Blog group 78

Joost van der Weerd `j.vanderweerd@student.tudelft.nl` (4704320)

Darwin Liu, `d.liu-13@student.tudelft.nl` (5171695)

Eric Kemmeren, `e.a.kemmeren@student.tudelft.nl` (5074223)

Jurriaan Buitenweg, `j.r.buitenweg@student.tudelft.nl` (4965221)

1 Introduction

In this blog, we are reproducing the paper "Towards Artistic Image Aesthetics Assessment: a Large-scale Dataset and a New Method" [7]. This paper proposed a combinatorial network on top of a Resnet50 backbone to achieve a 76.8 accuracy on image aesthetics assessment on 1-10 score basis. The training set was a dataset, created by the researchers themselves, namely "Boldbrush Artistic Image Dataset (BAID)", which consists of 60,337 artistic images covering various art forms, with more than 360,000 votes from online users having 25GB+ in size [7]. In this paper, we downloaded this complete dataset and tried the following:

1. Reproduce the exact results described in the paper, namely 76.8% accuracy with a pearson coefficient of 0.476 and a spearmen coefficient of 0.473 (Jurriaan Buitenweg).

2. New algorithm variant: Changing the loss function and normalisation layers in the SAAN model. (Darwin Liu)

3. Hyper parameter optimization: Applying successive halving for hyper parameter optimization at a low cost (Joost van der Weerd)

4. Performing an ablation study to research the effect of the combinatorial network proposed. (Eric Kemmeren)

2 Reproduction

The researchers claimed that by adding 3 layers. A style specific layer, generic feature layer and a layer to fuse these together, could provide high accuracy for image aesthetics assessment (figure 1).

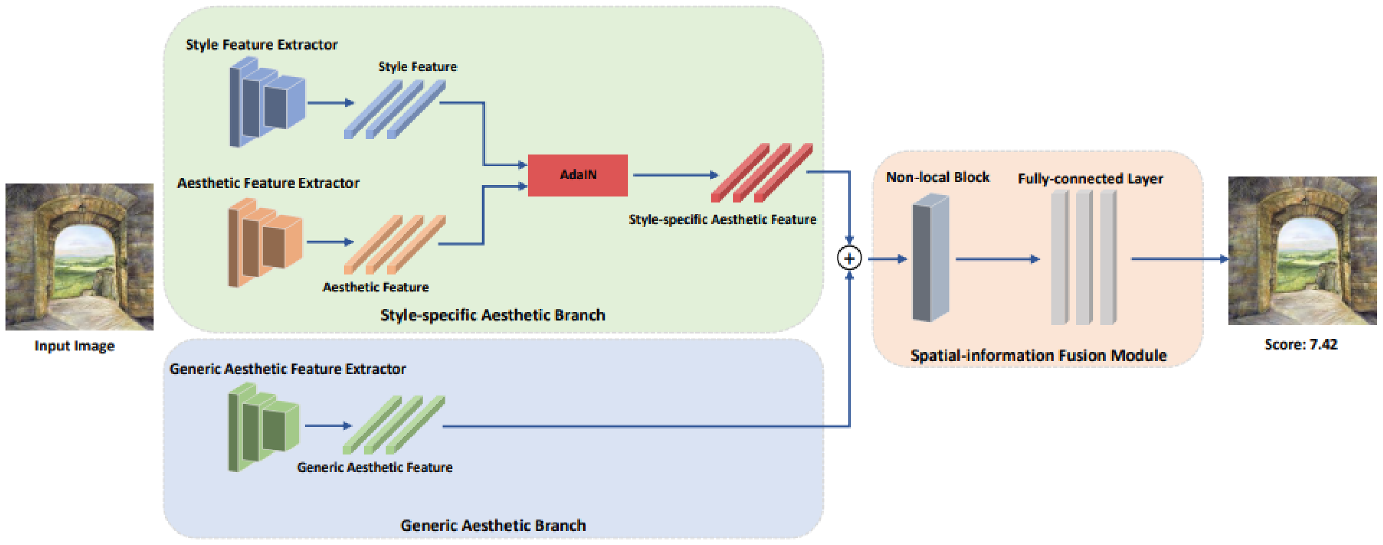
Style Feature Extractor
Style Feature
Aesthetic Feature Extractor
AdaIN
Style-specific Aesthetic Feature
Aesthetic Feature
Style-specific Aesthetic Branch
Input Image
Generic Aesthetic Feature Extractor
Generic Aesthetic Feature
Generic Aesthetic Branch
Non-local Block
Fully-connected Layer
Spatial-information Fusion Module
Score: 7.42

