MEMETECTOR: ENFORCING DEEP FOCUS FOR MEME DETECTION

Christos Koutlis, Manos Schinas, Symeon Papadopoulos,
ckoutlis@iti.gr, manosetro@iti.gr, papadop@iti.gr
CERTH-ITI, Greece

ABSTRACT

Image memes and specifically their widely-known variation *image macros*, is a special new media type that combines text with images and is used in social media to playfully or subtly express humour, irony, sarcasm and even hate. It is important to accurately retrieve image memes from social media to better capture the cultural and social aspects of online phenomena and detect potential issues (hate-speech, disinformation). Essentially, the background image of an image macro is a regular image easily recognized as such by humans but cumbersome for the machine to do so due to feature map similarity with the complete image macro. Hence, accumulating suitable feature maps in such cases can lead to deep understanding of the notion of image memes. To this end, we propose a methodology, called *Visual Part Utilization*, that utilizes the visual part of image memes as instances of the *regular image class* and the initial image memes as instances of the *image meme class* to force the model to concentrate on the critical parts that characterize an image meme. Additionally, we employ a trainable attention mechanism on top of a standard ViT architecture to enhance the model's ability to focus on these critical parts and make the predictions interpretable. Several training and test scenarios involving web-scraped regular images of controlled text presence are considered for evaluating the model in terms of robustness and accuracy. The findings indicate that light visual part utilization combined with sufficient text presence during training provides the best and most robust model, surpassing state of the art. Source code and dataset are available at `https://github.com/mever-team/memetector`.

Keywords Meme Detection · Visual Part Utilization · Trainable Attention · Vision Transformer

1 Introduction

Image memes have been established during the last years as a popular means of communication in social media. Their typical form, known as *image macros*[1], comprise images with overlay text at the top and/or bottom and are principally used to express a spectrum of concepts and emotions such as humour, irony, sarcasm and even hate. Memes and regular images have critical visual differences that render their discrimination an easy task for a human, such as the overlay text with a specific type of font size, color, family and position as well as the background image usually having a cultural reference or being memorable. In contrast, regular images potentially depict anything without certain constraints. In Figure 1, we exemplify one *image meme*[2] and one *regular image*[3] to showcase the differences between the two types of digital media.

Other forms of Internet memes also exist, for instance they might be plain text [3], tweet screenshots, social statement cards, logos [4] or images reusing memorable visual elements in different creative ways such as Bernie Sanders' mittens[4]. In addition, the adoption of different meme forms seems to be highly platform-dependent as community-specific vernaculars determine different meme cultures [4]. Here, we only address the detection of the typical Internet meme form, namely image macros being a combination of background images with superimposed text as Figure 1a.

---

[1] `https://en.wikipedia.org/wiki/Image_macro`
[2] Source: Facebook's HateFul Memes dataset [1]
[3] Source: Google's Conceptual Captions dataset [2]
[4] `https://en.wikipedia.org/wiki/Bernie_Sanders_mittens_meme`

(a) Image meme   (b) Regular image

Figure 1: Example image meme vs. a regular image.

In the framework of analysing digital social behavior and trends, image memes have attracted research interest [5, 6, 7, 8], mostly with a focus on deep learning models for image meme classification [9, 10, 11, 12] and more frequently for the detection of *hateful* image memes [13, 14, 15, 16]. The latter works utilize datasets with image memes and appropriate labeling, thus they do not put effort on detecting whether an input image is a meme or not. The detection of image memes and their discrimination from regular images is currently still a relatively understudied topic, and there are only few attempts in this direction [17, 18].

In this work, we present *MemeTector*, a model for efficiently classifying images as memes or regular ones. Utilizing this model in online social environments to retrieve memes can facilitate the monitoring and analysis of web trends and behaviors as well as the detection of harmful practices that are sometimes carried out with the use of memes, such as hate-speech and disinformation. To enforce the model's focus on critical visual cues that characterize both classes, we propose Visual Part Utilization (VPU), a methodology for artificial dataset creation from an existing image meme dataset [1] and a deep learning architecture called ViTa, employing a trainable attention mechanism on top of a Vision Transformer (ViT) [19]. Although we propose ViTa for meme detection it is general and can potentially be utilized to other tasks. Regarding VPU, from each image meme instance M_i of the initial dataset we extract the largest part that contains no text and utilize it as a regular image instance. We denote the latter with V_i and call it visual part of image meme M_i. In Figure 2, we present the construction process for set **M** of image memes and **V** of visual parts. In essence, a meme's background image is a regular image which is easily discriminated from the meme by humans while neural networks initially produce almost identical feature maps for both. VPU is thus considered here for effectively enhancing the learning of class distribution subtle peculiarities.

The paper is structured as follows. Section 2 reviews the related literature. Section 3 elaborates the proposed methodology. Section 4 describes the experimental setup. Section 5 presents the results. Section 6 concludes the paper.

2   Related work

Previous related studies focus mainly on classification of memes into categories such as hateful or offensive [20]. Due to their multi-modal nature the classification of image memes is most frequently treated as a multi-modal analysis problem by processing both visual and text content [9]. Other studies attempt to classify image memes in terms of their sentiment (positive, negative, neutral) and type of humour (e.g. sarcastic or motivational) [21]. However, the topic of this study precedes the work of classifying image memes into certain categories. First, one needs to know if an image is a meme or just a regular image before further analysing it.

