Assignment 2 Mixed Network Classification

Sicheng Guo	Yicheng Si	Kecheng Li
510191768	500170456	50009357
sguo0672	yisi0700	keli9616

Abstract

In this assignment, We applied DenseNet structure multi-layer network and Multi-head Self-attention transformer encoder and obtained a mix net model which can predict the labels and description captions of specific images. Several preprocesses such as lemmatization, embedding and image augmentation are implemented to ensure effectiveness and wide applicability. We've conducted different experiments of various combinations of hyper parameters to figure out the model with the best performance and efficiency. Furthermore, we conducted research and comparison of different structures such as ResNet to have a deeper understanding of the influence of diverse network structure on the prediction result.

Keywords: DenseNet, Multi-head Self-attention

1 Introduction

Nowadays, the problem of image classification with multi-label is much concern in the field of modern deep learning. With the idea of backpropagation and multilayer perceptron proposed by Hinton in 1986 [1], many classification algorithms have been proposed to learn with images to handle multi-label classification tasks like Convolutional neural networks (CNN) and Generative adversarial networks (GAN). However, there is no very mature model that can handle the tasks of learning pictures and picture descriptions together. Thus, the aim of this study is about introducing different classification algorithms and image as well as natural language preprocessing methods. After comparing analysis and ablation study, this report tend to find the best classifier to handle image classification with multi-label tasks.

In the field of image classification, a common type of work is to use both image data and image caption to train neural networks for classification, which often will obtain better performance. At the same time, the neural network used in deep learning may tend to be complex in structure. Thus this study is important because it is going to find a balance between classification performance and complex structure. In addition, nowadays, many classical classification models cannot perform perfectly in multi-label classification tasks. This study can supplement the current field of image classification through experiments, since there is not a perfect model that can handle both image captions and image data.

We will conduct experiments on a specific dataset, during the experiment we will apply different models and parameters, and after collecting and analyzing the results we can find the optimal classification model. The dataset for the experiment contain 30,000 training data and 10,000 testing data which are consist of images and the related captions. In order to achieve our study aim, we will not only use related techniques of natural language process, such as preprocessing methods such as stopwords removal, tokenisation and stemming, but also experiment with different word embedding pre-training models (such as glove-twitter-25 and glove-twitter-100). In addition, we will also apply the techniques of the densenet model framework and transformer in the image classification model.

2 Related Work

Since the rise of deep learning, the general recognition represented by ImageNet data set has achieved a significant improvement in accuracy by leaps and bounds. After the general recognition performance is gradually ”saturated”, researchers focus on the more difficult fine-grained image recognition and multi-label image recognition.

Early method is speak more by the label of the N independent binary classification, will be more label recognition as a binary classification of N independent, respectively to predict each category is more labels to identify the most simple method, but the problem with this approach is that, did not consider the characteristics of multiple labels to identify the task itself, also is the co-occurrence dependence, so the classification effect is poorer.

As a simple and effective processing method, Attention is widely used in various visual tasks such as recognition and segmentation. Therefore, some researchers also introduce Attention into the field of multi-label recognition to implicitly model the spatial relations between different labels. However, this kind of method is generally common and can also improve performance in other recognition tasks. Similarly, attention-related work in other recognition tasks can also be effective in multi-label recognition tasks[2].

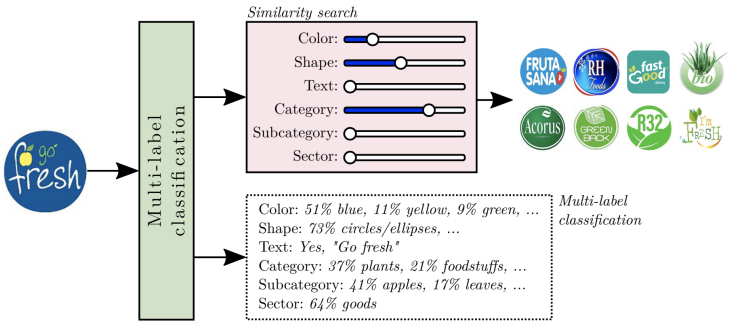

Figure 1: design of logo recognition method

A logo classification work presents a TIR method with a twofold purpose. A multi-label architecture is proposed that allows different labels to classify the characteristics of the logo, such as color, shape and image elements (semantics).

In this work, the authors propose a systematic multi-label classification and similar search for logo images. This approach allows the most similar identity to be obtained based on shape, color, line of business, semantics, general characteristics, or a combination of these characteristics established by the user. This is done by using a set of multi-label networks that specifically target certain characteristics of the logo.

