THE UNIVERSITY OF SYDNEY

The University of Sydney

COMP 5329 Deep Learning

Enhancing Multi-Label Image Classification with Integrated Visual-Textual Data Analysis: A Comparative Study of CLIP and Hybrid Neural Network Architectures

Team Members:
Calvin Li (490236424)
Zichao Wang (480004000)
Theresa Wang (490388169)

Lecturer:
Chang Xu

MAY 2024

Abstract

This study examines multi-label image classification challenges within a Kagle competition context, employing two distinct neural network architectures: the CLIP model, utilizing a transformer-based approach for integrating visual and textual data, and a hybrid model that combines GoogLeNet with LSTM to process image features alongside sequential text. The dataset, characterized by significant class imbalance, is addressed through sophisticated preprocessing techniques, the implementation of Focal Loss, and strategic data resampling to enhance model training.

Experimental results demonstrate the superiority of the CLIP model in terms of precision and validation accuracy. This model's success can be attributed to its ability to generalize effectively from diverse and extensive pre-training on internet-sourced data, which is essential given the multimodal nature and imbalance present in the dataset. The CLIP model's dual encoding capability efficiently handles both visual and textual inputs, providing a robust framework for understanding complex data interactions.

Additionally, the research highlights the critical role of precise model selection, advanced data preprocessing, and the optimization of loss functions and optimizers in improving performance in multi-label classification tasks. By leveraging textual information to refine visual data classification, the study advances computational tools and strategies, ensuring higher predictive accuracy and robustness in handling real-world data complexities.

Contents

1 Introduction **3**
- 1.1 Research Goal . . . 3
- 1.2 Research Importance . . . 3
- 1.3 Dataset overview . . . 4
  - 1.3.1 Label Imbalanced Issue . . . 4
- 1.4 Methods and Solutions . . . 5
  - 1.4.1 CLIP Model . . . 5
  - 1.4.2 GoogLeNet + LSTM . . . 5
  - 1.4.3 Addressing Imbalanced Label . . . 5

2 Related Works **6**
- 2.1 Multi Label Classification Task . . . 6
  - 2.1.1 Early Solutions and the Rise of CNNs . . . 6
  - 2.1.2 2014 - Innovations and Breakthroughs with GoogLeNet . . . 6
  - 2.1.3 2021 - The CLIP Model . . . 6
- 2.2 Vision Models . . . 7
  - 2.2.1 GoogLeNet . . . 7
  - 2.2.2 ResNet . . . 7
  - 2.2.3 EfficientNet . . . 8

3 Techniques **9**
- 3.1 Label One-Hot Encoding . . . 9
  - 3.1.1 Principle . . . 9
  - 3.1.2 Usage in Neural Networks: . . . 9
  - 3.1.3 Justification . . . 9
  - 3.1.4 Advantages or Novelty . . . 9
- 3.2 Caption Tokenization . . . 10
  - 3.2.1 Byte Pair Encoding for CLIP Model . . . 10
  - 3.2.2 GoogLeNet with LSTM Model Caption Tokenization . . . 10
  - 3.2.3 Justification . . . 11
  - 3.2.4 Advantages or Novelty . . . 12
- 3.3 Image Preprocessing . . . 12
  - 3.3.1 Justification . . . 13
  - 3.3.2 Advantages or Novelty . . . 13
- 3.4 Weighted Random Sampler . . . 13
  - 3.4.1 Principle . . . 13
  - 3.4.2 Justification . . . 14
  - 3.4.3 Novelty . . . 14
- 3.5 Optimizer . . . 14
  - 3.5.1 Adaptive Moment Estimation (ADAM) . . . 14
  - 3.5.2 AdamW . . . 15
  - 3.5.3 Justification . . . 15
  - 3.5.4 Advantages or Novelty . . . 16
- 3.6 Loss Function . . . 16
  - 3.6.1 Binary Cross-Entropy (BCE) Loss . . . 16
  - 3.6.2 Focal Loss . . . 16
  - 3.6.3 Justification . . . 16
  - 3.6.4 Advantages or Novelty . . . 17
- 3.7 GoogLeNet Design . . . 17

3.7.1 Principles . . . 17
3.7.2 Justification . . . 18
3.7.3 Advantages or Novelty . . . 18
3.8 LSTM . . . 18
3.8.1 Principles . . . 18
3.8.2 Justification . . . 20
3.8.3 Advantages or Novelty . . . 20
3.9 CLIP Model . . . 20
3.9.1 Principles of CLIP . . . 21
3.9.2 Justification . . . 22
3.9.3 Advantages or Novelty . . . 23

4 Experimentation and Results **23**
4.1 Experiment Models . . . 23
4.1.1 CLIP . . . 23
4.1.2 GoogLeNet + LSTM . . . 24
4.2 Hyperparameter Analysis . . . 25
4.2.1 GoogLeNet + LSTM . . . 25
4.3 Ablation Studies . . . 26
4.3.1 CLIP . . . 26
4.3.2 GoogLeNet + LSTM . . . 28
4.4 Model Comparison and Performance Evaluation . . . 29

5 Conclusion and Limitations **30**
5.1 Final Model Choice . . . 30
5.2 Limitations . . . 31
5.2.1 Complex Relationships and Bias . . . 31
5.2.2 Granular Detail Recognition . . . 31
5.2.3 Data Processing Enhancements . . . 31
5.2.4 Rigorous Validation Methods . . . 32

Appendices **34**

A Hardware and Software Specification **34**

B Instruction of Compiling Code **35**

C Disclaimer **36**

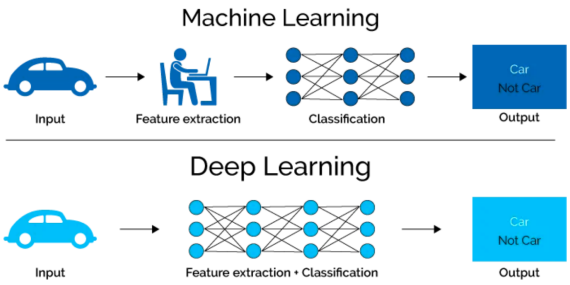
1 Introduction

1.1 Research Goal

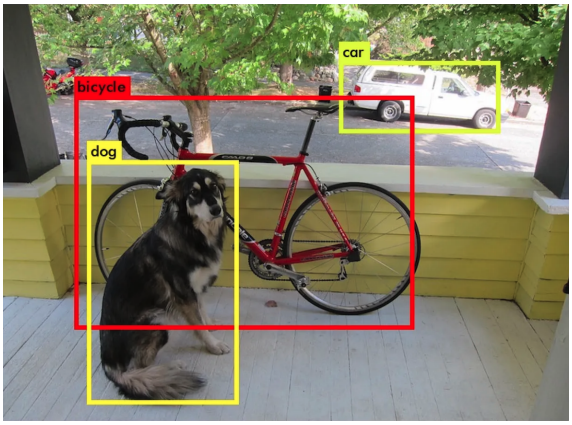
The study was conducted in accordance with the Kaggle Competition (Tao, 2024). This report focused on building multi-labelled image classifiers, which is essential for understanding the complexity of real-world data. Real-world data often consists of multiple elements that need to be identified and classified simultaneously, which is critical for applications that rely on accurate and efficient image analysis. For example, in environmental monitoring, satellite imagery may need to simultaneously identify multiple elements such as forest cover, urban areas, water bodies, and pollution levels. Each image contains multiple labels describing different environmental features, and a model that can accurately distinguish and classify these aspects in a single analysis is required.

This project revolves around the challenge of multi-label classification, linking each image in the dataset to multiple labels and succinct captions summarising the visual content. The main aim is to develop an image classifier that can accurately predict the labels of each image, potentially improving classification accuracy by considering textual data in the captions as well. This study goes beyond the traditional boundaries of single-label image classification into more complex and realistic multi-label scenarios, reflecting real-world situations where objects and scenes are rarely represented by a single label. Mastery of this task therefore allows students to acquire advanced skills in working with complex datasets and to develop models capable of interpreting and analysing images with a high degree of ambiguity and diversity.

1.2 Research Importance



(a) Machine Learning vs Deep Learning



(b) Multi-label classification Example

Figure 1: Deep Learning

In traditional machine learning, experts identify and extract features to simplify the data and make learning algorithms easier to understand. Deep learning learns features from data in an incremental way, without needing experts or feature extraction. Deep learning networks can learn complex patterns in data, which makes them very effective in tasks such as image recognition, natural language processing, and predictive modelling.

Our research focuses on the development of multi-label image classifiers, using advanced deep-learning networks such as ResNet, GoogLeNet, EfficientNet, and CLIP. These models perform well in recognising images, such as Google's self-driving cars, understanding human language like Amazon's Alexa, and performing predictive analytics like Netflix's recommender system(Leaky AI, 2023). This research is important because it exploits these transformative capabilities of deep learning, contributing to multi-label classification systems and the wider field of AI, potentially impacting areas as diverse as security, business, healthcare, and entertainment. This study is therefore of significant importance, as it encourages us to investigate the

field of deep learning and develop the skills to develop our deep neural networks.

1.3 Dataset overview

The dataset for this Kaggle competition (Tao, 2024) comprises a series of images each paired with a set of labels ranging from 1 to 19, with the exclusion of label 12, and a descriptive caption. As displayed by Figure 2, each image in the dataset can have multiple labels associated with it, making this a multi-label classification task. The captions provide a textual summary of the scenes depicted in the images, potentially offering additional contextual cues that can aid in the classification process. The dataset contains a total of

Figure 2: Display The Sample of Image Data

40,000 images and has been manually segmented into two distinct sets: training data, which comprises the first 30,000 images, and test data, which includes the subsequent 10,000 images. This separation facilitates the training and evaluation phases of the model development process.

1.3.1 Label Imbalanced Issue

As shown in Figure 3, the bar chart highlights a pronounced imbalance in the dataset. Class 1, is the most disproportionately represented, with over 20,000 images, while other classes, particularly for 2, 5, 9, 13, 11, and 14, are significantly underrepresented with fewer than 5,000 images each. This imbalance can lead to biased model training where the model overly specialises in recognising features of the dominant class at the expense of others.

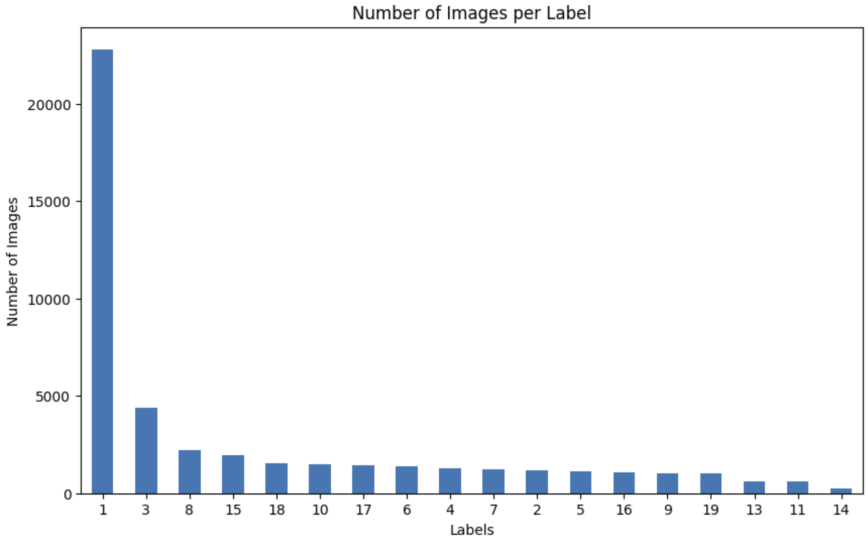


Figure 3: Number of Images per Label

1.4 Methods and Solutions

1.4.1 CLIP Model

CLIP (Contrastive Language–Image Pre-training) by OpenAI (Radford et al., 2021) is a cutting-edge model designed to learn visual concepts from natural language supervision. It uses a contrastive learning objective to jointly train an image encoder and a text encoder, which results in embeddings that can be effectively compared in a shared space.

