An AutoML-based Approach to Multimodal Image Sentiment Analysis

Vasco Lopes*, António Gaspar*, Luís A. Alexandre*, João Cordeiro†
*NOVA LINCS, Universidade da Beira Interior
†LIAAD, INESC TEC – Institute for Systems and Computer Engineering, Technology and Science
‡HULTIG – Centre of Human Language Technology and Bioinformatics
{vasco.lopes, antonio.pedro.gaspar, luis.alexandre, jpcc}@ubi.pt

Abstract—**Sentiment analysis is a research topic focused on analysing data to extract information related to the sentiment that it causes. Applications of sentiment analysis are wide, ranging from recommendation systems, and marketing to customer satisfaction. Recent approaches evaluate textual content using Machine Learning techniques that are trained over large corpora. However, as social media grown, other data types emerged in large quantities, such as images. Sentiment analysis in images has shown to be a valuable complement to textual data since it enables the inference of the underlying message polarity by creating context and connections. Multimodal sentiment analysis approaches intend to leverage information of both textual and image content to perform an evaluation. Despite recent advances, current solutions still flounder in combining both image and textual information to classify social media data, mainly due to subjectivity, inter-class homogeneity and fusion data differences. In this paper, we propose a method that combines both textual and image individual sentiment analysis into a final fused classification based on AutoML, that performs a random search to find the best model. Our method achieved state-of-the-art performance in the B-T4SA dataset, with 95.19% accuracy.**

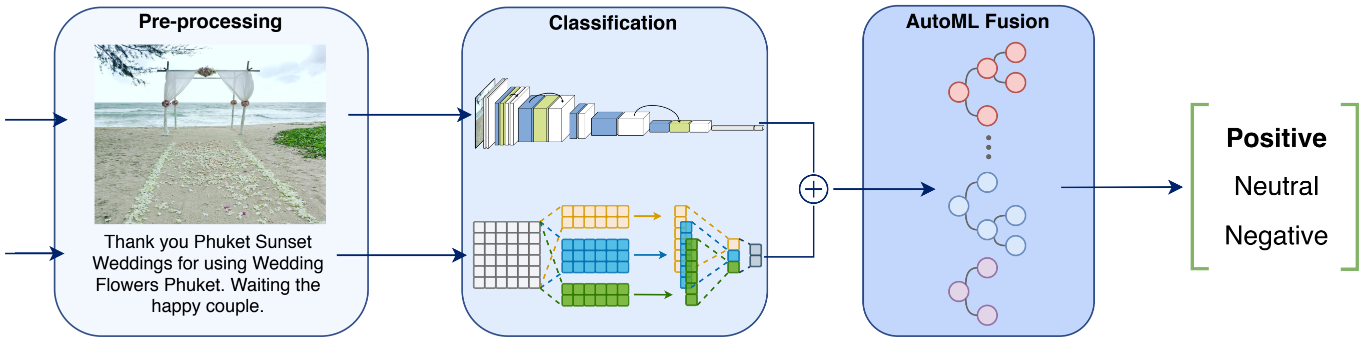
I. Introduction

Sentiment analysis is an ever-growing research topic, where the focus is to analyze the underlying sentiment of a given source of data, based on its subjectivity and context. Sentiment analysis is mostly performed using textual data, where the goal is to, based on a sentence or a text, determine the author's message polarity. The classification is generally binary - either negative or positive, or n-class classification, wherein, the most used one is a 3-class classification - negative, neutral and positive, using either machine-learning approaches, where a classifier is trained using a labelled corpus, or using lexicon-based approaches, where the textual information is classified based on its semantics or by using statistical approaches [1], [2]. Applications of sentiment analysis can be seen in many contexts, such as brand awareness [3], political voting intentions [1], customer satisfaction [4], [5] and in disaster relief [6].

The proliferation of social media opened doors for massive collection of data for sentiment analysis [7]. These are important channels of human communication, allowing for instantaneous spread of information. Twitter emerged as one focal point to capture and analyze data, in which users express their opinions, feelings and thoughts regarding entities or events. Detecting sentiments in Twitter differs from detecting sentiments in conventional text such as blogs and forums, due to the reduced size of the textual data, and because of the information context, addition of symbols in the form of emojis and irony and subjectivity. However, social media networks, such as Twitter, also provide the opportunity to congregate textual data with more information, usually in the form of images, videos or audio. The coupling of multiple data sources, allows the development of multimodal classification, in which a method leverages more than one type of data to perform classification [8]. This is significantly harder in the context of sentiment analysis, as extracting sentiments solely from textual information is easier than combining information from text and, for example, images. Even though a multimodal approach can improve the performance when compared to a sole text-approach [9], this is a challenging task, especially when using data acquired from social media, as the different data types are sparse and can have different contexts, present irony, different intentions and their combined evaluation is not trivial.

In this paper, we propose a novel multimodal sentiment analysis method, that uses both textual data and images from social media to perform 3-class classification regarding polarity. The proposed method consists of initial individual classification of the textual and image components, and then, based on Automated Machine Learning (AutoML), fuse both classifications into a final one. To perform the individual classifications, we leverage the power of deep neural networks. For this, we evaluated the performance of multiple networks and then selected the best for both the text and the image part. Then, to build the fusing method, we used AutoML to perform a random search to determine the best model to perform the final classification. We evaluated the proposed method in the task of multimodal sentiment analysis using a dataset containing over 470 thousand tweets, where each tweet is composed of both textual and image content.