The topic of image meme detection, i.e. automatically discriminating image memes from regular images, has not yet received considerable attention from the research community. To our knowledge, there is only one dataset for meme detection, namely the DankMemes dataset [17], which was released in 2020 but is publicly unavailable at the timing of writing the paper. DankMemes contains 2000 images related to the 2019 Italian government crisis, half of which are memes and the rest are regular images. In terms of competing approaches, only few meme detection methods have been recently presented [22, 23, 24]. Also, the similar problem of identifying satire images on social media is addressed in [18]. Finally, a few other attempts exist on the Internet, for instance in blog posts or GitHub repositories, but are not peer reviewed.[5,6]

This is the first paper to utilize the visual part of image memes as instances of the regular images class, thus enabling: (i) a 100% accurate albeit automatic annotation process through the usage of already existing meme classification datasets, as well as (ii) the creation of a dataset of 40,000 images (x20 larger than the DankMemes dataset). Additionally, we are

---

[5]https://github.com/matyasbohacek/meme-detection
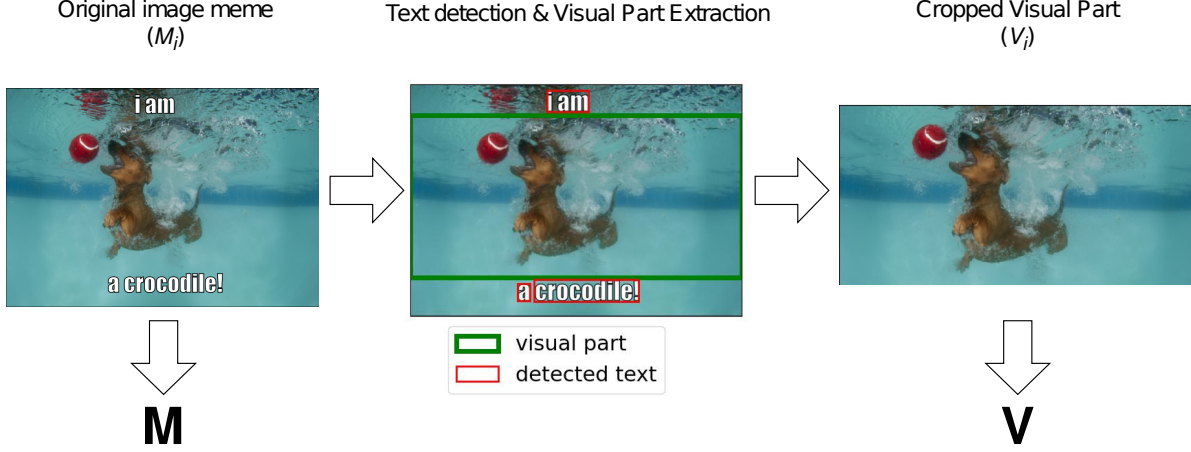[6]https://medium.datadriveninvestor.com/memes-detection-android-app-using-deep-learning-d2c65347e6f3

Figure 2: The proposed Visual Part Utilization process. The original image M_i, which belongs to the set of image memes **M**, is passed through the visual part extraction algorithm that identifies the corresponding visual part V_i, crops it and adds it to the set **V**. Best viewed in color.

the first to employ a supplemental attention mechanism on top of a ViT architecture that empowers deep focus through the combination of different levels of information granularity and interpretability of the results.

3 Methodology

Here we present MemeTector's building blocks, namely VPU and ViTa.

3.1 Visual Part Utilization

3.1.1 Extraction

To extract the visual part V_i of a given image meme M_i ($i = 1, \ldots, k$) one first needs to locate the text elements in it. To this end, we consider a state-of-the-art deep learning-based text detection model, called TextFuseNet [25]. This processes M_i and produces the set $\mathbf{B}_i$ of detected text bounding boxes. We only keep boxes corresponding to whole words since the bounding boxes of letters are not useful for our task. Then, we apply Algorithm 1 to find the largest rectangle that contains no text and consider that part of M_i as its visual part V_i.

A rectangle R covers a fraction p of the initial image area, namely $A_R = p \cdot W \cdot H$, where W and H are width and height of the initial image. Thus, R can have width $\frac{\sqrt{p} \cdot W}{r}$ and height $\sqrt{p} \cdot H \cdot r$ to consider different aspect ratios r and preserve the rectangle's area A_R. Given p, we can determine the upper and lower bounds of r based on the image size i.e. $\frac{\sqrt{p} \cdot W}{r} \leq W$ and $\sqrt{p} \cdot H \cdot r \leq H$ that entail $\sqrt{p} \leq r \leq \frac{1}{\sqrt{p}}$. Similarly, given p and r we can determine the upper and lower bounds of the rectangle's center $c_R = (f_W \cdot W, f_H \cdot H)$:

$$f_W \cdot W - \frac{W\sqrt{p}}{2r} \geq 0 \tag{1}$$

$$f_W \cdot W + \frac{W\sqrt{p}}{2r} \leq W \tag{2}$$

$$f_H \cdot H - \frac{\sqrt{p} \cdot H \cdot r}{2} \geq 0 \tag{3}$$

$$f_H \cdot H + \frac{\sqrt{p} \cdot H \cdot r}{2} \leq H \tag{4}$$

that entail $\frac{\sqrt{p}}{2r} \leq f_W \leq 1 - \frac{\sqrt{p}}{2r}$ and $\frac{r\sqrt{p}}{2} \leq f_H \leq 1 - \frac{r\sqrt{p}}{2}$.

