NLP_UIOWA at SemEval-2020 Task 8: You're not the only one cursed with knowledge - Multi branch model memotion analysis

Ingroj Shrestha and **Jonathan Rusert**
Department of Computer Science
University of Iowa
Iowa City, IA, USA
{ingroj-shrestha,jonathan-rusert}@uiowa.edu

Abstract

We propose hybrid models (HybridE and HybridW) for meme analysis (SemEval 2020 Task 8), which involves sentiment classification (Subtask A), humor classification (Subtask B), and scale of semantic classes (Subtask C). The hybrid model consists of BLSTM and CNN for text and image processing respectively. HybridE provides equal weight to BLSTM and CNN performance, while HybridW provides weightage based on the performance of BLSTM and CNN on a validation set. The performances (macro F1) of our hybrid model on Subtask A are 0.329 (HybridE), 0.328 (HybridW), on Subtask B are 0.507 (HybridE), 0.512 (HybridW), and on Subtask C are 0.309 (HybridE), 0.311 (HybridW).

1 Introduction

Background. With the increasing social media culture, the sharing of internet memes on social media platforms has grown immensely in the recent years. *Meme* is defined as the unit of cultural information that replicates and transmits with reliability and fecundity (Linxia and Ziran, 2006). Memes are generally an image paired with text, and used to express an array of ideas (e.g. humor, sarcasm). Memes can be derived from pop cultures, previous experiences, or even more abstract ideas. Memes have become a large part of internet culture, and can preserve viewpoints specific to the community from where it originated. Memes can be used to express humor, embarrassment, hate, and even more emotions. The creativity of memes, however, carry a downside. Hateful or offensive memes can also be created and can lead to an increase in hate crimes (Heikkilä, 2017; Sabat et al., 2019). As with hateful language, several social media platforms have been working on policies to control such hateful and offensive memes while being careful not to hinder the creativity of users' expressions through memes (Kastrenakes, 2019; Hutchinson, 2020; Heilweil, 2020).

One of the major steps in controlling the sharing of hateful memes is being able to successfully detect them. Detection of offensive content on social media is an ongoing task. Current attempts at detecting offensive memes is limited. Furthermore, detecting offensive memes is more challenging than detecting offensive text as it involves both visual and language understanding while the latter only requires language understanding. Currently, many sites rely on human moderators to identify and remove memes that express emotions that violate the platform's policy. However, with the increasing use of memes across social media platforms, handpicking offensive memes would require larger human resource and can cause problems in scalability. Automated systems to identify the emotion of a meme could help in a first line defense/analysis of memes and could help reduce the load on human moderators. We already see this hybrid approach being employed for offensive and hateful text detection on several social media platforms (Yenala et al., 2018; Zhang et al., 2018), so it is only natural to extend this approach to classifying memes as well.

In order to address the problem of detecting offensive memes as well as classifying types of memes in general, a group of organizers created a community driven task, SemEval 2020 Task 8 (*Memotion Analysis*). Sharma et al. (2020) brings attention of the research community towards automatic meme emotion

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 891–900
Barcelona, Spain (Online), December 12, 2020.

analysis and allows for the examination of multiple approaches. We approach this problem with a hybrid architecture of Convolutional Neural Network (CNN) for image classification and a Bidirectional Long Short Term Memory (BLSTM) neural network for text classification.

2 Proposed Approach

Our goal is to capture informative features from both images and text to help the system in its classification. To increase the usefulness of both image and text, we first fine tune a CNN on image classification and BLSTM on text classification separately, then use a validation set to score their respective performances. A CNN was chosen as CNNs have shown strong performance in image classification (Xin and Wang, 2019). Likewise, BLSTMs have shown strong performance on text classification, therefore we chose this for our framework[1]. We finally combine the CNN and BLSTM models using a hybrid approach.

2.1 Text classification

To classify the text, we implement a Bidirectional Long Short Term Memory (BLSTMs) with pretrained word embeddings. Figure 1 represents the BLSTM architecture we used.