1.4.1.1 Justification and Motivation The use of the CLIP model is driven by its inherent capability to handle both images and text simultaneously, aligning the embeddings from the two different modalities in a shared latent space. This makes CLIP particularly suited for tasks where understanding the relationship between text and images is crucial. Given the dataset includes captions summarizing the images, CLIP can utilize this descriptive text to enhance its image classification predictions, potentially outperforming traditional methods that treat image and text data separately. The motivation for using CLIP lies in its pre-trained nature and the efficiency of its contrastive learning framework, which has been demonstrated to perform robustly across a variety of visual tasks, particularly in zero-shot or few-shot scenarios.

1.4.2 GoogLeNet + LSTM

GoogLeNet, or Inception v1, is a deep convolutional neural network architecture known for its efficiency in image recognition tasks due to its inception modules that allow it to learn multi-level features. LSTMs are a type of recurrent neural network (RNN) specially designed to handle sequences, making them ideal for processing textual data over time.

1.4.2.1 Justification and Motivation The combination of GoogLeNet with LSTM is motivated by the need to effectively process and synthesize both the visual elements of the images and the sequential nature of the textual captions. GoogLeNet extracts intricate features from the images, which are then fed into the LSTM to be combined sequentially with the features derived from the text captions. This hybrid model aims to capture both the static features present in the images and the dynamic, contextual information provided by the text, potentially leading to a more nuanced understanding of the content and its corresponding labels. The motivation here is to leverage the LSTM's ability to remember long-term dependencies, thereby making connections between the visual cues and textual descriptions that might be temporally spaced in the data.

1.4.3 Addressing Imbalanced Label

To address the pronounced class imbalance observed in the dataset, several strategic measures were implemented during the neural network training process.

Focal loss was utilized to modulate the influence of well-classified examples, thereby placing greater emphasis on misclassified cases from minority classes. This approach effectively shifts the model's focus towards challenging examples, which is vital in scenarios dominated by a single class.

In conjunction with modifying the loss function, the dataset underwent weighted resampling to balance the class representation in the training set. By adjusting the sampling probabilities inversely proportional to class frequencies, each class was given equal importance during model training. This ensured that the neural network was exposed to a more uniform distribution of classes, reducing the likelihood of bias towards the majority class.

Furthermore, the implementation of AdamW as the optimizer introduced a decoupled weight decay regularization, aiding in the mitigation of overfitting to the dominant class. This was complemented by the use of a Cosine Annealing Learning Rate scheduler, which varied the learning rate throughout the training process. Such adjustments allowed the model to explore a wider range of solutions, enhancing its ability to generalize across less frequent classes and avoid local minima that might emphasize the majority class.

2 Related Works

2.1 Multi Label Classification Task

2.1.1 Early Solutions and the Rise of CNNs

Early solutions for multi-label image classification primarily relied on enhancing feature extraction capabilities, with convolutional neural networks (CNNs) playing a pivotal role in this progress. Classic CNN architectures such as AlexNet, VGG, and ResNet extract low-level to high-level features through stacked convolutional layers. However, these models often faced challenges due to their large number of parameters and high computational costs.

2.1.2 2014 - Innovations and Breakthroughs with GoogLeNet

2.1.2.1 The Inception Module The key innovation in GoogLeNet is its Inception module, consisting of multiple convolution layers with different kernel sizes operating in parallel to capture features at various scales. That kind of parallelism makes the network of very low computational cost yet still very effective at image recognition. The 1x1 convolutions actually reduce the number of dimensions before going for more computationally expensive, larger convolution kernels. In fact, in this way, the approach saves on computations and also acts as a regularizer to avoid overfitting. The GoogLeNet architecture aimed to stack together multiple Inception modules such that the final network had an enormous depth to increase its accuracy. Moreover, the introduction of these auxiliary classifiers in the intermediate layers helped to solve the problem of vanishing gradients during training.

2.1.2.2 Application of GoogLeNet in Multi-Label Classification The GoogLeNet (Inception v1) marked a significant improvement in CNN architecture, particularly in handling multi-label classification tasks that require simultaneous identification of multiple objects within an image. The unique Inception module of GoogLeNet allows the model to capture image features at different scales using convolution kernels of various sizes (1x1, 3x3, 5x5) and max-pooling layers within the same layer. This modular design not only enhances the network's expressive power but also effectively controls the growth of computational complexity and parameters.

GoogLeNet was a real breakthrough work in deep learning. The architecture did very well on benchmark image classification tasks, including the ImageNet Large Scale Visual Recognition Challenge, and really set the stage for future advancements in neural network design. The resource efficiency in GoogLeNet motivated researchers in further exploration of resource-friendly network architectures, eventually resulting in models like ResNet and MobileNet.

2.1.3 2021 - The CLIP Model

Despite the significant performance improvements with GoogLeNet, its generalization capability when dealing with real-world data remained limited. In 2021, OpenAI introduced the CLIP (Contrastive Language–Image Pre-training) model, a pre-trained model designed to measure the compatibility between given images and text captions. CLIP was trained on a massive dataset of 400 million image-caption pairs collected from the internet, being one of the first models to do (OpenAI, 2023). CLIP can associate images with a set of class labels and also with textual descriptions containing any English words.

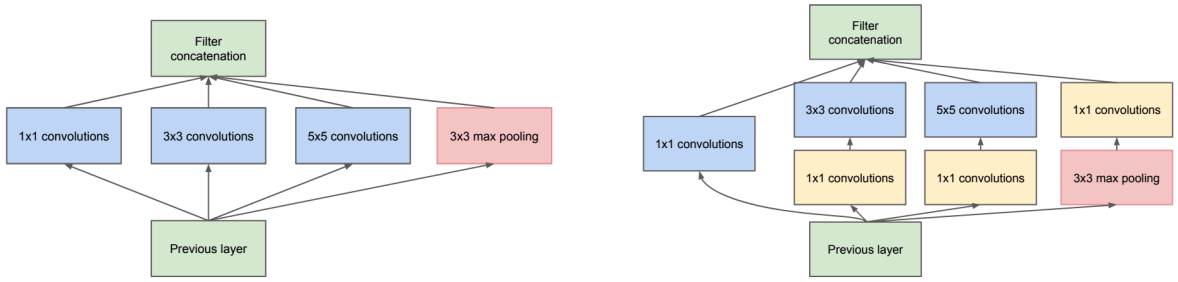
2.1.3.1 Contrastive Learning and Broad Adaptability CLIP uses contrastive learning to learn visual concepts from natural language descriptions. It trains image and text encoders in parallel, enabling the understanding of unannotated image-text pairs. This design enhances the model's adaptability and interpretability, especially in handling diverse multi-label image classification tasks.

2.1.3.2 Impact and Advantages of CLIP The CLIP represents a shift in deep learning from Single-Modality Image processing to the multimodal understanding of images and language. CLIP addresses the issue of overfitting to specific datasets in traditional models and demonstrates the capability to handle unseen data in practical applications. This is particularly important for multi-label image classification, where real-world data often has rich and complex labels.

2.2 Vision Models

2.2.1 GoogLeNet

GoogLeNet, also known as Inception v1, is a deep convolutional neural network developed by Google researchers. It is renowned for its complexity and depth while being highly efficient in computational terms. The key innovation of GoogLeNet is the introduction of the "Inception module," which performs convolutions in parallel using filters of varying sizes and then concatenates the outputs. This design allows the network to capture information at various scales. Additionally, GoogLeNet introduced global average pooling, which significantly reduces the number of parameters in the network and helps prevent overfitting. GoogLeNet was the winner of the 2014 ILSVRC (ImageNet Large Scale Visual Recognition Challenge)(Goodfellow et al., 2014).



(a) Inception module, naïve version

(b) Inception module with dimension reductions

Figure 4: Inception module(Source: image from the original paper)

2.2.2 ResNet

ResNet, short for Residual Network, is a convolutional neural network (CNN) architecture that was developed to facilitate the training of networks that are substantially deeper than those previously used. Introduced by He et al. in 2015(He et al., 2015), ResNet addresses the vanishing gradient problem by incorporating "residual connections" or "skip connections". These connections allow gradients to flow directly through the network by skipping one or more layers. This architecture significantly enhances the training speed and accuracy of deep neural networks. ResNet gained popularity after winning the ILSVRC 2015 competition and is notable for its use of residual connections that enable the network to learn a residual function with reference to the layer inputs, rather than learning an unreferenced function. This approach helps to prevent the vanishing gradient issue that can occur in very deep neural networks. ResNet models come in various depths including versions like ResNet50, ResNet101, which indicate the number of layers in the network.

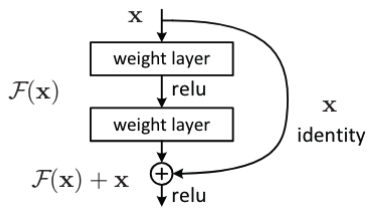


Figure 5: Residual Learning(Source: image from the original paper)

2.2.3 EfficientNet

EfficientNet is a more recent innovation in CNN architectures, introduced by Mingxing Tan and Quoc V. Le in 2019. It systematically scales network width, depth, and resolution with a set of fixed scaling coefficients, which leads to much better efficiency and accuracy. EfficientNets use a compound coefficient ϕ to uniformly scale the network size in a principled way, as explained by the formula

$$depth : d = \alpha^{\phi}, \quad width : w = \beta^{\phi}, \quad \text{and} \quad resolution : r = \gamma^{\phi},$$

such that the condition

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \tag{1}$$

is satisfied, and

$$\alpha \geq 1, \quad \beta \geq 1, \quad \gamma \geq 1. \tag{2}$$

where each parameter is scaled according to a power of ϕ. This approach achieves much better performance than previous CNNs with fewer parameters and requiring less computation, thereby maintaining a balance between the network's complexity and resource efficiency.

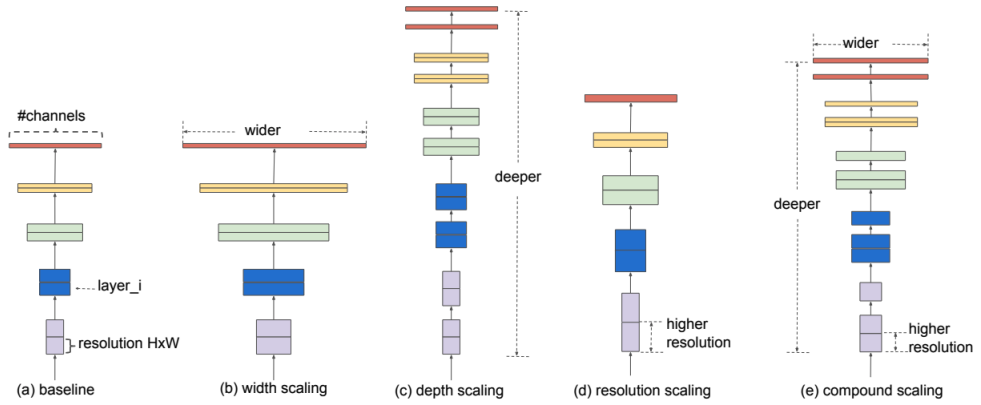


Figure 6: Different scaling methods vs. Compound scaling (Source: image from the original paper)

3 Techniques

3.1 Label One-Hot Encoding

3.1.1 Principle

In the experiment, each instance can have multiple labels simultaneously. The `MultiLabelBinarizer` from `sklearn` is used to transform the list of labels for each instance into a binary vector. Let $\mathscr{L} = \{l_1, l_2, \ldots, l_m\}$ be the set of all possible labels. For a given instance with labels $\{l_{i1}, l_{i2}, \ldots\} \subseteq \mathscr{L}$, the one-hot encoded vector y is:

$$y_j = \begin{cases} 1 & \text{if } l_j \in \{l_{i1}, l_{i2}, \ldots\} \\ 0 & \text{otherwise} \end{cases}$$

This transformation is applied to all instances in the dataset, converting categorical labels into a binary matrix suitable for multi-label classification tasks.