Figure 1: The SAAN architecture proposed in [7]

The hardest part about reproduction was staying within the kaggle quota, because training the whole BAID dataset (25+GB) takes more than 30 hours, which exceeds the kaggle quota. We implemented a small change in the code which made it possible to split up the training over multiple kaggle runs, obtaining the fully trained model as stated in the paper [7]. To our surprise, when running the trained model on the test set, the accuracy was the same as stated in the paper, but the correlation coefficients differed. Training and validation loss also seemed to stagnate after 20 epochs (figure 2), this also resulted in the accuracy correlation measurements to be almost identical when testing the model which was trained for 100 epochs and the model which was trained for 20.

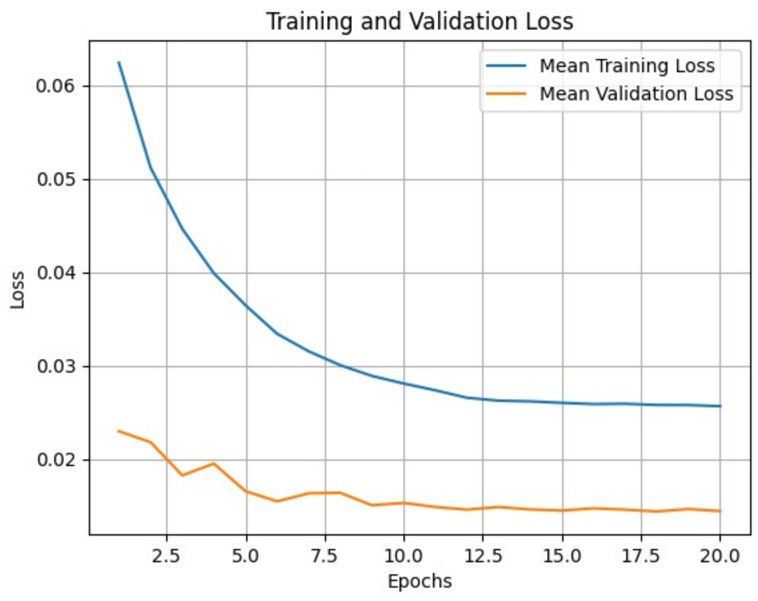

Figure 2: Averaged training and validation loss over epochs

Model	**Accuracy**	**Pearson Coefficient**	**Spearman coeffeicient**
Pretrained by paper author	76.8%	0.476	0.473
Manually Trained (until epoch 100)	76.8%	0.416	0.413
Manually Trained (until epoch 20)	76.7%	0.414	0.409
Only Resnet (Random weights)	40.6%	0.11	0.18

Spearman and Pearson both measure correlation. Because these values are both lower compared to the given model by the author, this means there is less correlation between predictions and true labels compared to the original dataset.

Using random weights for the SAAN architecture and essentialy only using the ResNet Backbone, does show a big decrease in accuracy, so this combination of style and feature extraction does increase the accuracy for image aesthetic assessment.

This inability to exactly reproduce the results in [7] is one of the concerning trends described in [1], namely ”Failure to identify the sources of empirical gains, e.g. emphasizing unnecessary modifications to neural architectures” and a reproducibility issue stated in [2], which stated that only providing the code is not enough to exactly reproduce results, like in this project. Its hard to trace back where this small change in correlation coefficients exactly comes from but we think it could be due to different weights at initialization time.

3 New Algorithm Variant

The paper about artistic image aesthetics assessment (AIAA) [7] bases its success on two major elements:

1. Their newly proposed dataset which is a large-scale AIAA dataset: the Boldbrush Artistic Image Dataset (BAID)
2. Their own model (SAAN) which consists of separate components each designed to perform a certain subtask in order to assess an artistic image on its aesthetics.