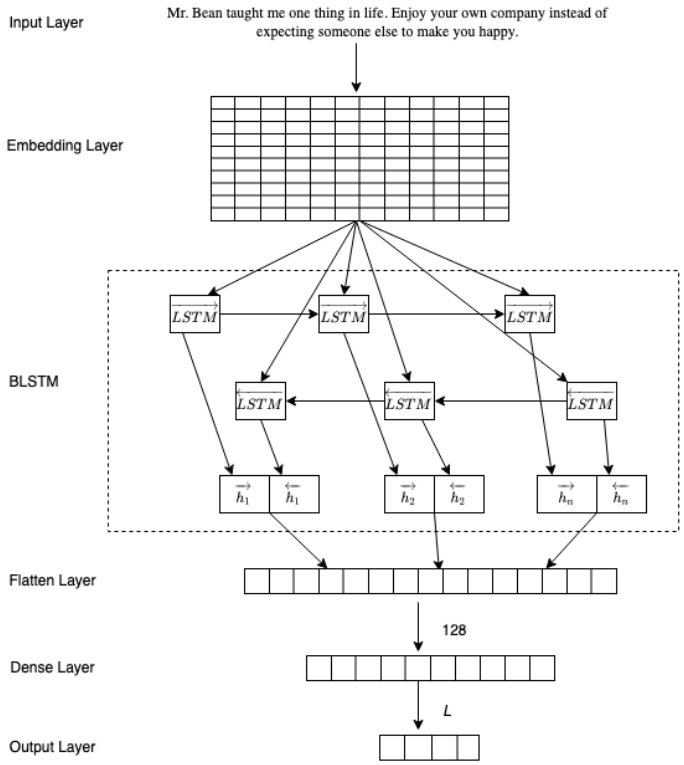

Figure 1: BLSTM architecture

Embedding layer. The embedding layer converts the input text (*input layer*) to a real valued vector using pre-trained word embeddings[2] of dimension 200. The pre-trained word embeddings are obtained from Glove (Pennington et al., 2014) word embeddings trained on English Gigaword[3] and Wikipedia data. For the words not in the vocabulary, we randomly initialed the word embedding. After preprocessing, we find the longest text size (V). The input text that is shorter than the longest text size is padded with zeros at the end. Next, the *embedding layer* output is fed into BLSTM layer.

BLSTM Layer. Long Short-Term Memory (LSTMs) build on top of traditional RNNs, by adding 4 gates through which input travels: *ignoring (i)*, *memory(c)*, *forgetting (f)*, and *selection (o)*. These

[1]We also ran preliminary tests on Logistic Regression models for both image and text, but were outperformed by the CNN and BLSTM.

[2]https://nlp.stanford.edu/projects/glove/

[3]https://catalog.ldc.upenn.edu/LDC2011T07

gates aim to help the system remember the important parts of input, while forgetting the non-relevant parts. *Ignoring* gates out the non relevant information from predictions. To add in longer term memory, a *memory* mechanism is applied. Tied with the memory gate, the *forgetting* mechanism is used to help to filter irrelevant previous prediction with old memory. *Selection* gate looks at possible predictions and gates them before allowing the system to make a final prediction. The gates are represented by the following equations:

$$\text{LSTM} : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l$$
$$\begin{pmatrix} i_t^l \\ f_t^l \\ o_t^l \\ g_t^l \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$
$$c_t^l = f_t^l \odot c_{t-1}^l + i_t^l \odot g_t^l$$
$$h_t^l = o_t^l \odot \tanh(c_t^l)$$

where *sigm*, and *tanh* are sigmoid and tanh activation functions,respectively. $\odot$ represents element-wise multiplication, h_t^l represents the hidden state at time step *t* for layer *l*, and h_t^{l-1} is the output from embedding layer $\in \mathbb{R}^{V*200}$ for $(l = 1)$.

A BLSTM, a 2 directional LSTM which reads the sentence in normally (forward direction) i.e., $\overrightarrow{LSTM}$, and reads the sentence in backward direction i.e.,$\overleftarrow{LSTM}$. The final learned representation of text from BLSTM layer is $\overrightarrow{LSTM} \oplus \overleftarrow{LSTM}$, where $\oplus$ refers to concatenation.

Dense Layer. The output of BLSTM layer is flattened and fed to a *dense layer* of size 128 and then fed to an output layer of size *L* with softmax activation, where *L* is the number of classes *c*.