3.1.2 Usage in Neural Networks:

These one-hot encoded vectors are used as target labels in training the neural network. During the training process, the network learns to predict the probability of each label being applicable to an instance. The final layer of the network typically uses a `sigmoid` activation function to output these probabilities, which are then compared to the one-hot encoded vectors to compute the loss and update the model parameters.

3.1.3 Justification

3.1.3.1 Compatibility with Neural Networks One-Hot Encoding converts categorical labels into binary vectors, making them suitable for neural network training. Neural networks require numerical input, and this method ensures that each label is represented in a form that the network can process.

3.1.3.2 Multi-Label Classification In multi-label classification, each instance can belong to multiple classes simultaneously. One-Hot Encoding allows for the representation of multiple classes by using binary vectors, where each position in the vector corresponds to a specific label. This approach effectively handles the complexity of multi-label tasks.

3.1.3.3 Loss Calculation Using One-Hot Encoded vectors as target labels allows for straightforward loss calculation. The neural network's predictions (probabilities for each label) can be directly compared to the One-Hot Encoded vectors, facilitating the computation of loss and enabling effective backpropagation.

3.1.4 Advantages or Novelty

3.1.4.1 Combining with MultiLabelBinarizer With the application of MultiLabelBinarizer from sklearn, the method transforms labels effortlessly into One-Hot Encoded vectors. This consolidation ensures uniformity in the preprocessing pipeline and makes sure that the encoding of labels is also consistent for all examples.

3.1.4.2 Sigmoid Activation for Multi-Label Tasks One of the important things that the sigmoid activation at the final layer is most suitable for multi-label classification is due to the fact that, unlike the softmax, it is used for mutually exclusive classes. A sigmoid activation allows independent probability predictions for each class, which is perfectly in line with the nature of multi-label classification.

3.1.4.3 Applicability to Various Datasets The method works well with diversified datasets having many labels. It could be used effectively for many problem domains where target instances can belong to one or more pre-defined categories: text classification, image tagging, etc. It uses the computational power of neural networks to manage large, complex datasets with multiple labels, ensuring that the model learns the intricate relationships and patterns between each class.

3.2 Caption Tokenization

3.2.1 Byte Pair Encoding for CLIP Model

CLIP uses a tokenizer that is based on Byte Pair Encoding (BPE), a subword tokenization method. This method is particularly effective for handling the vocabulary diversity and morphological richness found in natural language (Zouhar et al., 2023).

BPE works by iteratively merging the most frequently occurring pair of bytes or characters in a dataset to form new tokens. This process continues until a specified vocabulary size is reached or no more frequent pairs can be found. Mathematically, if S is the set of individual characters in the dataset and V is the vocabulary built by iteratively merging pairs of elements, BPE defines a merge operation $\oplus$ such that:

$$V = S \cup \{x \oplus y \mid x, y \in S \text{ and } x \oplus y \text{ is frequently occurring}\}$$

The tokenization of a caption c can be seen as a mapping f from the string of text to a sequence of tokens t from the vocabulary V:

$$t = f(c, V)$$

This function f breaks down the caption into subwords or tokens based on the learned merges that best represent the recurring patterns in the training text data.

3.2.1.1 Padding and Batching: For batch processing in neural networks, it's critical that all input data tensors have the same dimensions. Therefore, after tokenization, captions are padded to ensure uniform length across a batch.

Let $\{t_1, t_2, \ldots, t_n\}$ be the tokenized outputs for a batch of n captions, where each t_i is a vector of token indices. Let L be the length of the longest token vector in the batch. Padding each token vector t_i to length L can be represented as:

$$t_i' = t_i \oplus \text{pad}(L - |t_i|)$$

Here, pad(k) represents a sequence of k padding tokens (commonly the token index for a special padding symbol), and $\oplus$ denotes concatenation.

3.2.2 GoogLeNet with LSTM Model Caption Tokenization

The caption preprocessing in this model involves more detailed and depth processing to enhance the model's understanding and handling of text. First, all text is converted to lowercase to eliminate the variability of the case, standardizing how the model processes the data. Following this, text is cleaned by removing all non-alphabetic characters using regular expressions, thus clearing noise and irrelevant information, ensuring the quality and accuracy of the text data. After cleaning, the text undergoes another round of tokenization, this time splitting the cleaned strings into a list of words, preparing for further semantic processing. Finally, using the pre-trained GloVe word vectors, embeddings are generated for each word. These embeddings, rich in semantic information, are directly used in model training and significantly improve the model's ability to interpret text.

3.2.2.1 Word Embedding for Caption Tokenization

The GoogLeNet with LSTM model uses a tokenizer based on pre-trained word embeddings from the GloVe (Global Vectors for Word Representation) model. This method effectively captures semantic information and relationships between words, providing a rich representation of textual data.

Preprocessing Captions are preprocessed by converting to lowercase, removing non-alphabetic characters, filtering stopwords, and applying lemmatization. This ensures that the text is normalized and cleaned before tokenization.

Vocabulary Creation A vocabulary is created from the most frequent words in the captions, including special tokens such as `[PAD]` for padding and `[UNKNOWN]` for unknown words.

Mapping Words to Indices Each word in the vocabulary is assigned a unique index. If a word is not present in the vocabulary, it is mapped to the `[UNKNOWN]` token.

Embedding Table Construction An embedding table is created using pre-trained GloVe embeddings. Each word index corresponds to its embedding vector.

3.2.2.2 Example of Tokenization Process

Consider a caption: "A dog playing with a frisbee."

1. **Preprocessing**: The caption is converted to lowercase and cleaned:

   "A dog playing with a frisbee" $\rightarrow$ ["dog", "play", "frisbee"]

2. **Tokenization**: The words are mapped to their corresponding indices in the vocabulary:

   ["dog", "play", "frisbee"] $\rightarrow$ [`idx_dog`, `idx_play`, `idx_frisbee`]

3. **Padding**: If the maximum caption length in the batch is 5, the tokenized caption is padded:

   [`idx_dog`, `idx_play`, `idx_frisbee`, `pad_idx`, `pad_idx`]

4. **Embedding**: The tokenized and padded caption is converted into an embedding vector using the embedding table E:

$$E[t_i'] = [E[\texttt{idx\_dog}], E[\texttt{idx\_play}], E[\texttt{idx\_frisbee}], E[\texttt{pad\_idx}], E[\texttt{pad\_idx}]]$$

This embedding vector is then used as input to the LSTM network in the GoogLeNet with LSTM model.

3.2.2.3 Word Embedding Alignment

To align the word embeddings with the vocabulary, the embedding matrix W is created where each row i corresponds to the embedding of word w_i:

$$W = \begin{pmatrix} E[w_1] \\ E[w_2] \\ \vdots \\ E[w_n] \end{pmatrix}$$

This matrix W is used to convert token indices into their respective embedding vectors during the forward pass of the model.

3.2.3 Justification

The Byte Pair Encoding (BPE) method used in the CLIP model effectively handles the diversity and morphological richness of natural language, which is crucial for multi-label image classification. In this context, descriptive labels may include various forms of words and expressions, such as technical terms, compound words, or new vocabulary. BPE builds the vocabulary by merging frequently occurring pairs of bytes or

characters, allowing the model to adapt to and recognize complex lexical structures, thereby enhancing the accuracy of processing and classifying image labels.

The caption preprocessing steps in the GoogLeNet with LSTM model, which include converting text to lowercase, removing non-alphabetic characters, and applying lemmatization, help clean and normalize the input text. In multi-label image classification, this thorough normalization ensures consistency and comparability among different labels, making the model more effective in learning and recognizing image-related tags.

3.2.4 Advantages or Novelty

A significant advantage of BPE is its ability to handle new or rare words, enabling the model to effectively manage unknown words. This capability is particularly important in multi-label image classification, where image descriptions may include uncommon or specialized vocabulary. Additionally, the BPE method reduces the complexity of the model and enhances its generalization capabilities, which are essential for handling large-scale and diverse datasets.

In the GoogLeNet with LSTM model, the incorporation of GloVe word embeddings represents a key strength. These pre-trained word vectors capture deep semantic relationships between words, significantly boosting the model's ability to understand complex text. In multi-label image classification, labels often have semantic associations and exhibit rich semantic layers, such as the tags "cat," "pet," and "animal" appearing together. The GloVe embeddings, with their rich semantic information, aid the model in capturing these complex relationships, thereby improving the accuracy of classification.

3.3 Image Preprocessing

Resize In image preprocessing, resizing images is absolutely crucial to ensure that all input images meet the size requirements of machine learning models. This step standardizes input sizes and enhances model training performance and efficiency. Since most neural network models require specific input dimensions, such as 224x224 or 256x256 pixels, resizing all images to these dimensions is a requirement(Patel, 2021). During this process, it's vital to maintain the original aspect ratio of the images to prevent unrealistic distortions of the content. The right approach is to scale the images proportionally so that the longest side matches the target size, and then perform a center crop to fit the square format. This typically involves focusing on the central part of the image and trimming the edges. Doing so not only meets the model's input specifications but also preserves the authenticity and crucial visual information of the images, significantly affecting the model's ability to handle a diverse range of real-world images.

ImageNet In our image preprocessing, we utilize the statistical data "mean and standard deviation" from the ImageNet dataset for normalization, which enhances the model's generalization capabilities and overall performance when handling new data. ImageNet is a widely studied extensive image database, and these statistics not only facilitate the effective use of pretrained weights but also ensure that the colors in the boundary regions of images remain consistent with the rest of the image during affine transformations, thereby reducing learning interference. Our models, such as ResNet, GoogLeNet, and EfficientNet available in torchvision, have all been pretrained on ImageNet. By applying the same mean and standard deviation in preprocessing that were used during pretraining, we ensure the consistency of input data and the efficacy of model training.

ToTensor plays a crucial role in the field of image processing, especially when using the PyTorch framework. It converts PIL images or numpy arrays (ndarray) into PyTorch's Tensor format, a step that is vital for any model because all PyTorch models require input data to be in Tensor form. This operation not only automatically transforms pixel values from integers (typically ranging from 0 to 255) into floating-point numbers but also normalizes them to the [0.0, 1.0] range. Such normalization is critical for ensuring the effectiveness and numerical stability of gradient descent algorithms.

Conversion to RGB In the preprocessing of images for the CLIP model, converting all input images to the RGB format is crucial to ensure uniformity across the dataset. Different source images might come

in a variety of color formats, such as grayscale or CMYK, which can lead to inconsistencies in how the model interprets the data. By standardizing to RGB, the preprocessing ensures that the features extracted and learned by the model are based on a consistent color space. This is particularly important in a model like CLIP, which integrates and compares textual and visual information. Ensuring that all images are processed in RGB format supports the model's ability to accurately correlate and learn from the color data, enhancing its performance on color-sensitive tasks.

3.3.1 Justification

The image preprocessing techniques employed in multi-label image classification ensure that the models handle the complexities of varied visual inputs efficiently. Resizing images to uniform dimensions, such as 224x224 or 256x256 pixels, is essential for maintaining uniformity across the dataset, which aids neural networks in processing data efficiently and maintaining the aspect ratio preserves crucial visual information necessary for the accurate identification of objects within complex, multi-label images. Normalization using ImageNet statistics aligns new images with a pretrained model's expectations, enhancing the model's ability to generalize by standardizing image pixel values to a consistent scale. Conversion to Tensor format within the PyTorch framework ensures that all images are in the correct data structure and scale for effective processing, crucial for stability during training. For the CLIP model, converting all images to RGB format standardizes color information across all images, ensuring the model processes visual data consistently, a critical factor for tasks where image and text data must correlate closely.

