a variation

[2] https://github.com/gokulkarthik/hateclipper

of *Hate-CLIPer* that instead uses *align-fusion*. This variant takes the diagonal of the FIM and builds the classifcation model on this alone. This reduces the dimensionality from a FIM matrix of size n^2 to a vector of size n. The authors report that align fusion produced the best performance in terms of AUROC and cross-fusion performed best in terms of micro-F1 score.

3.2. Reducing HateCLIPper

In this paper we seek to determine the effects of reducing the size of *HateCLIPper*. This can be done using a number of strategies, align-fusion being the stategy proposed by [25]. We evaluate a number of alternative strategies using both cross-fusion and align-fusion, and compare their training time and performance to that of the original models. The two primary strategies are (i) dimensionality reduction of the projection layer by using a reduced CLIP base model and (ii) using shared parameters through the use of convolutional layers.

(i) Reduced CLIP model: The most obvious first approach is to simply reduce the size of the base CLIP model used to encode image and text features, leading to a lower dimensional projection layer. In our experiments, this change reduces the number of parameters by roughly half.

(ii) Convolutional layers: The next approach to parameter reduction is the use of convolution layers rather than fully-connected layers after the fusion of the FIM. One of the core benefits of CNNs is parameter sharing, which reduces the number of trainable parameters and allows the addition of pooling layers after the convolutional layers in order to shrink feature maps [44]. Here we use the same architecture as HateCLIPper up to and including FIM, however we apply convolutional layers to the FIM rather than fully connected layers. We experiment with both cross-fusion and align-fusion. Specifically, when using align-fusion, the convolutional block will apply 1D convolutions to the diagonal of the FIM (which is a vector of size n). In the cross fusion variant, a block using 2D convolutions can be applied directly to the $\mathrm{n}\times\mathrm{n}$ FIM - substantially reducing the number of parameters necessary to train.

4. Experimental Setup

This section will describe the dataset used for experiments, the specific implementation details and hyperparameters of all models, and the evaluation stratgy employed.

4.1. Dataset

The *Facebook Hateful Memes Dataset* (HMD) is a widely used benchmark for hateful meme detection due to the controlled data collection process. During the creation of the HMD , a strict definition for hate speech was employed and annotators were trained for four hours with three pilot runs [11]. This set was then used as a part of a competition hosted by Facebook AI (now Meta AI)[3]. The dataset comes with training, development, and test splits –totalling 10,000 memes. Additionally, so-called "benign confounders" were included: Each hateful meme includes an alternative meme that is non-hateful by changing either the text or the image (Figure 2).

For all experiments, the dataset HMD was used, downloaded directly from the authors' original source[4] and PyTorch `Dataset`[5] was used to create the datasets for training and inference. When creating the PyTorch `Dataset`, tensors were generated using the `CLIPProcessor` model to retrive the pixel values for the images and `CLIPTokenizer` to tokenize the text using max length padding.

[3]https://ai.meta.com/blog/hateful-memes-challenge-and-data-set/
[4]https://hatefulmemeschallenge.com/
[5]https://pytorch.org/tutorials/beginner/basics/data_tutorial.html

(a) Benign text confounder

(b) Benign image confounder

Figure 2: Example of Benign confounders turning the meme from Figure 1 into non-hateful ones (Images sourced from [10, 11]).

4.2. Models

Depending on the model used for the training (large vs. base), the underlying pre-trained CLIP model was exchanged when instantiating the processor and tokenizer. All pre-trained models were loaded using HuggingFace's *transformers*[6] library.

4.2.1. Hate-CLIPper

The *Hate-CLIPper* re-implementation is based on the hyperparameters outlined in its original paper as well as the code provided by the authors [7] with the exception that *PyTorch* was used instead of *PyTorch Lightning* initially. Table 1 lists only parameters that differ by model, thus the following information is not specifically stated for each model: The learning rate and the weight decay are of the same value of 0.0001 for all models, the optimizer used is AdamW [45], the loss function is Binary Cross Entropy and the models were trained over 20 epochs. The pre-output layer consists of one fully-connected layer. The following dropout [46] is used: 0.2 for the projection layers, 0.4 for the pre-output layer and 0.1 after the ReLU activation in the pre-output layer. The dimensionalities used in *Hate-CLIPper* are of size $\mathrm{n} = 1024$ for the projected image encodings $\mathrm{p_i}$ and text encodings $\mathrm{p_t}$ due to encoders provided in *clip-vit-large-patch14*. Using the best-performing cross-fusion variant results in a flattened vector of size n^2 [25].