2.2 Image classification

We implemented a Convolutional Neural Network (CNN) for the image classification task. Figure 2 represents CNN architecture we used.

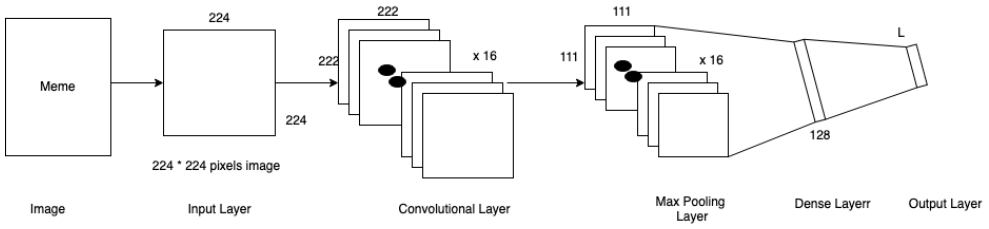

Figure 2: CNN architecture

Input Layer. The first layer of CNN network is the *input layer*, which takes images, resizes them to a dimension of *w* * *w*, where *w* = 224. We then fed the image to the *convolutional layer* for feature extraction.

Convolutional Layer. In the convolutional layer, we use a *k* * *k* filter with a stride of *s=1* and zero padding *p=0* to produce a feature map of size $\left(\frac{w-k+2*p}{s} + 1\right)$, where *k=3*. The convolutional layer uses $n_{ch} = 16$ output channels. So, the final output of convolutional layer $(conv_{out})$ is $n_{ch} * \left(\frac{w-k+2*p}{s} + 1\right) * \left(\frac{w-k+2*p}{s} + 1\right)$.

Max Pooling Layer. A max pooling of size $j * j$ is applied to the output from *convolutional layer*, where $j = 2$. The resulting output is $\left(n_{ch} * \frac{conv_{out}}{j} * \frac{conv_{out}}{j}\right)$.

Dense Layer. The output from *max pooling* is flattened and fed into a *dense layer* consisting 128 neurons with ReLU activation. Finally, the output is fed to the output layer of size *L*. The output layer uses softmax activation function to provide the probability distribution s_p for each class prediction *(y)*.

2.3 Hybrid approach

In order to balance text and visual features for prediction, we use a hybrid approach. The hybrid approach is shown in Figure 3. In the hybrid approach, we give each system, BLSTM and CNN, a weight for their predictions, α and β respectively.

Hybrid Model Weighted (HybridE). In this approach, we set $\alpha = \beta = 1$. We obtain probability distribution for each class using softmax activation. We then compute element-wise sum of probability distribution of each class obtained from two architectures (CNN and BLSTM). Finally, we take argmax of combined probability distribution to predict final class for a meme.

Hybrid Weighted Average (HybridW)[4]. In this approach, the contribution (i.e., softmax distribution) of each class is weighted by the performance (macro F1) of models α (BLSTM), and β (CNN). The performance of models are evaluated on the *validation* set (described further in section 3.2). Finally, we take argmax of the weighted probability distribution to obtain a final class for a meme.

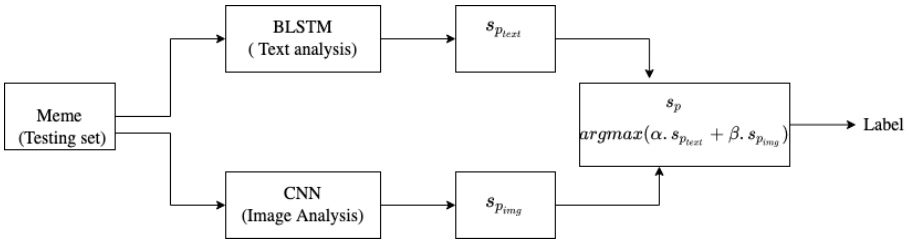

Figure 3: Hybrid

2.4 Hyper-parameters/Tuning.

We experimented with different epochs (10, 15, 20) and batch sizes (64, 100, 150). We found an epoch of 10 and batch size 64 (text) and 100 (image) are optimal. We use a dropout of 0.2 (CNN) and 0.5 (BLSTM) in penultimate layer to handle the issue of model overfitting. For BLSTM, we use a hidden size (*n*) of 64. The model learns optimal parameters minimizing cross-entropy loss shown in equation 1a ($L = 2$), equation 1b ($L > 2$). We use Adam optimizer with a learning rate of 0.001. We implemented the system using PyTorch[5].