3.3.2 Advantages or Novelty

The image preprocessing techniques provide numerous advantages. Resizing images facilitates the effective use of Inception networks by providing consistency in input data size, essential for the accurate application of filters and feature extraction. Using ImageNet's mean and standard deviation for normalization leverages robust benchmarks for data preprocessing, enhancing model performance on diverse datasets by normalizing images in a way that closely matches the training conditions of widely used pre-trained networks. Tensor conversion ensures that all images are optimally prepared for processing in the PyTorch environment, supporting the computational needs of deep learning algorithms and enhancing overall training efficiency. Additionally, the RGB conversion for the CLIP model ensures the visual data's color consistency is maintained, crucial for the model to effectively integrate and analyze the combined textual and visual inputs, improving its performance on tasks that require detailed image-text interrelations.

3.4 Weighted Random Sampler

3.4.1 Principle

In Pytho, the DataLoader is an iterable over a dataset. It iterates over dataset samples and feeds the data into machine learning models. The Weighted Random Sampler is a sampler object available in PyTorch, which defines the sampling strategy for items within the DataLoader's dataset iteration. This sampler ensures that each batch contains a proportional representation of all classes by oversampling minority classes, as illustrated in Figure 7.

To handle class imbalance, the Weighted Random Sampler obtains all the target classes from the DataLoader and utilises the class weights inputted during the implementation phase. The formula for each class weight is

$$w_i = \frac{1}{n_i}$$

where w_i represents the weight assigned to the i^{th} class, and n_i denotes the number of items in class i.

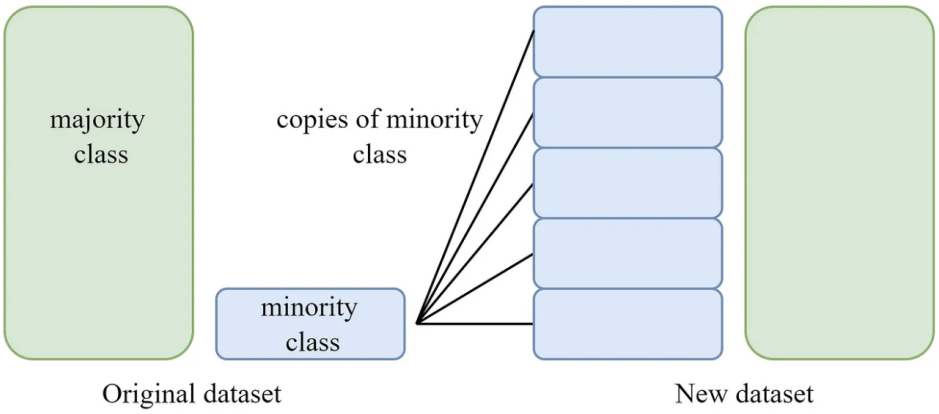


Figure 7: Oversampling Method

Each item in the dataset is mapped to the weight of its respective class. Sampling from the target classes is then performed based on these class weights, where each class weight represents the probability of sampling that class. By utilising the weighted approach, labels that appear more frequently in the dataset are assigned with smaller weights, reducing their likelihood of being sampled, in order to generate balanced datasets within each batch.

Additionally, the Weighted Random Sampler has the ability to draw samples with or without replacements. Without replacement, if a sample is drawn from the dataset, it cannot be drawn again. Sampling with replacements allows for oversampling of the dataset, effectively increasing the frequency of minority class items to address class imbalances.

3.4.2 Justification

The Weighted Random Sampler is used to handle class imbalance during the training of deep learning networks.

Class imbalance occurs when some classes are overwhelmingly more represented in the dataset than others. As described in Section 1.3.1 and Figure 3, the dataset used in this multi-label classification task is disproportionately represented, with class 1 dominating over other class labels.

Without intervention, trained classifiers may be more focused on the larger classes, specialising in recognising their features and neglecting the minority classes. As a result, these classifiers may not generalise well to predict unseen data.

3.4.3 Novelty

The advantage of using the Weighted Random Sampler is that it helps to overcome the class imbalanced problem and prevent the trained model from becoming overly specialised in learning the features of dominant classes and ignoring the features of infrequent classes. The sampler ensures the model is trained on a representative subset of the data and learns the features of each class equally, increasing the prediction ability for all classes.

3.5 Optimizer

3.5.1 Adaptive Moment Estimation (ADAM)

Adam is a popular optimizer in the field of deep learning because it combines the advantages of two other extensions of stochastic gradient descent: AdaGrad and RMSProp. Its primary advantage lies in its ability

to handle sparse gradients on noisy problems. Adam updates the weights w using the following iterative formulas:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$w_{t+1} = w_t - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

Here, g_t represents the gradient of the loss function with respect to the weights at timestep t, m_t and v_t are estimates of the first and second moments of the gradients, β_1 and β_2 are the exponential decay rates for these moment estimates (commonly set to 0.9 and 0.999, respectively), η is the learning rate, and ε is a small scalar used to prevent division by zero.

3.5.2 AdamW

AdamW is a modification of the original Adam algorithm that decouples the weight decay from the optimization steps. This change addresses an issue in the Adam optimizer where the weight decay is integrated into the gradient updates, potentially leading to suboptimal training behavior, especially in the context of regularization. AdamW modifies the weight update rule of Adam as follows:

$$w_{t+1} = w_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} + \lambda w_t \right)$$

Where λ is the weight decay coefficient. This modification effectively separates the weight decay term from the gradient-based updates, applying it directly as part of the weight update step.

The choice of Adam and AdamW as optimizers in the research is justified based on their robustness and performance in dealing with complex datasets that involve high-dimensional data and require efficient handling of sparse gradients. Adam is chosen for its adaptability to various types of data and its ability to stabilize the training process in face of noisy gradients. AdamW, on the other hand, is particularly selected for its refined approach to integrating weight decay, which is vital for maintaining generalization in larger or more complex models. These characteristics make both optimizers highly suitable for deep learning tasks that involve intricate interactions between model parameters, as is typical in multi-modal datasets involving images and text.

3.5.3 Justification

ADAM and AdamW optimizers are highly reasonable choices for multi-label image classification due to their ability to handle high-dimensional data and sparse gradients effectively. ADAM's adaptability to various data types and stabilization of the training process in noisy conditions justify its use in this context. AdamW's decoupling of weight decay from optimization steps further enhances regularization, ensuring better generalization in complex models.

3.5.4 Advantages or Novelty

The main advantages of ADAM are its adaptability to various data types and stabilization of training amid noisy gradients. Moment estimates allow dynamic learning rate adjustment, handling high-dimensional and sparse gradient data effectively, typical in multi-label classification. This adaptability makes ADAM highly effective for tasks involving intricate parameter interactions.

The novelty of AdamW lies in its weight update rule, including a separate weight decay term. This refinement prevents suboptimal training seen in original Adam due to integrated weight decay. By separating weight decay from optimization, AdamW ensures better generalization and performance, particularly in large-scale models. This is advantageous for multi-label classification, where robust models are essential. Combining dynamic learning rate adjustment and improved weight decay handling makes AdamW an innovative optimizer for multi-label classification challenges.

3.6 Loss Function

3.6.1 Binary Cross-Entropy (BCE) Loss

BCE Loss is a standard loss function used for binary classification tasks, which has been extended to multi-label classification by treating each label independently (Ruby and Yendapalli, 2020). It measures the error between the true label and the predicted probability for each class, penalizing incorrect predictions. This ensures that the model learns to output probabilities close to the true distribution of each label.

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(p_i) + (1-y_i) \cdot \log(1-p_i)]$$

Where α_t is a weighting factor for the class, γ is a focusing parameter to adjust the rate at which easy examples are down-weighted, p_t is the model's estimated probability for each class being true.

3.6.2 Focal Loss

In this multi-label classification task, class imbalance is an issue where label 1 is predominantly overwhelming which leads to some labels being underrepresented. Focal Loss helps mitigate this by dynamically scaling the loss contribution of each example based on how well it is classified, thereby ensuring that the model pays more attention to difficult and minority-class instances (Lin et al., 2017). This leads to better performance in terms of precision and recall, particularly for less frequent labels.

$$\text{Focal Loss} = -\alpha_t \cdot (1-p_t)^{\gamma} \cdot \log(p_t)$$

Where α_t is a weighting factor for the class, γ is a focusing parameter to adjust the rate at which easy examples are down-weighted, and p_t is the model's estimated probability for each class being true.

Focal Loss is designed to address class imbalance by focusing more on hard-to-classify examples. It modifies the traditional cross-entropy loss by adding a modulating factor $(1-p_t)^{\gamma}$, which reduces the loss contribution from well-classified examples and emphasizes the harder ones. The parameter γ controls the focusing strength, while α_t helps to balance the importance between different classes.

3.6.3 Justification

Binary Cross-Entropy (BCE) Loss is suitable for multi-label image classification as it handles multiple labels independently. This approach measures the error between the true and predicted probabilities for each class, ensuring accurate predictions across all labels. Its design is ideal for multi-label scenarios where each image can belong to multiple classes simultaneously.

Focal Loss addresses class imbalance, a common issue in multi-label datasets. By dynamically scaling the loss contribution based on classification difficulty, it ensures the model focuses on hard-to-classify and minority-class instances. This makes Focal Loss particularly reasonable for enhancing performance on underrepresented labels.

3.6.4 Advantages or Novelty

BCE Loss's advantage lies in its simplicity and effectiveness in managing independent binary predictions for each label, ensuring accurate probabilities without interference.

Focal Loss introduces novelty with its modulating factor $(1 - p_t)^\gamma$, which reduces the loss from well-classified examples and emphasizes harder ones. This adaptive mechanism improves the model's attention to difficult and minority-class instances. The focusing parameter γ allows fine-tuning of the model's sensitivity, enhancing robustness and generalization in imbalanced datasets, making Focal Loss a particularly advantageous method.

3.7 GoogLeNet Design

3.7.1 Principles

GoogLeNet is a 22-layer deep convolutional neural network based on the Inception architecture. As shown in Figure 8, it comprises a total of 9 Inception modules, which are arranged into 3 groups and separated by max pooling for dimensionality reduction.

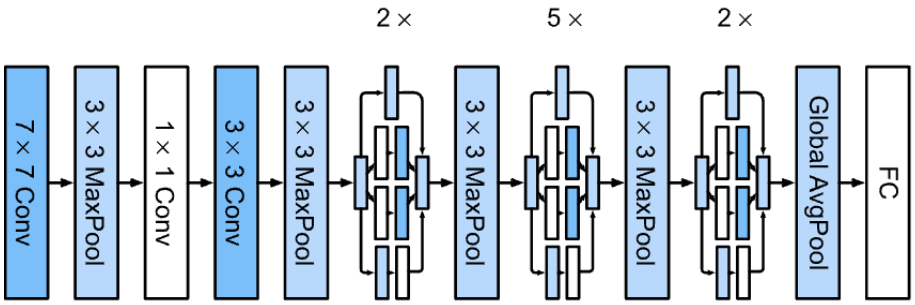


Figure 8: GoogLeNet Architecture

3.7.1.1 Inception Modules Inception block allows the network to choose between various convolutional filter sizes within each block. The Inception architecture aims to identify an optimal local sparse structure in the CNN that can be approximated by dense components.

The structure of the inception block is illustrated in Figure 5. It is a combination of three different kernel sizes (1x1, 3x3, and 5x5) along with max pooling. All convolutions, including those within the Inception modules, use the rectified linear activation function. Additionally, 1x1 convolutions are introduced to reduce the number of parameters in the architecture.

The design of the Inception model is based on a layer-by-lay construction approach. The correlation of the last layer is analysed, and highly correlated units are clustered together. These clusters are connected to the previous layer and form the units for the next layer.

