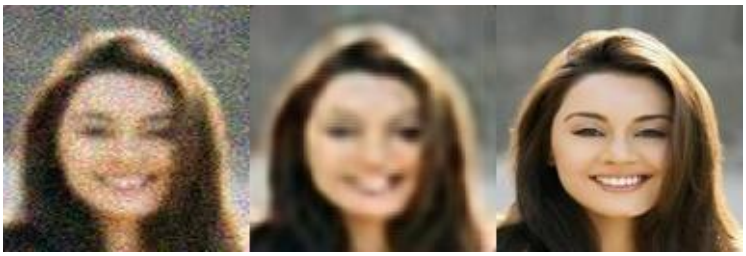
Multi-Prior Learning via Neural Architecture Search for Blind Face Restoration

Group 11

1. Task Definition, Evaluation Protocol, and Data

Task Definition:Blind face restoration refers to generating high-quality face images from low-quality face images. Compared with general image restoration of uncertain items, blind face restoration can use prior facial features, such as parsing map, heatmap, facial dictionary. As shown in Figure 1, the input of a low-quality image will generate an enhanced image. Compared with the input, the output image will be closer to the original image and have higher definition[1].

Figure1: Concatenation figure of lq, output and gt



Low_quality Output Ground truth

Evaluation Protocol:

1. PSNR:Peak Signal-to-Noise Ratio (PSNR), an objective standard for evaluating images, has many application scenarios. The whole meaning is to reach the peak signal of the noise ratio, psnr is generally used for an engineering project between the maximum signal and the background noise. Usually after image compression, the output image is usually different from the original image to some extent. In order to measure the image quality after processing, we usually refer to the PSNR value to measure whether a certain processing procedure is satisfactory.
2. SSIM:The full name of SSIM is structural similarity index, which is structural similarity, which is an index to measure the similarity of two images. The metric was first proposed by the Laboratory for Image and Video Engineering at UT Austin. And if the two images are before and after compression, then the SSIM algorithm can be used to evaluate the quality of the compressed image.

Dataset:

CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset with more than **200K** celebrity images, each with **40** attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including 202599 facial images[2].

Due to time and machine constraints, we could not use such a huge data set to train the model, so we used 500 high-definition pictures as the training set

2. Neural Network / Machine Learning Model

The authors propose two different neural architecture search models for a convolutional neural network. The first being a Face Restoration Searching Network, FRSNet. The input into this network is a degraded low quality image. To get this degraded image, multiple types of degradation techniques like adding noise, blur and compression artifacts are applied to a high quality image to get a full low quality image. The model follows an unsupervised learning with this low quality image.

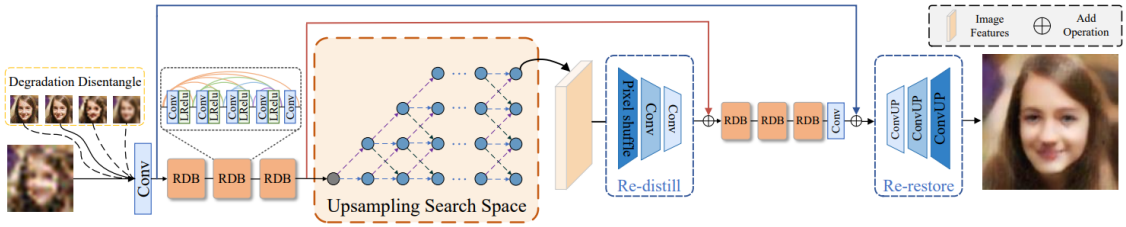


Fig. 2. Overview of the proposed FRSNet. The residual-based backbone RDB (residual dense block) extracts facial features that are processed by a decoding architecture. We utilize the NAS framework to select the optimal decoding architecture from an upsampling search space. We illustrate the degradation disentanglement strategy on the left.

Features are first extracted from this image from 3 layers of residual dense blocks. This residual network is a type of convolutional neural network. Each residual block takes the input from the previous layer and runs it through 5 rounds of convolutions each with a kernel size of 3 with a stride of 1 and padding of 1. These convolution layers each are followed by an LReLU activation to the next convolution layer. The LReLU activation is a deviation from the ReLU activation function in which negative values are given some small positive slope to pass some negative values instead of having all negative values be fixed to 0. In this implementation, the slope applied in 0.2. This is to prevent the dying ReLU problem[3]. In a residual block, each layer has what is called a skip connection to all the layers that come after. This allows a direct connection and gives a contiguous memory mechanic[4].