4.2.2. Hate-CLIPper reduced

Based on the re-implementation, a reduced model using OpenAI's smaller base model was trained in an attempt to reduce the number of parameters. *clip-vit-base-patch32* encoders can be used, resulting in smaller dimensionalities of $\mathrm{n} = 768$. *Hate-CLIPper reduced* refers to the same architecture as *Hate-CLIPper* but using CLIP's base model and smaller dimensionalities.

To determine the optimal hardware on which to train the models, this architecture was trained on a TPU v2 and the batch size was increased to 1,024. The training time was approximately 90 minutes. Training *Hate-CLIPper reduced* for approximately 5 minutes on an A100 GPU uses 0.98 compute units ($\frac{1}{12} \times 11.77$ compute units/hour) as opposed to 2.64 compute units on the TPU v2 ($\frac{3}{2} \times 1.76$ compute units/hour). Therefore, all following experiments were conducted on an A100 GPU and using a TPU was deemed unviable for this task.

[6] https://huggingface.co/docs/transformers/main/en/model_doc/clip
[7] https://github.com/gokulkarthik/hateclipper

4.2.3. CNN models

Three different variations of CNN-based models were trained, referred to as *CNN V1–V3* in Table 1. All of which use CLIP's smaller base models to encode the inputs with projections of dimension of 768.

1. *V1* uses align fusion, resulting in a vector of length 768. This is sent through three one-dimensional convolutional layers doubling the numbers of channels each time followed by ReLU activations. After the convolutional block, the outputs are flattened and sent through a linear output layer to retrieve the logits for the binary classification. No dropout was used aside from 0.2 on the projection layers as in *Hate-CLIPper*.
2. The convolutional architecture of *V2* is similar to *V1*. The only difference is the change of the fusion method from align fusion to cross fusion, leading to an input matrix of dimension 768^2. Therefore, two-dimensional convolutions were used given the matrix input. Due to the squared input size a reduced batch size of 32 was used.
3. ***V3*** was designed to more closely align with the *Hate-CLIPper* architecture by reducing the number of convolutional layers from three to one and reintroducing dropout. When *Hate-CLIPper* was trained by the original authors, experiments were conducted with both one and three pre-output layers, whereas the architecture with one layer performed slightly better (+0.46% on the AUROC [25]).

4.3. Evaluation

Based on the HMD , human performances were established with an accuracy of 84.7% [11]. AUROC is considered the main metric for the challenge. Due to easier interpretation and its balanced test set, accuracy is recommended to be reported as well [11]. All metrics are calculated with the TorchMetrics[8] library using `AUROC`, `Accuracy` and `BinaryF1Score`. The same splits were used as provided by the HMD . For calculating the validation during the training, the unseen development set (540 memes) was used, while for the evaluation of the performance the unseen test set (2,000 memes) was used. The reported inference time was measured in a CPU only environment.

5. Results

The evaluation of all models are shown in Table 1 while Figure 3 shows the Reciever Operating Characteristic (ROC) curves. We note that the objective of these experiments is not to determine the best model for the task in terms of performance, but to analyse the diffence in performance relative to the parameter reduction. We start by discussing models using cross-fusion and then move on to those using align-fusion.

5.1. Cross-fusion Models

Hate-CLIPer with Cross-fusion: While the replicated *Hate-CLIPper* using cross fusion reached neither the human level accuracy nor the AUROC of 85.12 reported by *Hate-CLIPper's* authors [25] using the configurations described in Section 4, this model still achieves the highest performance compared to all other methods applied in this paper. This is expected given the extremely large number of parameters in *HateCLIPper*'s cross-fusion model. In the next sections we will describe the effects of parameter reduction using the mechanisms outlined in Section 3.

