# A Lightweight Deep Learning Framework using Resource-Efficient Batch Normalization for Sarcasm Detection

**Article** *in* International Journal of Scientific Research in Computer Science Engineering and Information Technology · July 2025

DOI: 10.32628/CSEIT251143

CITATIONS
0

READS
15

**1 author:**

Jiby Mariya Jose
Independent Researcher
**6** PUBLICATIONS **6** CITATIONS

SEE PROFILE

# A Lightweight Deep Learning Framework using Resource-Efficient Batch Normalization for Sarcasm Detection

Jiby Mariya Jose
published by - IJSCSEIT

**Abstract**

Communication is not always direct; it often involves nuanced elements like humor, irony, and sarcasm. This study introduces a novel two-level approach for sarcasm detection, leveraging Convolutional Neural Networks (CNNs). Convolutional neural networks (CNNs) are crucial for many deep learning applications, yet their deployment on IoT devices is challenged by resource constraints and the need for low latency, particularly in on-device training. Traditional methods of deploying large CNN models on these devices often lead to suboptimal performance and increased energy consumption. To address this, our paper proposes an energy-efficient CNN design by optimising batch normalisation operations. Batch normalisation is vital for deep learning, aiding in faster convergence and stabilising gradient flow, but there has been limited research on creating energy-efficient and lightweight CNNs with optimised batch normalisation. This study proposes a 3R (reduce, reuse, recycle) optimisation technique for batch normalization. This technique introduces an energy-efficient CNN architecture. We investigate the use of batch normalization optimization to streamline memory usage and computational complexity, aiming to uphold or improve model performance on CPU-based systems. Additionally, we evaluate its effectiveness across diverse datasets, focusing on energy efficiency and adaptability in different settings. Furthermore, we analyze how batch normalization influences the performance and effectiveness of activation functions and pooling layers in neural network designs. Our results highlight batch normalization's ability to enhance computational efficiency, particularly on devices with limited resources.

*Keywords:* Lightweight Convolutional Neural Networks (CNN), Lightweight Diabetic Retinopathy Detection, Lightweight Mineral Classification, Batch Normalisation (BN), Edge devices, Energy Efficiency.

## 1. Introduction

Convolutional neural networks (CNNs) have become ubiquitous in modern deep learning applications, excelling in tasks such as image classification, object detection, and semantic segmentation [1]. However, the widespread adoption of CNNs faces challenges, especially during training when applied to IoT devices [2], which typically operate under limited resource and power. The traditional approach of deploying large parameterized CNN models directly on these devices[3] not only results in suboptimal performance but also leads to increased energy consumption and reduced battery life [4]. In the current landscape, solution providers are confronted with a dilemma: either train and deploy CNN models in the cloud [5], encountering latency issues[6] and escalating operational costs, or conduct training on high-performance servers[7], contributing to excessive energy consumption and electronic waste [8], [9]. As the IoT ecosystem continues to expand, the need for energy-efficient, on-device training methods becomes imperative.

Batch normalisation is a pivotal optimisation technique in deep learning [10], addressing key challenges encountered during the training of neural networks. By mitigating internal covariate shift [11], batch normalisation normalises inputs to each layer [12], facilitating faster convergence [13]. This normalisation process also stabilises gradient flow, preventing the issue of vanishing or exploding gradients [14] and enabling more efficient learning [15]. Overall, batch normalisation plays a critical role in enhancing the stability [16], efficiency, and performance of deep neural networks, making it an indispensable component in modern deep learning research and applications [17].However, there is a lack of study on the design of batch normalisation for energy-efficient and lightweight CNNs.To address this research gap, the paper introduces an energy-efficient convolutional neural network (CNN) design by optimising batch normalisation operations.

The research focused on employing a multimodal approach to detect sarcasm as a practical application for evaluating the effectiveness of the proposed CNN architecture. Sarcasm is a distinct form of communication that targets the intended recipient directly [18] rather than broadcasting a message broadly into the environment [19]. Unlike straightforward messages that evoke universal emotions such as happiness, sadness, anger, or neutrality in all listeners, sarcasm functions differently. It elicited varied emotional responses in listeners based on their individual contextual understanding, past experiences, current mindset, and ability to grasp sarcastic cues [20], [21]. The impact of a sarcastic message largely depended on the receiver [18]. For someone with a playful and fun-loving nature, a sarcastic message could be received positively, evoking amusement or laughter. Conversely, for a serious-minded individ-

*Email address:* jiby.jose1404@gmail.com (Jiby Mariya Jose)

ual, the same message might have been perceived negatively, potentially leading to feelings of resentment or envy towards the speaker [22], [23]. Thus, sarcasm[24] had a nuanced effect that varied significantly depending on the recipient's perspective [25].
In the realm of human-machine interaction, understanding and identifying sarcasm from a listeners perspective is crucial. Sarcasm in communication often leads to misunderstandings within society [26]. The way a sarcastic message was perceived and responded to was[27] influenced by the dynamics between the speaker and the[28] listener. Therefore, understanding these[29] nuances was essential for effective and respectful communication, minimising the risk of misinterpretation, and fostering[30] clearer interpersonal interactions. As technology progresses, automated sarcasm detection has become increasingly critical for enabling seamless interactions between humans and machines and for advancing the development of emotionally intelligent systems [31]. However, the current research landscape lacks exploration[32] into direct speech-only-based studies and expression identification for multimodal sarcasm detection, as contemporary advancements primarily focus on digital input communication mainly using text input [33].
Challenges in existing applications of multimodal systems include:

- Text-Centric Approach: Existing multimodal automatic sarcasm detection models identified sarcasm by extracting textual content [34], which increased overall complexity. Consequently, there is a lack of study on developing multimodal sarcasm detection models that could classify outputs directly using audio fetched from the environment [35].

- Power-Intensive and Energy-Hungry: The current multimodal sarcasm detection model utilised multiple CNNs or transformers [36], resulting in increased power consumption by devices due to running these large parameterized models [37]. And hence, there is a lack of research on energy-efficient and lightweight solutions for deploying CNNs for multimodal sarcasm detection.

- Lack of Adaptability on CPU: Existing multimodal sarcasm detection models were trained on GPU-based systems [38]–[40], and there is a lack of emphasis on support for general-purpose CPU-based systems.

For this study, we consider the following research questions:

- How do different activation functions and pooling layers affect the number and characteristics of non-trainable parameters in CNN models?

- What is the relationship between non-trainable parameters in CNN models and their impact on the energy consumption of CNN models?

- How do non-trainable parameters impact CNN model performance when applied to different domains with varying data characteristics?

The structure of this research paper is as follows: Section 2 provides a review of existing literature. Section 3 outlines the problem statement, while Section 4 elaborates on our dataset creation process. In Section 5, we delve into the proposed system, and Section 6 presents the results and experiments conducted.

## 2. Related Works

### *2.1. Batch Normalization Optimization*

In the current state-of-the-art, batch normalisation optimisation is generally found in two different perspectives in the literature. It can be broadly classified into two different perspectives: hardware-based and software-based.
In the first approach, researchers considered optimising batch normalisation from a hardware perspective. The work [71] proposed the one-pass normalizer (OPN) accelerator, which aimed to address memory inefficiencies in batch normalisation (BN) during deep neural network (DNN) training. By introducing a novel one-pass computation approach based on sampling-based range normalisation and sparse data recovery techniques, OPN significantly reduced off-chip memory access, enhancing training efficiency while maintaining performance. Additionally, the OPN circuit employed channel-wise constant extraction, leading to a more compact design and substantial reductions in gate count and power consumption. Another work [72] designed a differential amplifier for batch normalisation to compensate for the inference accuracy loss at the lower end of the voltage range. Another work [73] approximately calculated BN to achieve a better tradeoff between hardware efficiency and performance for DNN on-device training processors implemented on an FPGA. However, these batch normalisation techniques did not guarantee a reduction in parameters to fit resource-constrained devices and were also difficult to scale and upgrade [74], [75].
In the second approach, researchers utilised software-based solutions to optimise batch normalization. The work by [76] proposed a drop block layer combined with batch normalisation to reduce trainable parameters. Another work by [77] proposed a batch-normalised long short-term memory (BN-LSTM) network to allow faster convergence of the model. Additionally, [78] proposed shifting the operation of batch normalisation to the logits to rectify the predictions of detectors biassed by long-tail distribution. In another effort, [79] aimed to improve the stability of the model through batch normalization. Furthermore, [80] introduced adaptive batch normalisation, which enabled fast and adaptive determination of channel numbers for each convolutional layer.However, the software-based work in this section focused on memory efficiency and faster training through reduction of trainable which even resulted in accuracy drop.And hence there was a lack of research directed towards designing batch normalisation with a focus on energy efficiency and non-trainable parameter reduction for CPU-on-edge devices[81].