Assuming that the components of SAAN really do as they say in the paper, we thought of a few small changes to the model to increase its performance.

In SAAN are two normalization layers. Normalization layers have some useful benefits such as fast convergence and more stability during training of a model [3]. The standard normalization method is batch normalization (BN) and is also used in this model. We propose to use a group normalization (GN) layer in this model. GN is a method to normalize the outputs of a layer. The standard way to normalize is batch normalization, however the downsize of batch normalization is that it requires large a batch size to obtain a good performance. For larger inputs, such as images, this might be a bottleneck to the model, as that would require a lot of memory and computing power. GN however, is independent on batch size and still achieves similar performance. Furthermore, GN is also somewhat mimics nature, such as the grouping of certain colors in your eyes [6]. These benefits make GN an interesting substitute to batch normalization. By using GN after combining all features

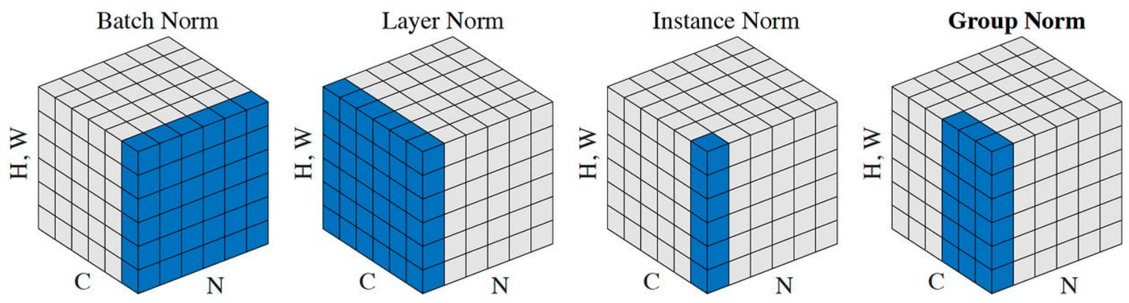

Figure 3: A visualization of four different normalization methods. The standard way of normalizing output features is Batch Norm, which normalizes over all the input samples. Group Norm however normalizes over a group of output features [6].

extracting by the style-specific aesthetic branch and the generic aesthetic branch, it is similar to what happens when a person assesses an artistic image. You can assess a single image a time, but the combination of features is what makes an artistic image aesthetic. A visual representation of GN and BN and two other methods are shown in figure 3.

GN does add an extra hyper parameter to the model which is the amount of groups to normalize over. For our model we choose 16 groups. The number of output features has to be divisible by the number of groups and furthermore it seemed like a good number of features you would rate a painting with.

As mentioned before the model uses normalization two times. We decided to try to only change the first normalization layer to GN and also both. The disadvantage of only changing one layer to GN is that the model still remains batch size dependent. The results are shown in table 1.

The model are trained until epoch 44, because that is the longest amount of time a notebook can run in a single run on kaggle and at that point the loss function has converged already. In the original code the learning rate also decays every 10 epochs and that stops after epoch 40. For GN only, we did not have enough hours left on kaggle, but it was already sufficient to train it until epoch 34 anyway. As can be seen, GN does provide better results

Model	**Accuracy**	**Pearson Coefficient**	**Spearman Coefficient**
GN and BN trained until epoch 44, batch size 64	76.6%	0.415	0.414
GN and BN trained until epoch 44, batch size 16	77.6%	0.471	0.460
GN only trained until epoch 34, batch size 16	77.7%	0.486	0.493

Table 1: Results of models trained using GN layeres

while also decreasing the batch size. It seems even the combination GN and BN achieves better results if the batch size is smaller.

