report

github

https://github.com/yyy53449599/ai

Abstract

This experiment aims to conduct a three-class emotion classification task using a multimodal approach combining images and text. Fourteen models were trained using different strategies for fusion, followed by training a final fusion model for prediction. The data was first preprocessed by extracting it from folders, sorting it, and converting it into tensors. Exploratory Data Analysis (EDA) was performed, and the data was transformed into a format suitable for training. Subsequently, training was conducted with a total of five models and four different training strategies. These models were combined to train the fourteen models, each with its own unique characteristics. In order to conduct ablation studies, two single-modal models were included among the five models. After the final fusion, a validation accuracy of 67% was achieved.

Data Preprocessing

Firstly, we need to extract the images and text from the training files. We extract the file numbers from the train dataset and retrieve the corresponding files from the data folder. The images and text are read in sequential order. When reading the images, we record the original dimensions and resize them to a uniform size of 224x224 for further processing. The images are then converted into tensors. Next, we preprocess the text.

Since I will be using the BERT model for text training, I directly encode the text using the BERT base encoder for convenient storage. As I am conducting the training on the Colab platform, I store these two tensors on Google Drive. Label storage is also performed, where different labels are extracted. However, a mapping is created such that negative sentiment is represented as 0, neutral sentiment as 1, and positive sentiment as 2. The labels are also stored on the cloud drive.

The same preprocessing steps are applied to the test dataset.

EDA

I recorded the dimensions of the images and the lengths of the text, and the results are shown in the following figures:

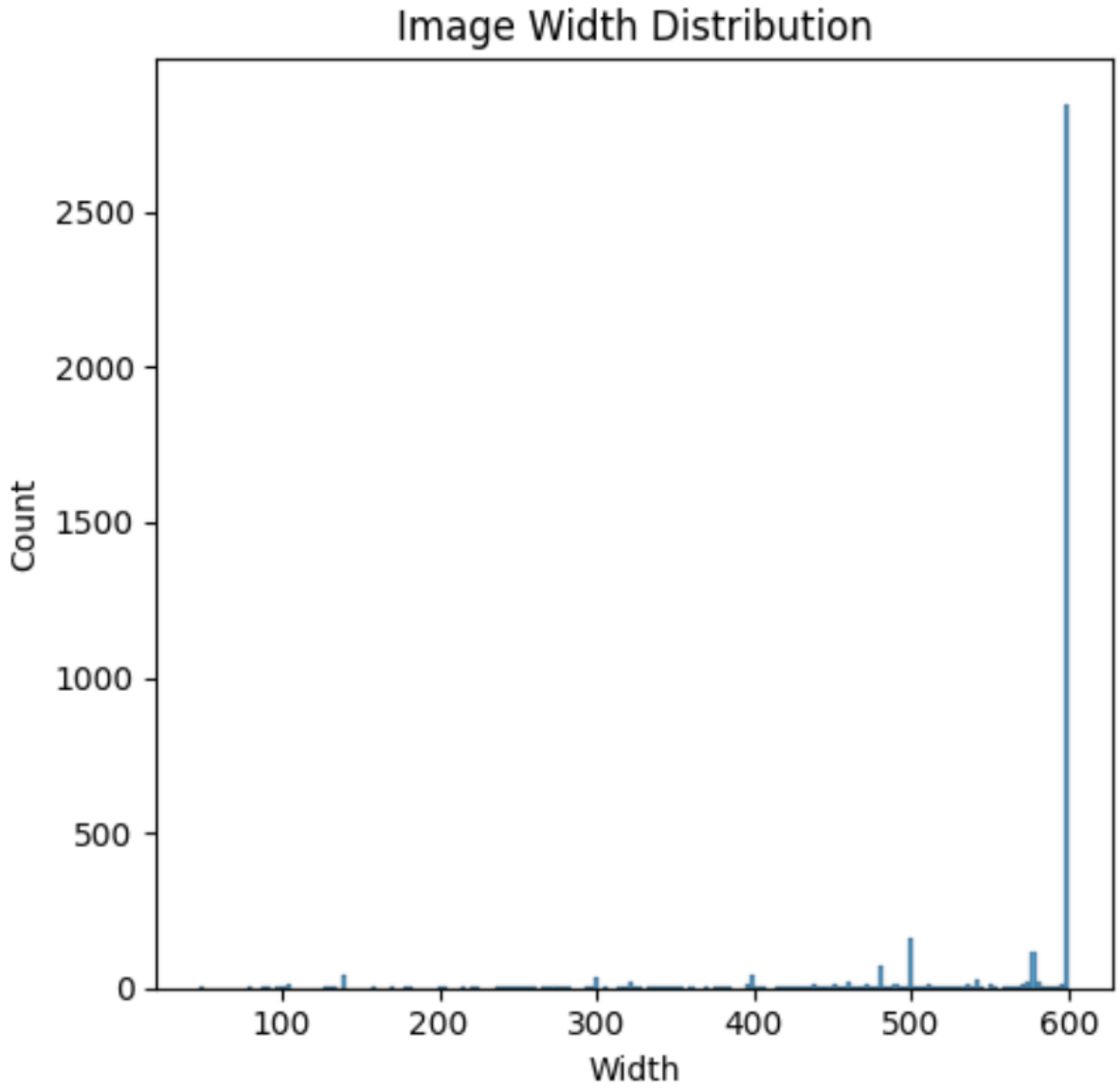


Image Width Distribution
Count
2500
2000
1500
1000
500
0
100
200
300
400
500
600
Width

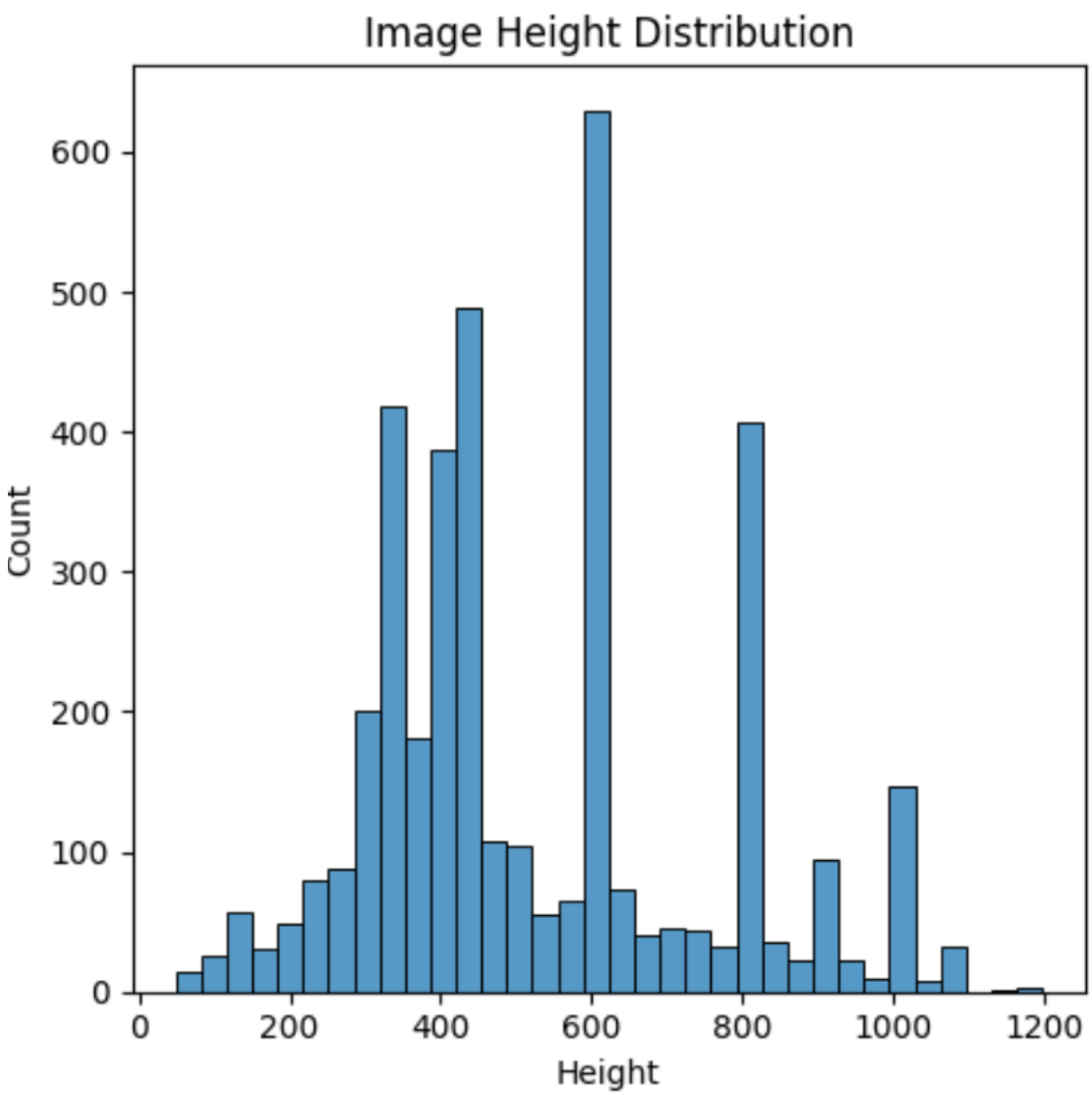
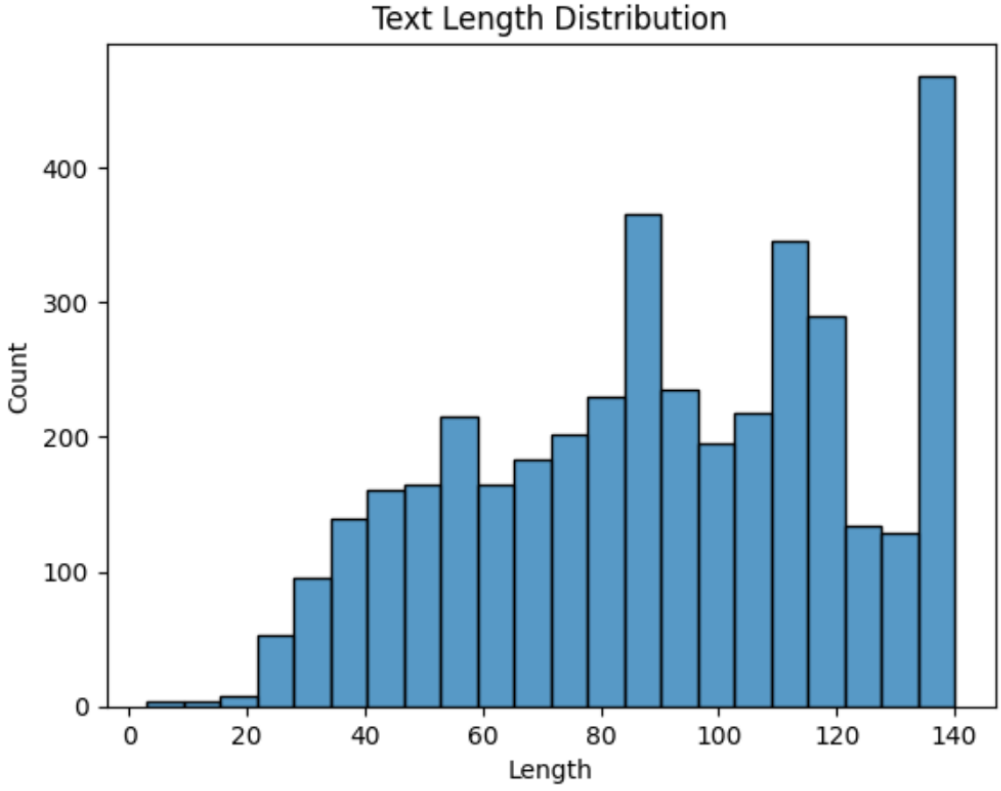