Algorithm 1: Visual part extraction

---

input : M_i
output : V_i
$W, H \leftarrow size(M_i)$;
$\mathbf{B}_i \leftarrow \text{TextFuseNet}(M_i)$;
$\mathcal{R} \leftarrow \emptyset$;
for $p \in [0.1, 0.9]$, $r \in \left[\sqrt{p}, \frac{1}{\sqrt{p}}\right]$, $f_W \in \left[\frac{\sqrt{p}}{2r}, 1 - \frac{\sqrt{p}}{2r}\right]$, $f_H \in \left[\frac{r\sqrt{p}}{2}, 1 - \frac{r\sqrt{p}}{2}\right]$ **do**
    $R \leftarrow \left(f_W \cdot W - \frac{W\sqrt{p}}{2r}, f_H \cdot H - \frac{\sqrt{p} \cdot H \cdot r}{2}, f_W \cdot W + \frac{W\sqrt{p}}{2r}, f_H \cdot H + \frac{\sqrt{p} \cdot H \cdot r}{2}\right)$;
    if $R \cap B = \emptyset$, $\forall B \in \mathbf{B}_i$ **then**
        $\mathcal{R} \leftarrow \mathcal{R} \cup \{(R, p)\}$;
$p_{max} \leftarrow \max\left(\{p \mid (R, p) \in \mathcal{R}\}\right)$;
$R_{V_i} \leftarrow RandomSample\left(\{R \mid ((R, p) \in \mathcal{R}) \wedge (p = p_{max})\}\right)$;
$V_i \leftarrow crop(M_i, R_{V_i})$;

Consequently, we consider 17 equidistant values for p and 10 equidistant values for r, f_W and f_H covering the corresponding ranges. So, $17,000$ rectangles R are created per M_i from which we initially select the non-overlapping with any $B \in \mathbf{B}_i$, and out of those select the ones with the maximum area. Finally, we randomly chose one rectangle R_{V_i} and crop the corresponding part from M_i in order to get its visual part V_i.

3.1.2 Utilization

We utilize the extracted visual parts V_i of image memes M_i as regular image instances in order to force the model's focus on the critical parts that discriminate them. More precisely, we consider the set $\mathbf{M} = \{M_i\}_{i=1}^k$ that contains image memes and the set $\mathbf{V} = \{V_i\}_{i=1}^k$ that contains the corresponding visual parts. In Figure 2, we illustrate the construction process of $\mathbf{M}$ and $\mathbf{V}$ through an example. To assess the extent to which VPU is useful we also conduct experiments mixing instances of $\mathbf{V}$ and web-scraped regular images for the construction of regular images class $\mathbf{R}$. Additionally, given the inherent text presence in image memes another crucial aspect to consider is the extent to which text presence in regular images affects the model's robustness. Hence, we consider two more sets as pools for $\mathbf{R}$ construction, namely $\mathbf{R}_p = \{R_i^p\}_{i=1}^k$ for web-scraped regular images with text presence and $\mathbf{R}_a = \{R_i^a\}_{i=1}^k$ for web-scraped regular images with text absence. The model's objective is to correctly classify the instances of the two sets, $\mathbf{M}$ and $\mathbf{R}$, with:

$$\mathbf{R} = \{V_i\}_{i=1}^{k \cdot (1 - P_W)} \cup \{R_i^p\}_{i=1}^{k \cdot P_W \cdot P_T} \cup \{R_i^a\}_{i=1}^{k \cdot P_W \cdot (1 - P_T)} \tag{5}$$

where P_W and P_T denote the fraction of web-scraped regular images out of the total number of regular images and the fraction of web-scraped regular images with text presence out of the total number of web-scraped regular images, respectively. In that way, $\mathbf{M}$ and $\mathbf{R}$ preserve the same cardinality k.

3.2 Model architecture

We propose Vision Transformer with Trainable Attention (ViTa), that augments ViT [19] by an attention module leveraging information from multiple processing stages. This approach was first successfully tested on CNNs [26].

3.2.1 ViT

The input image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ is reshaped into a sequence of flattened 2D patches $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where H is height, W is width, C is number of channels, P is patches' side length and $N = HW/P^2$ is number of patches. Then, $\mathbf{x}_p$ is linearly projected to D dimensions through a dense layer, a learnable *class* token is added to the sequence and learnable 1D position embeddings are added to the $N + 1$ tokens, resulting in the Transformer encoder inputs $\mathbf{z}_0 \in \mathbb{R}^{(N+1) \times D}$. Consequently, L Transformer encoder layers process the inputs to produce the final vector representations:

$$\mathbf{z}_l^{'} = \text{MSA}(\text{LN}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1} \tag{6}$$

$$\mathbf{z}_l = \text{MLP}(\text{LN}(\mathbf{z}_l^{'})) + \mathbf{z}_l^{'} \tag{7}$$

where MSA is Multiheaded Self-Attention with h heads [27], LN is Layer Normalization [28], MLP is a Multilayer Perceptron with two GELU [29] activated layers of $2 \cdot D$ and D number of units respectively and $l = 1, \ldots, L$. Finally,

Figure 3: Example images from the Hateful Memes (a, b, c) and the Google's Conceptual Captions with (d, e, f) and without (g, h, i) text presence.

a general representation $\mathbf{y}$, describing the whole image is extracted by passing $\mathbf{z}_L^0$, namely the *class* token's embedding after L Transformer encoder layers, through Layer Normalization:

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{8}$$

3.2.2 Attention module

ViT contains multiple self-attention layers in which the *class* token's embedding receives information from the patch embeddings of the same layer. However, it lacks an attention module that combines information from past layers that capture semantics of different granularity levels. To this end, we first compute a compatibility score between $\mathbf{y}$ and the patch embeddings of odd layers $\{\mathbf{z}_1^{1:N}\}, \{\mathbf{z}_3^{1:N}\}, \ldots, \{\mathbf{z}_n^{1:N}\}$ ($n = L - 1$ if L is even and $n = L$ if L is odd), by:

$$s_l^i = \; < \mathbf{v}, [\mathbf{y}; \mathbf{z}_l^i] > \tag{9}$$

where $i \in \{1, \ldots, N\}$, $l \in \{1, 3, \ldots, n\}$, $[\cdot; \cdot]$ denotes concatenation and $\mathbf{v} \in \mathbb{R}^{2D}$ is a learnable vector. The attention weights are calculated through softmax as:

$$a_l^i = \frac{\exp(s_l^i)}{\sum_{i=1}^{N} \exp(s_l^i)} \tag{10}$$

and the context vectors are simply the weighted average of the corresponding layer's patch embeddings $\mathbf{c}_l = \sum_{i=1}^{N} a_l^i \cdot \mathbf{z}_l^i$.