The contributions of this paper, can be summarized as follows: 1) we conduct a comparison regarding the performance of sentiment analysis in textual data from Twitter; 2) we compare different state-of-the-art deep learning models in image sentiment analysis, and 3), we propose a novel fusion method that combines the individual classifications into a final one, by levering an optimal model generated with AutoML, which resulted in state-of-the-art accuracy on the B-T4SA



Pre-processing
Thank you Phuket Sunset Weddings for using Wedding Flowers Phuket. Waiting the happy couple.
Classification
AutoML Fusion
Positive
Neutral
Negative

Fig. 1. Proposed multimodal architecture. The first container represents the pre-processing component, that receives an image and associated text, and pre-processes it to remove noise and non-important data. The second container shows both classification components, where the image and the text are classified individually using CNNs. The third container receives the concatenation of the individual classifications, and performs a final classification using the optimal model searched - represented by a Gradient Boosting Machine (GBM) in the image.

dataset.

The remainder of this work is organized as follows. Section II, contextualizes the related work in multimodal sentiment analysis, and the use and application of AutoML methods. Section III, details the proposed method. Presenting the architecture of our proposal, including a detailed description of the text and image analysis methods, as well as the AutoML component. In Section IV, we introduce the datasets used, a description of the conducted experiments, and discuss the results. Finally, Section V presents a conclusion.

II. Related Work

A. Multimodal Sentiment Analysis

Even though the vast majority of the sentiment analysis proposals focus on single model sentiment analysis, mainly using textual information [1], [10], there are interesting proposals that try to fuse more than one source of information to perform multimodal sentiment analysis. The method proposed in [11], combines features from audio and video, and fuses them with text features to estimate the sentiment of youtube movie reviews. In [12], the authors propose Tensor Fusion Network, which is a network capable of fusing features extracted from different sources of data, into a single tensor, allowing sentiment analysis both in separate and conjoined. Taking a different approach [9], performs multimodal sentiment analysis by conducting hierarchical fusion of the different features by first fusing them into pairs, and then combining them into one. In [13], the authors proposed Visual Aspect Attention Network, in which the goal is to use images as attention mechanisms to aid in detecting important sentences in documents. To perform such operation, images are analyzed using a CNN, and the output is used as weights in a word encoder. In [14], the authors propose a multimodal method that performs individual analysis of both the image and the text, and then performs a weighted average over the individual predictions to perform a final classification. However, while producing the individual predictions, the authors also introduce *Image Content Analysis*, a second method to classify the images, which is based on detecting the most predominant object on the image and classifying the image based on the probability of that object appearing in a given class in the training set.

The proposed method is more closely related with the one proposed in [14] in the sense of using two processing branches: one for text and one for image, and a fusion method to perform the final classification. However, instead of using heuristics based on probabilities to classify the text and perform the fusion, we use deep neural networks to perform the individual classifications, and AutoML to create an efficient fusion method.

B. AutoML

AutoML [15], focuses on developing approaches that provide efficient methods to design machine learning workflows without extensive need for human intervention or optimization processes [16]. There have been proposals for solving the optimization problem of designing machine learning workflows using random search [17], evolutionary strategies [18], bayesian optimization [19] and reinforcement learning [20]. In [21], the authors extended Auto-Weka in order to use AutoML to design methods to detect railway track defects. In [22], AutoML is leveraged to improve the classification component of CNNs. This method was built by training CNNs until convergence and then partially removing their classification component and replacing it by a searched model, resulting in performance improvements both in accuracy and inference time. Finally, in [23], AutoML is used to forecasting bank failures by performing automated feature extraction.

Our work has similarities with Neural Architecture Search (NAS) [24], in the sense that the focus is in building the classifier and not the entire process of a machine learning workflow, but the main difference is that we do not limit our search to neural networks, rather, we allow more machine learning models to be searched.

Some of the differentiating points of our proposal are the fact that it performs a simple and efficient fusion, based on the individual analysis of the text and image components, and that it takes advantage of AutoML for finding the final classifier that works on top of the fused features.

III. Proposed Method

A. Architecture

The proposed method is composed of three stages: pre-processing, individual classifications, and the fusion stage. The entire architecture of the method can be seen in Fig. 1. For both individual classification components, we implemented several state-of-the-art methods to perform a comparison and select the most performant one on the validation set, to be integrated into the proposed architecture.

In the following sections, we present the implementation details of both the image (III-B) and text classification (III-C), as well as the implementation details for the fusion mechanism (III-D). Moreover, in each section, we further explain the pre-processing components of the image and the text.

B. Image Sentiment Analysis

To classify the post's polarity using image data, we explored the use of state-of-the-art CNNs that perform feature extraction and classification over the inner representations created. For this, we focused on using two architectures that are known to do well in different image analysis tasks: ResNet and DenseNet. The remainder of this section details the CNNs used, as well as the pre-processing steps for cleaning the input images.

1) Preprocessing:

We first apply a resize operation, changing the image size to $224 * 224$, to uniformize the dataset, as images from the Twitter dataset have different dimensions. Then, each image is normalized using the mean and standard deviation of each channel in the whole dataset. This is applied by subtracting the mean and dividing by the standard deviation in each channel of each image: $img_i = (o_i - \mu_i)/\sigma_i, \quad i = 1,..,c$, where i represents the channel, o the original image, μ is the mean of the dataset and σ is the standard deviation of the dataset.

2) Models:

ResNet: Residual Neural Networks (ResNet) [25], introduced the idea of skip connections on CNNs. The "identity shortcut connection" present in ResNets, usually skipping two or three layers in the model, allowing for efficient training of deep CNNs, which are known to suffer from the vanishing gradient problem. This problem is present in the back-propagation of the calculated gradients to earlier layers. As the gradient is propagated backwards, repeated multiplications might produce very small gradients, resulting in performance degradation. By having residual connections, residual nets learn residual functions concerning the layer inputs. Furthermore, instead of learning a direct mapping using stacked-layers, residual connections let these layers learn residual mappings. This means that instead of having a layer learning a desired mapping $H(x)$, regarding to the input x, residual connections allow to reframe this mapping as $F(x) := H(x) - x$, which can then be reframed to $H(x) := F(x) + x$. More, if the identity mapping is optimal, residuals can be pushed to zero, meaning that residual networks will, at least, have the same performance of networks using stacked-layers without residual connections.