Hate-CLIPer Reduced: Using the reduced base CLIP model decreased both the performance (6.20% decrease in terms of AUROC) and training time (85.51% decrease). Given an analysis of the confusion matrices for both the seen and unseen test set, most misclassifications can be found in the False Negatives. For measuring how well the positive class `1` (Hateful) is detected, the sensitivity ($\frac{\mathrm{TP}}{\mathrm{TP+FN}}$) is calculated.

[8] https://lightning.ai/docs/torchmetrics/stable/

Table 1
Evaluation of the trained models on the HMD

	Hate-CLIPper	Reduced	CNN V1	CNN V2	CNN V3
Trained on	A100 GPU	A100 GPU	A100 GPU	A100 GPU	A100 GPU
Fusion method	Cross	Cross	Align	Cross	Cross
P.o. layer	1x linear + dropout	1x linear + dropout	3x 1D conv	3x 2D conv	1x 2D conv + dropout
P.o. input dim	1024^2	768^2	768	768^2	768^2
Parameters	1'075'580'929	453'970'945	990'861	5'703'561	2'164'245
Batch size	64	64	64	32	64
Training time	39min 0s	5min 39s	3min 42s	7min 23s	4min 37s
Inference time	37min 57s	2min 42s	3min 22s	3min 10s	2min 29s
AUROC	82.13	77.04	65.81	72.14	72.47
Accuracy	75.70	71.45	64.40	69.00	68.90
F1	64.88	55.49	38.40	50.00	52.59
Size	5.6 GB	2.25 GB	578.6 MB	596.5 MB	255.4 MB

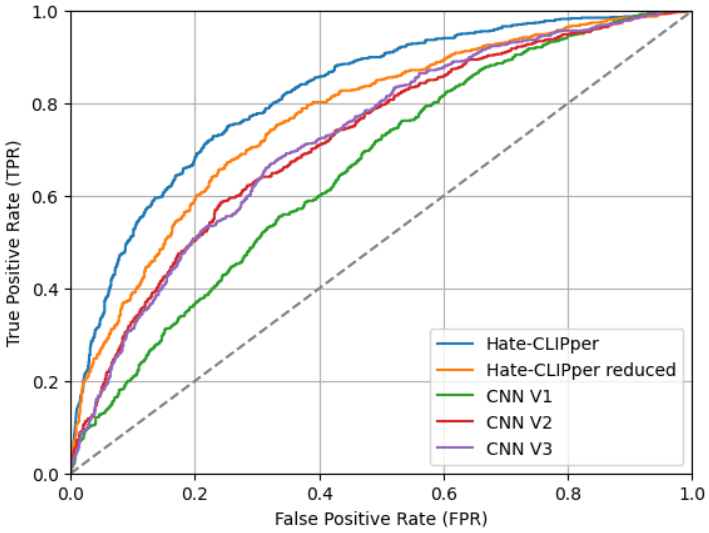

Figure 3: ROC curve of all trained models as a part of this paper.

For *Hate-CLIPper* it is 59.97 while for *Hate-CLIPper Reduced* it is 47.47. This highlights that an enormous reduction in parameters will not necessarily result in a similarly large reduction in performance.

CNN models: The largest CNN-based model, *CNN V2*, performs comparably to *CNN V3* which is just under half the size of *V2*. Additionally, *CNN V3* only sees a 5.93% decrease in AUROC compared to *Hate-CLIPer Reduced* despite being 99.52% smaller.

The F1 score for *CNN V3* was best compared to all other CNN based models. The sensitivity was measured at 46.0, the confusion matrix is shown in Figure 4. These results taken together with HateCLIPper Reduced appear to suggest that there is not a linear relationship between the number of parameters and the performance of the model in terms of AUROC, with their being a diminishing return in terms of performance as the scale of the parameters increases.