This work presents a TIR method with a twofold purpose. A multi-label architecture is proposed that allows different labels to classify the characteristics of the logo, such as color, shape and image elements (semantics)[3].

The architecture combines, in a weighted manner, the most distinctive features of the representation flags learned by a series of multi-label classification networks dedicated to detection.

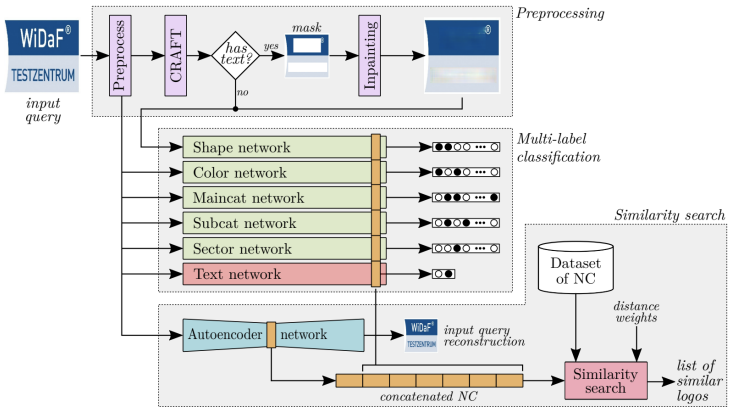

Figure 2: Scheme of the proposed method.

3 Techniques

3.1 Data Preprocess

3.1.1 Image Preprocess

Transforms.ColorJitter was used to change the attributes of the image to improve the recognition of the image. In the process of image preprocessing, the features of the image can be more easily extracted by the deep learning algorithm through RandomCrop and RandomHorizontalFlip and image enhancement methods.

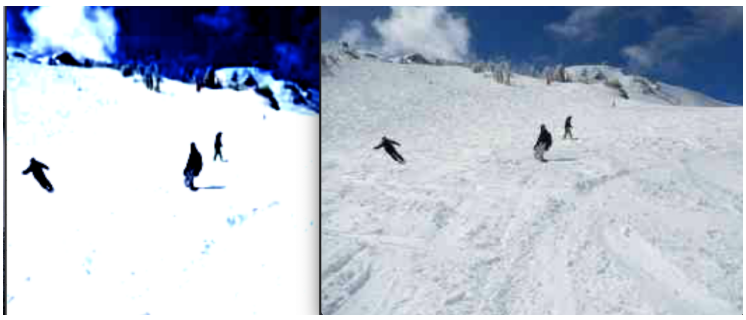

Figure 3: Change of tne image after the preprocess.

3.1.2 Label Preprocess

There are labels from 1-19 (12 is missing).The labels are converted into one-hot code, which benefit model's calculation of BCE loss function during back propagation.

3.1.3 Caption Preprocess

Punctuation removal

A regular expression is a logical formula used to manipulate strings. It uses predefined characters and the combination of these characters to form a "regular string". The "regular string" is used to filter strings. In order to make word training more efficient, regular expressions are used to clear all punctuation marks, so that later preprocessing results will be better.

Stopwords removal and Wrong spell

Stopwords are the most common words in any natural language. They tend to perform a structural role rather than semantic. So for the purpose of analysing text data and building NLP models, these stopwords are removed for a better performance. When checking the data, I found some misspellings, which would lead to the increase of the number of words and the wrong judgment in the training process, so I selected some words and added them to the stop words list in this experiment.

Case-folding

Conduct case-folding by reducing all letters to lower case on the text.

Lemmatization

Lemmatization is the process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form. After this process, the inflection can be removed. The algorithm can focus more one the word, instead of different type of the word.

Tokenization

Through the pre-processing process above, each sentence has become relatively clean. Tokenization can separate each word in a sentence, which is convenient for word vector transformation of each word. Besides, Padding is an important step before input model. In this step, a list of words was generated to analyze and delete words that appeared only once, thus reducing the amount of training.

Word Embedding

NLP is a method of word processing in the language model. This technique maps a word or phrase to an N-dimensional numerical vector. The core of this technique is a mapping relationship. By loading the word vector model pre-trained by Genism, every word is vectorized, and the text is transformed into the form of vector numbers, so that the computer can better understand the correlation between words.

Padding

In order for the data to be able to be entered into the model, it is necessary to see each sentence controlled to the same length, which is accomplished using the fill. After this step, each sentence will have the same length.