Image Height Distribution
Count
Height
0
100
200
300
400
500
600
0
200
400
600
800
1000
1200


Text Length Distribution
Count
Length
0
100
200
300
400
0
20
40
60
80
100
120
140

We can observe that the distribution is highly uneven, especially for the images. Additionally, most of the images have dimensions greater than 224x224. This implies that our training performance will likely be affected because compressing the images to a fixed size will result in the loss of high-frequency information. On the other hand, the distribution of text lengths appears to be more normal compared to the images. When using the BERT encoder, I chose automatic truncation and padding to ensure that the sequence lengths are consistent.

Model 1

For the multimodal task, the most straightforward approach is to use separate encoders for each modality and concatenate the encoded feature vectors before passing them through an MLP (Multi-Layer Perceptron). Firstly, let's discuss the choice of encoders.

To ensure training efficiency and avoid memory overflow, I selected a pre-trained ResNet50 model as the encoder for images. As for the text, I chose the pre-trained BERT base model. Initially, I considered using the popular transformer-based models, such as Swin Transformer, which have gained popularity in recent years. Since the image size for this task is larger than before, the limitations of CNN-based models, such as insufficient global observation capabilities, become apparent. Transformer-based models have outperformed CNN-based models in various competitions, especially Swin Transformer, which pays attention to both global and local details through its hierarchical design. However, these models require significant computational resources and memory and are sensitive to certain training hyperparameters, making training challenging. To save resources and time, I opted for the pre-trained ResNet50 model, as it is fast and already pre-trained. Although its performance may be slightly lower than Swin Transformer, the difference is not significant.