The model also uses a regression type loss function, namely the the mean squared error. We first study what happens when we use a different type of regression loss function, the Huber loss function. This is a combination of a mean absolute error and mean squared error loss. One thing to note is that assessment of the aesthetics is very subjective, which means that different people might think really differently about certain images. This means that there are probably more outliers in the data. Mean squared error is sensitive to large outliers and mean absolute error handles it better. For small errors however, mean squared error loss performs better. The Huber loss function combines these two aspects and might improve the model. Mathematically it is given by:

$$f(y_{pred}, y_{true}) = \begin{cases} \frac{1}{2}(y_{pred} - y_{true})^2, & \text{if } y_{pred} - y_{true} \leq \delta \\ \delta(|y_{pred} - y_{true}| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{1}$$

We tried the Huber loss with different normalization methods. In the table 2, the results are summarized.

The results show that only changing the MSE loss to the Huber loss does not change the model performance significantly, but it does have a slightly higher accuracy. However in

Model	**Accuracy**	**Pearson Coefficient**	**Spearman Coefficient**
GN and BN and Huber Loss trained until epoch 44, batch size 64	77.5%	0.445	0.448
All GN and Huber Loss trained until epoch 44, batch size 16	77.4%	0.468	0.481
All BN and Huber Loss trained until epoch 44, batch size 16	76.9%	0.420	0.425

Table 2: Results of models trained with a Huber loss function

combination with GN, it performs better than the original model. It is however not better than the models with GN and a MSE Loss function as in table 1. It seems GN is the largest factor in improving the model.

4 Hyper parameter optimization

The training of a deep learning model can sometimes be conceived as more of an art form than a science. And since the training of models is very resource and time intensive it can be hard to find a suitable set of hyper parameters to get the required/best performance of your model. Therefore a lot of algorithms have been proposed to speed up or reduce the cost of the training I will be focusing on the technique called 'successive halving'.

4.1 Successive halving

Successive halving focuses on keeping the cost of hyper parameter optimization low. The algorithm starts with model selection this can be done in numerous ways i.e. grid or random search. Then in the first iteration all models are trained a set amount. Then the best $1/\eta$ are selected and the number of training cycles is increase by η times. For each iteration k there are then an equal amount of training cycles to be done since the number of models decreases in the same rate as the amount of training cycles increases Equation (2). A graphical view of this relation is shown in Figure 5.

$$\begin{aligned} \#models_k &= \frac{\#model}{\eta * k} \\ \#epochs_k &= \eta * k \end{aligned} \tag{2}$$

In this case when there is a limiting factor on the number of training cycles that are possible still a large hyper parameter space can be searched. Also by tuning η a choice can be made about how big the hyper parameter space should be and how aggressive it should prune the candidate models. The more aggressive (bigger η) the pruning the more training cycles the best model gets and the bigger the hyper parameters space can be, but the more likely you are to discard the wrong model. A less aggressive pruning (lower η) will more likely end up with the 'best' model but will have less training cycles to train it and have a smaller hyper parameter space to start from.