Embedding

Global Vectors for Word Representation(Glove)[4] is an unsupervised learning algorithm to obtain Word vector Representation. Various pre-trained word vectors are provided on GitHub. Through the later comparative experiment, the glove-Twitter-100 performed best in this experiment. There are 400 Thousand Vectors have been pre-trained in Glove-Twitter-100.

3.2 Network Structure

3.2.1 DenseNet169 Structure

DenseNet is a modified structure of CNN which consists of several dense blocks. The input of layers inside the DenseNet are the concatenation of feature maps form previous layers.[5] The structure can be simply represented as the following diagram.

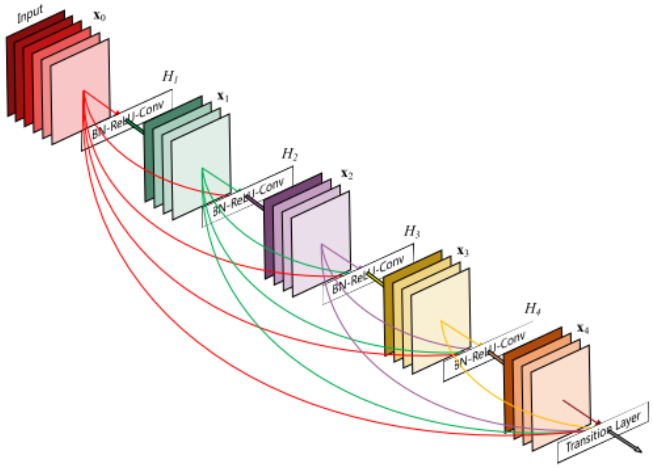

Figure 4: DenseNet Architecture

Unlike the summation to connect all preceding feature adopted by ResNet, DenseNet concatenate all features from preceding layers, alleviating the vanishing-gradient problem, strengthening feature propagation, encouraging feature reuse, and substantially reducing the number of parameters. The table below shows DenseNet structure with different dense blocks.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 5: Different DenseNet Architecture

According to research on comparison of different architectures, we found out that DenseNet169 has a better performance on image classification.[6] Therefore, we use it as a part of our model, which is more efficient in terms of parameters and computation for the same level of accuracy, compared with ResNet.[7]

3.2.2 Multi-head Self-Attention Transformer

Multi-head Self-attention transformer architecture refers to an attention mechanism relating different positions of a single sequence in order to compute several representations of the sequence. The calculation of representative value can be described as the figure below.

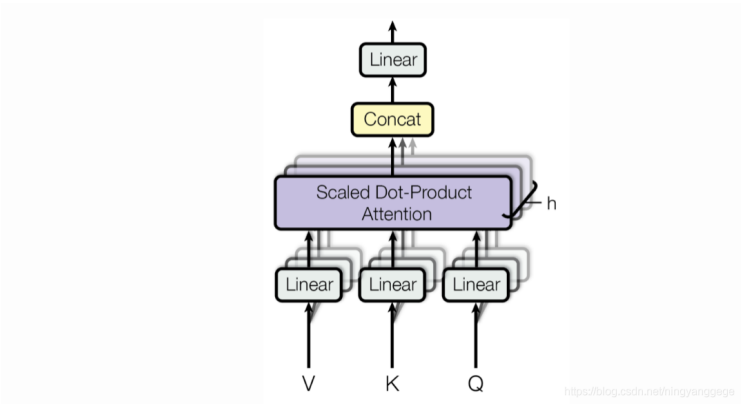

Figure 6: Multi-head Self-attention

Several heads will stimulate the complexion of models because queries, keys and values will be repeatedly generated several times and perform the attention function in parallel, generating multi-dimensional representative values for each sentence, benefiting features extraction and model augmentation.[8] Therefore, we implemented multi-head self-attention to captions and concatenate outputs to the images to train the model and predict.

3.2.3 Optimization

We used Adam optimization to fix weights during every epoch because it takes advantage of RMSProp and Momentum, enabling the model easier to obtain the global optimization during training. Moreover, the algorithm is efficient and intuitive to implement, which needs low memory less tuning than any other optimization algorithm. The updating process for Adam optimizer can be represents as the following formulas.[9]

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

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t \tag{2}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t \tag{3}$$

It can be found that Adam optimizer performs well when the dataset is large and complicated. Besides, it considers both of momentum and exponentially average decay of past squared weights gradient, making the update head towards global optimization while avoiding stuck in the local optimization.

3.2.4 BCE Loss

BCE loss refers to binary cross entropy loss between targets and input possibilities, which can be described by the following formula.