5.2. Align-fusion Models

Despite the comparably small reduction in performance that the proposed cross-align based models provide relative to the reduction in parameters, we note that the authors of *Hate-CLIPer* suggested an

CNN V3			
TARGET / OUTPUT	Hateful	Non-hateful	SUM
Hateful	345 17.25%	217 10.85%	562 61.39% 38.61%
Non-hateful	405 20.25%	1033 51.65%	1438 71.84% 28.16%
SUM	750 46.00% 54.00%	1250 82.64% 17.36%	1378 / 2000 68.90% 31.10%

Figure 4: Confusion matrix of the *CNN V3* model reporting high specificity but low sensitivity.

alternative to cross-fusion in order to reduce model parameters - align-fusion, as discussed in Section 3. Remarkably, despite this model reducing the parameters from 1.1 billion when using cross-fusion to 5 million for the best performing models, they report comparable performance for the best align-fusion model, despite an enormous parameter reduction. Taken together with our above results using cross-fusion, it appears that the mechanism by which the parameter reduction is achieved holds more weight than the parameter reduction itself. Given the success of align-fusion, it may be a more obvious approach for practitioners using this model than cross-fusion models. As discussed in Section 3, with this in mind, we performed an additional experiment where we applied 1D convolution to features resulting from align-fusion. Though not re-implemented for this paper, the authors report an AUROC of 85.8 and approximately 5 million parameters. *CNN V3*, using align fusion with just under a million parameters provides an AUROC of 65.81. This indicates that the performance reductions may align more closely to the decrease in parameters as the overall scale of the models decreases. This suggests that there may be a cut-off under which additional parameter reductions have a large effect on performance for this task.

6. Conclusion

In this paper we have analysed the effects of different mechanisms of parameter reduction for a the hate speech detection model, *Hate-CLIPper*. We have found that large parameter reductions are not associated with comparably large reductions in performance for larger scale models. This suggests that a larger set of parameters for this task may provide diminishing returns. On smaller scale models, we see a larger effect on performance when reducing the number of parameters. Additionally, when comparing the performance of *HateCLIPper* using align-fusion described by [25], we see that their model achieves a higher level of performance compared to the reduced cross-fusion based models proposed here, suggesting that the mechanism of parameter reduction may influence the performance of the model more than the parameter reduction itself. This indicates that a more careful analysis of feature fusion and subsequent operations may be an effective means of predictably downscaling large-scale models.

References

[1] Z. Wan, X. Wang, C. Liu, S. Alam, Y. Zheng, J. Liu, Z. Qu, S. Yan, Y. Zhu, Q. Zhang, M. Chowdhury, M. Zhang, Efficient large language models: A survey, Transactions on Machine Learning Research (2024). URL: https://openreview.net/forum?id=bsCCJHbO8A, survey Certification.

[2] J.-W. Chung, Y. Gu, I. Jang, L. Meng, N. Bansal, M. Chowdhury, Perseus: Removing energy bloat from large model training, arXiv preprint arXiv:2312.06902 (2023).

[3] N. Ahmed, M. Wahed, The de-democratization of ai: Deep learning and the compute divide in artificial intelligence research, arXiv preprint arXiv:2010.15581 (2020).

[4] P. C. d. Q. Hermida, E. M. d. Santos, Detecting hate speech in memes: a review, The Artificial intelligence review 56 (2023) 12833–12851.

[5] Y. Chen, F. Pan, Multimodal detection of hateful memes by applying a vision-language pre-training model, PLOS ONE 17 (2022) 1–12. doi:10.1371/journal.pone.0274300.

[6] C. Newton, The trauma floor: The secret lives of facebook moderators in america, https://www.theverge.com/2019/2/25/18229714/cognizant-facebook-content-moderator-interviews-trauma-working-conditions-arizona, 2019. Accessed: 2024-10-08.

[7] Checkstep, Checkstep: AI Content Moderation Services Tool Platform, 2024. URL: https://www.checkstep.com/.

[8] Membrace, Membrace - AI Content Moderation and Improvement Tools, 2024. URL: https://membrace.ai/.

[9] T. Gillespie, Content moderation, ai, and the question of scale, Big Data & Society 7 (2020). doi:10.1177/2053951720943234.