Table 1: Comparison of multimodal sarcasm detection algorithms spanning from 2011 to 2024. Here, [A] indicates achieving accuracy above 80 percent, and [M] denotes the incorporation of multimodal input.

| S.No. | [M] | Algorithm | [A] | Dataset |
|---|---|---|---|---|
| [41] | Text+Tweet | Softmax attention layer BiLSTM | Yes | Twitter |
| [41] | Hindi text+English text | Soft-attention based bi-directional LSTM | Yes | Twitter |
| [42] | Voice cues+Eye movements | Multiclass Neural Network | Yes | MUStARD, Memotion |
| [43] | Phrase+Text | Parallel LSTM | Yes | MUStARD |
| [44] | Text+Acoustic | LSTM+Hierarchical attention mechanism | Yes | MaSaC |
| [45] | Image,Text | Guide attention+Gate mechanism | Yes | MUStARD |
| [46] | Text+Knowledge | Pre-trained COMET | Yes | Twitter,Reddit |
| [47] | Text,Emoji | Emoji-aware-multitask learning | Yes | SEEmoji MUStARD |
| [48] | Text+Image | LSTM+GRU+CNN | Yes | Reddit |
| [49] | Audio+Text | Multi-level late-fusion learning framework+residual connections | Yes | MUStARD |
| [50] | Image+Text | Granularity-fusion Module | Yes | English PHEME, Chinese WeiBo |
| [51] | Image+Text | Crossmodal Bipolar Attention Network | Yes | MUStARD |
| [52] | Text,Image | Encoder-Decoder Network | Yes | Dataset-1 |
| [53] | Text,Emoji | Encoder–decoder + multitask learning | Yes | MUStARD, Memotion |
| [54] | Text+Emoji | CNN+LSTM | Yes | Sarc-H |
| [55] | Text+Image | Multitask interaction learning | Yes | Memotion, MUStARD |
| [56] | Image+Text | Attention fusion network | Yes | CMU-MOSI, CMU-MOSEI |
| [57] | Image+Text | Fusion Transformer network | Yes | MUStARD |
| [58] | Image+Text | ResNet CNN+Text Encoder | Yes | MET-Meme |
| [55] | Audio+Text | Cross-modal target attention mechanism + FCN | Yes | Memotion and MUStARD |
| [53] | Audio+Text | Encoder–Decoder | Yes | MUStARD, Memotion, CMU-MOSEI and MELD |
| [59] | Image+Text | Pre-trained BERT+GAT | Yes | SemEval-2018 |
| [60] | Image+Text | Quantum Probability | Yes | MUStARD, MELD |
| [61] | Image+Text | Pre-trained BERT+VGG19+RESNET50 | Yes | Arafacts |
| [62] | Image+Text | Transformer+GCN | Yes | Reddit |
| [63] | Text+Emoji | CNN+ RoBERTa | Yes | Reddit, SARC |
| [64] | Image+Text | Contrastive learning | Yes | MUStARD |
| [65] | Image+Text | Bi-LSTM variational autoencoder | Yes | MUStARD, Reddit |
| [66] | Image+Text | Cross-modal infiltration+multimodal inconsistency learning | Yes | Twitter, Weibo, Politifact |
| [67] | Text+Emoji | GT-BiCNet+ALABerT | Yes | Twitter |
| [40] | Text+Image | Multi-modal Mutual Learning | Yes | Twitter |
| [38] | Text+Image | Multi-Head Self-Attention+Multiway Feed Forward Network | Yes | Twitter |
| [68] | Text+Audio | Transformer encoder+Gating Network | No | MUStARD, Memotion |
| [69] | Image+Text | Crossmodal Transformer model | Yes | Twitter |
| [70] | Image+Text+Audio | Segmented attention mechanism | Yes | MUStARD |

### *2.2. Multimodal Sarcasm Detection*

In the current state-of-the-art, automatic sarcasm detection using multimodal data can be broadly categorised into text-based approaches and non-textual-based approaches.

For the first approach, researchers implementing multimodal sarcasm detection used text as one of the modalities. The work by [82] proposed a graph-oriented contrastive learning strategy to fuse and distinguish sarcastic clues in both textual and visual modalities. Another work by [83] proposed a multimodal fusion and multitask learning algorithm with a Seq2Seq structure to capture sentiment and sarcasm features. Additionally, [84] extracted text from images for cross-modal sarcasm detection to maximise information extraction. However, the current state of the art in text-based approaches requires users to communicate through digital mediums, which are therefore unsuitable for real-time detection [85].

In the second approach, researchers used non-textual inputs. The work by [42] identified voice cues and eye movements as influential factors in detecting sarcasm. While these solutions could be applied in real-time contexts, they relied significantly on the need for facial images in decision-making. As a result, there is a lack of studies on sarcasm detection using audio and facial expressions.

A chronological summary of multimodal research from 2011 to 2024, compiled from SCI-indexed journals of ACM, IEEE, Elsevier, and Springer is shown in table 1. Table 1 indicated that there was a lack of audio and facial expression-based multimodal sarcasm detection applications that were energy-efficient and compatible with resource-constrained devices.

## 3. Proposed System

### *3.1. Problem Statement*

In the context of multimodal sarcasm classification, our aim was to develop an energy-efficient model that incorporates multiple input types by optimising the architecture of batch normalisation in a convolutional neural network.

The number of parameters for existing batch normalisation can be computed as follows: Given that there are typically $D_{\text{in}}$ features in the input volume, the number of parameters in the batch normalisation layer can be calculated as follows: Therefore, the total number of parameters represented as $\theta_{\text{bn}}$ in the batch normalisation layer is:

$$\theta_{\text{bn}} = 4 \times D_{\text{in}} \tag{1}$$

We aim to minimize the total energy consumption $E$ over the time interval $[0, T]$ by optimizing the batch normalization parameters.

$$E = \int_0^T P_M(\theta_{\text{bn}}(t) + \theta_{\text{N-bn}}(t))\, dt \tag{2}$$

where $P_M$ represents the power consumption rate of the CNN model.And $\theta_{\text{N}}$ represents the total parameters.

Where,

$$\theta_N = \theta_{N-bn} + \theta_{bn} \tag{3}$$

And our goal is to minimise :E by reducing the parameters of the batch normalisation layer $\theta_{bn}$.

### *3.2. Dataset Creation*

Videos served as representations of the real world, embodying multiple modalities [86]. Thus, we opted to train our model using video datasets. However, the existing video-based dataset lacked an adequate number of sarcastic instances. Consequently, we embarked on designing a specialised dataset focused on sarcasm. Our custom dataset underwent several pivotal steps to guarantee its authenticity and pertinence. We utilised TV series renowned for their sarcastic humour, relying on YouTube and IMDb ratings as benchmarks. We specifically targeted series with widespread acclaim to ensure a varied depiction of sarcasm. Episodes featuring notable instances of sarcasm were meticulously chosen based on the ratings and tags

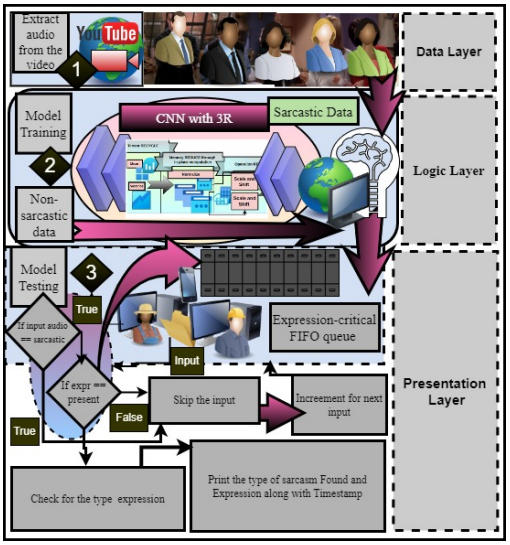


Figure 1: 3-Layer architecture of the proposed expression-critical sarcasm detection

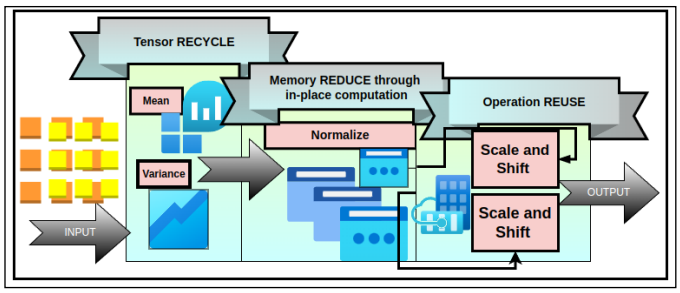


Figure 2: Abstract representation of 3R batch normalisation

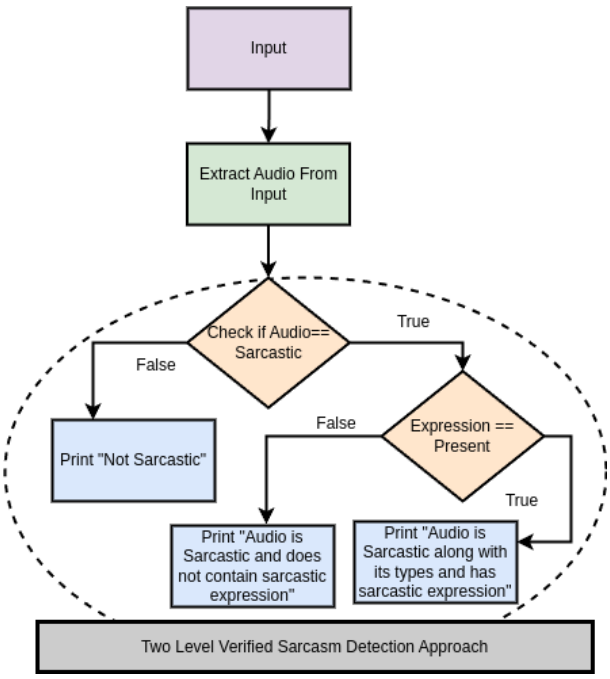


Figure 3: Proposed Two- Authenticated Sarcasm Detection

provided by the above applications. Leveraging the popularity of TV series celebrated for their sarcastic wit, such as "Golden Girls," "Friends," "Sarcamholics," "Chandler Bing," and "Modern Family," we meticulously assembled a dataset. Our selection criteria prioritised series with IMDb ratings exceeding 8.0 and tagged as sarcastic by both applications.Initially, for creating the dataset, we gathered videos using PyTube and extracted audio using MoviePy.