The ResNet architectures, with different depths, implemented were: ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152, using the parameters defined in the original paper [25].

DenseNet: Densely Connected Convolutional Networks (DenseNet) [26] improved upon ResNet by proposing a type of CNN that utilizes dense connections between layers, using Dense Blocks. In this networks, every layer obtains additional inputs from all preceding layers, instead of traditional layers, in which their input is the output of the last layer or the output of the last layer plus a short connection. By allowing multiple parallel connections, DenseNets preserve the feed-forward network scheme, by having all layers obtaining concatenated outputs of all preceding layers, and passing its own feature-maps to subsequent layers. This allows every layer to receive a "collective knowledge" from all past layers, mitigating the vanishing-gradient problem, encouraging feature reuse and allowing for a reduced number of parameters in each layer, since the feature maps continuously expand by concatenation with previous feature maps.

DenseNet models achieved state-of-the-art results in many image tasks while requiring less computation and fewer model parameters than previous state-of-the-art CNNs. In this work, we have implemented DenseNet161 using the parameters detailed in the original paper [26].

For both ResNet and DenseNet models implemented, we have also applied transfer learning and conducted an experiment, in which the initial layer of each model was modified to have 4 input channels, instead of the original 3. A detailed explanation of these experiments is presented in Section IV-C.

C. Text Sentiment Analysis

This section presents the operations performed over textual data and the models used to do so. For every deep learning model with an Embedding Layer, a GloVe 200 dimension pre-trained on Twitter Embedding was used [27].

1) Pre-processing:

To effectively classify text, before inputting the text into a classifier, a pre-processing step takes place. This is done because: 1) the type of text obtained from social-media varies substantially and contains noise: syntactic, semantic and grammar errors, mainly due to size constraints, typing speed and slang; 2) standardize data, so that classifiers can more easily learn patterns; 3) be in concordance with the input constraints of Word Embeddings layers and different classifiers.

So, the steps to clean the textual data were: 1) transform HTML codes into words and symbols; 2) remove stop words using NLTK functionalities; 3) transform every word to lower case; 4) remove occurrences of more than three equal sequential characters into a maximum of two (e.g., *"sooo happppyy"* becomes *"so happy"*); 5) remove links and specific social media user-mentions (both the mention and the "RT" word from Twitter); and finally, 6) punctuation was removed.

2) Models:

VADER: Valence Aware Dictionary for Sentiment Reasoning (VADER) [28], is a lexicon and rule-based method designed to classify the polarity of text from social media. It uses a list of lexical features, such as words, which are labelled accordingly to their polarity. The output of VADER is based on the probability of the sentence belonging to either the positive, negative or neutral class. This method was created having as basis a set of human-curated lexicon sentiment analysis.

TextBlob: TextBlob is a library that implements methods for processing textual data [29]. Like VADER, TexBlob sentiment analysis is based on a set of lexicons that are labelled accordingly to their polarity. This method performs an average over all the lexicons that represent the words in the input sentence and outputs a polarity between -1 and 1. Using this polarity, we defined that values under -0.1 are classified as having a negative meaning, over 0.1 are classified as positive, and the remainder is classified as neutral.

FastText: FastText is a shallow network architecture that can be trained in a reduced time, when compared to deeper architectures, whilst achieving competitive results in multiple text processing tasks [30]. The idea behind FastText is to have an architecture, based on the Continuous Bag of Words (CBOW) model [31] to represent words, being then followed by linear classifiers that classify the input.

The FastText architecture designed in this work consists on an Embedding layer followed by two Linear layers, the first having input size equal to the embedding size and output equal to 256, whilst the second one has input size of 256 and outputs the classification vector.

LSTM: The Long-Short Term Memory architecture (LSTM) [32], is a variation of the traditional Recurrent Neural Networks (RNNs), and was created to mitigate the exploding and vanishing gradient problems found in simple RNNs. An LSTM layer consists of a predefined set of recurrent blocks, each one of them containing cells. These cells have three types: the input, output and forget gates and serve the purpose of updating, or not, the cell state, erasing its memory and deciding if the cell output should be available. With this architecture, an LSTM layer can store information for later use, preventing gradients from vanishing during the learning process, and can also determine what information to ignore, therefore, allowing the network to remember important information for a longer period of time.

Formally, an LSTM layer computes for each element in the input sequence:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \tag{1}$$

$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) \tag{2}$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \tag{3}$$

$$c_t = f_t \circ c_{(t-1)} + i_t \circ g_t \tag{4}$$

$$h_t = o_t \circ \tanh(c_t) \tag{5}$$

where h_t is the hidden state at time t; c_t is the cell state at time t; x_t is the input at time t; $h_{(t-1)}$ is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0, and i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, respectively. σ is the sigmoid function, and $\circ$ is the element-wise product.

The architecture implemented consist of an Embedding layer, followed by an LSTM layer with an input size equal to the dimension of the Embedding and with the number of features in the hidden state equal to 256. This is then followed by a Linear layer with input size equal to the number of features in the LSTM (256) and output size equal to the number of classes.