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = -w_n \left[y_n \cdot \log x_n + (1 - y_n) \cdot \log (1 - x_n) \right] \tag{4}$$

Since we changed the labels to one-hot code, the output should be the probability of the existence of each 20 labels. (excluding 12 since it is missed) We applied Sigmoid function to transfer the network output to numbers between 0 and 1 at the last layer of our model to represent the probability because multiple labels can exist at the same time. After that, BCE loss can be calculated according to the probability obtained from Sigmoid function, assisting in fixing weights during back propagation, which is suitable for objectives in this assignment.[10]

4 Experiment

4.1 Accuracy and Efficiency

The model is a combination of Multi-head Self-Attention Transformer and DenseNet16. After training with the best hyper-parameter, the highest accuracy of the final model was 0.8516 on the training set, and it took six minutes to evaluate the training of an Epoch. The selection and training convergence of Epoch are shown in the following section.

4.2 Ablation

In the preprocessing, many preprocessing methods are directly applied to the text, so whether to study the role of each and processing module is considered in this part. In this part, the three and processing methods were removed respectively before training. After training with the same EPOCH and other hyperparameters, the following results were obtained.

Table 1: Caption Preprocess Experiment

Removed	Loss	Precision	Recall	F1 Score
Baseline(all 3)	0.0729	0.832	0.8303	0.8311
Stopwords	0.0726	0.8229	0.8372	0.83
Lemmatization	0.073	0.8352	0.8298	0.8325
Wrong-spell	0.0757	0.8204	0.8289	0.8246

It can be seen that after the removal of lemmatization, the training effect has been slightly improved. Therefore, this process has been removed after comparison. It can be seen that not all preprocessing methods will have a positive impact on the data.

4.3 Comparison Analysis

The influence of different parameters and preprocessing methods on training has been compared in the previous part. After comparing, two pre-processing methods are used.For hyper-parameter the Threshold value is 0.3, and the Dropout value is 0.3. For details, please refer to the above parts. In the selection of results, the optimal parameters were found by comparing F1-score and Precision.

4.4 Hyper parameters

4.4.1 Threshold

According to BCE loss definition, we set a threshold for output probability to determine if the each label exist for each image respectively. For example, if the output of one label is 0.5 and threshold is 0.4, the label will be kept during training because the output is larger than the threshold. To figure out which threshold is the most suitable for the model training, we've conducted several experiments with different threshold values. The following table shows the results of different experiments.

Table 2: Threshold Experiment

Threshold	Loss	Precision	Recall	F1 Score
0.3	0.0685	0.819	0.8391	0.8289
0.4	0.071	0.884	0.7973	0.8394
0.5	0.0715	0.904	0.7859	0.8408

As the table shows, the precision is increasing and recall is descending when threshold value grows, which probably due to the greater constraint on prediction brought by higher threshold. Even though

the percentage of correct prediction among overall prediction increase, the number of correct prediction over all dataset reduce, which may brought by ignoring potential correct labels. Here we choose 0.3 as the threshold.

4.4.2 Dropout Rate

Dropout is a regularization technique for reducing overfitting in neural network by preventing co-adaptations on training data. Randomly chosen neurons will be deactivated during training. In this assignment, the dropout rate was initially set to 0.1 and we found out that validation error will increase after the 7th epoch while the training error still descend, which is an overfitting sign. Therefore, several experiments with different dropout rate have been conducted to figure out the suitable dropout rate for the model. The following table shows the results of experiments.

Table 3: Dropout Rate Experiment

Dropout	Loss	Precision	Recall	F1 Score
0.1	0.0711	0.8927	0.7894	0.8378
0.2	0.0671	0.9119	0.7942	0.849
0.3	0.0647	0.8675	0.8363	0.8516

As the table shows, the F1 score increase when the dropout rate increase, which may indicate that a larger rate will mitigate overfitting and makes model perform better on validation data. Hence, the generalization error will decrease to some extent. Here we choose 0.3 as the dropout rate.

4.4.3 Head Amount

In Multi-head Self-attention transformer, each sentence input will be represented by several head values extracted from the specific calculation of relationships among different words in each sentence. It can focus on particular sections of images with consideration of descriptive captions, gaining more features of images. We've conducted several experiments with different head amount to figure out the most suitable amount of heads for model training. The following table shows the experiments results.