Table 1: Composition scenarios for regular images class ($\mathbf{R}$) construction based on P_W and P_T.

P_W	0%	33%				67%				100%			
P_T	0%	0%	33%	67%	100%	0%	33%	67%	100%	0%	33%	67%	100%

3.2.3 Classification module

The concatenation of all context vectors $\mathbf{c} = [\mathbf{c}_1; \mathbf{c}_3; \ldots; \mathbf{c}_n] \in \mathbb{R}^{D'}$ ($D' = ([n/2] + 1) \cdot D$, where $[\cdot]$ denotes integer part) is processed by three dense layers for the final prediction, the first two are GELU activated and the last has one sigmoid unit:

$$y = \text{sigmoid}(\mathbf{w}_3 \cdot \text{GELU}(\mathbf{w}_2 \cdot \text{GELU}(\mathbf{w}_1 \cdot \mathbf{c}))) \tag{11}$$

where $\mathbf{w}_1 \in \mathbb{R}^{2048 \times D'}$, $\mathbf{w}_2 \in \mathbb{R}^{1024 \times 2048}$, and $\mathbf{w}_3 \in \mathbb{R}^{1024}$.

4 Experimental setup

4.1 Datasets

To form a suitable dataset that can be used for the task of meme detection, we merge instances from existing datasets that contain image memes and regular images respectively.

For the image meme class $\mathbf{M}$, we consider the Hateful Memes Dataset [1]. This is a multimodal dataset for hateful meme detection containing 10,000 image memes. We do not take into account the nature of these memes, being hateful or not, in our analysis rather we use them all under the class of image memes. Figure 3 (a)-(c) illustrates three indicative examples.

For the regular images class $\mathbf{R}$, apart from the VPU methodology, explained in Section 3.1, that we apply on the Hateful Memes dataset to obtain $\mathbf{V}$, we also consider part of the widely-used web-scraped Google's Conceptual Captions dataset [2]. Specifically, we randomly sample images in order to construct $\mathbf{R}_p$ and $\mathbf{R}_a$ (see Section 3.1.2), with the text presence property automatically assessed through TextFuseNet [25] if it detects at least one text instance. Figure 3 presents three indicative examples with ((d)-(f)) and without ((g)-(i)) text presence.

4.2 Sample mixing and splitting

As a starting point for the image meme class $\mathbf{M}$, we consider the 10,000 instances of the Hateful Memes dataset. Then, we extract the visual parts of these image memes resulting in 9,984 images considered as regular to form the set $\mathbf{V}$. The mean area fraction across all $V_i \in \mathbf{V}$ is 64.3%. For the remaining 16 images the VPU algorithm was unable to find a rectangle with no overlap with text. Thus, for all four sets $\mathbf{M}$, $\mathbf{V}$, $\mathbf{R}_p$ and $\mathbf{R}_a$ we consider the same size of k=9,984 instances. To do so, we discard the same 16 image memes from $\mathbf{M}$ and sample from Google's Conceptual Captions dataset k=9,984 instances to form $\mathbf{R}_p$ and another k=9,984 instances to form $\mathbf{R}_a$ respectively.

For sample mixing, in order to form the class of regular images $\mathbf{R}$ we only need to determine P_W and P_T (see Equation 5). Also, to assess the impact of both VPU and text presence in the model's performance we consider several dataset composition scenarios $S_i=(P_W, P_T)$, with i=1,...,13, for $\mathbf{R}$, presented in Table 1.

Furthermore, we consider the same scenarios both on the training and test sets and experiment with *crossed scenarios* (S_i, S_j), e.g. the model is trained on $S_1=(P_W$=0%, P_T=0%) but evaluated on $S_{13}=(P_W$=100%, P_T=100%), resulting in 13·13=169 crossed scenarios to analyse. For sample splitting, we initially select 85% training, 5% validation and 10% test samples for each set $\mathbf{M}$, $\mathbf{V}$, $\mathbf{R}_p$, $\mathbf{R}_a$ and construct $\mathbf{R}$ per split. For $\mathbf{M}$ and $\mathbf{V}$ we consider the same index split in order not to include the visual part V_i in one split e.g. training, and the initial image meme M_i in another, e.g. test. The training and validation sets always derive from the same scenario S_i, while the evaluation is performed on all test scenarios S_j.

4.3 Competitive models

Meme detection is an understudied research topic. However, published studies exist such as leveraging ImageNet pre-trained ResNet features [22], combining ResNet, AlexNet and DenseNet fine-tuned on the task [23], and considering VGG16 and ResNet fine-tuned on the task [24]. These works consider text representations as additional inputs yet do not assess their impact. In this study, we only focus on the analysis of the visual signal for three reasons:

1. meme detection is a visually driven task as the critical information regarding existing text is the fonts' size, color, family and position rather than the actual text content

2. reliably recognizing text in image memes (let alone regular images) is a challenging task on its own, so an error-prone text recognition component would add another layer of unnecessary complexity

3. manual text recognition by humans, such as in the dataset used by the previously mentioned studies, is unrealistic for automatic meme detection.

Hence, we consider state of the art and baseline models from the image classification domain as competitive methods. More precisely, we consider VGG16 [30], ResNet50 [31], EfficientNetB5 [32], and ViT[7] [19].

4.4 Training details

For ViT and ViTa, we consider input size (H, W, C)=$(250, 250, 3)$ and patch size P=25 that entail N=100, projection dimension D=64, L=8 Transformer encoder layers and h=4 MSA heads. In total ViTa has 3.4M parameters trained from scratch as we also do for ViT. We consider the pre-trained on ImageNet weights for EfficientNetB5, ResNet50 and VGG16, discard the last layer, add a dense sigmoid activated layer to all and train only this very last layer.

