Supercharged Picture Labels: Diving into Multimodal Learning

Hualin Zhou 530445609 Sicheng Liu 520096356 Rafiul Nakib 440590457

COMP5329 Deep Learning

What We're Looking At

We're taking a crack at a task that involves labeling lots of different things (it's called **multi-label classification**) targeting a set of **10,000** image-text pairs. This train data comes from a much bigger collection of 30,000 items, each having one of **18** unique labels. The exciting part? We're jumping into the world of **multi-modal learning.** It's a fresh approach in machine learning where we combine data from different sources. In this case, we're merging pictures and text to do our multi-modal classification. Now, to make this work, we're harnessing the power of deep learning architectures. Picture a sophisticated system designed to process visual data (that's our **EfficientNet**) and another one (called **ALBERT**) that's a whizz at handling text data. We're looking to conquer some of the tough bits involved in grabbing and bringing together key features from both images and text. It's got pretraining-finetuning stages, residual connections (these are like shortcut links that help in learning), bidirectional transformer layers (imagine them like two-way highways for data). Our big dream? To amp up our understanding of multi-modal learning.

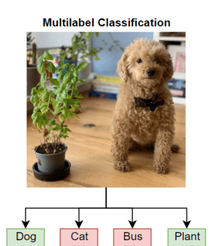

Figure 1. An example of multi-label classification. Credit to MathWorks

How We Did It

- **Getting the Data Ready:** First off, we had to clean up our data a bit. For the text data, we used a process called tokenization and padded out the sequences. For the image data, we used normalization and augmentation. And we didn't forget to standardize the image data after we re-scaled it.
- **Model Selection** For our multi-label classification task, we brought in two pre-trained models. First up, we have the Albert model. Think of it as a lean version of BERT - fewer parameters but still a solid performer. Its job was to understand and classify the text data. Then we have the EfficientNet-B3 model, a convolutional neural network (CNN) known for doing a great job and doing it efficiently. It's in charge of pulling out the high-level features from images.
- **Multimodal Model** When we say 'multimodal', we mean the model can juggle multiple types or modes of information. These models are a big deal in areas like natural language processing (NLP) and computer vision (CV) where you've got all sorts of data to process at the same time. Like, a model processing both image and text data can come up better result than models only works with one type of data. Two approaches to this model are:
  - ***Early Fusion*** Here, we take the features we pull from both text and image data and blend them early in the process. We then take the resulting feature map and use it for classification. Check it out in **Figure 2**.
  - ***Late Fusion*** In this approach, we treat image and text data as two separate entities. We make predictions for each, independently, and then stir these predictions together to make the final call. You can see in **Figure 3**.
- **Using Binary Cross Entropy Loss:** When it comes to calculating loss for our binary prediction tasks, we turned to Binary Cross Entropy (BCE) loss. This method is super useful when you're dealing with just two potential results (true or false). What makes it even better is that it's able to handle the multi-label nature of our work, calculating loss for each label separately.

$$\text{BinaryCrossEntropy}(y, \hat{y}) = -\left(y \log(\hat{y}) + (1-y) \log(1-\hat{y})\right) \quad (1)$$