LSTM-Attn: The second LSTM architecture we implemented, LSTM-Attn, to perform sentiment analysis in the text is based on the aforementioned LSTM architecture, but with the addition of an Attention layer [33] between the LSTM layer and the Linear layer. The idea behind the Attention mechanism is to increase the importance of specific parts of the input sentence [34]. There are two types of attention mechanisms, the global and the local ones. The difference is that in the global mechanisms, all the hidden states of the previous layer are considered for deriving the context vector, whilst on the local attention mechanisms, only some hidden states are considered [35]. In this implementation, the Attention layer is a global attention mechanism that computes the soft alignment score between the output of the LSTM and its final hidden state.

The final architecture for LSTM-Attn is: an Embedding layer, followed by an LSTM layer with input size equal to the embedding dimension and 256 as the number of hidden features. Following this, comes the Attention layer that receives the output from the LSTM and the last hidden LSTM state, and outputs a new hidden state with the same size as the output of the LSTM. This then serves as input to the Linear layer, which outputs the classification vector.

Bi-LSTM: Bi-directional LSTMs (Bi-LSTM) were created as an improvement over the vanilla LSTMs in tasks where full sequences are present, and context is important [36]. The idea behind Bi-LSTM is to present the information forward and backwards to two separate LSTM networks that are both connected to the same output layer. Having such bi-directional processing on the input information means that the network has, for each input (embedding or word), sequential and context information about it.

The architecture implemented starts with an Embedding layer, followed by a bi-directional LSTM layer with input size equal to the embedding dimension and 256 as hidden features. Then, we use the output from the Bi-LSTM layer and perform both an average pool and a max pool, which are concatenated together and fed into a Linear layer of input size $256 * 4$ and output of 64. Then, a ReLU operation is performed, followed by a dropout, with $p = 0.1$. The result of this goes to a Linear layer that outputs the classification vector.

RNN: Recurrent Neural Networks were designed to allow neural networks to have temporal information, which simple neural networks cannot have [37]. Basically, RNNs form a chain structure in which each node receives as input the output from the predecessor node and one part of the input sequence (e.g., a word or a vector). Each node outputs a value, both to the successor node and to the next layer.

So, what an RNN layer does is, for each input element, it computes:

$$h_t = \tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh}) \tag{6}$$

where h_t is the hidden state at time t; x_t is the input at time t, and $h_{(t-1)}$ is the hidden state of the previous layer at time $t-1$ or the initial hidden state at time 0.

The architecture implemented is an Embedding Layer followed by a multi-layer Elman RNN with 2 layers, input size equal to the dimension of the embedding and with a hidden size of 256. The output of this layer is then inserted into a Linear layer that outputs the classification vector.

RCNN: The idea behind Recurrent Convolutional Neural Network (RCNN) [38], is to apply a recurrent network structure to text classification that requires no human-designed features. The recurrent structure captures contextual information as far as possible when learning word representations whilst having less noise when compared to methods that use neural networks that rely on window-based processing. To store the context, the RCNN uses a bi-directional recurrent structure and employs a max-pooling layer to capture key features present in the text automatically.

The architecture implemented consists of an Embedding Layer, a bi-directional LSTM Layer with input size equal to the dimension of the embedding, hidden size of 256 and a dropout of 0.8. The final embedding vector is the concatenation of its embedding and left and right contextual embeddings, which in this case is the hidden vector of the LSTM. This concatenated vector is then passed to a Linear Layer which maps the input vector back to a vector with a size equal to the hidden size of the LSTM, 256. This is passed through a 1D Max Pooling Layer, and finally, the output from this layer is sent to a Linear Layer that maps the input to a classification vector.

TextCNN: The Convolutional Neural Networks for Sentence Classification (TextCNN), performs convolutions, of different kernel sizes, on textual data [39]. This is done by performing convolutions over the embedding matrix that represents the input sentences. The convolutions performed are parallel and independent of one another. The architecture of TextCNN implemented is an Embedding Layer followed by 5 convolutional blocks. Each one of those blocks consists of a 2D Convolutional Layer followed by a ReLU activation function and a 1D Max Pooling Layer. The output of all blocks is concatenated into a single vector that goes through a dropout phase with $p = 0.8$, and the result then goes to a Linear Layer that returns the classification vector.

Based on TextCNN, we defined a second CNN-based network to perform text classification, which we named **sCNN**. The architecture is similar, but instead of having 5 blocks of Convolutions-ReLU-Max Pooling, it has only 3 with kernel sizes of 1, 3 and 5 respectively.

VDCNN: Inspired by the results that deep convolutional networks have on image classification tasks [40], the authors of Very Deep Convolutional Networks for Text Classification (VDCNN) [41] designed a similar deep neural network for the problem of classifying text. In VDCNN, many convolutions with small kernel sizes (size 3) are stacked to form a deep network, where shortcuts are present for keeping contextual information and solving the vanishing gradient problem. VDCNN employs convolutional blocks that consist of a sequence of two convolutional layers, each one followed by batch normalization and a ReLU activation.

We implemented 4 VDCNN architectures with different depths: 9, 17, 29 and 49. Every architecture starts with an Embedding layer, followed by a 1D Conv layer with input size equal to embedding size and output size of 64. Then, they have a set of Convolution Blocks. The number of Convolution Blocks depends on the depth of the architecture, but can be seen in [41]-Table 2. After the Convolution Blocks, comes a K-Max Pooling layer, a Linear layer with input of $512 * k$, where k is the number selected for the pooling layer, (2), and output of 2048. Following it, a Linear Layer with 2048 as input and output is inserted and finally, a Linear Layer with an input size of 2048, outputs the classification vector.

D. Fusion Method

The focus of the fusion method is to use the individual classifications of the text and image components to perform a final classification. The goal with the fusion approach is to leverage context knowledge of both sources, to outperform individual classifications.