All models are trained for 20 epochs with batch size 64, using the AdamW optimizer [33] with weight decay 1e-3 and the binary cross-entropy loss function. The first 10% of the iterations are linear warm-up steps with learning rate $\lambda(t)$ from 0 to 1e-3 which then decays as below:

$$\lambda(t) = \frac{1\text{e-}3}{(1 + d \cdot t \cdot 1.001^t)} \tag{12}$$

where t is the iteration and d=1e-3/20. We checkpoint our models based on validation accuracy. For ViT and ViTa, we preprocess input images first through normalizing to [0,1] and then through standardizing the values by subtracting the mean and dividing by the standard deviation per channel (computed on the training set). For the other models, we use the standard preprocessing pipeline provided by keras.

All models are evaluated on various test settings using the binary accuracy metric.

5 Results

5.1 Ablation study

In Table 2, we illustrate the performance of the proposed MemeTector model on all crossed scenarios between training and test settings. For easier interpretation of the results, we denote best performance per test scenario with bold letters. First, we observe that in almost all crossed scenarios MemeTector obtains high accuracy scores ranging between 89.0 and 97.8 (mean 94.98 and standard deviation 1.47). The S_9=(P_W=67%, P_T=100%) training scenario not only provides better performance on average and the highest accuracy values in most test scenarios (10 out of 13), but it preserves model robustness against all test scenarios as well. On the contrary, the highest variability in performance is observed in the S_2=(P_W=33%, P_T=0%) training scenario that provides high accuracy in test scenarios with P_T=0% yet low accuracy in test scenarios with text presence. The best performance is achieved in the (S_7, S_1) scenario, where MemeTector is trained on (P_W=67%, P_T=33%) and evaluated on (P_W=0%, P_T=0%). The worst performance is obtained with (S_1, S_{13}), where MemeTector is trained on (P_W=0%, P_T=0%) and evaluated on (P_W=100%, P_T=100%). This is expected, as feeding the model with samples of different nature makes it generalize better to easy scenarios such as S_1, while evaluating it on a dissimilar and harder setting compared to what it was trained on leads to lower performance. It is also remarkable that when evaluating on S_{10}-S_{13} scenarios, training on S_9 outperforms the models trained on S_{10}-S_{13}. This fact showcases the usefulness of the VPU methodology in training set construction.

In Table 3, we present comparative results with regards to VPU. The first four columns show the maximum model performance when trained without the use of VPU, and the last row shows the corresponding results of MemeTector when trained using the VPU. Additionally, the last column of Table 3 provides the average model performance. It is observed that VPU improves our model performance and also when incorporated, MemeTector performs better than competition in 3 out of 4 test scenarios as well as on average.

---

[7]For ViT we consider the same architecture as ViTa discarding only the attention module, which means that **y** is directly passed to Equation 11 replacing **c**. This of course changes the size of $\mathbf{w}_1$ which for ViT belongs to $\mathbb{R}^{2048 \times D}$.

Table 2: MemeTector's performance in terms of binary accuracy over all crossed scenarios. Best performance per test scenario is denoted with bold letters.

Test scenario

P_W	P_T	0%	33%				67%				100%			
		0%	**0%**	**33%**	**67%**	**100%**	**0%**	**33%**	**67%**	**100%**	**0%**	**33%**	**67%**	**100%**
0%	**0%**	94.1	93.9	93.4	93.1	92.7	93.6	92.9	92.1	90.7	93.1	92.2	90.7	89.0
33%	**0%**	96.8	96.5	95.5	94.5	94.2	95.6	93.9	93.1	91.9	95.2	92.9	91.4	89.6
	33%	96.5	96.7	95.7	95.1	95.0	96.4	95.1	94.4	94.0	96.2	94.8	93.9	92.7
	67%	96.4	96.2	95.7	95.4	95.2	96.0	95.5	94.7	93.9	95.3	94.6	93.7	92.3
	100%	96.6	96.6	96.2	95.6	95.4	96.5	95.4	94.9	94.3	95.9	94.9	93.8	92.6
67%	**0%**	96.4	96.5	95.9	95.5	95.2	96.2	95.2	94.7	94.2	96.2	94.8	93.9	92.7
	33%	**97.8**	**97.4**	**96.8**	96.3	96.2	**97.3**	96.0	95.9	95.3	96.7	95.5	94.7	93.8
	67%	96.4	96.3	95.9	95.7	95.4	96.4	95.7	95.0	94.9	96.3	95.4	94.5	94.0
	100%	97.2	97.1	**96.8**	**96.6**	**96.7**	96.6	**96.1**	**96.3**	**95.8**	**97.0**	**96.3**	**96.0**	**95.5**
100%	**0%**	94.5	95.2	94.5	93.9	93.3	95.5	94.3	93.7	92.4	95.5	93.9	92.8	91.4
	33%	96.0	96.0	95.8	95.5	95.3	96.2	95.9	95.4	95.2	96.7	95.8	95.4	94.7
	67%	94.5	95.2	95.0	94.8	94.7	95.7	95.5	95.1	94.7	95.0	95.2	94.9	94.
	100%	94.6	94.9	94.6	94.7	94.6	95.2	95.0	94.8	94.9	95.1	95.1	94.8	94.5

Training scenario

Table 3: Impact of VPU usage in terms of aggregated accuracy. $A(S_i,S_j)$ denotes accuracy of the model when trained on S_i and evaluated on S_j. In all cases $10 \leq i,j \leq 13$.