The different kernel size assists the analysis of images by capturing both lower-level and higher-level features. Each unit from the prior layer is assumed to represent some regions of the image. Units with high correlation in the lower layers, which are closer to the input image, concentrate in neighbouring regions, and can be further analysed by 1x1 convolutions in the next layer. However, a few other image features are spatially spread out and require analysis by larger convolutions. Therefore, three different kernel sizes are utilised to analyse various image features effectively.

3.7.1.2 Auxiliary Classifiers

Auxiliary classifiers are implemented to prevent the gradient vanishing problem commonly observed in networks with large depths. These classifiers are connected to the intermediate layers of the model. Figure 9 illustrates the architecture of the auxiliary classifier.

During training, the loss computed from the auxiliary classifiers is added to the total network loss, with a lower weight of 0.3. This approach encourages discrimination in the lower layers of the network and increases the gradient signal that is propagated back. The auxiliary classifiers are discarded during model prediction.

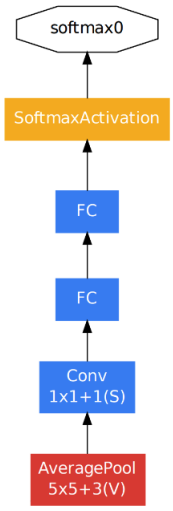


Figure 9: Auxiliary Classifier

3.7.2 Justification

GoogLeNet is chosen for its high performance in image classification and detection. The model is the winner of the ILSVRC (ImageNet Large Scale Visual Recognition Challenge), and its proven architecture design is suitable to be used for the multi-label classification task in this experiment.

3.7.3 Advantages or Novelty

One of the advantages of GoogLeNet is its efficiency. Compared to other models that can go very deep in its depth and contain lots of parameters, like ResNet50 and Resnet101, GoogLetNet can be run on devices with limited computational resources and a low memory footprint. The dimensionality reduction achieved through 1x1 convolutions in the Inception modules helps to reduce the number of parameters, significantly reducing computational complexity and minimising the risk of overfitting due to an excessive number of parameters.

Additionally, before the final classifier outputs image predictions, an average pooling layer and a linear layer are implemented. The design enables the convenience in adaptation and fine-tuning of the network for other datasets.

3.8 LSTM

3.8.1 Principles

Long Short-Term Memory (LSTM) is an improved version of RNN which enables the capture of long-term memory for sequence prediction tasks like text analysis. Figure 10 illustrates the architecture for LSTM.

LSTM contains three main components: an input gate, a forget gate, and an output gate.

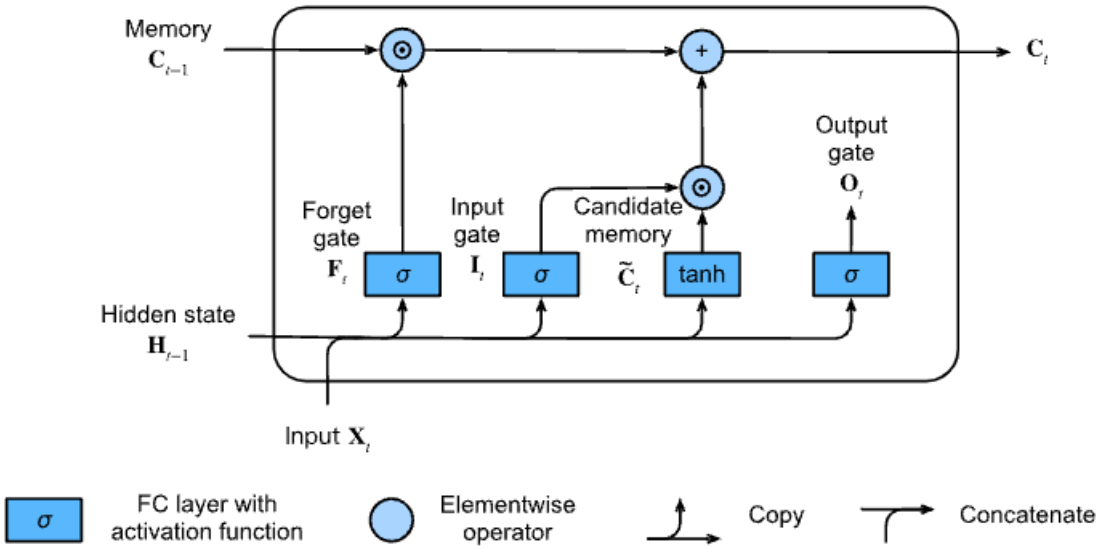


Figure 10: LSTM Architecture

The cell state is the flow of information from the previous memory C_{t-1} to the current memory C_t. Visually, it is the top horizontal line in Figure 10. By utilising the gates, LSTM has the ability to manipulate the information flow process and remove or add information to the cell state.

3.8.1.1 Forget Gate Forget gate determines what information is removed from the cell state. The decision is made by a sigmoid layer. It calculates information from the hidden state H_{t-1} and the input X_t, and outputs a value ranging between 0 and 1 for each number in the cell state C_{t-1}. 0 indicates a complete removal of information from the previous cell state C_{t-1}, and an output of 1 keeps all the information from the previous memory C_{t-1}.

The forget gate output F_t is mathematically defined as

$$F_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f)$$

where W_{fh} is the weight matrix associated with the hidden state h_{t-1}, W_{fx} is the weight matrix associated with the input x_t, and b_f is the bias term.

3.8.1.2 Input Gate The main purpose of the input gate is to decide what new information is added to the cell state. The input gate contains two components.

The input gate layer, denoted by I_t, is a sigmoid layer which decides the values to be updated.

It is mathematically represented as:

$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i)$$

where W_{ih} is the weight matrix associated with the hidden state h_{t-1}, W_{ix} is the weight matrix associated with the input x_t, and b_i is the bias term.

The value of i_t is bounded between 0 and 1, where 0 indicates no addition and 1 keeps all the information to be added.

A tanh layer produces a vector of new candidate values, noted by $\tilde{C}$, to be added to the state. The candidate cell state $\tilde{C}$ is calculated by:

$$\tilde{C}_t = \tanh(W_{hh}h_{t-1} + W_{hx}x_t + b_h)$$

where W_{hh} is the weight matrix associated with the hidden state h_{t-1}, W_{hx} is the weight matrix associated with the input x_t, and b_h is the bias term.

$\tilde{C}$ is bounded between -1 and 1 as it is an output from the tanh function. A negative value indicates a subtraction of information from the cell state, and a positive value suggests that information is added to the cell state.

The two components of the input gate are combined to update the cell state. The new cell state C_t forgets the information determined by the forget gate and adds the information from the input gate. This is mathematically defined by:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

3.8.1.3 Output Gate The output gate determines which information is used to update the hidden state.

Firstly, a sigmoid layer is utilised to decide what part of the cell state to be outputted, calculated by the equation:

$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_i)$$

where W_{oh} is the weight matrix associated with the hidden state h_{t-1}, W_{ox} is the weight matrix associated with the input x_t, and b_o is the bias term.

Then, the hidden state is calculated by

$$h_t = o_t + \tanh(C_t)$$

3.8.2 Justification

LSTM is suitable for analysing and learning image captions for multi-label classification tasks. Unlike RNN, LSTM can effectively memorise text patterns, retain relevant information, and discard irrelevant information. Leveraging its memory capabilities, LSTM can use multiple word strings to classify text into different classes. With the use of appropriate embeddings to convert text into vectors, the performance of LSTM can be further improved.

3.8.3 Advantages or Novelty

The major advantage of LSTM is its ability to capture long-term dependencies. This ability allows the model to capture information from sequential data, such as image captions. By utilising the forget gate, input gate, and output gate, the LSTM model can select the most important information to memorise, hence improving the prediction performance.

3.9 CLIP Model

The integration of language and vision in machine learning has led to significant advancements in multimodal learning. One of the notable contributions in this domain is the development of the CLIP (Radford et al., 2021). CLIP represents a paradigm shift in bridging the gap between visual and textual data through a novel training approach and architecture design (Klingler, 2023).

3.9.1 Principles of CLIP

3.9.1.1 Architecture CLIP employs two encoders: an image encoder and a text encoder. The image encoder uses a convolutional neural network (CNN), such as ResNet, to transform visual inputs into high-dimensional vectors that capture essential visual features (Radford et al., 2021). Concurrently, the text encoder, typically based on transformer architectures like BERT, processes textual descriptions into corresponding high-dimensional vectors. These vectors are produced in a shared vector space, enabling the direct alignment and comparison of textual and visual representations.

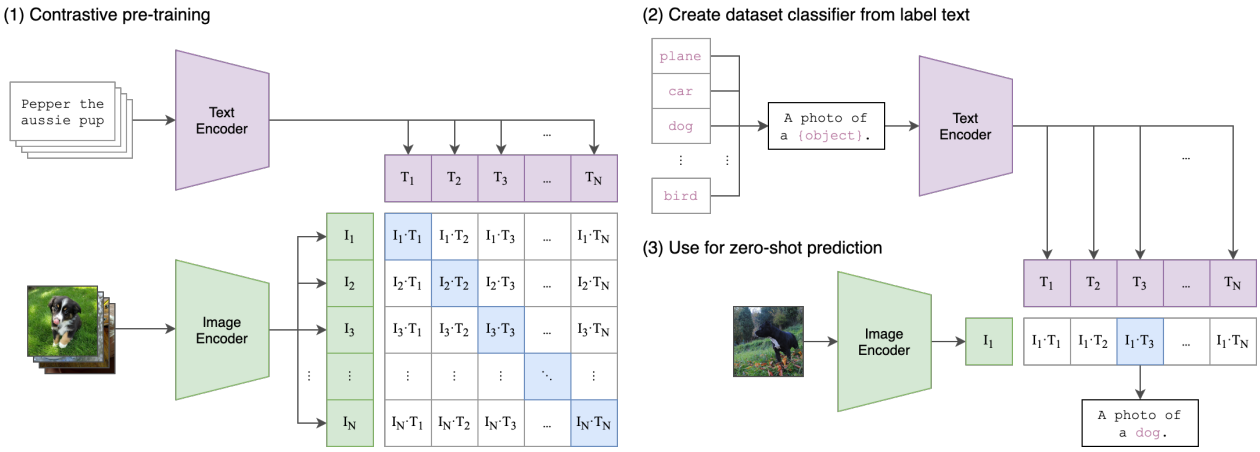


Figure 11: CLIP Training Process

3.9.1.2 Training Process

Step 1: Contrastive Pre-training As shown in Figure 12, CLIP undergoes pre-training on a vast dataset comprising 400 million internet-sourced image-text pairs (Chuang et al., 2020). This stage involves presenting the model with correctly matched image-caption pairs and deliberately mismatched pairs to foster a shared latent space for embeddings.

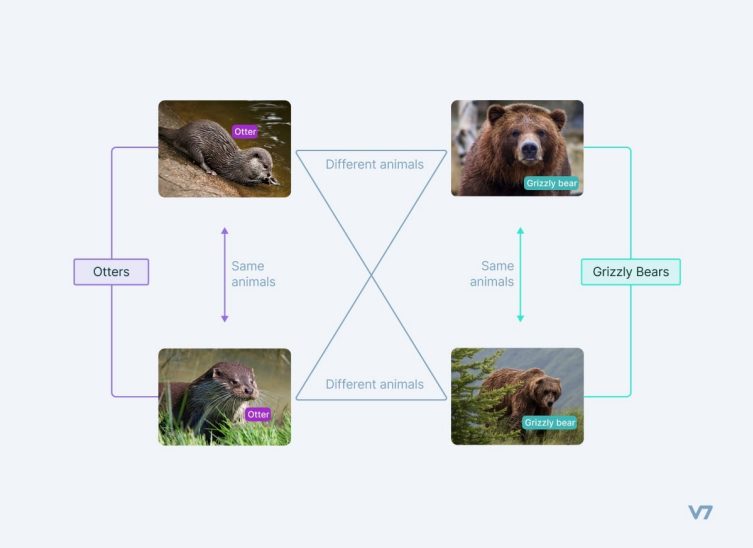


Figure 12: Example of Contrastive Learning

Step 2: Dataset Classifier Creation from Label Text As Shown in Figure 11, for each image, multiple textual descriptions are crafted—including accurate and inaccurate variants—to establish a combination of

positive and negative training examples. These are processed by the text encoder to create class-specific embeddings, leveraging a contrastive loss function that penalizes mismatches and rewards correct associations.