To create the classifier, we based our approach on AutoML [15], where the goal is to create an optimal model to classify a given dataset, without requiring extensive human modelling. First, we can define a machine learning model $\mathbb{L}$, as a mapping from the space of datasets, D, and architectures, A, to the space of models $M, \mathbb{L} : D \times A \to M$. For any given dataset $d \in D$, and architecture $a \in A$, the mapping returns the solution to the problem, which consists of minimizing a loss function, $\mathcal{L}$, with regularization mechanisms, R, with respect to the model, m, with parameters θ, architecture a, and using the training data, $d^{(train)}$ [42]:

$$\mathbb{L}(a, d^{(train)}) = \underset{m^{(a,\theta)} \in M^{(a)}}{\arg\min} \mathcal{L}(m^{(a,\theta)}, d^{(train)}) + R(\theta) \tag{7}$$

So, we can define our problem as a nested optimization problem, where the goal is to find an optimal model to classify the sentiment based on the individual classifications, d, and a search space A: $a^* \in A$, that maximizes the objective function $\mathcal{O}$, on the validation set:

$$a^* = \underset{a \in A}{\arg\max}\, \mathcal{O}(\mathbb{L}(a, d^{(train)}), d^{(valid)}) \tag{8}$$

Since our problem relies on fusing individual classifications of both text and image into a final one, the first step is to get d based on Y_{img} and Y_{text}, where Y represents the classification vector, and img and $text$ represent the classifiers, in order to fuse them into a unique feature map: $X = Y_{img} \bigoplus Y_{text}$, where X will be the input for the optimization problem (final classifier). Note that in our problem, $\mathcal{O}$ is defined as accuracy in the task of 3-class sentiment classification.

To search for the optimal machine learning model, we based our solution on [43], by performing an automatic random search [17] over a set of several machine learning algorithms and their inner parameters. So, in this work, to search the optimal model and its inner parameters, we performed a random search over the space that includes the following models: a random forest, an extremely-randomized forest, a random grid of generalized linear models, a random grid of XGboost, a random grid of gradient boosting machines (GBM), a random grid of deep neural networks. After searching these models, 2 stacked ensembles were created, the first one comprised of all models evaluated, and the other one, containing the best model of each type. In the end, the model with the best performance on the validation set is the one selected to be in the architecture of the proposed method.

IV. Experiments

A. Datasets

As aforementioned, our approach tackles the problem of multimodal sentiment analysis, using both textual and image information. For this, we focused on using the B-T4SA dataset, which is a dataset comprised of Twitter information, in which every sample has both text and image. Furthermore, to conduct experiments using transfer-learning [44], we incorporated two more datasets: *Stanford Sentiment Treebank* (SST-5), for the text classification, and *Flickr and Instagram Dataset*, for the image classification component.

Following is a detailed explanation of all the datasets used.

1) B-T4SA: B-T4SA is a subset of T4SA, consisting of 470 thousand samples, each one containing both text and image information. All classes are balanced, and the splits are stratified. The train set consists of approximately 80% of the dataset, while both the validation and test sets have 10% each. B-T4SA was created to solve the problems of T4SA, such as duplicated entries, small sentences, malformed images, and unbalanced classes [45]. In Figure 2, we show an example of an image and the corresponding text, for each class (negative, neutral, positive).

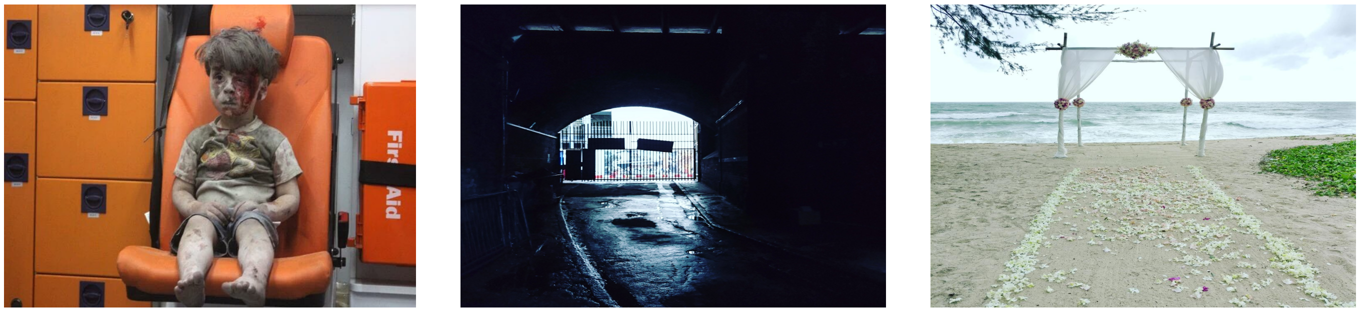
2) Stanford Sentiment Treebank: The Stanford Sentiment Treebank (SST) consists of sentiment labels for 215,154 phrases in the parse trees of 11855 sentences [46]. The dataset can be presented in the form of binary classification, either negative or positive, or in a fine-grained way, using a 5-class classification: very negative, negative, neutral, positive, and very positive. The latter is denominated SST-5, and is widely used to evaluate text sentiment classifiers. SST-5 was used in this work as a way to initially train a model, before training it on a final dataset. This allows performing fine-tuning on the model, by transferring the knowledge from the first dataset to the second one, avoiding initializing the model weights randomly [47]. Denote that, as this dataset has 5 classes, when the models were transferred to B-T4SA, the last layer was removed and substituted by a new one with only 3 output classes.

3) Flickr and Instagram: To perform transfer-learning on the image classifier, we have also used the Flickr and Instagram dataset [48], which is composed of 23308 labelled images of 8 different classes of emotions - amusement, anger, awe, contentment, disgust, excitement, fear and sadness. The goal with this dataset was to pre-train the image classifiers, which upon convergence, are transferred to B-T4SA, by replacing the last classification layer to an identical one with only 3 output classes.