### *3.3. Data Preprocessing*

After creating the dataset, we extracted audio from the video dataset. We chose to work with the audio dataset instead of directly training on video data due to its lightweight nature compared to video. We employed supervised learning to develop a classification model using a dataset containing 7,648 samples of both sarcastic and non-sarcastic audio. For training the model for identifying non-sarcastic audio, we considered the CREMA-D dataset [87]. Based on our analysis of the CREMA-D dataset, we did not find any audio files containing sarcasm. We examined the presence of words ending with long syllables [88] and found no instances of sarcasm. The data was divided into a 75:25 ratio for training and testing, respectively. We used one-hot encoding to label the outputs, distinguishing between the two classes: sarcastic and non-sarcastic. To capture relevant speech characteristics, we utilised the Librosa library in Python to extract MFCC, ZCR, spectral centroid, roll-off, bandwidth, and Mel-frequency cepstral delta coefficients, following established methodologies outlined in previous research [89]–[91]. Subsequently, we designed a CNN-based architecture for training the model, with further details provided in subsequent discussions.

### *3.4. Proposed Architecture*

In this section, we designed a multiple-modality-based lightweight sarcasm decision model. For this, we proposed a two-level verified sarcasm detection named as Dual-Input Authenticated or two-factor authenticated approach using CNN with 3R batch normalization. Figure 1 illustrates the complete architecture of the proposed system. Here, we divided our entire architecture into three stages: the first layer described the dataset collection and processing illustrated in Section IV; the second stage defined the model design and training with CNN with 3R batch normalisation; and finally, the third stage illustrated the model inference. Detailed explanations of each stage were given in the subsequent sections.

#### *3.4.1. Two-Level Verified Sarcasm Detection*

Sarcasm was a challenging expression to identify due to its nuanced and subjective nature. Its impact varied significantly from person to person, making its detection a critical task. In this study, we proposed a two-level verified sarcasm detection method that integrated audio input analysis with facial expression recognition to enhance the accuracy of sarcasm detection.Here we also show the sub-types of sarcasm from the audio along with the facial expression detected.Figure 3 depicts the flow chart of the proposed algorithm.

**Algorithm 1:** Sarcasm Expression Detection

**Input:** Image frame from video
**Output:** Detected critical expression (Raised Eyebrow, Smirk, Sneer)
face_cascade ← Load haarcascade for face detection;
**for** *each* $(x, y, w, h)$ *in faces* **do**
    roi_gray ← Extract region of interest (ROI) in grayscale;
    eyes ← Detect eyes within roi_gray;
    **if** *len(eyes) == 2* **then**
        Check for Raised Eyebrow based on eye positions;
    **end**
    **else if** *len(eyes) == 1* **then**
        Check for Smirk based on mouth position;
    **end**
    **else if** *len(eyes) == 0* **then**
        Check for Sneer based on mouth position;
    **end**
**end**

In our approach, we initially trained an audio CNN with the proposed 3R batch normalisation layers using the dataset described above and then tested it for sarcastic expression. We utilised OpenCV libraries for face detection and extracted facial expressions from the detected faces.We selected the expression to detect sarcasm based on previous studies [92]–[96].The detailed algorithm for expression detection is outlined below. If the input passed both factors, it was fed into a FIFO queue.

**Algorithm 2:** Expression-critical sarcastic message identification using FIFO queue

**Input:** $Q[n]$: Size of FIFO queue,rear
**Input:** videofiles: Files to be tested for sarcasm
**Output:** Updated queue $Q$
**if** **rear ≠ size of $Q - 1$* **then**
    *sarcasm_audio* ← extract audio from video file;
    **foreach** *input* **do**
        **if** *audio==sarcastic ∧ expression present!=0* **then**
            Enqueue *sarcasm_audio* into $Q$;
        **end**
    **end**
    **if** *size of $Q$ > *rear* **then**
        Dequeue the oldest message from $Q$;
    **end**
**end**
**else**
    Dequeue the oldest message from $Q$;
**end**

*3.4.2. Proposed 3R Batch Normalisation*

In this study, we introduced the Reduce-Reuse-Recycle (3R) batch normalisation to reduce the parameters associated with batch normalisation within the network. As illustrated in Fig. 2, the proposed model performed the following operations:

- Tensor RECYCLE: Here, the mean and variance tensors are recycled to avoid memory allocation and deallocation. Instead of allocating new memory for each forward pass, the same memory locations are recycled, leading to efficient memory management.
- Memory REDUCE through in-place computaion:In traditional batch normalisation, the computation of mean and variance usually entails generating intermediate tensors, which elevates memory usage, particularly for large batches or high-dimensional inputs. However, in our implementation, mean and variance are directly computed on the input tensors in place, eliminating the necessity for extra memory allocation.
- Operation REUSE:The reuse of operations for scaling and shifting across different branches of the input tensor eliminates the need to compute these operations separately for each branch. This approach enables a more efficient and concise implementation by applying the same operations to different parts of the input tensor.

Let $I \in \mathbb{R}^{H_{\text{in}} \times W_{\text{in}} \times C_{\text{in}}}$ denotes an input feature map produced by $L_{n-1}$, and $(M, N)$ denotes the output from the layer $L_{n-1}$. We feed the output $(M, N)$ into the batch normalisation layer. Here, $M$ denotes the output size from the layer, and $N$ denotes the channel size. For each layer, we compute $M$ and $N$ as follows:

$$M = \frac{H_{\text{in}} - F + 2P}{S} + 1 \tag{4}$$

Where:

- $M$: the output size from the layer
- $N$: the channel size
- $H_{\text{in}}$: height of the input feature map
- $W_{\text{in}}$: width of the input feature map
- $C_{\text{in}}$: number of channels in the input feature map
- $F$: filter size
- $S$: stride
- $P$: padding

Traditional batch normalisation calculates the mean ($\mu$) and variance ($\sigma^2$) of each feature channel across the entire batch. Mathematically, for a given input tensor $X$, the mean ($\mu$) and variance ($\sigma^2$) are calculated as follows:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} X_i \tag{5}$$

,

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)^2 \tag{6}$$

- $N$ is the total number of elements in the batch.
- $X_i$ represents each element in the batch.

After calculating the mean and variance, the input tensor is normalised using these values. The normalisation step ensures that the output has a zero mean and unit variance. However, this resulted in $4 \times C_{\text{in}}$ parameters due to four sets of learnable parameters present in the traditional one. We hence designed an enhanced batch normalisation by reducing the parameter to $2 \times C_{\text{in}}$ through the computation of the mean ($\mu_c$) and variance ($\sigma_c^2$) per channel ($c$) across all spatial dimensions ($H \times W$). The modified equations are as follows:

$$\mu_c = \frac{1}{N \times H \times W}\sum_{i=1}^{N}\sum_{j=1}^{H}\sum_{k=1}^{W} x_{ijkc} \tag{7}$$

$$\sigma_c^2 = \frac{1}{N \times H \times W}\sum_{i=1}^{N}\sum_{j=1}^{H}\sum_{k=1}^{W}(x_{ijkc} - \mu_c)^2 \tag{8}$$

After mean and variance computation, we normalise and further branch the input and perform scale and shift operations to reduce the time to compute batch processing. Overall, Our batch normalisation block effectively reduces the parameter count of the batch normalisation layer and thereby the convolutional neural network.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv1D | (None, 44, 2048) | 12,288 |
| MaxPooling1D | (None, 22, 2048) | 0 |
| CustomBatchNormalization | (None, 22, 2048) | 4,096 |
| Conv1D | (None, 22, 1024) | 10,486,784 |
| MaxPooling1D | (None, 11, 1024) | 0 |
| CustomBatchNormalization | (None, 11, 1024) | 2,048 |
| Conv1D | (None, 11, 512) | 2,621,952 |
| MaxPooling1D | (None, 6, 512) | 0 |
| CustomBatchNormalization | (None, 6, 512) | 1,024 |
| LSTM | (None, 6, 256) | 787,456 |
| LSTM | (None, 128) | 197,120 |
| Dense | (None, 128) | 16,512 |
| Dropout | (None, 128) | 0 |
| Dense | (None, 64) | 8,256 |
| Dropout | (None, 64) | 0 |
| Dense | (None, 32) | 2,080 |
| Dropout | (None, 32) | 0 |
| Dense | (None, 2) | 66 |