Now that initial features have been extracted from the residual dense blocks, a neural architecture search framework is used to select the best upsampling path from the design upsampling search space that will give the best feature transformation operations. The main goal in this upsampling search space is to go from an input resolution that matches the given low quality image, to a resolution that is 2, 4, and 8 times resolved. The upsampling search space proposed in this paper consists of 6 layers and each layer with the 4 possible upsampling rates (1, 2, 4, 8). Each connection to a cell in the search space has a scalar representing the weight of the previous cell where these scalars are normalized with softmax. Each cell in this search space is a result of the the cell in the layer before at the same upsampling rate, the cell in the layer before with the upsampling rate x 2, and the cell the layer before with the upsampling rate / 2.

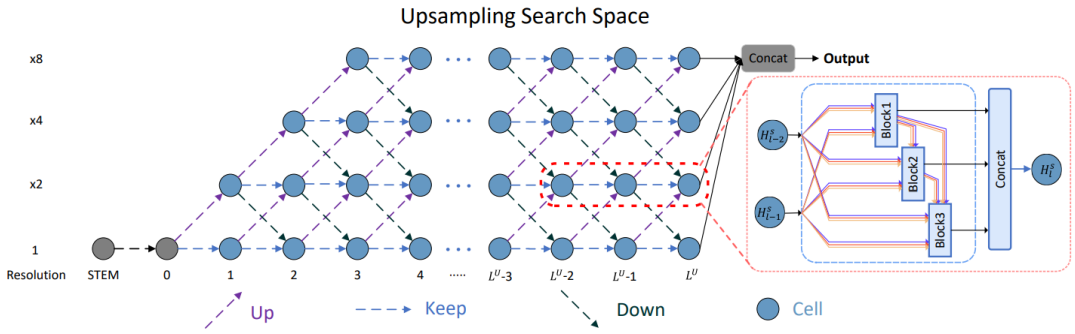


Fig. 3. Overview of the proposed upsampling search space. We illustrate the network-level search space on the left and the cell-level search space on the right. The number of layers L^U is set to 6.

The next step in this model is a redistill phase. The output of the upsampling search space first goes through pixel unshuffle. This is where the dimensions are mapped onto 1 layer given an upscale factor, in this case 8. This resulting image then goes through 2 levels of convolution with the same LReLU activation function mentioned above. Then, this result is combined with the low quality features extracted from before the upsampling search space to enforce the original feature extraction preventing any major loss. This is then run through an additional 3 layers of residual dense blocks for further feature extraction. The final step of this model is then the re-restore phase. This result of the residual network is combined with the original low quality image to undergo three levels of convolution upsampling with a kernel size of 3 with a stride of 1 and padding of 1. The result of this is the final high quality image.

The next proposed model is the Multiple Facial Prior Search Framework, MFPSNet. This is the model the experiment is based on. This model has the exact same steps as the FRSNet, the only difference being a fusion model between the upsampling search space and the de-distill phase. This is where the concept of using image priors is introduced. The paper chose three different priors typically used in the face recognition and resolution community, parsing maps, heatmaps, and facial dictionaries. For each of these priors, an upsampling search space with a NAS method similar to the one described above is used. The result of these three architecture searches are then fused with the output of the original space that extracted the facial features of the original input image. The result of this fusion model is then what is an input to the re-distill phase. The goal of this fusion search space is to aggregate the information from these 4 types of image features. Again, NAS is utilized to get the optimal path and the search space is set up identical to before although 8 layers are used this time.