B. Text Analysis

To select the best text sentiment analysis model to use in the multimodal architecture, we conducted a set of experiments with all the models implemented. We evaluated each model three times in the task of classifying sentiments in the B-T4SA dataset, using the Adam optimizer [49], and the Cross-Entropy loss.

The mean accuracy and standard deviation in each one of the sets are shown in Table I, where the first block represents the results for two models that can be used as a library in python (VADER and TextBlob), the second block represents the deep learning models, and the third block represents the best model of the previous block with fine-tuning. In the first block, we show two results for both methods, including with and without the pre-processing step. The "-PP" represents the results with clean data, which achieved better results than without any data cleaning, showing that cleaning textual data to remove noise will allow methods to yield better results. In the second and third blocks, all experiments were conducted using pre-processed data. Here, it is possible to see that, except for FastText that achieved approximately 42% and LSTM that was incapable of learning to solve the task (even with different learning rates, hidden features and optimizers), all models achieved a mean accuracy of over 90%. This is well above the plug-and-play methods, and the previous state-of-the-art [14], which used TextBlob methods to achieve 64.27% accuracy. The best result from these methods was obtained with the RCNN, which achieved a mean accuracy of 94.61%, outperforming all other methods. This can be justified due to the strong capability of RCNNs to evaluate a word, based on its embeddings, coupled by the right and left contexts, which are extracted using recurrent structures. This combination allows features to be extracted more accurately when working in problems that require context, of which sentiment analysis is heavily dependent on. On the third block of the table, the results of fine-tuning the RCNN model are presented. In this case, the fine-tuning was performed by initially training the model on the SST-5 dataset, and then replacing its final classification layer to output three values instead of five. More, "RCNN-sst *ft* B-T4SA FC" represents training initially on SST-5 and then train only on the Linear Layers of the model using B-T4SA, whereas "RCNN-sst *ft* B-T4SA" represents fine-tuning the entire model on B-T4SA, after training it on SST-5. By conducting such experiments, it is possible to see that fine-tuning the entire model yields better results when compared to only transfer learning and fine-



(a) **Negative**: "His eyes speak of the horror of war that no one should go through. War doesn't help, it only kills. Pls Stop."

(b) **Neutral**: "And they say it's grim up north..."

(c) **Positive**: "Thank you Phuket Sunset Weddings for using Wedding Flowers Phuket. Waiting the happy couple."

Fig. 2. Examples of images and the correspondent texts of the three different classes presented in the dataset. (a), presents a negative example; (b), a neutral one, and (c), a positive sample.

TABLE I

MEAN ACCURACY AND STANDARD DEVIATION ON THE TRAIN AND VALIDATION SET OF THE B-T4SA DATASET WITH DIFFERENT METHODS FOR LANGUAGE PROCESSING. THE FIRST BLOCK SHOWS THE RESULTS USING METHODS THAT ARE AVAILABLE AS PYTHON LIBRARIES. THE SECOND BLOCK SHOWS THE RESULT OF OUR IMPLEMENTATIONS OF DIFFERENT DEEP LEARNING METHODS. IN THE THIRD BLOCK, WE SHOW THE RESULTS OF THE BEST METHOD FROM THE SECOND BLOCK (RCNN), PRE-TRAINING ON SST AND FINE-TUNED ON B-T4SA. EACH MODEL WAS EVALUATED THREE TIMES, UNDER THE SAME CONDITIONS.

Method	**Mean Accuracy (%)** Train	Validation
VADER	41.04 ± 0	41.02 ± 0
VADER-PP	56.84 ± 0	56.82 ± 0
Textblob	64.22 ± 0	64.27 ± 0
Textblob-PP	64.88 ± 0	64.78 ± 0
FastText	42.86 ± 0.03	42.76 ± 0.05
LSTM	33.33 ± 0.04	33.13 ± 0.00
LSTM-Attn	97.36 ± 0.03	93.48 ± 0.57
BI-LSTM	96.56 ± 0.97	94.35 ± 0.07
RNN	90.72 ± 0.78	91.24 ± 0.48
RCNN	98.12 ± 1.10	**94.61** ± 0.03
TextCNN	95.47 ± 0.86	93.73 ± 0.00
sCNN	90.69 ± 0.10	92.69 ± 0.02
VDCNN9	94.18 ± 0.81	93.33 ± 0.23
VDCNN17	88.29 ± 2.28	92.05 ± 0.52
VDCNN29	93.90 ± 0.65	93.19 ± 0.18
VDCNN49	92.61 ± 0.33	92.81 ± 0.11
RCNN-sst *ft* B-T4SA FC	86.73 ± 0.69	86.51 ± 0.67
RCNN-sst *ft* B-T4SA	**98.60** ± 1.29	94.60 ± 0.03

tune the last classification layers. However, these results do not improve upon the normal model, with weights initialized randomly.

From this experiment, we selected RCNN as the text classifier to be used in the proposed method, since it presented the best results in the validation set. Even though the validation set cannot be seen as a surrogate of the test set, it is the best way to evaluate how a model will perform in unseen data, without introducing biases by using the test set, which is only used at the end of the entire training process (when all the methods for the proposed method are selected).