Table 2: Model Architecture

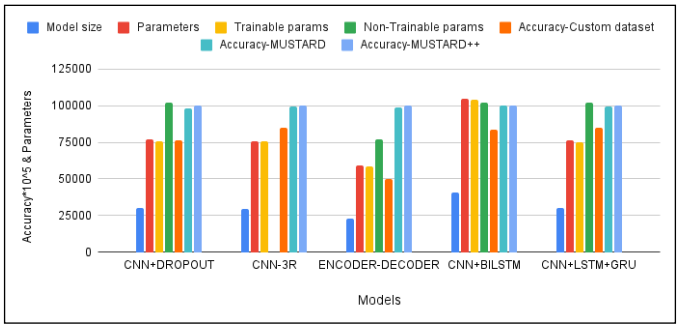


Figure 4: Accuracy and parameter size comparison for sarcasm applications

## 4. Experiments And Results

To validate our proposed method, we established an experimental platform and conducted various tests utilising diverse datasets. Our network model underwent comparison with others to gauge its effectiveness. Additionally, we investigated the effects of custom and traditional batch normalisation on the pooling layer and activation functions to assess classification accuracy and compression effects. These experiments were carried out on a computer equipped with an AMD Ryzen CPU running at 1965.9 MHz, as well as on a Google Cloud CPU. Energy consumption was measured using pyRAPL, a software toolkit that estimates power consumption during Python code execution, leveraging Intel's 'Running Average Power Limit' (RAPL) technology. The subsequent sections delve into the experiments conducted in detail.

### *4.1. Two-level verified Sarcasm Detection*

To demonstrate the effectiveness of our proposed model, illustrated in Fig. 3, we performed a comparative analysis with other networks, showcasing their classification accuracy using our custom dataset, MUSTARD [33], and MUSTARD++ [97]. The models we compared include CNN [98], CNN+Dropout, Encoder-Decoder, CNN+L1, CNN+LSTM [99], and CNN+BiLSTM. As shown in Fig. 4, our proposed network model achieves higher accuracy while utilising fewer parameters across the datasets. Additionally, our model maintains this high accuracy with a reduced number of network parameters compared to the other models.

### *4.2. Different energy-efficient techniques*

To evaluate the computational demands of our model and contrast them with alternative methods, we conducted a benchmarking study by varying the optimisation algorithms Adagrad [100], Adam [101], and SGD [102], both with and without custom batch normalization. This evaluation aimed to gauge the model's performance under various training conditions. We incorporated quantization and pruning techniques into our model on the CIFAR-10 dataset and performed experiments on Google Cloud CPU to evaluate its adaptability across different platforms. We explored various pruning methods, including neuron pruning, magnitude-based weight pruning, filter sparsity

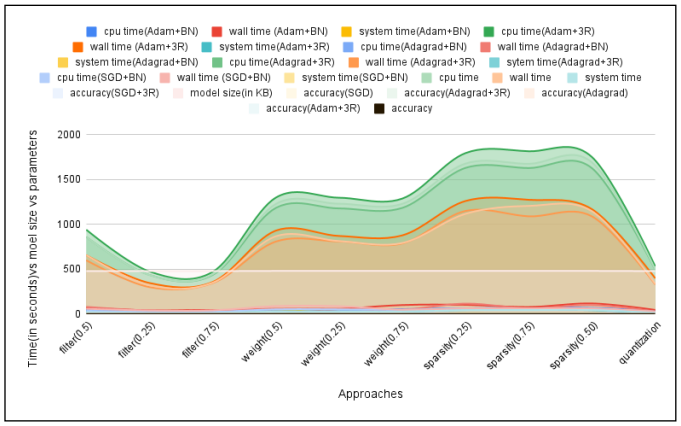


Figure 5: Time, accuracy and model size comparison of 3R with Pruning and Quantization with different optimizers.

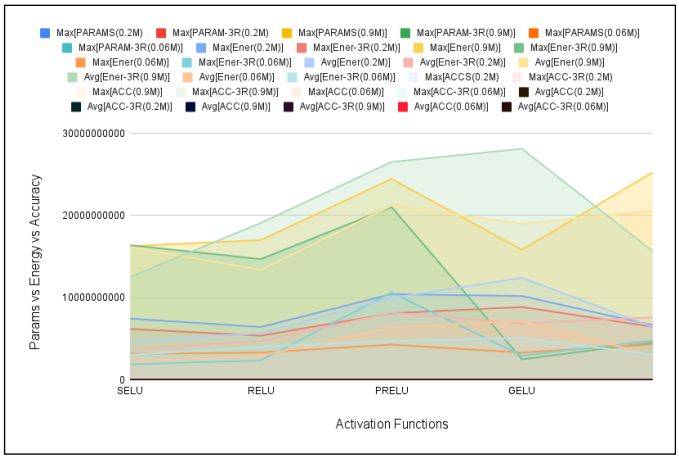


Figure 6: Energy consumption of different pooling function

pruning, and float16 model weight quantization. For our experiments, we utilised the network architecture shown in Table **??**, adapting it for CIFAR-10 by excluding the LSTM layer, as the dataset does not contain time-varying inputs. This comprehensive evaluation provided insights into the effectiveness and efficiency of our proposed model compared to established energy-efficient techniques.

The outcomes of the experiments on CIFAR10, as shown in Figure 5, revealed that the model size and parameter count remain consistent across different optimizers. However, metrics such as system time, CPU time, wall time, and accuracy vary depending on the optimizer used. This study leads us to conclude that in deep learning, the model size and parameter count do not directly determine the execution time and accuracy.

### *4.3. Different Application with 1D input*

Our experimental findings indicate that the proposed solution is appropriate for a two-class audio classifier, particularly for conducting energy-efficient CPU-on-device training. To evaluate its suitability for multiclass audio classification tasks, a comprehensive study is provided below.

#### *4.3.1. Different pooling function*

In this section, we investigate the adaptability of the model to different pooling functions and conduct a study on branch instructions and cache misses to analyse the bottlenecks during data transfer during training. Figure 10 illustrates that the proposed batch norm consumed less energy as compared to the CNN utilising the traditional batch norm. Figure 9 illustrates Branch instructions were greater in max pooling, while branch misses were higher in average pooling. Cache misses were generally higher in average pooling, except in the case of the PReLU activation function, where it had the fewest cache misses. Max pooling exhibited the highest cache misses. The model with 0.06 million parameters had the fewest CPU cycles, cache misses, and branch misses in both max and average pooling scenarios.

Figure 10 presents a comparison of energy consumption values

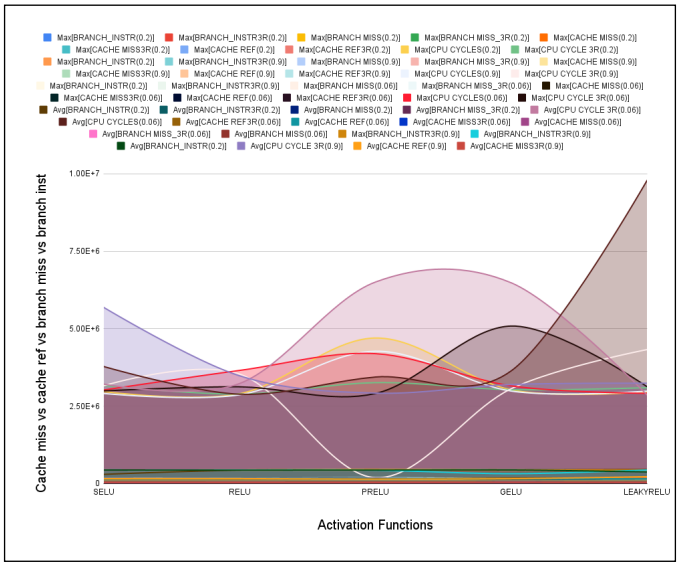


Figure 7: Cache references and cache misses with different activation functions with respect to to max and average pooling

across various activation functions and model sizes, considering both max pooling and average pooling scenarios. For each activation function, maximum and average energy consumption values are provided for different model sizes, ranging from 0.2M to 0.9M parameters. The results indicate notable variations in energy consumption across activation functions and pooling methods. In general, energy consumption tends to increase with larger model sizes and is influenced by the choice of activation function and pooling method. These findings offer valuable insights into optimising energy efficiency in neural network models by selecting appropriate activation functions and pooling strategies based on specific application requirements and computational resources. It can be concluded that the model size has a significant impact on cache misses and references.