	VPU	$\max_i \left(A(S_i,S_{10})\right)$	$\max_i \left(A(S_i,S_{11})\right)$	$\max_i \left(A(S_i,S_{12})\right)$	$\max_i \left(A(S_i,S_{13})\right)$	$\sum_{i,j} A(S_i,S_j)/16$
VGG16	no	97.35	94.69	93.29	92.49	92.53
ResNet50	no	**98.05**	**96.30**	95.90	95.25	94.24
EfficientNetB5	no	96.55	94.04	93.89	93.99	92.41
ViT	no	96.00	95.75	95.20	94.44	94.24
ViTa (ours)	no	96.70	95.85	95.40	94.74	94.69
		$A(S_9,S_{10})$	$A(S_9,S_{11})$	$A(S_9,S_{12})$	$A(S_9,S_{13})$	$\sum_i A(S_9,S_i)/4$
ViTa (ours)	yes	97.00	**96.30**	**96.00**	**95.50**	**96.20**

5.2 Comparative study

Based on the analysis presented in Section 5.1, we employ our best configuration to compare with the competitive models presented in Section 4.3. The most robust training scenario that also provides best performance in most evaluation scenarios is S_9=(P_W=67%, P_T=100%). Hence, we consider this configuration for comparison with state of the art, namely the proposed MemeTector model architecture trained using 33% VPU created regular images and 67% web-scraped images that all contain text. Additionally, we train and evaluate VGG16, ResNet50, EfficientNetB5 and ViT on all crossed scenarios.

In Table 4, we present the average model performance on all crossed scenarios where the proposed methodology achieves the highest score. Moreover, Table 5 presents the fraction of crossed scenarios where MemeTector surpasses the performance of competitive models. Although the average difference between our model and the second to best, namely ViT is only 0.81%, MemeTector actually outperforms the latter in 87.57% of the crossed scenarios. Similarly, the proposed methodology outperforms the baselines in the majority of cases.

Table 4: Models' performance in terms of average binary accuracy across all crossed scenarios.

model	accuracy
VGG16	91.36%
ResNet50	92.31%
EfficientNetB5	90.05%
ViT	94.17%
MemeTector (ours)	**94.98%**

Table 5: Fraction of crossed scenarios that the MemeTector model surpasses competitive models.

model	fraction
VGG16	147/169=86.98%
ResNet50	109/169=64.50%
EfficientNetB5	162/169=95.86%
ViT	148/169=87.57%

5.3 Attention plots

Figure 4 illustrates attention plots from MemeTector's trainable attention mechanism. Given that this mechanism attends back to four layers, here we show the average attention weights across these four layers. As we observe in image memes the MemeTector attends mostly on the areas where text is present almost ignoring the background content. However, it does not attend at the whole sentences but only at a few seemingly random parts of them which means that analysing the font morphology (being the same throughout each sentence) provides sufficient information for accurate discrimination. Similarly, humans do not need to focus on every part of image memes that contains text to make an informed and accurate decision. In regular images, MemeTector focuses on the main depicted concepts as well as the text if it is present and prominent. Presumably, the reason for not classifying regular images with prominent text presence as image memes is the fonts' morphology and their position.

5.4 Use case on Twitter images

We also evaluate MemeTector on images from Twitter in order to assess its applicability on a practical use case. Specifically, we consider three relevant queries, namely "meme", "dankmemes", "memesdaily" and download 19,502 recent tweets, on 15 April 2022. Out of the collected tweets, only 6,256 contain an image: 2,071 from "meme", 1,660 from "dankmemes" and 2,525 from "memesdaily" query respectively. We download these images, drop the duplicates leaving 3,199 images for analysis and provide them as input to the MemeTector. Another seven of the downloaded images being gray-scale are also discarded because our model has been trained on RGB images only.

The model detected 1,342 memes (42%) and 1,850 regular images (58%). To provide quantitative results, we manually labeled the Twitter images, compared with MemeTector's predictions and assessed the model's accuracy. We found

Figure 4: Attention plots from MemeTector's trainable attention mechanism. All predictions are correct. The upper 10 are image memes, while the lower 10 are regular images. From regular images the first 5 have text absence while the last 5 have text presence.

TP=877, FP=396, TN=1,454 and FN=465, which amount to a 73% accuracy. Note that although the used queries are related to image memes there are regular images retrieved. The performance is reduced in the uncontrolled and noisy real-world data as expected but it can still be considered successful especially when considering the quite different characteristics of the images used for training the model.

In Figure 5, we illustrate indicative correct and erroneous predictions from the experiment on Twitter images. The correctly classified image meme in Figure 5a has a similar format to MemeTector's training samples, while the format of Figure 5b with text over objects or persons is not present in the training set but the model still recognizes it from the fonts morphology and background image semantics. There are many other meme formats not present in the training set that the model recognizes as well. The correctly classified regular images do not confuse the model, even though they contain text. This is due to the incorporation of text containing regular images at training. The misclassified regular images, contain meme-like overlay text and text at the top, which appears to mislead MemeTector. The first misclassified image meme (Figure 5g) contains only two numbers as overlay text which is not a common meme form in the training set. However, the second (Figure 5h) has similar structure to the training samples and MemeTector focuses on the fonts, but their morphology is different and that might be the reason for the detection miss.