$$-ylogs_p + (1-y)log(1-s_p) \tag{1a}$$

$$\sum_{c=1}^{L} y_{o,c} log(s_{p_{o,c}}) \tag{1b}$$

3 Subtasks and Dataset

SemEval 2020 Task 8 involves an overall task of analysis of memes, which is divided into three subtasks – Sentiment analysis (Subtask A), Humor classification (Subtask B), and Scale of semantic classes (Subtask C).

3.1 Subtasks Description

Subtask A. Subtask A requires a system to identify if a meme is positive, negative, or neutral (multi-class classification).

Subtask B. Subtask B involves identification of humor expressed in meme (sarcastic, humorous, offensive, motivational). This involves four binary classifications, where each of the humor is classified as being present (e.g., *sarcastic*), or absent/not (e.g., *not sarcastic*). Overall, it is multi-label classification task.

Subtask C. Subtasks C involves multi-class, multi-label classification. This is an extension to Subtask B, where a system requires to quantify the extent to which a particular effect is being expressed (scale of semantic) in a meme. With one exception (motivational), the type of humor expressed is scaled from 0

[4]The hybrid weighted average was performed after the Evaluation period ended.

[5]https://pytorch.org/

to 4 – not (0), slightly (1), mildly (3), and very (4). Motivational is categorized as *motivational* or *non motivational*.

Architecture for subtasks. For each subtask, we use the same architecture (corresponding architecture for text and image analysis – Section 2), changing only the size of L (the number of classes). For Task A, we use $L = 3$. For Task B, we perform four binary classifications with $L = 2$ for each humor expressed. For Task C, we perform four multi-class classifications with $L = 4$ for each semantic class.

3.2 Dataset Description

Training set. The organizers provided a training dataset for development of automatic meme analysis. The training sets consist of 6992 memes. Each meme consists of five classifications (semantic classes) - humor, sarcasm, offensive, motivational, and overall sentiment, with scale of semantic classes. These classifications corresponds to subtasks (Section 3.1). A distribution of these sets is found in Table 1.

Testing set. The testing set consists of 1878 memes. The text was missing for several memes in the testing set, so we added these in manually by transcribing from the provided image.

sentiment	
positive	4160
negative	631
neutral	2201

(a) Subtask A

semantic	present	not
humorous	5337	1649
sarcastic	5443	1543
offensive	4277	2709
motivational	2465	4521

(b) Subtask B

semantic	not(0)	slightly(1)	mildly(2)	very(3)
humorous	1649	2452	2236	649
sarcastic	1543	3503	1546	394
offensive	2709	2591	1465	221

(c) Subtask C

Table 1: Class frequency distribution (Training set)

3.3 Training set Evaluation

To test our approach, we leveraged the training set, and performed train-validation split (80%-20%) to find macro and micro F1 scores. We first describe the steps employed to work with the training data, then give the results on the set.

Data Preprocessing. Though we presumed that the provided dataset would be set up to accommodate each subtask in SemEval, this was not the case. This caused us to employ some preprocessing steps to make the data more in line with the aforementioned subtasks. We remove six instances from the training set as the text was not available for those instances. Similarly, when working with the CNN, we found an image *got_GOT-Meme-9* failed to load, being corrupt. So, we remove the image from the training set.

Assigning Labels. Recall that Subtask A is a multi-class classification problem requiring for the memes to be classified into positive, negative or neutral. The training dataset contained 5 labels: very_positive, positive, neutral, negative, very_negative. We reduced the number of labels by collapsing the very_positive memes into the positive category, and followed the same with the very_negative memes to meet classification requirements in Subtask A. Again as previously noted, in Subtask B, a given meme can have one of multiple binary classification labels. For example, a meme can be humorous or non humorous. The same meme can be sarcastic or non sarcastic, offensive or non-offensive and motivational or non motivational. Each of these binary classification problem in Subtask B has multiple labels except for the motivational classification, which is why for the first three classifications task we combined the labels to fit them for binary classification. We combined funny, very funny, and hilarious into *humorous*, general, twisted meaning, and very twisted into *sarcastic* and slight, very offensive, and hateful offensive into *offensive*. For Subtask C, the labels required no conversion.