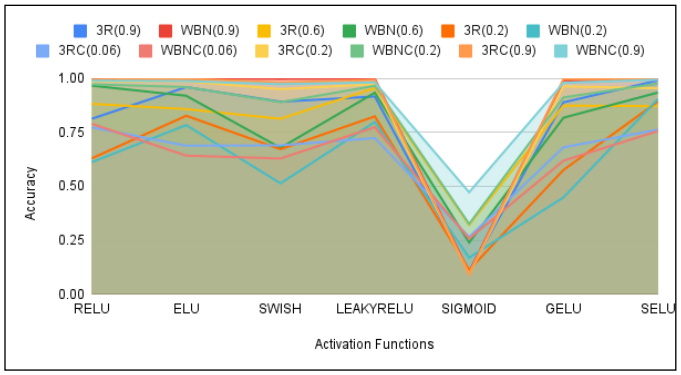


Figure 8: Accuracy comparison for different activation functions with different model sizes

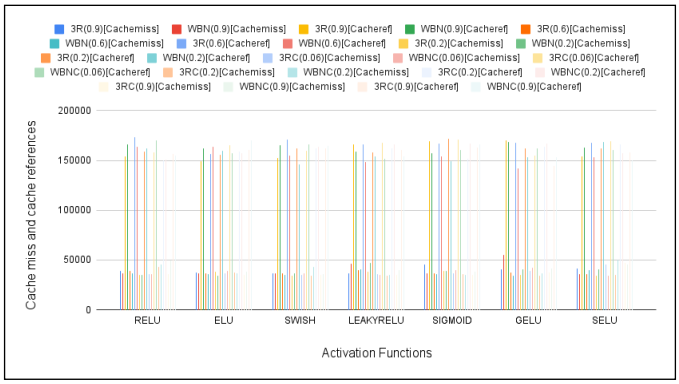


Figure 9: Cache references and cache misses with different activation functions

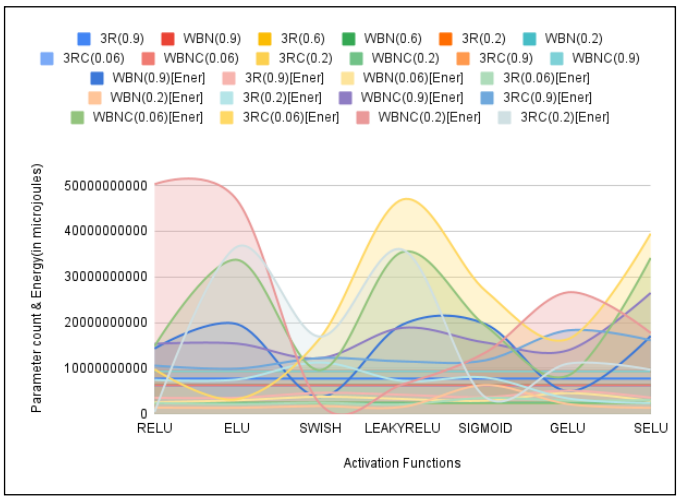


Figure 10: Energy comparison for different activation functions with different model size

### *4.3.2. Different Activation Function*

In this section, we investigated the adaptability of the model to different activation functions. We implemented the CNN architecture shown in Table **??** with varying sizes of X as 1024, 512, and 256, resulting in 0.9, 0.2, and 0.06 million parameters. These experiments were conducted on an AMD CPU using the TensorFlow framework. Additionally, we evaluated DRAM access based on cache references as part of this study. To verify the reproducibility of the results, we conducted experiments on two audio datasets: CREMA-D and RAVDESS.

Figures 8, 10, and 9 display the results of our experiments. Figure 8 showed that models with more parameters achieved higher accuracy. Our proposed model consistently outperformed others across all parameter variations with the CREMA-D dataset, while for the RAVDESS dataset, it achieved better accuracy only with 0.2 million parameters. Figure 10 illustrates that our proposed system consumed the least energy on the CREMA-D dataset compared to models using traditional batch normalisation on the RAVDESS dataset. This variation in accuracy and energy consumption may be attributed to the difference in dataset sizes, with the CREMA-D dataset being much smaller than the RAVDESS dataset.

Figure 9 illustrates the data transfer in the CNN. It was shown that, compared to the RAVDESS dataset, the CREMA-D dataset experienced significantly fewer cache misses and cache references when using our optimised batch normalisation, as opposed to the traditional batch normalization. In general, the CREMA-D dataset exhibited an overall increase in cache misses compared to the RAVDESS dataset. From this section, we concluded that different activation functions exhibit varying impacts on cache misses and energy consumption, and a reduction in cache misses does not always lead to reduced energy consumption.

### *4.4. Different Application with 2D Input*

To evaluate the adaptability and reproducibility of our proposed work in a different domain, we tested our model with 2D input. Given the increasing need for AI-based solutions in medical fields, we specifically examined whether our lightweight model supports applications in the medical domain, focusing on diabetic retinopathy detection. We selected this task due to its significance in medical diagnosis and the potential impact of AI-driven solutions on improving healthcare outcomes. We collected the dataset from Kaggle for the task.

Here we studied 10 pretrained models of TensorFlow. Table 3 illustrates the results. The results indicate that despite having the fewest parameters and the smallest model size, our proposed model achieved comparable accuracy to pretrained models. Although training from scratch took slightly more time, it's evident that energy consumption isn't directly tied to model size or parameter count. Variables such as cache references, cache misses, branch instructions, and CPU cycles remain largely independent of model size. This independence could be attributed to the efficiency of the model's architecture and optimisation techniques, highlighting the importance of design and implementation choices in determining computational performance.

### *4.5. Different Framework*

In our previous experiments, we demonstrated the versatility and effectiveness of our proposed model within the TensorFlow framework. Now, we aim to extend our investigation by evaluating its transferability to the PyTorch framework. To accomplish this, we conducted experiments focusing on the task of

Table 3: Evaluation of 3R Batch Normalization on Diabetic Retinopathy

| Model | Accuracy | Params | Non- Trainable | Trainable | Model Size | Energy | Cache Miss | Cache Ref | CPU Time | Wall Time | System Time | Branch Instr. | CPU Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VGG16 | 0.9268 | 20,024,384 | 20,024,384 | 0 | 76.39 MB | 32,381,004,504 | 38,281 | 160,095 | 2h 50min 31s | 19min 46s | 12min 16s | 482,374 | 4,461,369 |
| CNN-3R | 0.8546 | 658,420 | 0 | 658,420 | 2.51 MB | 23,297,113,923 | 38,982 | 154,114 | 1h 52min 2s | 16min 21s | 28min 1s | 479,873 | 5,764,402 |
| CNN+BN | 0.8025 | 658,612 | 192 | 658,420 | 2.51 MB | 16,706,036,092 | 38,962 | 159,572 | 1h 40min 27s | 16min 48s | 19min 19s | 481,266 | 3,866,706 |
| MobileNet | 0.9151 | 3,228,864 | 3,228,864 | 0 | 12.32 MB | 3,403,040,108 | 36,235 | 167,799 | 24min 55s | 3min 31s | 4min 34s | 480,679 | 3,204,176 |
| MobileNetV2 | 0.9043 | 2,257,984 | 2,257,984 | 0 | 477.25 KB | 3,705,314,942 | 36,004 | 167,876 | 28min 5s | 4min 17s | 7min 16s | 477,105 | 3,030,330 |
| NASNetMobile | 0.8851 | 4,269,716 | 4,269,716 | 0 | 16.29 MB | 5,021,896,895 | 37,627 | 154,526 | 38min 38s | 4min 52s | 4min 12s | 481,545 | 3,692,236 |
| MobileNetV3Small | 0.5441 | 939,120 | 939,120 | 0 | 3.58 MB | 1,281,377,506 | 35,797 | 165,497 | 7min 36s | 1min 32s | 34.2 s | 479,178 | 3,512,460 |
| EfficientNetB0 | 0.5315 | 4,049,571 | 4,049,571 | 0 | 15.45 MB | 5,737,505,120 | 35,275 | 166,614 | 42min 26s | 6min 36s | 12min 22s | 477,994 | 3,368,960 |
| EfficientNetV2B0 | 0.509 | 5,919,312 | 5,919,312 | 0 | 22.58 MB | 3,967,996,365 | 36,575 | 169,415 | 29min 59s | 3min 56s | 2min 45s | 475,178 | 3,390,313 |
| EfficientNetB1 | 0.5012 | 6,575,239 | 6,575,239 | 0 | 25.08 MB | 7,930,393,227 | 37,270 | 165,767 | 1h 5s | 8min 59s | 14min 49s | 476,883 | 3,203,157 |
| DenseNet121 | 0.9039 | 7,037,504 | 7,037,504 | 0 | 26.85 MB | 11,854,832,289 | 35,608 | 166,096 | 1h 23min 45s | 11min 44s | 16min 47s | 476,758 | 3,163,149 |
| EfficientNetV2B1 | 0.488 | 6,931,124 | 6,931,124 | 0 | 26.44 MB | 5,501,710,705 | 38,841 | 171,803 | 40min 43s | 5min 35s | 4min 1s | 476,949 | 3,174,329 |

Table 4: Comparison of CPU time, wall time, system time, memory usage, CPU usage, and accuracy for different models over 20 epochs.