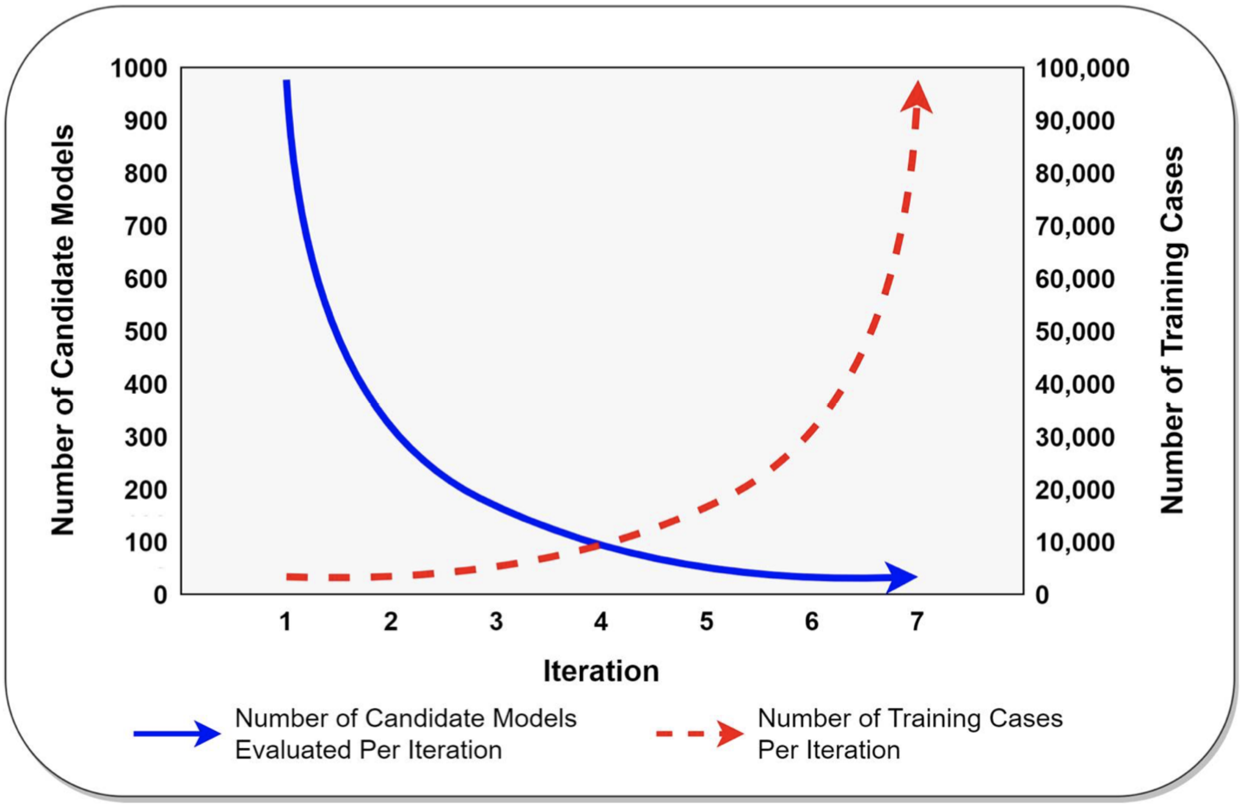

Figure 4: Successive halving (Taken from Soper [4])

4.2 Implementation

I will use grid search along the two hyper parameters batch size and learning rate. The models and their parameters are shown in table 3. We will start with 16 models with one epoch. After that the best 8 will be kept and will be trained again with 2 epochs. This will continue till only one model remains.

	1e-3	1e-4	1e-5	1e-6
64	Model 0	Model 1	Model 2	Model 3
32	Model 4	Model 5	Model 6	Model 7
16	Model 8	Model 9	Model 10	Model 11
8	Model 12	Model 13	Model 14	Model 15

Table 3: Model parameter table learning rate vs. batch size

Since our training time is very limited I have chosen for 5 iterations where the population is halved and the next iteration the training epochs is doubled. The best model will than reach 31 epochs. As we have seen in the reproduction the model training flattens out after 20 epochs so this should still result in a competitive model. The total is 80 epochs which is really close to the maximum training time we have.

4.3 Results

The results have been plotted in Figure (5). After the first training iteration we lose models (1,2,3,7,10,11,14,15) we can immediately see that the learning rates of 1e5 and 1e6 where to small since we almost lose all those models. After the second iteration we lose models (0, 6, 8, 12) After the third iteration we lose models (9, 13) After the fourth iteration we lose model 5 The final model is model 4 with a learning rate of 1e-3 and a batch size of 32. Model 4 ends with an overall accuracy of 76.34375%. Pearson coefficient of 0.375 and a Spearman coefficient of 0.414. This is already very close to the results from the paper with 69 epochs of less training. Another important thing to note is that the models that reached the end all have different hyper parameters than the model that is trained by the author (learning rate 1e5 and batch size 64). This indicates that maybe more performance can be achieved by training the model with the hyper parameters of model 4 (learning rate = 1e4 and batch size = 32).