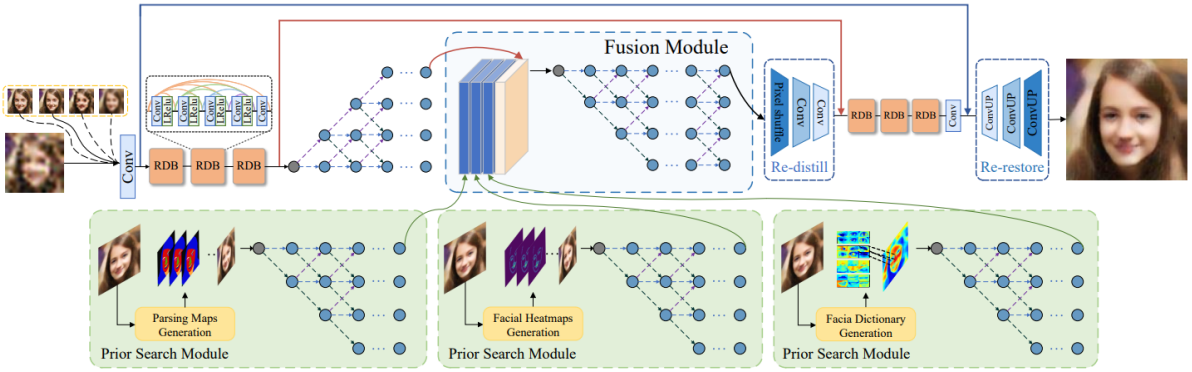


Fig. 4. Overview of the proposed MFPSNet. The MFPSNet is on the basis of the FRSNet and adds the additional prior search module and the fusion module. Three prior search modules extract features from different facial priors including parsing maps, facial heatmaps and facial dictionary, while the fusion module incorporates the prior features into facial features.

Each prior adds additional information for a better result. The parsing maps are generated through an external library[5]. This does classification on the basis of specific features, for example right eye, left eye, and nose. A mask is generated for each major feature and then flattened into one image to create a final combined image of major features masked out. The heatmaps follow a gaussian distribution following the main feature facial contours of the face. These are generated from a face-alignment module which collects 2d points following the contours of the major facial features[6]. The final prior used is a facial dictionary. This again is generated from an external module[7]. Here a dictionary is created for each major facial feature from its own training set and then created for a specific image. The exact implementation of these prior generations are out of the scope for this paper and project and is used as a tool without modifications.

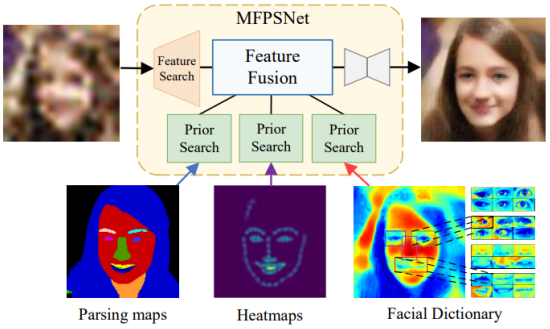


Fig. 1. An illustration of the proposed MFPSNet. Given a low quality face image, and aided by three typical facial priors (parsing maps, facial heatmaps and facial dictionary), MFPSNet restores a high quality face correspondingly.

The loss function used is a multi-prior loss function that combines the L1 loss and perceptual loss. The L1 loss is used to measure the pixel-wise difference between the restored image and the ground truth, while the perceptual loss is used to capture the high=level features of the image, such as color and texture.

3. Experiment

Preparations for the experiments:

Bug Fixing:

There were a surprisingly high number of bugs present in the code making it non-runnable until they were fixed. These fixes included:

i. Adding the solver argument to the train_args.py file (required for the
ii. Removing the baseline_path parameter in FFHQ_train_MFPS.py
iii. Adding the heat_feature variable which contains the heatmap information and dict_feature variable which contains the facial dictionary information to the step_in list which passes this information to the next pass.

I. Experiment 1: Choice of Optimizer

Design

Research Question	How does the choice of optimizer affect the performance of the proposed method for blind face restoration using multi-prior learning?
Hypothesis	Optimizers with an adaptive learning rate like Adam would perform better than optimizers with a fixed learning rate like Stochastic Gradient Descent.
Independent Variables	opt.solver (default value = 'adam' referring to the Adam optimizer. Other value could be 'sgd' referring to the Stochastic Gradient Descent optimizer)
Control Variables	1. step_in (list of all features to consider when building the model. Default value = [autoSR_out, parse_feature, heat_feature, dict_feature]. autoSR_out contains the output of a previous step, parse_feature contains the information of the parsing maps, heat_feature contains the information of the heatmaps, and dict_feature contains the information of the facial dictionaries.) 2. opt.nEpocs (number of epochs; default value = 104) 3. opt.lr (learning rate; default value = 0.0001) 4. opt.batch_size (training batch size; default value = 8)
Dependent Variables	1. mean_psnr (mean Peak-Signal-To-Noise ratio) 2. mean_ssim (mean Structural Similarity Index)

Methodology

1. A condition to check if the parameter "solver" had the value "sgd" was added. If yes, the Stochastic Gradient Descent optimizer would be used instead of the default Adam optimizer.
2. First, the solver was set to the default "adam". To achieve this, the training was run using train.py without passing the solver argument.