Table 4: Head Amount Experiment

Embedding	Head	Loss	Precision	Recall	F1 Score
glove-twitter-25	5	0.071	0.884	0.7973	0.8394
glove-twitter-50	5	0.068	0.8788	0.8162	0.8464
glove-twitter-50	10	0.0712	0.8409	0.8336	0.8372
glove-twitter-100	5	0.0665	0.9033	0.8074	0.8526
glove-twitter-100	10	0.0696	0.8522	0.8314	0.8417

According to the results, different combinations of embedding models and head amount contribute diversely to the model performance. With the length of embedding vector increasing, the F1 score grows at the same time. On the other hands, more heads may not mean an improvement of model since the score relatively descend when the amount of heads raise. Therefore, choosing the suitable combination is of the most significance during training. Here we select glove-twitter-100 model and 5 heads.

4.4.4 Epoch

For the epoch parameters selecting, we selected the optimal model selected through the above experiments, including the word vector pre-training model using glove-twitter-100, and the model

framework of Densenet169. We initialize the number of epoch to 20 in order to find the inflection point of overfitting through the elbow diagram.

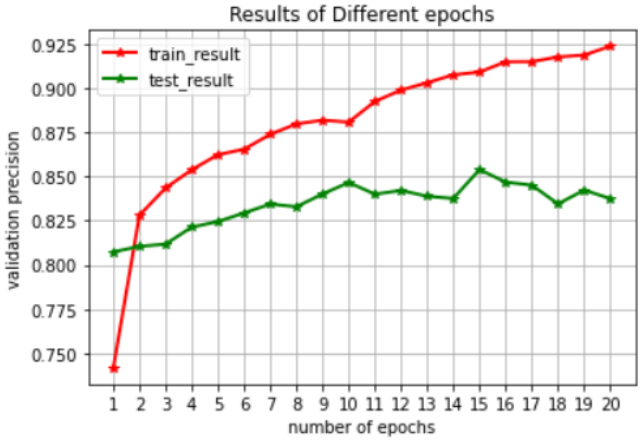

Figure 7: Result figure of different epochs

From the figure above, we can obviously see that the precision results on the validate data sets has been increasing from around 0.800 until it peaked at 0.8539 at the 15th epoch; after that, the model performance on the validate data began to gradually deteriorate. Therefore, our project team judged that our optimal model achieved the best effect at the 15th epoch. After that, although the accuracy of the training set has been increasing, it is likely to have an overfitting effect. Thus in the final epoch selection we chose a value of 15 which can lead to a more appropriate result.

5 Conclusion and Discussion

This study conducted experiments for a given multi-label image classification task. For annotated image classification, we adopted different techniques and models, and integrate them to improve the performance of the final model. Through extensive analysis such as ablation study, comparison analysis and hyperparameters selection, we found the best model. In terms of caption preprocessing, through ablation test, a suitable NLP preprocessing method was found, namely stopword removal and tokenisation. In the comparison analysis stage, we tested different image processing models such as CNN and Densenet, and finally found a suitable The model framework for our classification task, namely densenet169. At the same time, we also found suitable parameter values through hyperparameters, especially in preventing overfitting (through epochs analysis and adding dropout).

This study finds a suitable classifier to handle image classification with multi-label tasks by performing classification tasks on a specific image dataset. To some extent supplement the current field of image classification through experiments, given that many classical classification models lack the stable ability to handle both image captions and image data.

However, it has some limitations that could be addressed in future research. For example, in the future research, one may:

- Try a more efficient word vector processing method for image captions. Instead of using a pre-trained word embedding model, try to train the word embedding model based on the given data.
- Since the experiment is carried out through colab, the pictures are cropped and feature captured for the efficiency of the experiment. If you have enough time and resources, you can try to learn from more complete data.
- Use complex model architecture like Bi-LSTM which can have a closer look at the correlation between caption and image.

References

[1] E. R. David, E. H. Geoffrey, and J. W. Ronald, "Learning representations by back-propagating errors," 1986.

[2] F. Zhu, H. Li, W. Ouyang, N. Yu, and X. Wang, "Learning spatial regularization with image-level supervisions for multi-label image classification," 2017.

[3] M. Bernabeu, A. J. Gallego, and A. Pertusa, "Multi-label logo recognition and retrieval based on weighted fusion of neural features," 2022.

[4] T. Shi and Z. Liu, "Linking glove with word2vec," 2014.

[5] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2016.

[6] K. K. Bressem, L. C. Adams, C. Erxleben, B. Hamm, S. M. Niehues, and J. L. Vahldiek, "Comparing different deep learning architectures for classification of chest radiographs," Aug 2020.

[7] H. Gao, "The efficiency of densenet," Jun 2018.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[9] A. Gupta, "A comprehensive guide on deep learning optimizers," 2021.

[10] D. J. Huang, "Loss function — crossentropyloss vs bceloss in pytorch; softmax vs sigmoid; loss calculation," 2021.