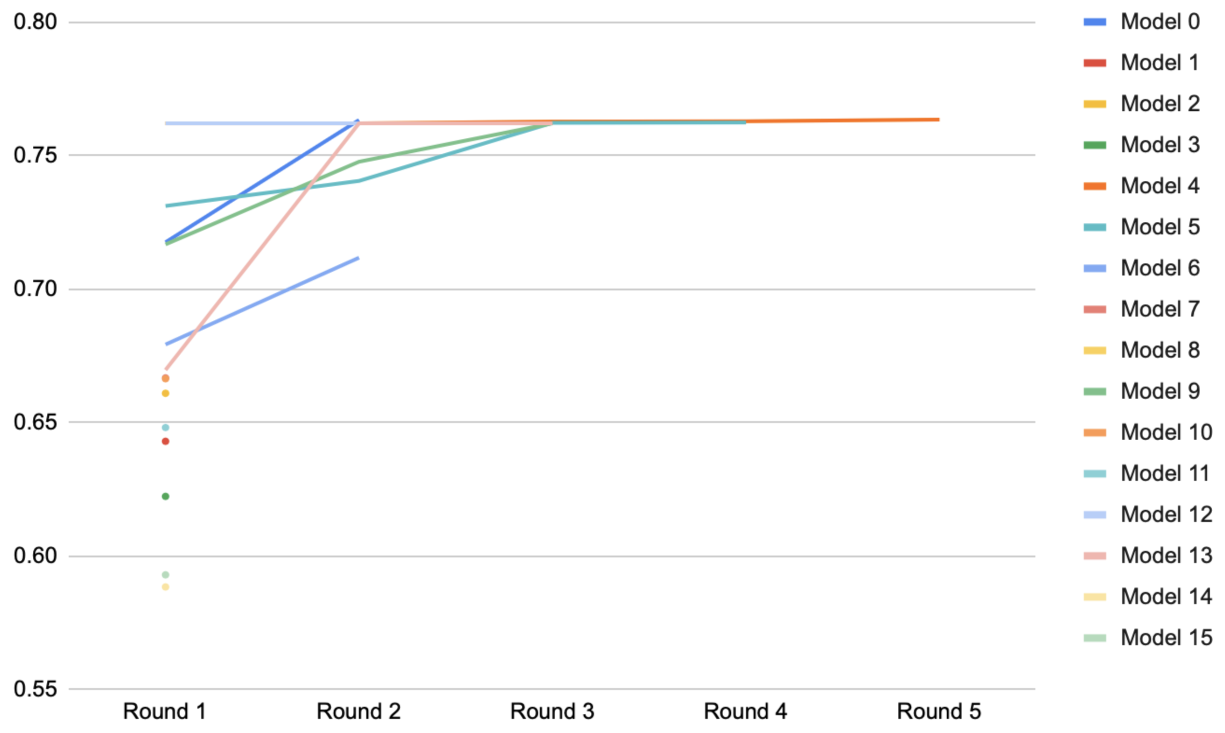

Figure 5: Evolution of models over iterations

5 Ablation

As can be seen in figure 6, the architecture consists of three parts, the style-specific aesthetic branch, the generic aesthetic branch and the spatial-information fusion module. The last mentioned module is based on the paper ”Non-local neural networks.” [5], this is why we decided to evaluate the influence of the first two branches by performing an ablation study. The paper also performed an ablation study with these two parts, but it is important to verify these results.

To test the influence of the style-specific branch, the model.py file was updated to remove this branch, and update the number of features for following layers in the network. The