3. The program was run for 104 epochs with a default batch size of 8.
4. The weight decay was set to 0.001.
5. Once done, the demo.py file was run to create the output images using the 104th (last) checkpoint.
6. Once all the output images were generated, the evaluation.py file was run to find the PSNR and SSIM scores for this experiment.
7. This process was now repeated for the Stochastic Gradient Descent optimizer case.
8. The training was run using train.py with the argument: –solver sgd.
9. The weight decay was set to 0.001 while all other parameter values resembled the adam optimizer parameter values.
10. The output images were then obtained along with the PSNR and SSIM scores.

II. Experiment 2: Number of Priors

Design

Research Question	How does the number of priors used in multi-prior learning affect the performance of blind face restoration?
Hypothesis	As the number of priors reduces, the performance of the multi-prior learning will also reduce.
Independent Variables	1. step_in (list of all features to consider when building the model. Default value = [autoSR_out, parse_feature, heat_feature, dict_feature]. autoSR_out contains the output of a previous step, parse_feature contains the information of the parsing maps, heat_feature contains the information of the heatmaps, and dict_feature contains the information of the facial dictionaries.)
Control Variables	1. opt.solver (value = ‘adam’ referring to the adam optimizer) 2. opt.nEpocs (number of epochs; default value = 104) 3. opt.lr (learning rate; default value = 0.0001) 4. opt.batch_size (training batch size; default value = 8)
Dependent Variables	1. mean_psnr (mean Peak-Signal-To-Noise ratio) 2. mean_ssim (mean Structural Similarity Index)

Methodology

1. The model was first trained and tested on all the priors to obtain the PSNR and SSIM values.
2. Next, the parse_feature was removed from the step_in list which contains all the priors that are used for training.
3. The training was then done using all the default parameter values. After this, the testing was also done.
4. Once all the output images were generated, the evaluation.py file was run to find the PSNR and SSIM scores for this experiment.

5. This process was then repeated by removing the heat_feature while keeping the parse_feature and the dict_feature.
6. The process was once again repeated, this time by removing the dict_feature and keeping the parse_feature and the heat_feature.

4. Experimental Results and Discussion

Our initial hypothesis were as follows:
Hypothesis1 : Optimizers with an adaptive learning rate like Adam would perform better than optimizers with a fixed learning rate like SGD.

For this hypothesis we trained our model on both these optimizers and compared the results for the PSNR and SSIM. The table below gives us a visualization of the baseline results and table 2 here gives us the results after our experiment.

Metric	Value
PSNR	21.62917900356466
SSIM	0.3977689239446125

Table 1: Baseline results

Optimizer	Iteration	Metrics	Value
Adam	0	PSNR	23.24370832010578
		SSIM	0.8162360358860138
	1	PSNR	23.920010644479802
		SSIM	0.8249151840725553
SDG	0	PSNR	23.21562114157254
		SSIM	0.6579162815169355
	1	PSNR	23.751039662523144
		SSIM	0.678114967817401