(a) meme→meme   (b) meme→meme   (c) regular→regular   (d) regular→regular

(e) regular→meme   (f) regular→meme   (g) meme→regular   (h) meme→regular
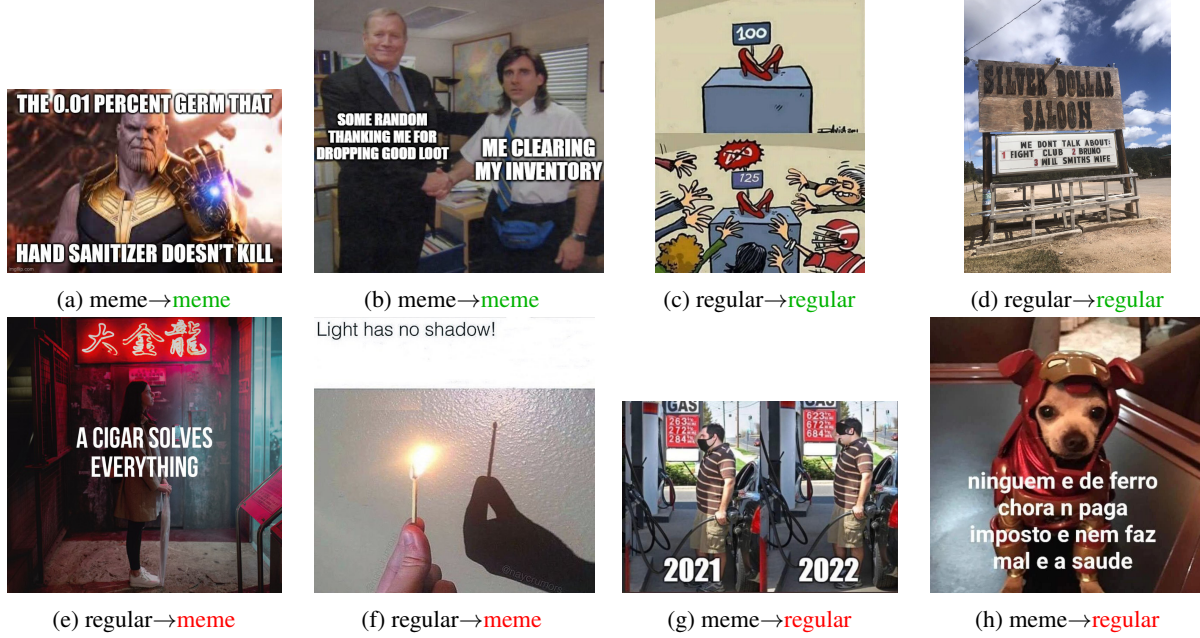
Figure 5: Twitter images classified by MemeTector: (a) through (d) are correct predictions, while (e) through (h) are wrong predictions. (a), (b), (g) and (h) are image memes, while (c), (d), (e) and (f) are regular images. The ground truth label is presented in the left of the arrow, and the MemeTector prediction on the right.

6 Conclusions

In this work, we address the problem of image meme detection. We introduce a novel artificial dataset creation process termed Visual Part Utilization (VPU) that first extracts the visual part of an image meme and then utilizes this new image as an instance of the regular images class. Additionally, we propose a trainable attention mechanism on top of a ViT architecture combining different levels of information granularity that led not only to improved performance but also to interpretability of the model's choices. The findings show that our model surpasses state of the art performance and also demonstrate the usefulness of incorporating VPU in the training of MemeTector. Finally, we validated the proposed methodology on a practical use case involving the retrieval and classification of image memes from Twitter.

Declarations

Data availability. All datasets used in this work are publicly available and have been properly referenced in the text.

References

[1] Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. The hateful memes challenge: Detecting hate speech in multimodal memes. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Annual Conference on Neural Information Processing Systems, NeurIPS 2020, Virtual Event, December 6-12, 2020*, pages 2611–2624. Curran Associates, Inc., 2020.

[2] P. Sharma, N. Ding, S. Goodman, and R. Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018*, pages 2556–2565. Association for Computational Linguistics, 2018.

[3] S. He, H. Yang, X. Zheng, B. Wang, Y. Zhou, Y. Xiong, and D. Zeng. Massive meme identification and popularity analysis in geopolitics. In *IEEE International Conference on Intelligence and Security Informatics, ISI 2019*, pages 116–121. IEEE, 2019.

[4] A. Olivieri, A. Noris, A. Theng, A. Berg, A. Rajhans, A. Holiavin, C. Arkenbout, G. D. Starita, K. Zheng, M. Bracale, M. Valli, N. Siddiqui, N. Welt, and S. Geboers. What is a meme, technically speaking? `https://wiki.digitalmethods.net/Dmi/WinterSchool2022WhatIsAMeme`, 2022.

[5] E. Segev, A. Nissenbaum, N. Stolero, and L. Shifman. Families and networks of internet memes: The relationship between cohesiveness, uniqueness, and quiddity concreteness. *Journal of Computer-Mediated Communication*, 20(4):417–433, 2015.

[6] S. Zannettou, T. Caulfield, J. Blackburn, E. De Cristofaro, M. Sirivianos, G. Stringhini, and G. Suarez-Tangil. On the origins of memes by means of fringe web communities. In *Proceedings of the Internet Measurement Conference, IMC 2018, Boston, MA, USA, 31 October - 2 November, 2018*, pages 188–202. Association for Computing Machinery, 2018.

[7] L. Xie, A. Natsev, J. R. Kender, M. Hill, and J. R. Smith. Visual memes in social media: Tracking real-world news in youtube videos. In *Proceedings of the 19th ACM International Conference on Multimedia, MM 2011, Scottsdale, AZ, USA, November 28 - December 1, 2011*, pages 53–62. Association for Computing Machinery, 2011.

[8] B. Dancygier and L. Vandelanotte. Internet memes as multimodal constructions. *Cognitive Linguistics*, 28(3):565–598, 2017.

[9] Tariq Habib Afridi, Aftab Alam, Muhammad Numan Khan, Jawad Khan, and Young-Koo Lee. A multimodal memes classification: A survey and open research issues. In Mohamed Ben Ahmed, İsmail Rakıp Karaș, Domingos Santos, Olga Sergeyeva, and Anouar Abdelhakim Boudhir, editors, *Innovations in Smart Cities Applications Volume 4*, pages 1451–1466. Springer International Publishing, 2021.

[10] Amalia Amalia, Amer Sharif, Fikri Haisar, Dani Gunawan, and Benny B Nasution. Meme opinion categorization by using optical character recognition (OCR) and naïve Bayes algorithm. In *2018 Third International Conference on Informatics and Computing (ICIC)*, pages 1–5, 2018.