3.4 Training set Results

We obtain results (Table 2) on gold standard training set using the aforementioned train-validation split.

Subtask A. CNN outperforms BLSTM (Table 2a) in both performance metrics (8.1% in macro F1 and 12% in micro F1). So, HybridE, which assigns equal weights to BLSTM and CNN, shows a reduction in overall performance, achieving a macro F1 of 0.3168 and a micro F1 of 0.4850.

Subtask B. BLSTM outperforms CNN by 12.3% in macro F1 (Table 2b). On the other hand, CNN outperforms BLSTM by 3.4% in micro F1. Due to equal weights to BLSTM and CNN, macro F1 for

HybridE is reduced, while micro F1 shows a slight improvement. The HybridE achieves a macro F1 and a micro F1 of 0.4753 and 0.6197, respectively.

Subtask C. Similar to Subtask B, BLSTM and CNN show a trade-off in the two performance metrics. Overall, the hybrid equal-weighted model achieves a macro F1 of 0.2858 and micro F1 of 0.4288.

System	Macro F1	Micro F1
BLSTM	0.3185	0.4535
CNN	0.3443	0.5079
HybridE	0.3168	0.4850

(a) Subtask A

System	Macro F1	Micro F1
BLSTM	0.5216	0.5986
CNN	0.4642	0.6187
HybridE	0.4753	0.6197

(b) Subtask B

System	Macro F1	Micro F1
BLSTM	0.3153	0.3838
CNN	0.2831	0.4278
HybridE	0.2858	0.4288

(c) Subtask C

Table 2: Results on Training set

3.5 Testing set Results.

The result for our proposed approach's performance on the Testing set for three subtasks are shown in Table 3. On the testing set, the proposed hybrid model (HybridE) achieves a macro F1 score of 0.3287, 0.5073, and 0.3087 on Subtask A, B, and C, respectively. The HybridE model outperforms baseline (in macro F1) in all of the subtasks (11.11% points (Subtask A), 0.71% points (Subtask B), 0.78% points (Subtask C)). On a similar line to HybridE, the weighted hybrid model (HybridW) outperforms baselines (provided by organizer) in both metrics.

We also can see that the HybridE favors BLSTM in macro F1 (performance is similar to BLSTM) and CNN in micro F1 (performance is similar to CNN). The weighted average approach (HybridW) shows little or no improvement over HybridE approach.

Subtask A. In contrast with the performance trend in Training set, BLSTM outperforms CNN by 4% in macro F1, while CNN outperforms BLSTM by 5% in micro F1 in Testing set (Table 3a). The HybridE favoring BLSTM, in terms of macro F1, shows an F1 score of 0.3287, which is similar to BLSTM. Likewise, HybridE achieves micro F1 of 0.5266 (similar performance to CNN). The HybridW shows no or little improvement in macro F1 and micro F1, respectively. HybridE performance (in macro F1) is 7.3% lower than the top system.

Subtask B. As with the Training set, BLSTM and CNN perform similarly on this subtask. On overall, the hybrid model (HybridE) achieves macro F1 and micro F1 of 0.5073 and 0.6330 respectively (Table 3b). The HybridW shows a slight improvement in macro F1, but no improvement on micro F1. HybridE performance (in macro F1) is similar to the top system.

Subtask C. Similar to Training set, BLSTM outperforms CNN in macro F1 by 5%, while CNN outperforms BLSTM in micro F1 by 8.5% (Table 3c). As mentioned earlier, HybridE favors BSLTM in macro F1, while it favors CNN in micro F1, achieving 0.3087 macro F1 (similar performance to BLSTM) and 0.4016 micro F1 (similar performance to CNN). The HybridW shows a slight improvement in both performance metrics. HybridE performance (in macro F1) is 4.3% lower than the top system.