| Epochs = 20 | | | | | | | Epochs = 15 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | CPU Time | Wall Time | System Time | Memory Usage (%) | CPU Usage (%) | Accuracy | Model | CPU Time | Wall Time | System Time | Memory Usage (%) | CPU Usage (%) | Accuracy |
| CONV+WBN | 1h 23min 26s | 1h 45min 25s | 13min 28s | 19 | 78.1 | 0.748 | CONV+WBN | 1h 1min 50s | 1h 19min 12s | 10min 35s | 11.6 | 76.5 | 0.667 |
| CNN-3R | 54.5 s | 6min 38s | 8.34 s | 19.8 | 77.3 | 0.812 | PROPOSED | 53min 56s | 1h 29min 44s | 30min 26s | 18.9 | 76 | 0.854 |
| CNN+BN | 1h 9min 37s | 1h 30min 55s | 13min 17s | 27.3 | 79.2 | 0.833 | CNN+BN | 46min 27s | 1h 2min 11s | 10min 37s | 18.8 | 77 | 0.896 |
| VGG16 | 3h 3min 29s | 3h 48min 25s | 35min 28s | 24.6 | 75.9 | 0.812 | VGG16 | 2h 41min 28s | 3h 22min 59s | 32min 47s | 30.1 | 77.4 | 0.792 |
| RESNET18 | 1h 22min 1s | 1h 45min 44s | 16min 12s | 20.1 | 76.6 | 0.896 | RESNET18 | 1h 4min 11s | 1h 21min 35s | 11min 17s | 34.6 | 78.5 | 0.875 |
| EFF-B7 | 2h 31min 7s | 4h 49min 15s | 2h 7min 32s | 27.9 | 74.9 | 0.625 | EFF-B7 | 2h 6min 56s | 3h 54min 20s | 1h 39min 8s | 19.7 | 76.4 | 0.792 |
| EFF-B6 | 1h 48min 33s | 3h 22min 52s | 1h 25min 24s | 22.1 | 74.8 | 0.708 | EFF-B6 | 3min 23s | 10min 4s | 1min 37s | 18.8 | 89.5 | 0.771 |
| EFF-B5 | 1h 25min 11s | 2h 49min 26s | 1h 15min 42s | 19 | 77.1 | 0.729 | EFF-B5 | 1h 9min 34s | 2h 17min 15s | 1h 23s | 18 | 77 | 0.771 |
| EFF-B4 | 58min 11s | 1h 58min 11s | 51min 48s | 26.6 | 77.9 | 0.771 | EFF-B4 | 48min 42s | 1h 39min 39s | 44min 3s | 17.3 | 78 | 0.667 |
| EFF-B3 | 46min 30s | 1h 38min 21s | 43min 33s | 14.6 | 75.6 | 0.75 | EFF-B3 | 35min 48s | 1h 12min 27s | 30min 48s | 23.4 | 78.7 | 0.646 |
| EFF-B2 | 34min 51s | 1h 11min 6s | 28min 4s | 16.2 | 77 | 0.729 | EFF-B2 | 23min 19s | 49min 45s | 20min 49s | 77.5 | 20.7 | 0.812 |
| EFF-B1 | 31min 5s | 1h 9min 12s | 30min 27s | 15.6 | 81.1 | 0.854 | EFF-B1 | 24min 50s | 52min 37s | 22min 6s | 18.1 | 77.8 | 0.875 |
| mobilenetv2 | 18min 36s | 45min 51s | 19min 15s | 18.2 | 81.6 | 0.812 | mobilenetv2 | 13min 46s | 32min 53s | 13min 38s | 17.6 | 80.5 | 0.854 |
| mobilenetv3_small | 4min 39s | 14min 1s | 2min 25s | 16.1 | 89.6 | 0.75 | mobilenetv3_small | 3min 23s | 10min 4s | 1min 37s | 18.8 | 89.5 | 0.771 |

mineral classification. In this study, we examined eleven pre-trained models available in the PyTorch library and utilised a Kaggle dataset for mineral classification [103]. Table 4 illustrates the results. We studied at different values of epochs 15 and 20, respectively. The results indicate a correlation between larger epoch numbers and increased CPU and memory usage, as well as improved accuracy. This observation suggests that epochs have a significant impact on both memory and CPU utilization. The reason behind this dependency lies in the iterative nature of training neural networks. With each epoch, the model undergoes multiple iterations over the entire dataset, adjusting its parameters to minimise the loss function. As a result, more epochs lead to increased computational demand, as the model performs more computations and requires more memory to store intermediate results. It's evident that our model, despite requiring less time for training, achieves good accuracy compared to other models. Furthermore, the relationship between epochs and accuracy underscores the importance of allowing the model to train for an adequate number of epochs to converge to an optimal solution. While increasing epoch numbers may incur higher computational costs, it often results in better model performance by allowing the model to capture intricate patterns in the data.In summary, the observed dependencies highlight the trade-off between computational resources and model performance, emphasising the need to carefully select the number of epochs based on available resources and desired accuracy levels.

## 5. Conclusion and Future Work

In this paper, we address the pressing need for energy-efficient CNN designs for IoT devices by proposing a novel approach to optimise batch normalisation operations. Our study demonstrated lightweight batch normalization. By focusing on multimodal sarcasm detection and other applications, we showed the effectiveness of our proposed CNN architecture in various different practical applications. As a future project, we plan to train our model in a controlled environment and plan to evaluate its energy consumption during real-time inference.

## 6. Acknowledgements

This research work was partially supported from GOOGLE Cloud Research Credits Programe under EDU Credit 324579202.

## References

[1] L. Alzubaidi, J. Zhang, A. J. Humaidi, *et al.*, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[2] Z. Zhang and A. Z. Kouzani, "Implementation of dnns on iot devices," *Neural Computing and Applications*, vol. 32, no. 5, pp. 1327–1356, 2020.

[3] J. M. Jose and S. Benedict, "Deepasd framework: A deep learning-assisted automatic sarcasm detection in facial emotions," in *2023 8th International Conference on Communication and Electronics Systems (ICCES)*, IEEE, 2023, pp. 998–1004.

[4] H.-I. Liu, M. Galindo, H. Xie, *et al.*, "Lightweight deep learning for resource-constrained environments: A survey," *ACM Computing Surveys*, 2024.

[5] K. Y. Chan, B. Abu-Salih, R. Qaddoura, *et al.*, "Deep neural networks in the cloud: Review, applications, challenges and research directions," *Neurocomputing*, p. 126 327, 2023.

[6] M. Christopher, J. M. Jose, T. Thomas, and S. Benedict, "Catboost and genetic algorithm implementations for university recommendation systems," in *2022 International Conference on Inventive Computation Technologies (ICICT)*, IEEE, Jul. 2022, pp. 436–443. DOI: 10.1109/ICICT54344.2022.9850819.

[7] J. M. Jose and J. Jeeva, "Energy-reduced bio-inspired 1d-cnn for audio emotion recognition," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 11, no. 3, Jun. 2023. DOI: 10.32628/CSEIT25113386. [Online]. Available: https://doi.org/10.32628/CSEIT25113386.

[8] H. Wu, X. Li, and Y. Deng, "Deep learning-driven wireless communication for edge-cloud computing: Opportunities and challenges," *Journal of Cloud Computing*, vol. 9, no. 1, p. 21, 2020.

[9] L. Ye, Z. Wang, Y. Liu, *et al.*, "The challenges and emerging technologies for low-power artificial intelligence iot systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 12, pp. 4821–4834, 2021.

[10] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: An empirical study of their impact to deep learning," *Multimedia tools and applications*, vol. 79, no. 19, pp. 12 777–12 815, 2020.

[11] M. Awais, M. T. B. Iqbal, and S.-H. Bae, "Revisiting internal covariate shift for batch normalization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, pp. 5082–5092, 2020.

[12] R. Banner, I. Hubara, E. Hoffer, and D. Soudry, "Scalable methods for 8-bit training of neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[13] M. M. Kalayeh and M. Shah, "Training faster by separating modes of variation in batch-normalized models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 6, pp. 1483–1500, 2019.

[14] H. Daneshmand, J. Kohler, F. Bach, T. Hofmann, and A. Lucchi, "Batch normalization provably avoids ranks collapse for randomly initialised deep networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 387–18 398, 2020.

[15] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" *Advances in neural information processing systems*, vol. 31, 2018.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pmlr, 2015, pp. 448–456.

[17] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "Normalization techniques in training dnns: Methodology, analysis and application," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 8, pp. 10 173–10 196, 2023.

[18] L. Huang and A. D. Galinsky, "Was that sarcastic or supportive? why receiving sarcasm improves perspective-taking," *Current Opinion in Psychology*, p. 101 709, 2023.

[19] A. Lieberman and J. Schroeder, "Two social lives: How differences between online and offline interaction influence social outcomes," *Current opinion in psychology*, vol. 31, pp. 16–21, 2020.

[20] S. Gates, "Recent dickens studies—2015," *Dickens Studies Annual: Essays on Victorian Fiction*, vol. 48, no. 1, pp. 285–394, 2017.

[21] Y. Yeshurun, M. Nguyen, and U. Hasson, "The default mode network: Where the idiosyncratic self meets the shared social world," *Nature reviews neuroscience*, vol. 22, no. 3, pp. 181–192, 2021.

[22] J. K. McNulty, "When positive processes hurt relationships," *Current directions in psychological science*, vol. 19, no. 3, pp. 167–171, 2010.

[23] D. Keltner, L. Capps, A. M. Kring, R. C. Young, and E. A. Heerey, "Just teasing: A conceptual analysis and empirical review.," *Psychological bulletin*, vol. 127, no. 2, p. 229, 2001.