Step 3: Zero-shot Prediction Post-training, the text encoder acts as a zero-shot classifier, enabling predictions on new images without fine-tuning. The image and text encoders optimize to heighten similarities between correct pairs and lessen them for incorrect ones, thus perfecting a multimodal embedding space where semantically related images and texts are closely aligned. The predicted class corresponds to the highest similarity score.

3.9.1.3 Integration Between Natural Language and Image Processing

CLIP's dual capability in mapping both images and texts into a unified vector space enhances the integration of natural language processing (NLP) and image processing tasks. This integration enables functionalities such as generating textual descriptions for images, classifying images using only textual cues in a zero-shot fashion, and editing images based on textual prompts. These capabilities establish a robust foundation for innovative applications in text-to-image generation and manipulation.

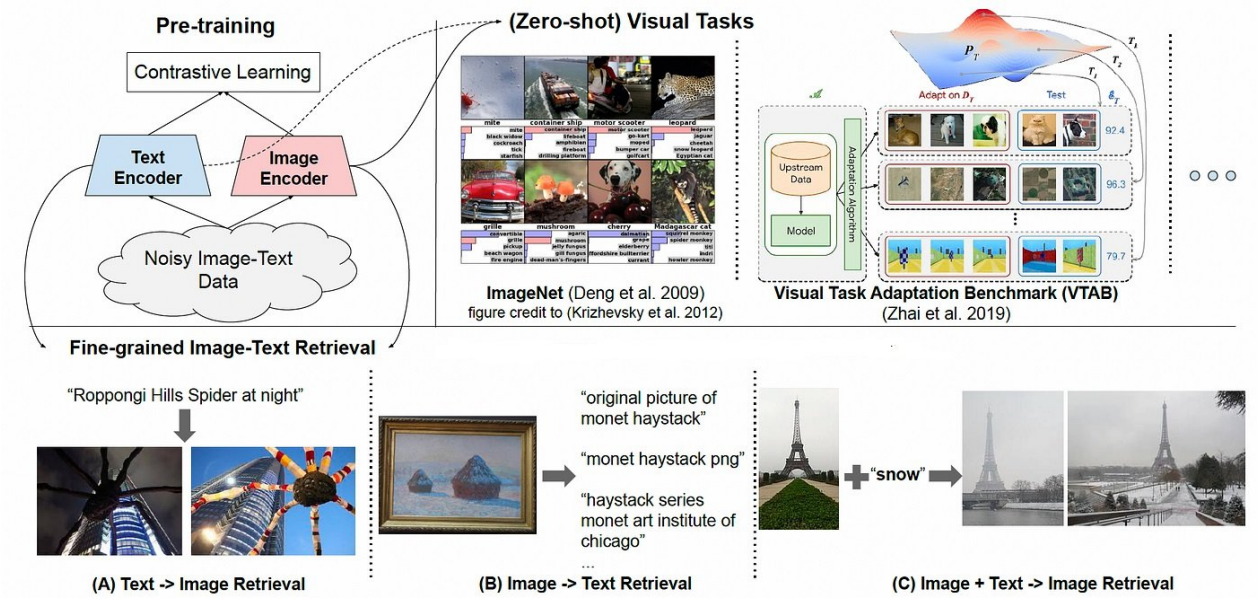


Figure 13: Example of CLIP Integrate NLP

3.9.2 Justification

CLIP model is chosen for its outstanding ability in handling multi-modal data and robustness to class imbalance mentioned earlier.

3.9.2.1 Handling Multi-Modal Data

CLIP is designed to handle and synthesize information from both images and text. Given that the dataset not only includes images but also accompanying captions, CLIP's dual-encoder structure (one for images and one for text) is ideal. This structure allows it to develop a joint understanding of visual and textual data, which is crucial for accurately interpreting the context provided by captions and associating it with the visual cues in the images.

3.9.2.2 Robustness to Class Imbalance

Given the earlier discussion about class imbalance in the dataset, CLIP's contrastive learning framework provides a distinct advantage. It focuses on aligning the correct image-text pairs among a vast dataset, which inherently boosts its capability to recognize minority classes by leveraging the descriptive power of text, thus mitigating some of the effects of skewed class distribution.

3.9.3 Advantages or Novelty

3.9.3.1 Zero-Shot Learning Capabilities One of the standout features of CLIP is its zero-shot learning capability. This feature allows the model to perform image classification tasks it hasn't been explicitly trained on by using natural language descriptions. In the context of the assignment, this means that even if certain label combinations are underrepresented in the training data, CLIP can still potentially classify them correctly based on its understanding of the captions.

3.9.3.2 Enhanced Feature Representation CLIP's use of advanced architectures for both its image and text encoders (like ResNet and transformers, respectively) means that it can extract and integrate complex features from both modalities. This integrated feature extraction is more powerful than using separate, unimodal methods, as it allows for a deeper and more nuanced understanding of the content, leading to more accurate classifications.

3.9.3.3 Scalability and Efficiency The pre-trained nature of CLIP offers another practical advantage: efficiency. Using a pre-trained model like CLIP can significantly reduce the time and computational resources required for training. Moreover, CLIP's ability to handle diverse and extensive datasets makes it scalable, a crucial factor for adapting to future expansions of the dataset or task.

4 Experimentation and Results

This study presents a comprehensive analysis comparing the performance of two models, CLIP and GoogLeNet with LSTM, for multi-label classifications.

The experimental approach analyses various model configurations, including trainable layers, dropout rates, and learning rates. Hyperparameter analysis is conducted to assess the impact of different parameter values on model performance, and an ablation study is performed to evaluate the contributions of model components on classification performance.

The experiment used a dataset of 29,996 images. 80% of the images are randomly allocated to be the training dataset and the remaining 20% of the images are selected to be the validation dataset, from which the model performance is calculated.

The analysis compares different configurations of the model and evaluates them using the F1 score, as it effectively balances both precision and recall. The comparison of the performance is made between the two different models and their configurations to identify the best-performing model and its structures.

4.1 Experiment Models

4.1.1 CLIP

4.1.1.1 Image Processing Data is managed through custom dataset classes that handle image loading, transformation, caption tokenization, and label encoding:

The input layer of the image encoder in CLIP is designed to accept images of a fixed size, typically $224 \times 224 \times 3$ for RGB images. This dimensionality ensures that the input tensor is consistent, crucial for batch processing in neural networks. The image data, now a $224 \times 224 \times 3$ tensor, passes through several convolutional, normalization, and activation layers, each mathematically transforming the input data into higher-level features. This can be represented for a convolutional layer as:

$$F_{\text{out}} = \text{GeLU}(W * F_{\text{in}} + b)$$

where F_{out} is the output feature map, W is the weight matrix of the convolutional filter, $*$ denotes the convolution operation, b is the bias, and GeLU is the non-linear activation function.

4.1.1.2 Base Model Sturcture In the context of this project, we harness CLIP's dual-encoder framework for multi-label image classification. This task presents unique challenges, particularly with the dataset which exhibits a significant class imbalance—specifically, the dominance of certain classes such as class 1. CLIP's ability to learn generalized visual-semantic embeddings allows us to address these challenges effectively. By leveraging both the visual features of the images and the contextual information provided by their captions, CLIP provides a robust framework for understanding and categorizing the complex interrelations between multiple labels in an image.

In the experiment, the CLIP model (ViT-B/32) was employed for a multi-label image classification task that integrates textual descriptions to enhance prediction accuracy. This model capitalizes on the strength of Vision Transformers for image processing combined with a language model for text (Radford et al., 2021). Our goal is to validate the efficacy of this approach on a dataset with complex labeling structures, where each image is annotated with multiple labels. The primary components and configurations of the CLIP model used in our experiments are as shown in Table 1:

Model Component	Value
Image Model	CLIP (ViT-B/32)
Text Model	Vission Transfomer (ViT)
Activation Function (Hidden Layers)	GELU
Activation Function (Output Layers)	Sigmoid
Loss Function	BCE
Optimiser	Adam
Batch size	64
Epoch	30
Learning Rate	0.001
patience (scheduler)	5

Table 1: Structure of Base CLIP Model

Experiment Design Moving forward, we aim to explore alternative optimizers like AdamW for potentially better weight regularization and Focal Loss to address challenges associated with class imbalance. These experiments will help refine our approach, aiming to enhance both the precision and recall of the model across various classes. Furthermore, We will continue using the Cosine Annealing scheduler and possibly explore other schedulers to optimize learning rates dynamically.

4.1.2 GoogLeNet + LSTM

A proposed model for multi-label classification tasks combines GoogLeNet with LSTM. GoogLeNet is implemented to train image datasets, while LSTM is utilised to extract textual information from image captions. The output of GoogLeNet and LSTM are combined with a fully connected layer. The prediction generated by the integrated model considers both visual and textual information, enhancing prediction accuracy and F1 score.

Table 2 outlines the components and hyperparameters of the proposed GoogLeNet + LSTM model. This model is referred to as the 'Base Model' in the later part of this report.

In the implemented model, GoogLeNet is initialized with pre-trained weights. During the training phase, all layers of the GoogLeNet, except for the last fully connected layer, have their weights frozen. This prevents updates to the pre-trained weights, and the parameters from the pre-trained model remain unchanged throughout the training process.

Model Component	Value
Sampler (Data Loader)	WeightedRandomSampler
Batch size	128
Epoch	30
Image Model	GoogLeNet
Text Embedding	GloVe
Text Model	LSTM
Dropout	0.2
Loss Function	FocalLoss
Optimiser	Adam
Learning Rate	0.001
patience (scheduler)	2

Table 2: Structure of GoogLeNet + LSTM Model

Text embedding converts textual data into numerical vectors, which then can be processed by machine learning algorithms. In this model, the 'glove-wiki-gigaword-50' embedding technique from GloVe is used. The pre-trained embedding technique contains 50-dimensional pre-trained word vectors from an uncased corpus of 6 billion tokens and a vocabulary of 400,000 words based on Wikipedia 2014 and English Gigaword Fifth Edition. As this embedding is also pre-trained, the weights of the embeddings are frozen, so they do not undergo updates throughout the training process. Then, LSTM is added to capture the information from the text embedding.

To reduce the risk of overfitting the model, a dropout probability of 0.2 is applied to both the GoogLeNet and LSTM of the model. FocalLoss is used as the loss function to measure the disparity between the prediction produced by the GoogLeNet + LSTM model and the true image labels. Adam is used as the chosen optimizer. The learning rate is initially set to 0.001. However, if there is no improvement of the metrics observed by the scheduler over 2 epochs, the learning rate is reduced by a factor of 0.5 to prevent the performance of the model from stagnating in the local minima.

4.2 Hyperparameter Analysis

4.2.1 GoogLeNet + LSTM

4.2.1.1 Dropout Rate Table 3 details the training and validation performance for different dropout rates. The model achieves optimal performance with a dropout rate of 0.2. Higher dropout rates create a greater regularisation effect, preventing the model from overfitting the training data. This is evident from the training F1 score, where the metric is highest with the lowest dropout rate. However, if the dropout rate is too high, the model may not learn a sufficient amount of data features to generate accurate classifications.

Dropout Rate	Validation F1 score	Training F1 score
0.2 (Base Model)	**0.75**	**0.93**
0.5	0.72	0.84
0.8	0.71	0.86

Table 3: GoogLeNet + LSTM Performance with Different Dropout Rate

4.2.1.2 Batch Size Table 4 lists the performance of the model with three different batch sizes. The highest validation F1 score is achieved with a batch size of 128.