System	Macro F1	Micro F1
Baseline	0.2176	0.3077
Top system	0.3547	0.4872
BLSTM	0.3287	0.4915
CNN	0.3161	0.5176
HybridE	0.3287	0.5234
Δ %*	- 7.3%	+ 7.4%
HybridW	0.3284	0.5266
Δ %*	- 7.4%	+ 8.1%

(a) Subtask A

System	Macro F1	Micro F1
Baseline	0.5002	0.5687
Top system	0.5183	0.6145
BLSTM	0.5085	0.5994
CNN	0.4874	0.6443
HybridE	0.5073	0.6330
Δ %*	- 2.1%	+ 3%
HybridW	0.5116	0.6286
Δ %*	- 1.3%	+ 2.3%

(b) Subtask B

System	Macro F1	Micro F1
Baseline	0.3009	0.3328
Top system	0.3225	0.3780
BLSTM	0.3122	0.3721
CNN	0.2968	0.4038
HybridE	0.3087	0.4016
Δ %*	- 4.3%	+ 6.2%
HybridW	0.3111	0.4024
Δ%*	- 3.5%	+ 6.5%

(c) Subtask C

Table 3: Results on Testing set (*Percentage change with respect to top system)

3.6 Class wise performance

Training set class wise results. The class wise performances for Subtask B and Subtask C on Training set are shown in Table 4 and Table 5, respectively (Note that since Subtask A only consists of one multi-class problem, the results are the same as shown in Table 2a). BLSTM performs better in some classes,

while CNN perform better in other classes. For example, BLSTM outperforms CNN in the class *Sarcasm* by 11% (Table 5a). However, CNN outperforms BLSTM in the class *Humor* by 3.2% (Table 5a). These results acted as a motivation for our weighted hybrid approach (HybridW).

System	Humor	Sarcasm	Offense	Motivation
BLSTM	0.542 6	0.498 9	0.527 9	0.516 8
CNN	0.475 1	0.494 4	0.476 3	0.410 8
HybridE	0.536 6	0.492 5	0.485 8	0.386 3

(a) Macro F1

System	Humor	Sarcasm	Offense	Motivation
BLSTM	0.656 7	0.650 9	0.541 7	0.545 1
CNN	0.613 2	0.733 5	0.524 4	0.603 7
HybridE	0.683 8	0.648 1	0.517 2	0.629 5

(b) Micro F1

Table 4: Class wise performance (validation set) for Subtask B

System	Humor	Sarcasm	Offense	Motivation
BLSTM	0.239 7	0.259 3	0.245 4	0.516 8
CNN	0.247 4	0.234 0	0.240 1	0.410 8
HybridE	0.251 5	0.257 6	0.247 9	0.386 3

(a) Macro F1

System	Humor	Sarcasm	Offense	Motivation
BLSTM	0.283 3	0.379 1	0.327 6	0.545 1
CNN	0.293 3	0.444 9	0.369 1	0.603 7
HybridE	0.297 6	0.440 6	0.347 6	0.629 5

(b) Micro F1

Table 5: Class wise performance (validation set) for Subtask C

Testing set class wise results. The class wise performances for Subtask B and Subtask C on Testing set are shown in Table 6 and Table 7, respectively. We can see a drop in macro F1 for some classes on combining the performances of BLSTM and CNN. For example, the macro F1 drops for the class *Sarcasm* in Subtask B (Table 6a). However, we also can see that hybrid approaches help improve the overall class wise performance for some classes. For example, macro F1 on the class *Offense* is 0.4928 and 0.4898 for BLSTM and CNN, respectively (Table 6a). When combining the BLSTM and the CNN results, there is an improvement in macro F1 score (HybridE: 2.3% over BLSTM and 3% over CNN, HybridW: 3.9% over BLSTM, and 4.6% over CNN). We can see similar observations for the class *Motivation* for Subtask B (Table 6a), and Subtask C (Table 7a). Overall, the effect of hybrid approach is somewhat mixed with respect to macro F1. We can see similar mixed performance with respect to micro F1 also (Table 6b and Table 7b).