[11] E. S. Smitha, S. Sendhilkumar, and G. S. Mahalaksmi. Meme classification using textual and visual features. In D. Hemanth and S. Smys, editors, *Computational Vision and Bio Inspired Computing, Lecture Notes in Computational Vision and Biomechanics*, volume 28, pages 1015–1031. Springer International Publishing, 2018.

[12] D. Gaurav, S. Shandilya, S. Tiwari, and A. Goyal. A machine learning method for recognizing invasive content in memes. In B. Villazón-Terrazas, F. Ortiz-Rodríguez, S. M. Tiwari, and S. K. Shandilya, editors, *Knowledge Graphs and Semantic Web, KGSWC 2020*, pages 195–213. Springer International Publishing, 2020.

[13] A. Aggarwal, V. Sharma, A. Trivedi, M. Yadav, C. Agrawal, D. Singh, V. Mishra, and H. Gritli. Two-way feature extraction using sequential and multimodal approach for hateful meme classification. *Complexity*, 2021:1–7, 2021.

[14] D. Kiela, H. Firooz, A. Mohan, V. Goswami, A. Singh, C. A. Fitzpatrick, P. Bull, G. Lipstein, T. Nelli, R. Zhu, N. Muennighoff, R. Velioglu, J. Rose, P. Lippe, N. Holla, S. Chandra, S. Rajamanickam, G. Antoniou, E. Shutova, H. Yannakoudakis, V. Sandulescu, U. Ozertem, P. Pantel, L. Specia, and D. Parikh. The hateful memes challenge: Competition report. In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 344–360. PMLR, 2021.

[15] Y. Zhou, Z. Chen, and H. Yang. Multimodal learning for hateful memes detection. In *IEEE International Conference on Multimedia Expo Workshops, ICMEW 2021, Virtual Event, July 5-9, 2021*, pages 1–6. IEEE, 2021.

[16] S. Khedkar, P. Karsi, D. Ahuja, and A. Bahrani. Hateful memes, offensive or non-offensive! In Ashish Khanna, Deepak Gupta, Siddhartha Bhattacharyya, Aboul Ella Hassanien, Sameer Anand, and Ajay Jaiswal, editors, *International Conference on Innovative Computing and Communications, ICICC 2022, Delhi, India, February 19–20, 2022*, pages 609–621. Springer Singapore, 2022.

[17] M. Miliani, G. Giorgi, I. Rama, G. Anselmi, and G. E. Lebani. DANKMEMES @ evalita 2020: The memeing of life: Memes, multimodality and politics. In V. Basile, D. Croce, M. Maro, and L. C. Passaro, editors, *Proceedings of the 7th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, EVALITA 2020, Torino, Italy, December 17th, 2020*. Accademia University Press, 2020.

[18] A. Sinha, P. Patekar, and R. Mamidi. Unsupervised approach for monitoring satire on social media. In *Proceedings of the 11th Forum for Information Retrieval Evaluation, FIRE 2019, Kolkata, India, December 12-15, 2019*, pages 36–41. Association for Computing Machinery, 2019.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[20] Shardul Suryawanshi, Bharathi Raja Chakravarthi, Mihael Arcan, and Paul Buitelaar. Multimodal meme dataset (MultiOFF) for identifying offensive content in image and text. In *Proceedings of the 2nd Workshop on Trolling, Aggression and Cyberbullying, TRAC 2020, Marseille, France, May, 2020*, pages 32–41. European Language Resources Association, 2020.

[21] Ingroj Shrestha and Jonathan Rusert. NLP_UIOWA at SemEval-2020 Task 8: You're not the only one cursed with knowledge - multi branch model memotion analysis. In *Proceedings of the 14th Workshop on Semantic Evaluation, SemEval 2020, Barcelona, Spain, December 12-13, 2020*, pages 891–900. International Committee for Computational Linguistics, 2020.

[22] S. Fiorucci. SNK @ DANKMEMES: Leveraging pretrained embeddings for multimodal meme detection. In V. Basile, D. Croce, M. Maro, and L. C. Passaro, editors, *Proceedings of the 7th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, EVALITA 2020, Torino, Italy, December 17th, 2020*. Accademia University Press, 2020.

[23] J. Setpal and G. Sarti. ArchiMeDe @ DANKMEMES: A new model architecture for meme detection. In V. Basile, D. Croce, M. Maro, and L. C. Passaro, editors, *Proceedings of the 7th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, EVALITA 2020, Torino, Italy, December 17th, 2020*. Accademia University Press, 2020.

[24] G. A. Vlad, G. E. Zaharia, D. C. Cercel, and M. Dascalu. UPB @ DANKMEMES: Italian memes analysis - employing visual models and graph convolutional networks for meme identification and hate speech detection. In V. Basile, D. Croce, M. Maro, and L. C. Passaro, editors, *Proceedings of the 7th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, EVALITA 2020, Torino, Italy, December 17th, 2020*. Accademia University Press, 2020.

[25] Jian Ye, Zhe Chen, Juhua Liu, and Bo Du. TextFuseNet: Scene text detection with richer fused features. In Christian Bessiere, editor, *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI 2020, Yokohama, Japan, January 7-15,2021*, pages 516–522. International Joint Conferences on Artificial Intelligence Organization, 2020.

[26] S. Jetley, N. A. Lord, N. Lee, and P. H. Torr. Learn to pay attention. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018*. OpenReview.net, 2018.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Annual Conference on Neural Information Processing Systems, NeurIPS 2017, Long Beach, CA, USA, December 4-9, 2017*, volume 30. Curran Associates, Inc., 2017.

[28] L. J. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint*, 2016.

[29] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint*, 2016.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*, 2015.

[31] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pages 770–778. IEEE, 2016.

[32] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, USA, June 9-15, 2019*, volume 97, pages 6105–6114. PMLR, 2019.

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.