Assignment 1 Report: SVD-CNNIQA

Wu Shuwen(521030910087)

***Abstract*—This report is about my IQA model design based on CNNIQA [1] and Singular Value Decomposition(SVD), inspired by SVDiff [2].**

I. Introduction

In my report, I'd like to introduce my idea of my model SVD-CNNIQA, which is trained starting from a pretrained model. The pretrained model is provided by the author of IQA-Pytorch, but is trained on another dataset. However, I infer that there's something common in doing IQA on different datasets, and the pretrained weight can be made better use of than directly train the vanilla CNNIQA. So I developed my SVD-CNNIQA to do this thing.

II. Approach

A. Singular Value Decomposition

I performed Singular Value Decomposition (SVD) on the weight matrices of a pre-trained CNNIQA model [3]. The weight matrix is denoted as W, and its SVD is $W = U\Sigma V^T$, where $\Sigma = diag(\boldsymbol{\sigma})$ and $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, ...]$. The SVD is a one-time computation and can be cached. For each decomposed layer W, only the $\Sigma = diag(\boldsymbol{\sigma})$ of the three matrices is marked as trainable. U and V^T are frozen in the training process.

B. CNNIQA

CNNIQA is a classic DL-based IQA model. Put forward in 2014, it is still widely in use today.

III. Implementation

I implemented my IQA model utilizing IQA-PyTorch [4] program. I registered AGIQA-3K [5] dataset and my IQA model in my own fork so that I can perform experiments on my IQA model and compare it with vanilla CNNIQA easily and fairly.

A. The Model Arch

I implemented the networks of my model in the python files './pyiqa/archs/myiqa_arch.py' and './pyiqa/archs/myiqa2_arch.py'.

1) SVDLinear Layer: SVDLinear layer is set to be a sub-class of the torch layer nn.Linear. The layer is first set to have the same feature as a vanilla Linear layer, but is decomposed by SVD the first time the function forward is called. SVD is implemented in the function perform_svd, utilizing 'torch.linalg.svd'. The whole layer is frozen except for a parameter 'delta', which is added to Σ(represented as S in my code).

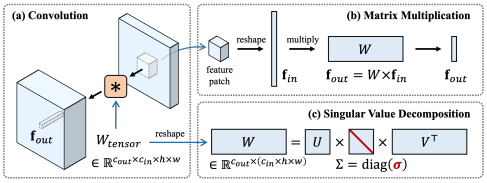

Fig. 1. Performing singular value decomposition (SVD) on weight matrices. In an intermediate layer of the model, (a) the convolutional weights Wtensor (b) serve as an associative memory. (c) SVD is performed on the reshaped 2-D matrix W

2) SVDConv2d Layer: This layer is almost the same as SVDLinear layer except that it is a sub-class of the torch layer nn.Conv2d.

3) Arch in Detail: For 'myiqa', I performed SVD on the three fully-connected layers. For 'myiqa2', however, I performed SVD on the Conv layer. For each model, I first loaded the pretrained CNNIQA weights to a vanilla CNNIQA. Then I created SVD layers and copied the weights to them.

B. Model Registration

I register my arch utilizing the api 'ARCH_REGISTRY.register()' provided by the program. Also, I added my model to the dictionary in the python file './pyiqa/default_model_configs.py', so that it can easily be set up using the training framework provided by the program.

C. Dataset Registration

I created meta-info file 'meta_info_AGIQA-3K.csv' and split-file 'AGIQA-3K.pkl' for AGIQA-3K dataset, so that I can register the dataset with an option file and access the dataset easily.

IV. Experiment

I conducted my experiment on my modified CNNIQA models and vnilla CNNIQA model using the same training settings. For these three models, I loaded the pretrained CNNIQA model provided by the author of the program. I used MSELoss in my training:

$$l(x, y) = L = \{l_1, ..., l_N\}, l_n = (x_n - yn)^2$$

For the optimizer, I utilized the AdamW [6] optimizer.

V. Results and Analysis

As is shown in the table, performing SVD on fc-layers of vanilla CNNIQA significantly improved the performance. On the other hand, performing SVD on the conv layer turns out to hold back the training process. I also tested HyperIQA on the same dataset, but it turned out that my improved CNNIQA is

still no match with it. However, the result does show that SVD does have potential in migrating similar pretrained models, as it did made good use of a pretrained model on another dataset in my experiment.

TABLE I
TRAINING RESULTS

Model	**Epochs**	**SRCC**	**PLCC**	**KRCC**
CNNIQA(Vanilla)	50	0.5429	0.6038	0.3753
CNNIQA(Vanilla)	100	0.5696	0.6305	0.3968
SVD on fc-layer(myiqa)	50	0.6173	0.7912	0.4371
SVD on fc-layer(myiqa)	100	0.6254	0.7959	0.4434
SVD on conv-layer(myiqa2)	50	0.5677	0.6870	0.3908
SVD on conv-layer(myiqa2)	100	0.5677	0.6870	0.3908
HyperIQA		0.8137	0.8801	0.6261

REFERENCES

[1] Convolutional Neural Networks for No-Reference Image Quality Assessment CVPR2014
[2] SVDiff: Compact Parameter Space for Diffusion Fine-Tuning ICCV2023
[3] https://github.com/chaofengc/IQA-PyTorch/releases/download/v0.1-weights/CNNIQA_koniq10k-e6f14c91.pth
[4] https://github.com/chaofengc/IQA-PyTorch
[5] AGIQA-3K: An Open Database for AI-Generated Image Quality Assessment.
[6] Decoupled Weight Decay Regularization