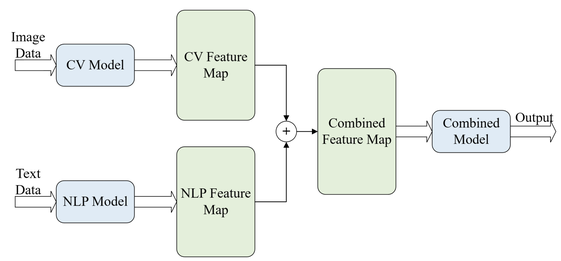

Figure 2. Models Architecture Early Fusion

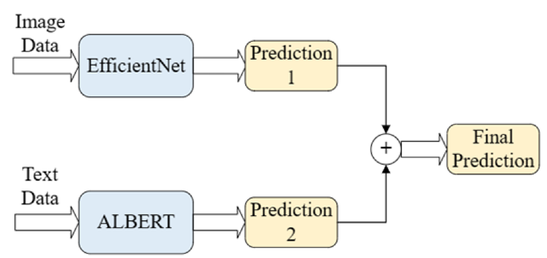

Figure 3. Models Architecture Late Fusion

What We Did

- **Exploring the models:** We put a bunch of models to the test (BERT, EfficientNet, ALBERT, and YOLOv5), but ended up going with ALBERT and EfficientNet. They had the performance we were looking for and a size that fit the bill. Then we looked at two different ways to fuse our multi-modal data: feature fusion and decision fusion. The best results? Combining the results did the trick!
- **Mixing Up the Results:** We cooked up a fresh new way to blend the results of our two models. Instead of relying on weighted models in late fusion, we took a more straightforward route – just directly mix the results from both models. So, for instance, if our image model (CV) pegs a sample as 1 and 3, and our text model (NLP) says it's 7, we combine those to get a final prediction of 1, 3, and 7. This straight-to-the-point approach is a bit different from weighted models, which can sometimes struggle to figure out when to favor which model.
- **Experimenting with Data Augmentation:** We wanted to see how different data augmentation methods would impact our chosen model. Turns out, our original method (random horizontal flip and random resize crop) was the winner!
- **Fine-tuning & Ablation Study:** With our model and data augmentation methods set, we went to town on hyper-parameter tuning. This involved running the model with different numbers of training epochs, various batch sizes, learning rates, and optimization algorithms. We also looked closely at the threshold for our multi-label classification task to get it just right.
- **Threshold Adjustment:** We don't set our threshold to a fixed number, because the model itself can not tell the output is too conservative or aggressive! Remember the final goal is F1-score, try adjusting the threshold to keep FP & FN as same as possible.

By going through these step-by-step experiments, we were able to whip up a super-efficient model that not only rocks in performance but also plays nice with the task's limitations. Pretty impressive, if you ask us!

What We Found

We got the final result by using ALBERT and EfficientNet with a results combination method. The hyper-parameters that achieved the best experimental results were set as in **Table 1.** The final experiment results are shown in **Table 2** and the confusion matrix is shown in **Figure 4.**

	EfficientNet	ALBERT
Learning Rate	0.0001	0.0001
Threshold	0.8	0.3 - 0.6
Epoch	8	13
Batch Size	25	100

Table 1. Best Hyper-parameter Setting

F1-score on training data	78
F1-score on Kaggle	87.47
Prediction time	10 min
Training time	20 min
Total Model size	86.1 MB

Table 2. Best Hyper-parameter Results

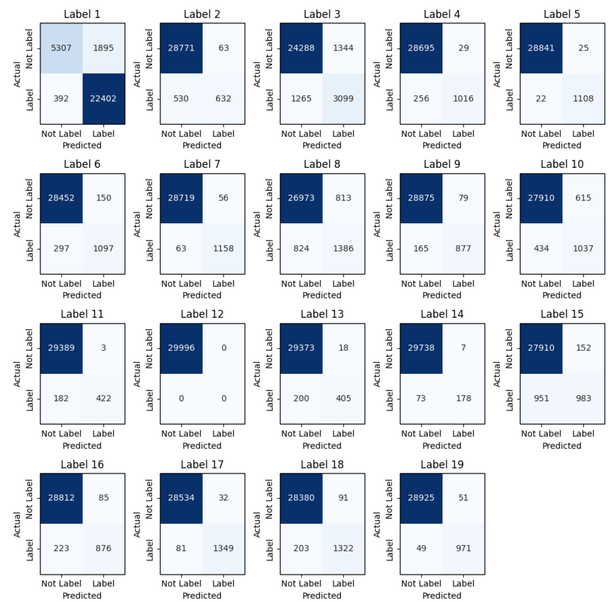

Figure 4. Models Architecture Late Fusion

Discussion and What's Next

We obtained an impressive F1-score of **87.47** on the test data using a model with a size of **86.1 MB**, satisfying the requirement of staying below 100 MB, with prediction time in only around a **10-minutes! (1.67 sample/s)** Talk about efficiency!

But we're not done yet! We've got big plans for enhancing our model's performance in the future. Here's what's on our radar:

- **We NEED Validation set:** We didn't use any validation set to check the loss, and the test score is like a random toss! We need to trade off our train data for a validation set.
- **Oversampling:** For some rarely occured labels, repeat training until performance saturated.
- **LSTM: The Mediator:** There is a lack of investigating the relationship between the labels, a LSTM is likely to learn how they appear in group! Hence boost the performance.