For the text modality, I used the BERT model. It is a relatively small model but performs well. After passing the two modalities through their respective encoders, I extracted the classification token (CLS) from the last layer of the BERT model for classification. This resulted in two feature vectors. I concatenated these vectors and passed them through an MLP. In the first layer of the network, I included a dropout layer to prevent overfitting.

Model 2

For the multimodal task, in addition to concatenating the feature vectors, we can also perform element-wise addition on the feature vectors. The encoders remain the same as mentioned above. First, we apply a linear transformation on the BERT output to change its dimension from 768 to 1000. Then, we add the two feature vectors together, with weights assigned to each modality. This approach allows the model to have a more intelligent fusion of modalities, and the weights can be learned during training. After the fusion, we pass the combined feature vector through an MLP.

Model 3

The essence of the first two models can be referred to as "early fusion" models. However, I believe that multimodal tasks can be categorized into three types: "early fusion," "mid fusion," and "late fusion." In "early fusion," as the name suggests, the modalities are fused before passing through the encoder. The original data from different modalities are combined and processed together before further training. One common method for early fusion is cross-attention, which I initially attempted but found that it did not yield satisfactory results, so I excluded it from consideration.

In "mid fusion," we first pass each modality through its respective encoder to obtain feature vectors, and then we fuse these feature vectors together. This is the approach I used for the first two models.

For the third model, I employed "late fusion." In this approach, each modality goes through its own MLP after the encoder, and then the resulting feature vectors are fused. I acknowledge that the results of this approach may not be as good as mid fusion because passing through separate MLPs will inevitably result in some information loss. However, in order to introduce an attention module, this was the best option. If I were to apply attention directly to the concatenated feature vectors with a length of 2000, the computational complexity would be too high, as the complexity of attention operations is O(n^2). Therefore, I chose to pass them through MLPs first and then perform fusion. This way, the length of the feature vectors is relatively smaller.

Model 4

To conduct an ablation study and assess the ability of individual modalities, I chose to perform image classification only using a pre-trained ResNet50 model.

Model 5

Similarly, this model focuses on the text modality alone, using the standalone BERT model.

a little problem

During the initial training of my model, I encountered a perplexing issue. As I trained the model for longer durations, its accuracy actually decreased. Initially, I partitioned the last 1000 samples of my dataset as the validation set. After training for a few epochs, the accuracy reached 60%. However, after 10 epochs, the accuracy dropped to 58%, and it was not due to overfitting. To investigate further, I conducted some tests and discovered that even when training for just one epoch, the accuracy remained at 60%. This raised concerns as the accuracy appeared consistently stable, without any fluctuations. Subsequently, I examined the model's predictions and found that it consistently predicted label 2, corresponding to positive sentiment. Initially, I suspected an error in my preprocessing step. However, upon inspecting all the labels, I discovered that out of the 4000 training samples, over 2000 were labeled as 2, while only four to five hundred were labeled as 1. Consequently, my model had learned to predict only label 2.

In order to validate my hypothesis and establish a more suitable evaluation approach, I contemplated controlling the distribution of labels in the validation set. Specifically, I considered extracting an equal number of label 2 and label 0 samples to match the quantity of label 1 samples. Although I haven't included the validation code here, it was an integral part of my investigation.

```
import torch
from torch.utils.data import TensorDataset, DataLoader

# 将输入数据、标签和结果合并为一个 TensorDataset
dataset = TensorDataset(val_image_tensor, val_input_ids, val_result_tensor)

# 对类别2进行抽样，使其数量与类别1相同
class2_indices = torch.where(val_result_tensor == 2)[0]
class1_indices = torch.where(val_result_tensor == 1)[0]
```

```
sampled_class2_indices = torch.randperm(len(class2_indices))
[:len(class1_indices)]

# 对类别0和类别1进行抽样，使其数量与类别1相同
class0_indices = torch.where(val_result_tensor == 0)[0]
sampled_class0_indices = torch.randperm(len(class0_indices))
[:len(class1_indices)]

# 合并抽样后的索引
sampled_indices = torch.cat([class0_indices[sampled_class0_indices],
                             class1_indices,
                             class2_indices[sampled_class2_indices]])

# 根据抽样后的索引创建新的抽样数据集
sampled_dataset = torch.utils.data.Subset(dataset, sampled_indices)

# 创建 DataLoader 加载抽样数据集
batch_size = 32
sampled_dataloader = DataLoader(sampled_dataset, batch_size=batch_size)

# 在验证集上进行评估
model1.eval()
total_correct = 0
total_samples = 0

with torch.no_grad():
    for batch_inputs, batch_labels, batch_results in tqdm(sampled_dataloader):
        batch_inputs = batch_inputs.to(device)
        batch_labels = batch_labels.to(device)
        batch_results = batch_results.to(device)
        outputs = model1(batch_inputs,batch_labels)
        _, predicted_labels = torch.max(outputs, 1)

        total_correct += (predicted_labels == batch_results).sum().item()
        total_samples += batch_labels.size(0)

accuracy = total_correct / total_samples
print(f"Validation Accuracy: {accuracy}")
```