A smaller batch size of 64 achieves a slightly lower performance, with a validation F1 score of 0.74. A larger batch size of 256 performs the worst among the three batch sizes examined, with a training F1 score of 0.83

and a validation F1 score of 0.7. Larger batch size can lead to poorer generalisation as it tends to converge to sharp minima of the loss function.

Batch Size	Validation F1 score	Training F1 score
64	0.74	0.92
128 (Base Model)	**0.75**	**0.93**
256	0.7	0.83

Table 4: GoogLeNet + LSTM Performance with Different Batch Size

4.2.1.3 Learning Rate Three learning rates are examined in the hyperparameter analysis, and the lowest learning rate achieves the best training and validation performance. This result aligns with expectations, as a lower learning rate allows the model to make smaller weight updates, reducing the likelihood of overshooting local minima during training. In contrast, a high learning rate increases the risk of the model overshooting the optimal minima of the loss function, resulting in poor convergence.

Learning Rate	Validation F1 score	Training F1 score
0.1	0.62	0.6
0.05	0.71	0.89
0.001 (Base Model)	**0.75**	**0.93**

Table 5: GoogLeNet + LSTM Performance with Different Learning Rate

4.2.1.4 Loss Function Two loss functions, BCE Loss and Focal Loss, are explored in this hyperparameter analysis. Table 6 compares the model performance between the two. BCE Loss and Focal Loss achieve similar training F1 scores of 0.92 and 0.93 respectively. However, the model trained with Focal Loss demonstrates significantly better performance during validation. This performance discrepancy indicates the importance of addressing class imbalance during the training process. Since Focal Loss is designed to address class imbalance and prioritise hard-to-classify classes, it achieves a better performance than BCE Loss by preventing overfitting to the dominant class's features.

Loss Function	Validation F1 score	Training F1 score
BCE Loss	0.5	0.92
Focal Loss (Base Model)	**0.75**	**0.93**

Table 6: GoogLeNet + LSTM Performance with Different Loss Function

4.3 Ablation Studies

An ablation study is performed to analyze the effects of each component within the model on the overall performance of the neural network. Different model features are removed, and F1 score is calculated to evaluate the model performance without certain components.

As CLIP and GoogLeNet+LSTM models have different model implementations, separate ablation studies were conducted for each. Section 4.3.1 details the findings of the ablation study for the CLIP model, and Section 4.3.2 explains the result for the GoogLeNet+LSTM model.

4.3.1 CLIP

The ablation study conducted on the CLIP model, leveraging both Adam and AdamW optimizers with BCE and Multi-Label Focal Loss functions, reveals insightful trends. The study's results, summarized in Table 7,

indicate variations in Validation Accuracy, Micro F1, and Macro F1 scores across different configurations.

Optimizer	Loss Function	Validation Accuracy	Validaion Macro F1	Validation Micro F1
AdamW	BCE	0.979	0.768	0.857
	Multi-Label Focal	**0.986**	**0.798**	**0.891**
Adam (Base)	BCE (Base)	0.976	0.763	0.849
	Multi-Label Focal	0.981	0.775	0.862

Table 7: Ablation Study for CLIP

4.3.1.1 AdamW with Multi-Label Focal Loss: This configuration achieved the highest Validation Accuracy (0.986) and Micro F1 score (0.891), suggesting superior performance in managing class imbalance and improving prediction precision across multiple labels. The Focal Loss, designed to focus more on misclassified instances, complements the AdamW optimizer, which incorporates weight decay to prevent overfitting, enhancing generalization.

4.3.1.2 Adam with Multi-Label Focal Loss: Shows a significant improvement over the base BCE loss, with a higher Macro F1 score of 0.862 compared to 0.849 with BCE. This indicates the effectiveness of Focal Loss in handling datasets with uneven label distribution, which is likely in multi-label settings where certain labels are more sparse.

Switching from Adam to AdamW with BCE loss sees an increase in both Validation Accuracy and F1 scores, highlighting AdamW's advantage in dealing with complex datasets typical of CLIP's training regime. CLIP models, pre-trained on a diverse range of internet-sourced text-image pairs, are optimized for generalization, and AdamW's weight regularization is particularly effective in leveraging this pretraining for fine-tuned tasks.

4.3.1.3 Conclusion

Optimizer Efficiency AdamW: Demonstrates superior performance over Adam, particularly with the Multi-Label Focal Loss. AdamW incorporates weight decay directly into the parameter updates, which helps in regularizing the model and reducing overfitting (Loshchilov and Hutter, 2017). This characteristic is particularly beneficial given the CLIP model's exposure to diverse training data, where maintaining generalization across varied inputs is crucial.

Loss Function Effectiveness Multi-Label Focal Loss: Shows marked improvements in both Micro and Macro F1 scores compared to BCE. The Multi-Label Focal Loss is an advanced adaptation of the traditional Focal Loss specifically tailored to address class imbalance in multi-label classification tasks, which is especially the case in the dataset (Mukhoti et al., 2020). The higher scores reflect its effectiveness in enhancing model sensitivity and precision across the less frequent, yet critical labels.

Impact on Validation Accuracy and F1 Scores: The improved Macro F1 scores under the Focal Loss setups suggest better handling of label imbalances and an enhanced ability to model the minority classes effectively. Given the nature of the CLIP model, which has been pre-trained on a mixture of easily and rarely seen image-text pairs, focusing on harder-to-classify examples again ensures the model will be less biased against dominant class '1' in the dataset.

Feature	Parameter	Validation F1 score	Training F1 score
Base Model	Deafult Parameter	0.75	0.93
WeightedRandomSampler	sampler = None	0.78	0.79
Dropout	p = 0	0.7	0.89
Learning Rate Scheduler	scheduler = None	0.73	0.88
GoogLeNet Trainable Layers	param.requires_grad = True	0.66	0.95
Embedding Trainable Weights	weight.requires_grad = True	0.7	0.92

Table 8: Ablation Study for GoogLeNet + LSTM

4.3.2 GoogLeNet + LSTM

Table 8 lists the training and validation F1 score for the GoogLeNet + LSTM model through ablation studies. The structure of the base model is listed in Section 4.1.2. The base model achieves a training F1 score of 0.93 and a validation F1 score of 0.75.

4.3.2.1 Weighted Random Sampler WeightedRandomSampler is a sampler used in DataLoader that helps to balance the class distribution within batches, thus preventing oversampling of a certain class due to imbalanced data during the training process. Removal of this sampler achieves a training F1 score of 0.79 and a validation F1 score of 0.78. The validation F1 score is higher than the score achieved by the base model. However, both training and validation data originate from the same dataset and they may inherit similar class imbalance characteristics. Therefore, the WeightedRandomSampler is implemented in the final prediction model to better accommodate potential class imbalances in unseen data, despite a lower validation F1 score in this ablation study.

4.3.2.2 Dropout Layer The dropout layer is a regularisation technique in both GoogLeNet and LSTM architectures to reduce overfitting. This feature can be removed by setting the dropout probability to 0, which converts the layer into an identity operation. This decreases the validation F1 score to 0.7 and the training F1 score to 0.89. Both of the training and validation F1 scores are less than the base model, suggesting an overfitting of the model when regularisation is omitted.

4.3.2.3 Learning Rate Scheduler Removing the learning rate scheduler eliminates the adjustment of the learning rate if no improvement in metrics is observed. The absence of learning rate adaption creates a higher risk of the model's performance stagnating in the local minima. As shown by the results of the ablation study, both the training and validation F1 scores are lower, at 0.73 and 0.88 respectively, compared to the F1 score achieved by the base model, where a learning rate scheduler is implemented.

4.3.2.4 GoogLeNet Trainable Layers In the base model, only the parameters of the last fully connected layer in GoogLeNet are updated. In this ablation study, the parameters of all layers in GoogLeNet can be updated, so GoogLeNet is more adaptable to the training dataset. As a result, the training F1 score is high at 0.95, but the validation F1 score is only at 0.66. Enabling all layers of the GoogLeNet to be trained leads to the model learning too much information about the training data and reducing its ability to generalize validation.

4.3.2.5 Embedding Trainable Weights The weights for the pre-trained text embedding are not updated in the base model during the training process. In the base model, the significance assigned to each word is based on the pre-training data rather than the dataset used in this study. In the ablation study, all weights in the text embedding layer can be updated, in order to investigate the impact of trainable embedding on model performance. Although the training F1 score is similar to the base model at 0.92, the validation F1 score declines to 0.7. The observation suggests that updating the weights for text embedding can lead to overfitting, and reduce the model's ability to generalize and make accurate predictions on unseen data.

4.4 Model Comparison and Performance Evaluation

The Final GoogLeNet and CLIP Models exhibit notable differences in their architecture and hyperparameters, which directly influence their performance metrics. This subsection will analyze these aspects and reveal how structural choices and parameter settings are correlated with the outcomes in validation accuracy, F1 score, precision, and recall.

Model Component	Final GoogLeNet	Final CLIP Model
Batch size	128	64
Epoch	30	30
Dropout	0.2	N/A
Loss Function	FocalLoss(Gamma = 0.7)	FocalLoss (Gamma = 0.7)
Optimiser	Adam	AdamW(weight decay=$1e-2$)
Learning Rate	0.001	0.001
Activation Function (Hidden Layers)	ReLU	GELU
Activation Function (Output Layers)	Sigmoid	Sigmoid
patience (scheduler)	2	5
Image Model	GoogLeNet	CLIP (ViT-B/32)
Text Embedding	GloVe	CLIP
Text Model	LSTM	Vision Transformer (ViT)

Table 9: Model Hyperparameter Comparison

The CLIP Model, leveraging a Vision Transformer (ViT) for both image and textual embeddings, achieves a significantly higher validation accuracy of 0.986 compared to GoogLeNet's 0.938. This discrepancy is largely attributed to the transformer's capability to process and integrate complex patterns across modalities more effectively than the CNN-based GoogLeNet. Transformers, such as ViT, inherently capture broader contextual relationships within the data, making them adept at handling the intricacies of a diverse dataset that includes a wide range of image-text pairings. This ability is further enhanced by the CLIP Model's use of AdamW optimizer with weight decay, which ensures better generalization by minimizing overfitting—a crucial advantage when dealing with extensive and varied pre-training datasets.

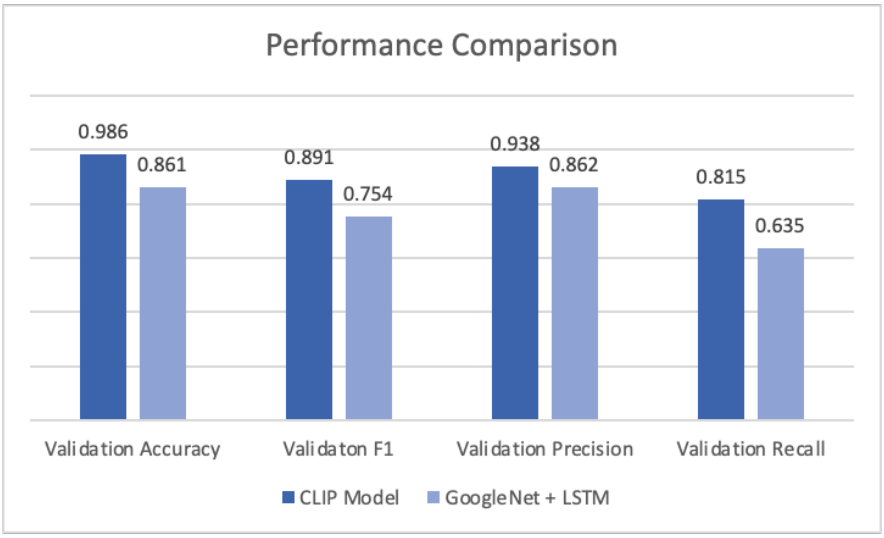


Figure 14: Model Performance Comparison

In terms of F1 scores, the precision for CLIP is considerably higher (0.938) compared to GoogLeNet (0.862), indicating CLIP's superior capability to correctly identify relevant instances across its predictions. This is a direct result of the effective regularization strategies employed by AdamW and the model's robust learning dynamics facilitated by the GELU activation function, which ensures smoother gradients and more stable updates during training.

