VRDL Homework Final Project

309553047 李育人 0716314 陳鎧勳 0716072 吳季嘉

GitHub link of my code

https://github.com/072jiajia/VRDL_FinalProject

Reference

https://github.com/PeiqinZhuang/API-Net
https://arxiv.org/pdf/2002.10191.pdf

Introduction

We join the competition “Bengali.AI Handwritten Grapheme Classification”. This challenge hopes to improve Bengali recognition.

There are some issues in this challenge, in training set, it only contains 1295 categories of words, but there are over 10000 categories of words in Bengali. It might have some issues like “if it contains grapheme root A in this word, it must contain vowel diacritic B also” in the training dataset.

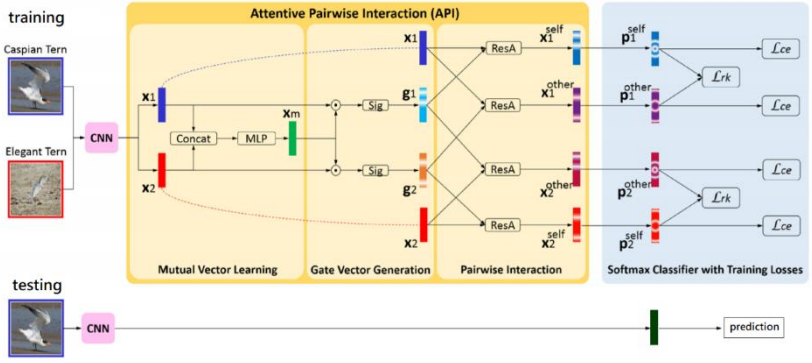
In this work, we used API-Net (Attentive Pairwise Interaction Network) to do the classification task. API-Net is a module which can attentively distinguish two fine-grained images via pairwise interaction. And we tried some method to deal with the problems above.

Related Work

Fine-Grained Classification

When learning to recognize some similar objects, a simple CNN model cannot learn contrastive clues easily, but humans can focus on contrastive clues between image pairs.

Proposed approach



We use API-Net with DenseNet121 as our model's backbone.

API-Net

In training phase, it focuses on contrastive clues between image pairs.

First, it extracts the features of input images, and finds the most similar image of each image. It uses Euclidean distance in the feature space to determine the similarity of two images.

Gate Vector. Then, as we have the features of itself and its most similar image's features, it uses the features to compute a gate vector, which can be seen as the discriminative attention of a view.

$$\mathbf{x}_m = f_m([\mathbf{x}_1, \mathbf{x}_2])$$

$$\mathbf{g}_i = sigmoid(\mathbf{x}_m \odot \mathbf{x}_i), \quad i \in \{1, 2\}$$

Pairwise Interaction. Next, use the Gate Vector to highlight the discriminative clues via residual attention.

$$\mathbf{x}_1^{self} = \mathbf{x}_1 + \mathbf{x}_1 \odot \mathbf{g}_1$$
$$\mathbf{x}_2^{self} = \mathbf{x}_2 + \mathbf{x}_2 \odot \mathbf{g}_2$$
$$\mathbf{x}_1^{other} = \mathbf{x}_1 + \mathbf{x}_1 \odot \mathbf{g}_2$$
$$\mathbf{x}_2^{other} = \mathbf{x}_2 + \mathbf{x}_2 \odot \mathbf{g}_1$$

The last layer of this module is a fully-connected layer which can do classification.

Pair Construction. The above process will be done twice for each input image, one of it is pairing with the most similar image in the same class, and the other one is pairing with the most similar image in the other classes.

Loss Function. The loss function is composed of two parts, cross entropy loss for each image's prediction, and Score Ranking Regularization for each pair of predictions. The Score Ranking Regularization is thought that the feature highlighted by the object in the same class should lead to higher score than the one highlighted by the object in the other classes.

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda \mathcal{L}_{rk}$$

$$\mathcal{L}_{ce} = -\sum_{i \in \{1,2\}} \sum_{j \in \{self, other\}} \mathbf{y}_i^\top \log(\mathbf{p}_i^j)$$

$$\mathcal{L}_{rk} = \sum_{i \in \{1,2\}} \max(0, \mathbf{p}_i^{other}(c_i) - \mathbf{p}_i^{self}(c_i) + \epsilon)$$

Training Phase

We trained grapheme root, vowel diacritic and consonant diacritic classifier separately, the loss function is the same as the loss API-Net uses.

Data Preprocessing

Because All of the images are black on white, it is easy to compute the center of the word, so we align the center of the word to the center of the image. Then transform the black parts(words) to 1s and other parts to -1s and add some noise to the image.

In training set, it only contains 1295 categories of words, but there are over 10000 categories of words in Bengali. It might have some issues like “if it contains grapheme root A in this word, it must contain vowel diacritic B also” in the training set.

To deal with this issue, when training vowel diacritic and consonant diacritic classifier, we only use part of training set. More details in training detail.

Validation Phase and Testing Phase.

Align the word to the center of the image and make prediction.

Training Detail

The total number of training epochs is 50. When training on {grapheme root, vowel diacritic, consonant diacritic} each batch contains {80, 11, 7} categories, in each category, there are {4, 32, 50} images. We use AdamW with Amsgrad and weight decay of 1e-4, initial learning rate of 1e-4 and adopt cosine annealing to adjust it. In data Preprocessing, cut off a block with size (1, 1) to (h/4, w/4) with probability 50% and add noise to part of image with size (1, 1) to (h/2, w/2) with probability 50%. With 50% adjusting the contrast of image by multiplying pixels by a scalar in [0.5, 1.5] and finally clip the values of pixels to [-1, 1].

Experimental results

This Challenge consists of two dataset, public dataset and private dataset. It is said that the two datasets consist of some words not appeared in the training dataset.

The performance of model is determined by weighted accuracy. {grapheme root, vowel diacritic, consonant diacritic} are weighted [2, 1, 1].

In this challenge, the final result is determined by Private Score.

	Private Score
deoxy (1st place)	0.9762
Ours	0.9270

Though our work didn’t pass the baseline, we found out that when training vowel diacritic and consonant diacritic classifier, if we drop out the data in which the same grapheme root always leads to the same vowel or consonant in the training set, it leads much better performance in private dataset than the one we don’t drop out.

	Private Score	Public Score
Drop Out Part of Data	0.9270	0.9533
Do Nothing	0.9095	0.9642

But the performance in public dataset get worse, we think it’s because the public dataset has almost the same data to our training data, so we instead didn’t train well to the words we dropped out.

Conclusion

In this work, we tried some method to deal with unseen objects problem and we adopt a module named API-Net to do fine-grained classification on the training dataset and make a prediction on the testing data set. we also adjusted the hyperparameters and tried several ways to improve the performance of our models.

Contribution

Tasks	**Contributors(%)**
Literature survey	309553047(34%), 0716314(33%), 0716072(33%)
Approach design	309553047(50%), 0716072(50%)
Approach implementation	309553047(50%), 0716072(50%)
Report writing	0716314(50%), 0716072(50%)
Slide mask and oral presentation	309553047(34%), 0716314(33%), 0716072(33%)