I believe that this code, which truly reflects real-world conditions, is the key to conducting accurate validation. I proceeded to test it using the previously trained model and, as expected, the accuracy significantly decreased.

Training Details and Different Training Options

In order to create a model that combines multiple models, it is important to ensure that the trained models possess distinct characteristics. To achieve this, I employed two different loss functions during training: Support Vector Machine (SVM) and Softmax. Considering the significant label imbalance I observed earlier, I utilized weights to address this issue. First, I determined the exact quantities of each label, and then calculated the ratios between them, which served as the weights. However, it is uncertain whether the imbalance in the test set or real-world scenarios truly reflects a higher prevalence of label 2. In such cases, applying weights may not be

appropriate. Therefore, the alternative training approach involved excluding the use of weights. Combining these options with the three primary models resulted in a total of twelve distinct models.

Regarding the two single-modality models, after conducting tests, it became evident that their accuracies were noticeably lower than the other models. Consequently, to prevent them from dominating the fusion process, I trained them using only one training method: softmax classifier with the inclusion of weights.

As for training specifics, I trained each model for 20 epochs using the ADAM optimizer with a learning rate of 0.0001. The batch size was set to 100, and training was performed on half of an A100 GPU. In addition to training the MLP, I also fine-tuned the ResNet and BERT models.

The Ultimate Model

Next, we proceed with the fusion of models. There are multiple approaches to model fusion, and in this study, I have chosen to concatenate the results of various models and add a linear layer to obtain the final model. However, training this model requires a significantly longer time due to the involvement of fourteen individual models. Moreover, it imposes a substantial memory burden. Thus, I had to restrict the training process by using a batch size of 10 and training for only five epochs.

The fusion of models presents a powerful technique that leverages the strengths of each individual model, leading to improved performance through a combination of diverse features and representations. By merging the predictions of multiple models, we can capture a broader range of information and enhance the overall predictive capability.

The concatenation of model outputs followed by the addition of a linear layer allows for the integration and transformation of the learned representations from each model. This approach enables the final model to extract and combine valuable information from different perspectives, promoting a more comprehensive understanding of the underlying data.

Future Directions for Improvement

If greater computational resources are available, there are several avenues for further improvement. One potential direction is to modify the backbone models, which encompass the image and text encoders. In the current implementation, I have utilized ResNet-50 as the image encoder and BERT as the text encoder.

To enhance the model's performance, upgrading the backbone models to more advanced architectures could be explored. For instance, replacing ResNet-50 with deeper and more sophisticated convolutional neural networks (CNNs), such as ResNet-101 or Inception-ResNet, may capture more intricate image features and result in improved representation learning.

Similarly, for the text encoder, BERT has demonstrated exceptional performance in various natural language processing tasks. However, alternative state-of-the-art architectures, such as GPT-3 or Transformer-XL, might be considered for even better contextual understanding and semantic representation of text data.

Summary

The experiment aimed to perform a three-class emotion classification task using a multimodal approach combining images and text. Fourteen models were trained by employing different fusion strategies, and ultimately a fusion model was trained for prediction. The data was preprocessed by extracting and converting it into tensors from folders. Exploratory data analysis (EDA) was conducted, and the data was transformed into a suitable format for training. Five models and four different training strategies were employed for training. The models were combined and each model possessed unique features. To conduct ablation studies, two unimodal models were included among the five models. Following the final fusion, a validation accuracy of 67% was achieved.