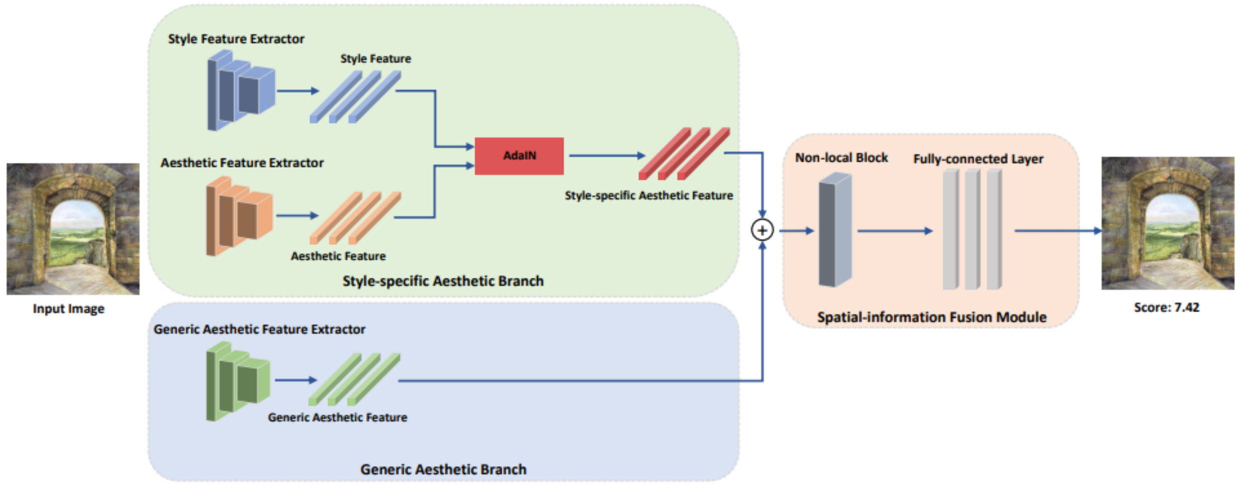
Style Feature Extractor
Style Feature
Aesthetic Feature Extractor
AdaIN
Style-specific Aesthetic Feature
Aesthetic Feature
Style-specific Aesthetic Branch
Input Image
Generic Aesthetic Feature Extractor
Generic Aesthetic Feature
Generic Aesthetic Branch
Non-local Block
Fully-connected Layer
Spatial-information Fusion Module
Score: 7.42

Figure 6: Overview of the SAAN architecture, from the paper

specifics can be found on the GitHub page. This process was repeated for the influence of the generic aesthetic branch. The result of training and evaluating the network can be found in table 4.

Table 4: Ablation results, Above SAAN are ablation results as run by the authors of the original paper, in the row containing SAAN are the results from the architecture proposed by the paper, followed by our own ablation study in the last two rows.

Method	**SRCC**	**PCC**	**Accuracy**
paper w/o style-specific branch	0.425	0.411	73.22%
paper w/o generic aesthetic branch	0.439	0.426	74.60%
SAAN	**0.473**	**0.467**	**76.80%**
Retested w/o style-specific branch	0.193	0.205	75.66%
Retested w/o generic aesthetic branch	0.439	0.453	77.20%

What is remarkable about these results, is that the accuracy improved when removing the generic aesthetic branch. This improvement of 0.40% is quite significant, especially taken into account that the result of our ablation study is not equal to the ablation study in the paper. The accuracy without the style-specific branch improved by 2.44%, but the SRCC and PCC decreased by 0.232 and 0.206 respectively, meaning that it found less correlation with the actual output. For the ablation study of the generic aesthetic branch, where the accuracy increased by 2.80% on the original ablation study, the SRCC stayed equal, and the PCC improved slightly with 0.027. Even though removing this branch increased the accuracy by 0.40% as mentioned, the SRCC and PCC did still decrease by 0.034 and 0.014.

6 Conclusion

We reproduced the SAAN architecture described in [7] and trained/tested on the BAID dataset provided. This resulted in the same accuracy as in [7] but different correlation coefficients. This could be due to the random weight initialized at the beginning. We also added a new variant to its algorithm, namely group normalization. This resulted in an accuracy increase of 0.1%. With successive halving, we performed hyper parameter optimization whie keeping the cost low. The ablation study showed that accuracy actually increased when removing the generic aesthetic branch from the SAAN [7] architecture.

References

[1] Zachary C. Lipton and Jacob Steinhardt. Troubling Trends in Machine Learning Scholarship. Technical report, July 2018. arXiv:1807.03341 [cs, stat] type: article.

[2] Edward Raff. A Step Toward Quantifying Independently Reproducible Machine Learning Research. Technical report, September 2019. arXiv:1909.06674 [cs, stat] type: article.

[3] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[4] Daniel S. Soper. Hyperparameter optimization using successive halving with greedy cross validation. *Algorithms*, 16:17, 2022.

[5] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[6] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[7] Ran Yi, Haoyuan Tian, Zhihao Gu, Yu-Kun Lai, and Paul L Rosin. Towards artistic image aesthetics assessment: a large-scale dataset and a new method. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22388–22397, 2023.