[24] J. M. Jose and J. Jose, "An efficient sarcasm detection in audio using parameter-reduced depthwise cnn," *International Journal of Innovative Research in Technology (IJIRT)*, vol. 12, no. 1, Jun. 2025, Independent Researcher, ISSN: 2349-6002.

[25] F. Quesque and Y. Rossetti, "What do theory-of-mind tasks actually measure? theory and practice," *Perspectives on Psychological Science*, vol. 15, no. 2, pp. 384–396, 2020.

[26] J. Cui, H. L. Colston, and G. Jiang, "Is that a genuine smile? emoji-based sarcasm interpretation across the lifespan," *Metaphor and Symbol*, vol. 39, no. 3, pp. 195–216, 2024.

[27] J. M. Jose, "Optimizing neural network energy efficiency through low-rank factorisation and pde-driven dense layers," *International Journal of Research Publication and Reviews*, vol. 2, no. 2, pp. 5483–5487, 2025, ISSN: 2022. [Online]. Available: `https://www.ijrpr.com`.

[28] S. Benedict, S. V. Reddy, M. Bhagyalakshmi, J. M. Jose, and R. Prodan, "Performance improvement strategies of edge-enabled social impact applications," in *2023 International Conference on Inventive Computation Technologies (ICICT)*, IEEE, 2023, pp. 1696–1703. DOI: `10.1109/ICICT57992.2023.10160004`. [Online]. Available: `https://doi.org/10.1109/ICICT57992.2023.10160004`.

[29] S. K. Kumar and J. M. Jose, "A survey on synthesizing images with generative adversial networks," *International Journal of Research Publication and Reviews*, vol. 2, no. 2, p. 5, 2021. [Online]. Available: `https://www.ijrpr.com`.

[30] E. O. Lange, J. M. Jose, S. Benedict, and M. Gerndt, "Automated energy modeling framework for microcontroller-based edge computing nodes," in *International Conference on Advanced Network Technologies and Intelligent Computing*, Springer Nature Switzerland, 2022, pp. 422–437. DOI: `10.1007/978-3-031-27082-5_32`. [Online]. Available: `https://doi.org/10.1007/978-3-031-27082-5_32`.

[31] D. Vinoth and P. Prabhavathy, "An intelligent machine learning-based sarcasm detection and classification model on social networks," *The Journal of Supercomputing*, vol. 78, no. 8, pp. 10 575–10 594, 2022.

[32] A. Jindal, J. M. Jose, S. Benedict, and M. Gerndt, "Lora-powered energy-efficient object detection mechanism in edge computing nodes," in *2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, IEEE, 2022, pp. 237–244. DOI: `10.1109/I-SMAC55078.2022.9984414`. [Online]. Available: `https://doi.org/10.1109/I-SMAC55078.2022.9984414`.

[33] S. Castro, D. Hazarika, V. Pérez-Rosas, R. Zimmermann, R. Mihalcea, and S. Poria, "Towards multimodal sarcasm detection (an _obviously_ perfect paper)," *arXiv preprint arXiv:1906.01815*, 2019.

[34] C. I. Eke, A. A. Norman, L. Shuib, and H. F. Nweke, "Sarcasm identification in textual data: Systematic review, research challenges and open directions," *Artificial Intelligence Review*, vol. 53, pp. 4215–4258, 2020.

[35] A. Al Maruf, F. Khanam, M. M. Haque, Z. M. Jiyad, F. Mridha, and Z. Aung, "Challenges and opportunities of text-based emotion detection: A survey," *IEEE Access*, 2024.

[36] A.-C. Băroiu and Ș. Trăușan-Matu, "Automatic sarcasm detection: Systematic literature review," *Information*, vol. 13, no. 8, p. 399, 2022.

[37] A. Jamali, S. K. Roy, and P. Ghamisi, "Wetmapformer: A unified deep cnn and vision transformer for complex wetland mapping," *International Journal of Applied Earth Observation and Geoinformation*, vol. 120, p. 103 333, 2023.

[38] H. Fang, D. Liang, and W. Xiang, "Single-stage extensive semantic fusion for multi-modal sarcasm detection," *Array*, vol. 22, p. 100 344, 2024.

[39] S. Sharma, S. Ramaneswaran, M. S. Akhtar, and T. Chakraborty, "Emotion-aware multimodal fusion for meme emotion detection," *IEEE Transactions on Affective Computing*, 2024.

[40] J. Wang, Y. Yang, Y. Jiang, M. Ma, Z. Xie, and T. Li, "Cross-modal incongruity aligning and collaborating for multi-modal sarcasm detection," *Information Fusion*, vol. 103, p. 102 132, 2024.

[41] D. Jain, A. Kumar, and G. Garg, "Sarcasm detection in mash-up language using soft-attention based bi-directional lstm and feature-rich cnn," *Applied Soft Computing*, vol. 91, p. 106 198, 2020.

[42] B. N. Hiremath and M. M. Patil, "Sarcasm detection using cognitive features of visual data by learning model," *Expert Systems with Applications*, vol. 184, p. 115 476, 2021.

[43] S. Das and A. K. Kolya, "Parallel deep learning-driven sarcasm detection from pop culture text and english humor literature," in *Proceedings of Research and Applications in Artificial Intelligence: RAAI 2020*, Springer, 2021, pp. 63–73.

[44] M. Bedi, S. Kumar, M. S. Akhtar, and T. Chakraborty, "Multi-modal sarcasm detection and humor classification in code-mixed conversations," *IEEE Transactions on Affective Computing*, vol. 14, no. 2, pp. 1363–1375, 2021.

[45] F. Yao, X. Sun, H. Yu, W. Zhang, W. Liang, and K. Fu, "Mimicking the brain's cognition of sarcasm from multidisciplines for twitter sarcasm detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 1, pp. 228–242, 2021.

[46] J. Li, H. Pan, Z. Lin, P. Fu, and W. Wang, "Sarcasm detection with commonsense knowledge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3192–3201, 2021.

[47] D. S. Chauhan, G. V. Singh, A. Arora, A. Ekbal, and P. Bhattacharyya, "An emoji-aware multitask framework for multimodal sarcasm detection," *Knowledge-Based Systems*, vol. 257, p. 109 924, 2022.

[48] P. Goel, R. Jain, A. Nayyar, S. Singhal, and M. Srivastava, "Sarcasm detection using deep learning and ensemble learning," *Multimedia Tools and Applications*, vol. 81, no. 30, pp. 43 229–43 252, 2022.

[49] N. Ding, S.-w. Tian, and L. Yu, "A multimodal fusion method for sarcasm detection based on late fusion," *Multimedia Tools and Applications*, vol. 81, no. 6, pp. 8597–8616, 2022.

[50] B. Li, Z. Qian, P. Li, and Q. Zhu, "Multi-modal fusion network for rumor detection with texts and images," in *International Conference on Multimedia Modeling*, Springer, 2022, pp. 15–27.

[51] T.-h. Cheung and K.-m. Lam, "Crossmodal bipolar attention for multimodal classification on social media," *Neurocomputing*, vol. 514, pp. 1–12, 2022.

[52] T. Yue, R. Mao, H. Wang, Z. Hu, and E. Cambria, "Knowlenet: Knowledge fusion network for multimodal sarcasm detection," *Information Fusion*, vol. 100, p. 101 921, 2023.

[53] Y. Zhang, J. Wang, Y. Liu, *et al.*, "A multitask learning model for multimodal sarcasm, sentiment and emotion recognition in conversations," *Information Fusion*, vol. 93, pp. 282–301, 2023.

[54] A. Kumar, S. R. Sangwan, A. K. Singh, and G. Wadhwa, "Hybrid deep learning model for sarcasm detection in indian indigenous language using word-emoji embeddings," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 22, no. 5, pp. 1–20, 2023.

[55] Y. Zhang, Y. Yu, D. Zhao, *et al.*, "Learning multi-task commonness and uniqueness for multi-modal sarcasm detection and sentiment analysis in conversation," *IEEE Transactions on Artificial Intelligence*, 2023.

[56] C. Zhu, M. Chen, S. Zhang, *et al.*, "Skeafn: Sentiment knowledge enhanced attention fusion network for multimodal sentiment analysis," *Information Fusion*, vol. 100, p. 101 958, 2023.

[57] J. Liu, S. Tian, L. Yu, X. Shi, and F. Wang, "Image-text fusion transformer network for sarcasm detection," *Multimedia Tools and Applications*, pp. 1–15, 2023.

[58] A. Alzu'bi, L. Bani Younis, A. Abuarqoub, and M. Hammoudeh, "Multimodal deep learning with discriminant descriptors for offensive memes detection," *ACM Journal of Data and Information Quality*, vol. 15, no. 3, pp. 1–16, 2023.

[59] Y. Zhang, D. Ma, P. Tiwari, *et al.*, "Stance-level sarcasm detection with bert and stance-centered graph attention networks," *ACM Transactions on Internet Technology*, vol. 23, no. 2, pp. 1–21, 2023.

[60] Y. Liu, Y. Zhang, and D. Song, "A quantum probability driven framework for joint multi-modal sarcasm, sentiment and emotion analysis," *IEEE Transactions on Affective Computing*, 2023.

[61] R. M. Albalawi, A. T. Jamal, A. O. Khadidos, and A. M. Alhothali, "Multimodal arabic rumors detection," *IEEE Access*, vol. 11, pp. 9716–9730, 2023.