System	Humor	Sarcasm	Offense	Motivation
BLSTM	0.5120	0.5265	0.4928	0.5028
CNN	0.4642	0.4915	0.4898	0.5042
HybridE	0.5121	0.5039	0.5043	0.5090
HybridW	0.5106	0.5109	0.5122	0.5127

(a) Macro F1

System	Humor	Sarcasm	Offense	Motivation
BLSTM	0.6289	0.6459	0.5639	0.5591
CNN	0.7178	0.7412	0.5800	0.5383
HybridE	0.7077	0.7109	0.5655	0.5479
HybridW	0.6922	0.7093	0.5634	0.5495

(b) Micro F1

Table 6: Class wise performance (Testing set) for Subtask B

System	Humor	Sarcasm	Offense	Motivation
BLSTM	0.2461	0.2384	0.2617	0.5028
CNN	0.2092	0.2406	0.2332	0.5042
HybridE	0.2441	0.2435	0.2384	0.5090
HybridW	0.2444	0.2478	0.2400	0.5127

(a) Macro F1

System	Humor	Sarcasm	Offense	Motivation
BLSTM	0.2918	0.3088	0.3285	0.5591
CNN	0.3387	0.3988	0.3392	0.5383
HybridE	0.3142	0.4004	0.3440	0.5479
HybridW	0.3147	0.4004	0.3450	0.5495

(b) Micro F1

Table 7: Class wise performance (Testing set) for Subtask C

4 Discussion

Trade off in the performance of BLSTM and CNN. As seen in Table 3, BLSTM shows better performance in macro F1, while CNN shows better performance in micro F1. Due to this, the hybrid model's performance is compromised.

Comparison of HybridE and HybridW. Overall, HybridW performs slightly better than HybridE (Subtask B and Subtask C) in terms of macro F1. Since there is no significant improvement, it is unclear that

adding extra weight really helps better to incorporate trade-off of BLSTM and CNN to capture more informative features.

Class imbalance and effect on performance metric. Macro F1 average computes F1 for each class and take average by treating all class equally. However, micro F1 average aggregates the contribution of each class, and then computes the average F1. From Table 1, we can see that the distribution of class is not balanced for each subtask. So, micro F1 scores are larger than macro F1 scores for each subtask (Table 3) since predictions favor the larger class.

Failure of transfer learning. For text analysis, we tried pre-trained BERT (Devlin et al., 2018). For image analysis, we tried VGG16 (Simonyan and Zisserman, 2014), and ResNet18 (He et al., 2016). We removed the last layer from each model and added a custom dense layer to fit the subtasks. We then fine-tune using the train set. However, each model overfitted. The overfitting issue might be due to complex architecture of pre-trained models, or due to failure to learn task specific features provided small train set.

5 Related Work

Since the multimodal social media content has seen a steady increase in the recent years, deriving the intended meaning from this content by establishing the connection between the image and the text has seen an increase in research. A limited research has been done in extracting meaning from social media images and texts, which includes identification of the humor, offensiveness or sentiment expressed in image, text or meme.

Humor Classification. Detection of humor in image and text has been approached by several teams in recent years. Chandrasekaran et al. (2016) analyze the humor present in abstract scenes at the scene-level and the object level and detect different types of humor depicted in the scenes. Tsakona (2009) mentions that the meaning and humor in a cartoon is expressed through verbal and visual mode. In order to capture the humor expressed in the cartoon, one has to pay attention to all the verbal and visual details of the cartoon.

Offense Classification. Recently, there has been a growing interest in identifying the offensive language of social media data. Chen et al. (2012) presents user-level offensive language detection on social media. This architecture uses features such as the user's writing style, structure, and specific cyberbullying for detecting offensiveness in the text. Wiegand et al. (2018) proposed a GermEval task for classifying offensive language as offensive, or other, and then further classify the offensive tagged language. More recently, Zampieri et al. (2019) ran a shared task, OffensEval, on detecting different classes of offensive text.

Sentiment Classification. Sentiment detection in the image or text has also seen a greater focus on research. Wang and Li (2015) mention that the accurate sentiment detection from internet images requires connection between visual and textual feature. They presented the Unsupervised Sentiment Analysis (USEA) framework to perform sentiment analysis on social media images in an unsupervised approach using both features mentioned earlier. Borth et al. (2013) present a method built upon web mining to automatically construct a visual detector library to detect Adjective Noun Pair in an image, which they used to identify the sentiment from visual content.