C. Image Analysis

Regarding the selection of the model to perform the image classification, we have evaluated the performance of multiple resnet architectures and densenet161 in the task of image sentiment analysis. In Table II, the results for our experiments are shown. Note that every network was evaluated using the same learning rate ($1e-3$), Adam optimizer and cross-entropy loss. The first row of the table represents if the experiment was done using transfer-learning, meaning that the models were initially trained on the Flicker and Instagram dataset. We have also evaluated how the different models behave with RBG images (1st and 4th experiment in the table, while the 4th uses pre-trained weights), and RBG and Local Binary Patterns (LBP) [50]. In the latter, we changed the models to receive 4 inputs and placed the LBP on the fourth channel (3rd experiment in the table). The goal of using LBP is to evidence hidden patterns that assist in detecting the image polarity. All the results show that classifying the sentiment of an image is difficult, mainly due to the subjectivity of the image and due to inter-class similarities, where images that have different classes can be visually similar. Neither the addition of the LBP nor pre-training the models on the Flicker and Instagram dataset improved the results when compared to using only RGB with randomly initialized weights. Furthermore, all models performed similarly, but ResNet34 was the best one, achieving 49.8% accuracy using RGB images, with or without pre-training. Even though ResNet18 had almost the same performance using pre-trained settings, we selected ResNet34 for the proposed method, as it consistently outperformed ResNet18.

D. Fusion Analysis

Based on the text classifier, RCNN, and the image classifier, ResNet34, we then evaluated the performance of the proposed method as a whole. For this, we initially searched for the optimal model, using the method described in III-D, allowing the search for a maximum of two hours. By doing this, the most performant model in the validation set was a GBM. By evaluating the entire proposed method on the test

TABLE II
ACCURACY (%) OF SEVERAL DEEP LEARNING NETWORKS IN THE TASK OF CLASSIFYING SENTIMENTS, BOTH IN THE TRAIN AND VALIDATION SET. THE FIRST ROW, PRE-TRAIN, INDICATES IF THE EXPERIMENT USES TRANSFER LEARNING. EACH EXPERIMENT COLUMN ALSO CONTAINS THE INFORMATION ABOUT THE DATA TYPE USED, EITHER RGB OR RGB PLUS LBP.

Pre-trained	×		×		✓	
Network	**RGB**		**RGB+LBP**		**RGB**	
	Train	**Val**	**Train**	**Val**	**Train**	**Val**
ResNet18	**47.4**%	47.7%	47.4%	47.9%	46.6%	49.7%
ResNet34	47.2%	**49.8**%	47.3%	**48.0**%	45.6%	**49.8**%
ResNet50	46.3%	46.4%	47.2%	47.4%	**48.5**%	48.7%
ResNet101	44.9%	45.1%	47.1%	47.1%	47.6%	47.7%
ResNet152	44.5%	44.5%	45.9%	45.9%	47.1%	47.5%
DenseNet161	46.9%	47.1%	**47.5**%	47.5%	47.2%	47.3%

TABLE III
ACCURACY (%) IN THE TEST SET FOR THE PROPOSED METHOD AND A BASELINE USING SVM.

Method	**Test Accuracy (%)**
SVM	95.16%
AutoML-based Fusion **(ours)**	**95.19%**

set, it achieved an accuracy of 95.19%, which the result is synthesized in Table III. To evaluate the performance of the proposed method, but with a different fusion classifier, we have also evaluated the use of a Support Vector Machine (SVM), which achieved a performance of 95.16% using all the settings of the proposed method. The difference between the GBM and the SVM classifier is small, 0.03%, but the AutoML searched method has several advantages. The first one is the time required to train, while the AutoML method of searching for methods only required two hours, SVM required several hours to train. More, SVMs tend not to scale well, as there are more features (in the order on thousands), they tend to become extremely slow to fit the data, whilst our proposed methodology to search for a classifier is extremely robust by comprising methods that can handle a large number of features without becoming untenable. However, the SVM baseline consolidates the proposed method, by showing that the proposed architecture works, even in the presence of different fusion classifiers.

To further validate our proposal, we compare our results with state-of-the-art methods, in which the results are present in Table IV. In this, it is possible to see that our proposal outperforms others. More, to further evaluate the effectiveness of our proposal, we have further tested the proposed method in [14], by replacing its text classifier by our RCNN, resulting in a 15.9% accuracy improvement, (represented in the table by Information Fusion [14] (TM)), but is still 18.8% below our proposed method. This consolidates that our proposed method of fusing the individual classifications and then performing a random search to find the optimal fusion classifier, is an efficient method.

TABLE IV
COMPARISON WITH THE RESULTS OF EXISTING METHODS WITH OUR EXPERIMENTS. THE TM, STANDS FOR SUBSTITUTING THE TEXT CLASSIFIER FROM [14] FOR THE ONE SELECTED IN OUR EXPERIMENTS.

Method	**B-T4SA Test Set Accuracy (%)**
Random Classifier	33.33%
Hybrid-T4SA FT-F [45]	49.90%
Hybrid-T4SA FT-A [45]	49.10%
VGG-T4SA FT-F [45]	50.60%
VGG-T4SA FT-A [45]	51.30%
Information Fusion [14]	60.42%
Information Fusion [14] (TM)	76.35%
SVM-fusion **(ours)**	95.16%
AutoML-based Fusion **(ours)**	**95.19%**

V. CONCLUSIONS

This paper proposes a novel method to perform multimodal sentiment classification of social media content. The proposed method consists of performing individual text and image classifications, which are then fused by an AutoML-generated model to perform a final classification. We explored several state-of-the-art classifiers for both text and image. More, with the proposed AutoML approach, our method was capable of finding an optimal model that outperformed the state-of-the-art in the B-T4SA dataset, which, due to its natural content, is very challenging and contains intra and inter-class subjectivity.