[10] Hateful Memes Challenge and dataset for research on harmful multimodal content, https://ai.meta.com/blog/hateful-memes-challenge-and-data-set/, 2020. Accessed: 2024-10-08.

[11] D. Kiela, H. Firooz, A. Mohan, V. Goswami, A. Singh, P. Ringshia, D. Testuggine, The hateful memes challenge: Detecting hate speech in multimodal memes, Advances in neural information processing systems 33 (2020) 2611–2624.

[12] L. Shang, C. Youn, Y. Zha, Y. Zhang, D. Wang, Knowmeme: A knowledge-enriched graph neural network solution to offensive meme detection, in: 2021 IEEE 17th International Conference on eScience (eScience), 2021, pp. 186–195. doi:10.1109/eScience51609.2021.00029.

[13] R. Zhu, Enhance multimodal transformer with external label and in-domain pretrain: Hateful meme challenge winning solution, CoRR abs/2012.08290 (2020). URL: https://arxiv.org/abs/2012.08290. arXiv:2012.08290.

[14] K. Karkkainen, J. Joo, Fairface: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation, in: Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2021, pp. 1548–1558.

[15] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, J. Dai, Vl-bert: Pre-training of generic visual-linguistic representations, in: International Conference on Learning Representations, 2020. URL: https://openreview.net/forum?id=SygXPaEYvH.

[16] Y.-C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, J. Liu, Uniter: Universal image-text representation learning, in: European conference on computer vision, Springer, 2020, pp. 104–120.

[17] Z. Gan, Y.-C. Chen, L. Li, C. Zhu, Y. Cheng, J. Liu, Large-scale adversarial training for vision-and-language representation learning, Advances in Neural Information Processing Systems 33 (2020) 6616–6628.

[18] F. Yu, J. Tang, W. Yin, Y. Sun, H. Tian, H. Wu, H. Wang, Ernie-vil: Knowledge enhanced vision-language representations through scene graphs, in: Proceedings of the AAAI conference on artificial intelligence, volume 35, 2021, pp. 3208–3216.

[19] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, et al., Oscar: Object-semantics aligned pre-training for vision-language tasks, in: the 16th European Conference on Computer Vision, Springer, 2020, pp. 121–137.

[20] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, K.-W. Chang, Visualbert: A simple and performant baseline