Sarcasm Classification. Though sarcasm is not always easy to identify online, researchers have attempted this with various approaches. Joshi et al. (2017) present a survey on various methods for automatic sarcasm detection. They link to many sarcasm papers which include the sarcasm datasets used (e.g. (Barbieri et al., 2014; González-Ibánez et al., 2011)) as well as sarcasm detection approachs leveraged (e.g. (Reyes and Rosso, 2014; Rajadesingan et al., 2015)).

6 Conclusion

We analyzed texts and images from memes using BLSTM and CNN, respectively. We then propose two hybrid approaches HybridE (equal weightage to prediction probability from BLSTM and CNN) and HybridW (weighted average based on performance of BLSTM and CNN) to identify humor, offensiveness, and sentiment expressed in memes. HybridE performs better overall than the individual systems,

however, HybridW shows a little or no improvement over HybridE.

Limitations and Future direction. We trained models for text and image analysis separately. Perhaps, we can feed the text output and image output into another dense layer (in a neural net). This approach might catch some features the first missed. Also, since the deep learning model shows better performance on a large data set, we could explore the problem on a larger data set.

References

Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. 2014. Italian irony detection in twitter: a first approach. In *The First Italian Conference on Computational Linguistics CLiC-it*, volume 28.

Damian Borth, Rongrong Ji, Tao Chen, Thomas Breuel, and Shih-Fu Chang. 2013. Large-scale visual sentiment ontology and detectors using adjective noun pairs. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 223–232.

Arjun Chandrasekaran, Ashwin K Vijayakumar, Stanislaw Antol, Mohit Bansal, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2016. We are humor beings: Understanding and predicting visual humor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4603–4612.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 71–80. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in Twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Niko Heikkilä. 2017. Online antagonism of the alt-right in the 2016 election. *European journal of American studies*, 12(12-2).

Rebecca Heilweil. 2020. Facebook is flagging some coronavirus news posts as spam. `https://www.vox.com/recode/2020/3/17/21183557/coronavirus-youtube-facebook-twitter-social-media`.

Andrew Hutchinson. 2020. Twitter Will Increase Its Use of Automation Tools as It Looks to Ensure Accuracy in COVID-19 Discussion. `https://www.socialmediatoday.com/news/twitter-will-increase-its-use-of-automation-tools-as-it-looks-to-ensure-acc/574263/`.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)*, 50(5):1–22.

Jacob Kastrenakes. 2019. Twitter says it now removes half of all abusive tweets before users report them. `https://www.theverge.com/2019/10/24/20929290/twitter-abusive-tweets-automated-removal-earnings-q3-2019`.

Chen Linxia and He Ziran. 2006. Analysis of memes in language. *Foreign Language Teaching and Research*, 2.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the eighth ACM international conference on web search and data mining*, pages 97–106.

Antonio Reyes and Paolo Rosso. 2014. On the difficulty of automatically detecting irony: beyond a simple case of negation. *Knowledge and Information Systems*, 40(3):595–614.

Benet Oriol Sabat, Cristian Canton Ferrer, and Xavier Giro-i Nieto. 2019. Hate Speech in Pixels: Detection of Offensive Memes towards Automatic Moderation. *arXiv preprint arXiv:1910.02334*.

Chhavi Sharma, Deepesh Bhageria, William Paka, Scott, Srinivas P Y K L, Amitava Das, Tanmoy Chakraborty, Viswanath Pulabaigari, and Björn Gambäck. 2020. SemEval-2020 Task 8: Memotion Analysis-The Visuo-Lingual Metaphor! In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain, Sep. Association for Computational Linguistics.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Villy Tsakona. 2009. Language and image interaction in cartoons: Towards a multimodal theory of humor. *Journal of Pragmatics*, 41(6):1171–1188.

Yilin Wang and Baoxin Li. 2015. Sentiment analysis for social media images. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1584–1591. IEEE.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.

Mingyuan Xin and Yong Wang. 2019. Research on image classification model based on deep convolution neural network. *EURASIP Journal on Image and Video Processing*, 2019(1):40.

Harish Yenala, Ashish Jhanwar, Manoj K Chinnakotla, and Jay Goyal. 2018. Deep learning for detecting inappropriate content in text. *International Journal of Data Science and Analytics*, 6(4):273–286.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European semantic web conference*, pages 745–760. Springer.