[62] A. Mohan, A. M. Nair, B. Jayakumar, and S. Muraleedharan, "Sarcasm detection using bidirectional encoder representations from transformers and graph convolutional networks," *Procedia Computer Science*, vol. 218, pp. 93–102, 2023.

[63] O. Vitman, Y. Kostiuk, G. Sidorov, and A. Gelbukh, "Sarcasm detection framework using context, emotion and sentiment features," *Expert Systems with Applications*, vol. 234, p. 121 068, 2023.

[64] X. Guo, A. Kot, and A. W.-K. Kong, "Pace-adaptive and noise-resistant contrastive learning for multimodal feature fusion," *IEEE Transactions on Multimedia*, 2023.

[65] S. Chatterjee, S. Bhattacharjee, K. Ghosh, A. K. Das, and S. Banerjee, "Class-biased sarcasm detection using bilstm variational autoencoder-based synthetic oversampling," *Soft Computing*, vol. 27, no. 9, pp. 5603–5620, 2023.

[66] L. Wu, Y. Long, C. Gao, Z. Wang, and Y. Zhang, "Mfir: Multimodal fusion and inconsistency reasoning for explainable fake news detection," *Information Fusion*, vol. 100, p. 101 944, 2023.

[67] V. Sukhavasi and V. Dondeti, "Effective automated transformer model based sarcasm detection using multilingual data," *Multimedia Tools and Applications*, pp. 1–32, 2023.

[68] Y. Zhang, Y. Yu, M. Wang, M. Huang, and M. S. Hossain, "Self-adaptive representation learning model for multi-modal sentiment and sarcasm joint analysis," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 20, no. 5, pp. 1–17, 2024.

[69] H. Liu, R. Wei, G. Tu, J. Lin, C. Liu, and D. Jiang, "Sarcasm driven by sentiment: A sentiment-aware hierarchical fusion network for multimodal sarcasm detection," *Information Fusion*, vol. 108, p. 102 353, 2024.

[70] Y. Li, Y. Li, S. Zhang, *et al.*, "An attention-based, context-aware multimodal fusion method for sarcasm detection using inter-modality inconsistency," *Knowledge-Based Systems*, vol. 287, p. 111 457, 2024.

[71] H. Dai, H. Wang, X. Zhang, and H. Sun, "Memory-efficient batch normalization by one-pass computation for on-device training," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2024.

[72] H. Zhang, Y. Shu, Q. Deng, H. Sun, W. Zhao, and Y. Ha, "Wdvr-ram: A 0.25–1.2 v, 2.6–76 pops/w charge-domain in-memory-computing binarized cnn accelerator for dynamic aiot workloads," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 10, pp. 3964–3977, 2023. DOI: 10.1109/TCSI.2023.3294296.

[73] B. Li, H. Wang, F. Luo, X. Zhang, H. Sun, and N. Zheng, "Acbn: Approximate calculated batch normalization for efficient dnn on-device training processor," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 6, pp. 738–748, 2023. DOI: 10.1109/TVLSI.2023.3262787.

[74] R. Khurana and S. Hodges, "Beyond the prototype: Understanding the challenge of scaling hardware device production," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–11.

[75] K. F. Goser, "Implementation of artificial neural networks into hardware: Concepts and limitations," *Mathematics and computers in simulation*, vol. 41, no. 1-2, pp. 161–171, 1996.

[76] T. V. Lakshmi and C. V. Krishna Reddy, "Classification of skin lesions by incorporating drop-block and batch normalization layers in representative cnn models," *Arabian Journal for Science and Engineering*, vol. 49, no. 3, pp. 3671–3684, 2024.

[77] H.-I. Lin, R. Mandal, and F. S. Wibowo, "Bn-lstm-based energy consumption modeling approach for an industrial robot manipulator," *Robotics and Computer-Integrated Manufacturing*, vol. 85, p. 102 629, 2024.

[78] L. Zhao, Y. Teng, and L. Wang, "Logit normalization for long-tail object detection," *International Journal of Computer Vision*, pp. 1–21, 2024.

[79] S. Tang, B. C. Khoo, Y. Zhu, K. M. Lim, and S. Yuan, "A light deep adaptive framework toward fault diagnosis of a hydraulic piston pump," *Applied Acoustics*, vol. 217, p. 109 807, 2024.

[80] C. Yang, Z. Lin, Z. Lan, R. Chen, L. Wei, and Y. Liu, "Evolutionary channel pruning for real-time object detection," *Knowledge-Based Systems*, vol. 287, p. 111 432, 2024.

[81] J. M. Jose, "Edge intelligence: Architecture, scope and applications," *International Journal of Research Publication and Reviews*, vol. 2, no. 2, p. 5, 2022. [Online]. Available: https://www.ijrpr.com.

[82] B. Liang, L. Gui, Y. He, E. Cambria, and R. Xu, "Fusion and discrimination: A multimodal graph contrastive learning framework for multimodal sarcasm detection," *IEEE Transactions on Affective Computing*, 2024.

[83] P. Tiwari, L. Zhang, Z. Qu, and G. Muhammad, "Quantum fuzzy neural network for multimodal sentiment and sarcasm detection," *Information Fusion*, vol. 103, p. 102 085, 2024.

[84] H. Fang, D. Liang, and W. Xiang, "Multi-modal sarcasm detection based on multi-channel enhanced fusion model," *Neurocomputing*, p. 127 440, 2024.

[85] S. Kusal, S. Patil, J. Choudrie, K. Kotecha, D. Vora, and I. Pappas, "A systematic review of applications of natural language processing and future challenges with special emphasis in text-based emotion detection," *Artificial Intelligence Review*, vol. 56, no. 12, pp. 15 129–15 215, 2023.

[86] L. K. Chan, N. G. Patil, J. Y. Chen, J. C. Lam, C. S. Lau, and M. S. Ip, "Advantages of video trigger in problem-based learning," *Medical teacher*, vol. 32, no. 9, pp. 760–765, 2010.

[87] H. Cao, D. G. Cooper, M. K. Keutmann, R. C. Gur, A. Nenkova, and R. Verma, "Crema-d: Crowd-sourced emotional multimodal actors dataset," *IEEE transactions on affective computing*, vol. 5, no. 4, pp. 377–390, 2014.

[88] H. S. Cheang and M. D. Pell, "The sound of sarcasm," *Speech communication*, vol. 50, no. 5, pp. 366–381, 2008.

[89] S. Abbas, S. Ojo, A. Al Hejaili, *et al.*, "Artificial intelligence framework for heart disease classification from audio signals," *Scientific Reports*, vol. 14, no. 1, p. 3123, 2024.

[90] S. A. Qureshi, L. Hussain, M. Rafique, *et al.*, "Eml-psp: A novel ensemble machine learning-based physical security paradigm using cross-domain ultra-fused feature extraction with hybrid data augmentation scheme," *Expert Systems with Applications*, vol. 243, p. 122 863, 2024.

[91] A. Sujeesha, J. Mala, and R. Rajan, "Automatic music mood classification using multi-modal attention framework," *Engineering Applications of Artificial Intelligence*, vol. 128, p. 107 355, 2024.

[92] S. Attardo, J. Eisterhold, J. Hay, and I. Poggi, "Multimodal markers of irony and sarcasm," 2003.

[93] S. Tabacaru and M. Lemmens, "Raised eyebrows as gestural triggers in humour: The case of sarcasm and hyper-understanding," *The European Journal of Humour Research*, vol. 2, no. 2, pp. 11–31, 2014.

[94] S. Tabacaru, "Faces of sarcasm," *The diversity of irony*, vol. 65, p. 256, 2020.

[95] J. Mounts, "A history of sarcasm: Effects of balanced use of sarcasm in a relationship," 2012.

[96] J. Kim, "How korean efl learners understand sarcasm in l2 english," *Journal of Pragmatics*, vol. 60, pp. 193–206, 2014.

[97] A. Ray, S. Mishra, A. Nunna, and P. Bhattacharyya, "A multimodal corpus for emotion recognition in sarcasm," *arXiv preprint arXiv:2206.02119*, 2022.

[98] F. Schmid, K. Koutini, and G. Widmer, "Dynamic convolutional neural networks as efficient pre-trained audio models," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[99] J. Zhao, X. Mao, and L. Chen, "Speech emotion recognition using deep 1d & 2d cnn lstm networks," *Biomedical signal processing and control*, vol. 47, pp. 312–323, 2019.

[100] K. Chandra, A. Xie, J. Ragan-Kelley, and E. Meijer, "Gradient descent: The ultimate optimizer," *Advances in Neural Information Processing Systems*, vol. 35, pp. 8214–8225, 2022.

[101] S. Bock, J. Goppold, and M. Weiß, "An improvement of the convergence proof of the adam-optimizer," *arXiv preprint arXiv:1804.10587*, 2018.

[102] J. Duda, "Sgd momentum optimizer with step estimation by online parabola model," *arXiv preprint arXiv:1907.07063*, 2019.

[103] Kaggle, *Minerals Identification Dataset*, `https://www.kaggle.com/datasets/asiedubrempong/minerals-identification-dataset`, 2024.