for vision and language, arXiv preprint arXiv:1908.03557 (2019).
[21] N. Muennighoff, Vilio: State-of-the-art visio-linguistic models applied to hateful memes, CoRR abs/2012.07788 (2020). arXiv:2012.07788.
[22] A. Aggarwal, V. Sharma, A. Trivedi, M. Yadav, C. Agrawal, D. Singh, V. Mishra, H. Gritli, Two-way feature extraction using sequential and multimodal approach for hateful meme classification, Complexity (New York, N.Y.) 2021 (2021) 1–7.
[23] P. Zhang, X. Li, X. Hu, J. Yang, L. Zhang, L. Wang, Y. Choi, J. Gao, Vinvl: Revisiting visual representations in vision-language models, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 5579–5588.
[24] Y. Chen, F. Pan, Multimodal detection of hateful memes by applying a vision-language pre-training model, PloS one 17 (2022) e0274300–e0274300.
[25] G. K. Kumar, K. Nandakumar, Hate-CLIPper: Multimodal hateful meme classification based on cross-modal interaction of CLIP features, in: L. Biester, D. Demszky, Z. Jin, M. Sachan, J. Tetreault, S. Wilson, L. Xiao, J. Zhao (Eds.), Proceedings of the Second Workshop on NLP for Positive Impact (NLP4PI), Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Hybrid), 2022, pp. 171–183.
[26] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al., Learning transferable visual models from natural language supervision, in: International conference on machine learning, PMLR, 2021, pp. 8748–8763.
[27] J. Mei, J. Chen, W. Lin, B. Byrne, M. Tomalin, Improving hateful meme detection through retrieval-guided contrastive learning, in: L.-W. Ku, A. Martins, V. Srikumar (Eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 5333–5347. doi:10.18653/v1/2024.acl-long.291.
[28] S. Suryawanshi, B. R. Chakravarthi, M. Arcan, P. Buitelaar, Multimodal meme dataset (MultiOFF) for identifying offensive content in image and text, in: R. Kumar, A. K. Ojha, B. Lahiri, M. Zampieri, S. Malmasi, V. Murdock, D. Kadar (Eds.), Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying, European Language Resources Association (ELRA), Marseille, France, 2020, pp. 32–41.
[29] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).
[30] Z. Lu, P. Du, J.-Y. Nie, Vgcn-bert: augmenting bert with graph embedding for text classification, in: Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part I 42, Springer, 2020, pp. 369–382.
[31] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[32] G.-A. Vlad, G.-E. Zaharia, D.-C. Cercel, M. Dascalu, Upb @ dankmemes: Italian memes analysis - employing visual models and graph convolutional networks for meme identification and hate speech detection, in: Proceedings of the Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian Final Workshop, Accademia University Press, 2020.
[33] C. Sharma, D. Bhageria, W. Scott, S. PYKL, A. Das, T. Chakraborty, V. Pulabaigari, B. Gambäck, SemEval-2020 task 8: Memotion analysis- the visuo-lingual metaphor!, in: A. Herbelot, X. Zhu, A. Palmer, N. Schneider, J. May, E. Shutova (Eds.), Proceedings of the Fourteenth Workshop on Semantic Evaluation, International Committee for Computational Linguistics, Barcelona (online), 2020, pp. 759–773. doi:10.18653/v1/2020.semeval-1.99.
[34] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.
[35] C. Sharma, D. Bhageria, W. Scott, S. PYKL, A. Das, T. Chakraborty, V. Pulabaigari, B. Gambäck, SemEval-2020 task 8: Memotion analysis- the visuo-lingual metaphor!, in: A. Herbelot, X. Zhu, A. Palmer, N. Schneider, J. May, E. Shutova (Eds.), Proceedings of the Fourteenth Workshop on Semantic Evaluation, International Committee for Computational Linguistics, Barcelona (online),

2020, pp. 759–773. doi:10.18653/v1/2020.semeval-1.99.
[36] A. Alzu'bi, L. Bani Younis, A. Abuarqoub, M. Hammoudeh, Multimodal deep learning with discriminant descriptors for offensive memes detection, J. Data and Information Quality 15 (2023). doi:10.1145/3597308.
[37] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach. arxiv [preprint](2019), arXiv preprint arXiv:1907.11692 (1907).
[38] R. Cao, R. K.-W. Lee, W.-H. Chong, J. Jiang, Prompting for multimodal hateful meme classification, in: Y. Goldberg, Z. Kozareva, Y. Zhang (Eds.), Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 321–332. doi:10.18653/v1/2022.emnlp-main.22.
[39] R. Mokady, A. Hertz, A. H. Bermano, Clipcap: Clip prefix for image captioning, arXiv preprint arXiv:2111.09734 (2021).
[40] R. Mokady, A. Hertz, A. H. Bermano, Clipcap: CLIP prefix for image captioning, CoRR abs/2111.09734 (2021). arXiv:2111.09734.
[41] Y. Hu, O. Stretcu, C.-T. Lu, K. Viswanathan, K. Hata, E. Luo, R. Krishna, A. Fuxman, Visual program distillation: Distilling tools and programmatic reasoning into vision-language models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 9590–9601.
[42] Y. Hu, O. Stretcu, C.-T. Lu, K. Viswanathan, K. Hata, E. Luo, R. Krishna, A. Fuxman, Visual program distillation: Distilling tools and programmatic reasoning into vision-language models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 9590–9601.
[43] Google Colaboratory, Google Colab, 2020. URL: http://colaboratory.google.com.
[44] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, L. Farhan, Review of deep learning: concepts, cnn architectures, challenges, applications, future directions, Journal of big data 8 (2021) 53–53.
[45] I. Loshchilov, Decoupled weight decay regularization, arXiv preprint arXiv:1711.05101 (2017).
[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The journal of machine learning research 15 (2014) 1929–1958.