ACKNOWLEDGMENTS

This work was supported by 'FCT - Fundação para a Ciência e Tecnologia' throught the research grant '2020.04588.BD', partially supported by NOVA LINCS (UIDB/04516/2020) with the financial support of FCT-Fundação para a Ciência e a Tecnologia, through national funds, partially supported the project MOVES-Monitoring Virtual Crowds in Smart Cities (PTDC/EEI-AUT/28918/2017) financed by FCT-Fundação para a Ciência e a Tecnologia, and partially supported by project 026653 (POCI-01-0247-FEDER-026653) INDTECH 4.0 – New technologies for smart manufacturing, cofinanced by the Portugal 2020 Program (PT 2020), Compete 2020 Program and the European Union through the European Regional Development Fund (ERDF).

REFERENCES

[1] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams engineering journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[2] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in *Mining text data*. Springer, 2012, pp. 163–222.

[3] B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury, "Twitter power: Tweets as electronic word of mouth," *Journal of the American society for information science and technology*, vol. 60, no. 11, pp. 2169–2188, 2009.

[4] D. Gräbner, M. Zanker, G. Fliedl, M. Fuchs *et al.*, "Classification of customer reviews based on sentiment analysis," in *ENTER*. Citeseer, 2012, pp. 460–470.

[5] S. Shayaa, N. I. Jaafar, S. Bahri, A. Sulaiman, P. S. Wai, Y. W. Chung, A. Z. Piprani, and M. A. Al-Garadi, "Sentiment analysis of big data: Methods, applications, and open challenges," *IEEE Access*, vol. 6, pp. 37 807–37 827, 2018.

[6] G. Beigi, X. Hu, R. Maciejewski, and H. Liu, "An overview of sentiment analysis in social media and its applications in disaster relief," in *Sentiment analysis and ontology engineering*. Springer, 2016, pp. 313–340.

[7] A. Giachanou and F. Crestani, "Like it or not: A survey of twitter sentiment analysis methods," *ACM Comput. Surv.*, vol. 49, no. 2, Jun. 2016.

[8] M. Soleymani, D. Garcia, B. Jou, B. Schuller, S.-F. Chang, and M. Pantic, "A survey of multimodal sentiment analysis," *Image and Vision Computing*, vol. 65, pp. 3–14, 2017.

[9] N. Majumder, D. Hazarika, A. Gelbukh, E. Cambria, and S. Poria, "Multimodal sentiment analysis using hierarchical fusion with context modeling," *Knowledge-based systems*, vol. 161, pp. 124–133, 2018.

[10] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1253, 2018.

[11] M. Wöllmer, F. Weninger, T. Knaup, B. Schuller, C. Sun, K. Sagae, and L.-P. Morency, "Youtube movie reviews: Sentiment analysis in an audio-visual context," *IEEE Intelligent Systems*, vol. 28, no. 3, pp. 46–53, 2013.

[12] A. Zadeh, M. Chen, S. Poria, E. Cambria, and L.-P. Morency, "Tensor fusion network for multimodal sentiment analysis," *arXiv preprint arXiv:1707.07250*, 2017.

[13] Q. Truong and H. W. Lauw, "Vistanet: Visual aspect attention network for multimodal sentiment analysis," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*. AAAI Press, 2019, pp. 305–312.

[14] A. Gaspar and L. A. Alexandre, "A multimodal approach to image sentiment analysis," in *Intelligent Data Engineering and Automated Learning – IDEAL 2019*. Cham: Springer International Publishing, 2019, pp. 302–309.

[15] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automatic Machine Learning: Methods, Systems, Challenges*. Springer, 2019.

[16] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, p. 106622, 2020.

[17] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[18] J. Liang, E. Meyerson, B. Hodjat, D. Fink, K. Mutch, and R. Miikkulainen, "Evolutionary neural automl for deep learning," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 401–409.

[19] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 826–830, 2017.

[20] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 784–800.

[21] S. Kocbek and B. Gabrys, "Automated machine learning techniques in prognostics of railway track defects," in *ICDMW*. IEEE, 2019.

[22] V. Lopes and L. A. Alexandre, "Auto-classifier: A robust defect detector based on an automl head," in *International Conference on Neural Information Processing*. Springer, Cham, 2020, pp. 137–149.

[23] A. Agrapetidou, P. Charonyktakis, P. Gogas, T. Papadimitriou, and I. Tsamardinos, "An automl application to forecasting bank failures," *Applied Economics Letters*, pp. 1–5, 2020.

[24] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, pp. 1–21, 2019.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[27] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[28] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth international AAAI conference on weblogs and social media*, 2014.

[29] S. Loria, "textblob documentation," Technical report, Tech. Rep., 2018.

[30] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 427–431.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[34] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[35] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421.

[36] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602 – 610, 2005, iJCNN 2005.

[37] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[38] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15. AAAI Press, 2015, pp. 2267–2273.

[39] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[41] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 1107–1116.

[42] M. Wistuba, "Transfer neural architecture search," *1st Workshop on Neural Architecture Search at ICLR 2020*, 2020.

[43] H2O.ai, *H2O AutoML*, June 2017, h2O version 3.30.0.1. [Online]. Available: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

[44] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[45] L. Vadicamo, F. Carrara, A. Cimino, S. Cresci, F. Dell'Orletta, F. Falchi, and M. Tesconi, "Cross-media learning for image sentiment analysis in the wild," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Oct 2017, pp. 308–317.

[46] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[47] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, p. 9, 2016.

[48] Q. You, J. Luo, H. Jin, and J. Yang, "Building a large scale dataset for image emotion recognition: The fine print and the benchmark," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, p. 308–314.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015*, Y. Bengio and Y. LeCun, Eds., 2015.

[50] K. Meena and A. Suruliandi, "Local binary patterns and its variants for face recognition," in *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, 2011, pp. 782–786.