However, while CLIP outperforms in precision, its recall of 0.815, though higher than GoogLeNet's 0.635, indicates a lesser disparity between these two metrics compared to GoogLeNet. The more balanced precision and recall in CLIP suggest its proficiency in not only identifying relevant instances but also in capturing a majority of the positive cases across the dataset. The lower recall in GoogLeNet can be linked to its CNN architecture, which may not generalize as well across the multimodal data and possibly gets biased towards more frequently occurring labels due to the dataset's imbalance.

The use of Focal Loss in both models aims to address the issue of class imbalance by focusing more on harder, misclassified cases, which likely contributes to the improvements in precision and recall over models that might use a simpler loss function like BCE. However, the architectural superiority of CLIP, combined with its advanced optimizer and activation functions, appears to harness the benefits of Focal Loss more effectively than GoogLeNet.

Furthermore, the dropout in GoogLeNet, intended as a regularization measure, may not be sufficient alone to combat the overfitting and generalization issues exacerbated by the class imbalance in the dataset. In contrast, the lack of dropout in CLIP is compensated by the inherent capabilities of the Vision Transformer and the AdamW optimizer, which together foster a more adaptable and resilient learning environment.

5 Conclusion and Limitations

5.1 Final Model Choice

The decision to adopt the Final CLIP model as the preferred architecture for this multi-label classification task is substantiated by a combination of its empirical performance and inherent architectural advantages, as detailed in Section 4.4 model comparison and performance evaluation.

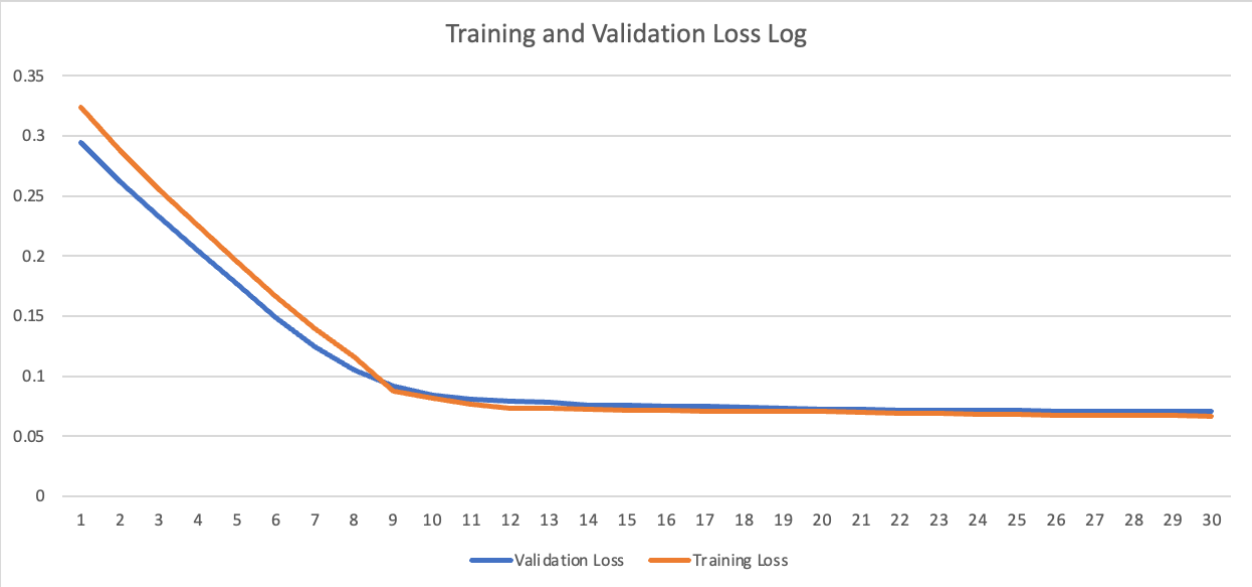


Figure 15: Training and Validation Loss Log

The provided Training and Validation Loss Log graph in Figure 15 displays a clear convergence pattern where both training and validation loss decrease significantly and stabilize as training progresses. Notably, the validation loss closely tracks the training loss, indicating that the CLIP model is not overfitting and generalizes well to unseen data.

To summarize, The Final CLIP model's superior handling of multi-modal data, robust performance across various metrics, and stable loss convergence all contribute to its selection as the model of choice. Its architectural strengths in processing and integrating complex data types provide a significant advantage over

traditional models like GoogleNet, which lacks the same level of integration and optimization for multi-modal datasets. Furthermore, the empirical evidence of its performance during the training and validation phases confirms its effectiveness and efficiency, making it an optimal choice for tackling the challenges presented in the dataset.

In the context of this Kaggle Competition, this model was tested on Kaggle and the correct labels are hidden.

Figure 16: Competition Test F1

This score aggregates the contributions of all classes to compute the average F1 score, thus giving a weight to each instance or prediction, proportional to its class size. Achieving a micro F1 score near 0.925 as shown in Figure 16 suggests several things:

High Precision and Recall Balance: This high score indicates a strong balance between precision (the ability of your classifier not to label a negative sample as positive) and recall (the ability of your classifier to find all the positive samples). It suggests that the model is generally effective at identifying relevant labels without excessively misclassifying negative cases as positive.

Effective Handling of Class Imbalance: Given that some labels are more frequent than others, as suggested by the label distribution in your dataset, a high micro F1 score indicates that your model manages to perform well across both frequent and infrequent labels.

5.2 Limitations

Several inherent and design limitations may impact the generalizability and effectiveness of final prediction results and can be further improved. These limitations are mostly due to the trade-off between computational efficiency and accuracy, which always limits people from further grinding.

5.2.1 Complex Relationships and Bias

CLIP's ability to interpret complex relationships, such as emotional expressions or abstract concepts, is limited, potentially leading to misclassifications in contexts that require a deep understanding of human experiences. Additionally, biases inherent in the pre-training data can lead to outcomes that might perpetuate and amplify societal biases, posing ethical concerns in sensitive applications like content moderation.

5.2.2 Granular Detail Recognition

While CLIP excels at broad, high-level tasks, it struggles with the finer nuances and subtle distinctions within complex images or texts. This limitation reduces its efficacy in applications requiring detailed analysis, such as specialized medical imaging or nuanced content filtering.

5.2.3 Data Processing Enhancements

Incorporating advanced data augmentation strategies could mitigate some of the dataset limitations we face, such as class imbalance. Techniques like synthetic data generation or sophisticated resampling could enhance the model's ability to generalize better from underrepresented data points.

5.2.4 Rigorous Validation Methods

Implementing comprehensive validation techniques, such as k-fold cross-validation, could provide a deeper insight into the model's performance across diverse dataset segments, helping to pinpoint areas of underperformance or overfitting.

References

Chuang, C.-Y., Robinson, J., Lin, Y.-C., Torralba, A., and Jegelka, S. (2020). Debiased contrastive learning. *Advances in neural information processing systems*, 33:8765–8775.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *arXiv preprint arXiv:1409.4842*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Klingler, N. (2023). Clip: Contrastive language-image pre-training (2024).

Leaky AI (2023). Why deep learning is the most important a.i. technology to learn now. Accessed: 2024-05-10.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Mukhoti, J., Kulharia, V., Sanyal, A., Golodetz, S., Torr, P., and Dokania, P. (2020). Calibrating deep neural networks using focal loss. *Advances in Neural Information Processing Systems*, 33:15288–15299.

OpenAI (2023). Clip: Connecting text and images. Accessed: 2024-05-10.

Patel, M. (2021). The complete guide to image preprocessing techniques in python. Accessed: 2024-05-05.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Ruby, U. and Yendapalli, V. (2020). Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(10).

Tao, L. (2024). Multi-label classification competition 2024.

Zouhar, V., Meister, C., Gastaldi, J. L., Du, L., Vieira, T., Sachan, M., and Cotterell, R. (2023). A formal perspective on byte-pair encoding. *arXiv preprint arXiv:2306.16837*.

Appendices

A Hardware and Software Specification

For the experiments detailed in this report, we used a MacBook Pro with Apple M1 Pro integrated chip and 16 GB RAM to train the model. This local environment provides a consistent and stable platform for model training and evaluation.

All experimental code and data were subsequently uploaded to Google Colab to facilitate sharing and accessibility for collaborative purposes.

However, it should be mentioned that variations in computational architecture and performance between the local MacBook Pro and the Google Colab environments may lead to discrepancies in results when the experiments are replicated or continued on Google Colab's cloud-based servers.

B Instruction of Compiling Code

Please go to this shared google colab

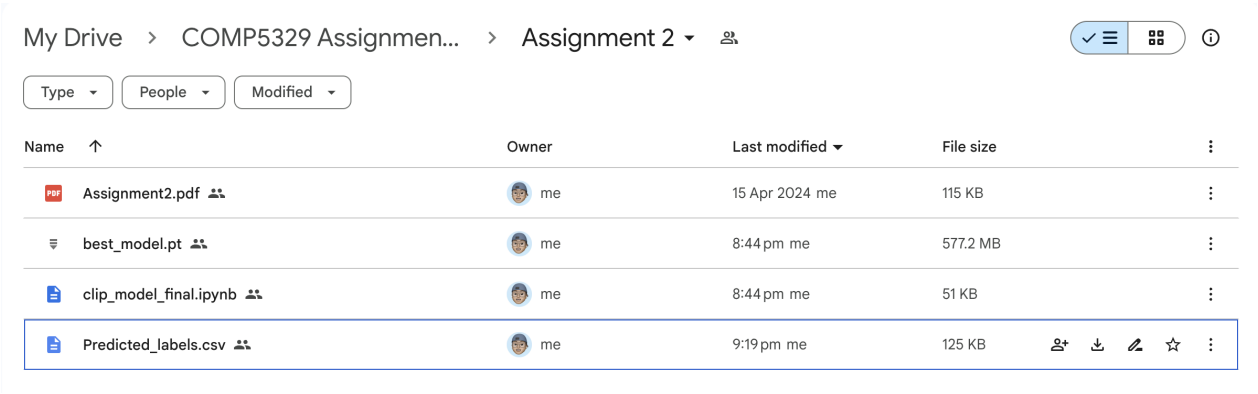


Figure 17: Screenshot of The Google Drive Space

There, you will see the assignment description which details the tasks required.

`best_model.pt` which are trained using the mentioned parameters. It can be loaded and tested without further training.

`clip_model_final.ipynb` contains the code in jupyter notebook format that loads, trains, and tests the data and model in one place. To run the code submitted, go to 'run time' and click run all for a model retraining process.

`Predicted_labels.csv` contains the prediction results on the test dataset.

Below is a snapshot of what the predicted labels file look like as per requirements in the assignment description

	A	B
1	ImageID	Labels
2	30000.jpg	1
3	30001.jpg	1
4	30002.jpg	1
5	30003.jpg	1
6	30004.jpg	1
7	30005.jpg	1
8	30006.jpg	11
9	30007.jpg	1 9
10	30008.jpg	1 9
11	30009.jpg	1
12	30010.jpg	1 10 3 8
13	30011.jpg	1 18 19
14	30012.jpg	7
15	30013.jpg	1
16	30014.jpg	1 8
17	30015.jpg	1
18	30016.jpg	1 3 2006
19	30017.jpg	1
20	30018.jpg	8

Figure 18: Enter Caption

C Disclaimer

Generative AI tools such as ChatGPT were used in helping to formulate words and phrases for a more formal way of expression.