Table 2: Experiment 1 results on testing both (Adam and SDG) optimizers

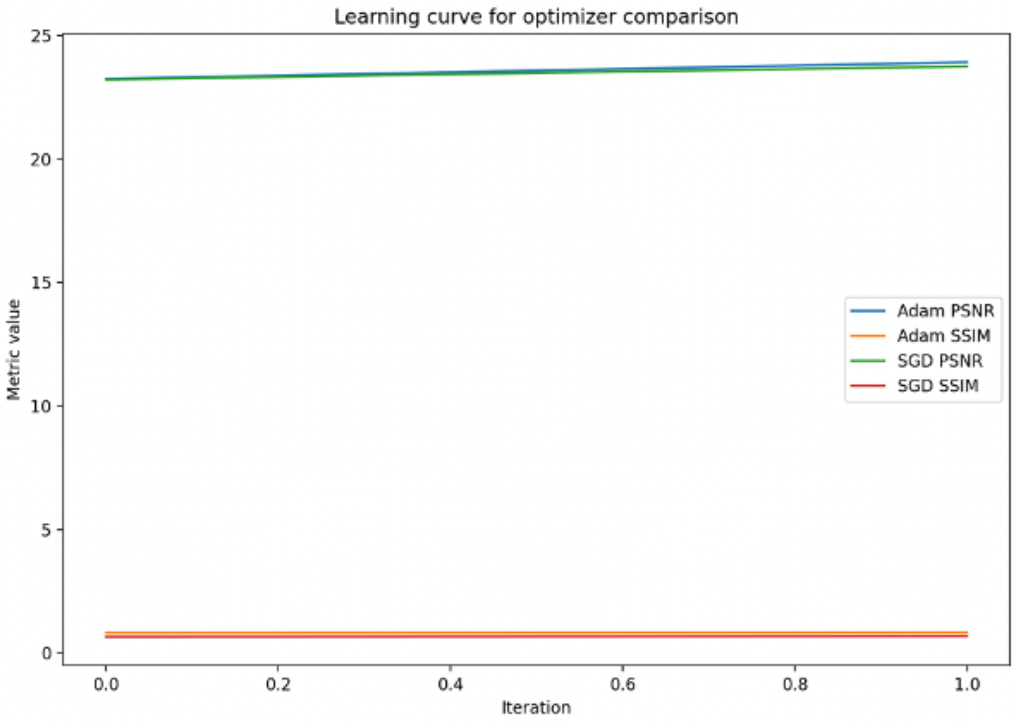


Fig: Learning curve for Adam optimizer and the SGD optimizer comparison

From the results of Experiment 1, it can be seen that the Adam optimizer performed better than the SGD optimizer in terms of PSNR and SSIM. However, it should be noted that the results for SGD were not far behind and further experimentation may be required to determine the optimal optimizer for this particular problem. But overall this result did support our initial hypothesis that Adam optimizer will perform better than the SGD.

It is important to note here that hyper parameters are such as the number of layers in the network, the number of channels in each layer, the learning rate, the batch size, etc.
As for the loss function, a new multi-prior loss function is used that combines the L1 loss and perceptual loss. The L1 loss is used to measure the pixel-wise difference between the restored image and the ground truth, while the perceptual loss is used to capture the high-level features of the image, such as color and texture.
The authors had also introduced a new attention-based weighting scheme to adjust the importance of each prior image during the training process which we will also be using for our model. This loss function will be useful for us with some slight changes.
The hyperparameters and the loss function have not been altered much as compared to the initial values but have been built upon.

Hypothesis 2: As the number of priors reduces, the performance of the multi-prior learning will also reduce.

The hypothesis led to the experiment 2 where we worked with face dictionaries, heat maps, parse maps. In each step we remove one of the prior values and test to see if the hypothesis is correct or not.

Prior(removed)	Iteration	Metric	Value
Face Dictionaries	0	PSNR	23.298488884987954
	1	SSIM	0.6601665661725232
	0	PSNTR	23.943895657763917
	1	SSIM	0.6823947103133946
Heat Maps	0	PSNR	13.959040662592686
	1	SSIM	0.4408279717936988
	0	PSNR	12.110697728872333
	1	SSIM	0.40386903497809606
Parse Maps	0	PSNR	14.672600915065615
	1	SSIM	0.4515721559874349
	0	PSNR	13.407175005594045
	1	SSIM	0.4259239965617286

Table 3 Experiment 2 results on removing one of the priors

From the above table (Table 3) we can see that Experiment 2 aimed to investigate the impact of removing each prior on the overall performance of the model. The results indicate that removing the face dictionary prior had a relatively small impact on performance, while removing either the heatmap or parse map had a much more significant impact. This suggests that these two priors may be more important for accurate image reconstruction. From this we can conclude that the hypothesis did hold true but it was only partially correct since only the factors which are more important will have an impact on the performance but if we remove multiple priors which are not very relevant to our research will not impact the performance much. Hence the hypothesis is only partially correct, but it does not hold true for all the cases.
The next steps for this would be to perform more analysis on these hypotheses to prove which values are most important ones and keep only those important priors.

References

[1]Yu, Yanjiang, et al. "Multi-Prior Learning via Neural Architecture Search for Blind Face Restoration." *arXiv preprint arXiv:2206.13962* (2022).

[2]Liu, Ziwei, et al. "Large-scale celebfaces attributes (celeba) dataset." *Retrieved August* 15.2018 (2018): 11.

[3]Paszke, A. et al., 2019. *Leakyrelu*. LeakyReLU - PyTorch 2.0 documentation. Retrieved from https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html#leakyrelu

[4]Yulun, Z. et al. (2018, March). *Residual dense network for image super-resolution*. arxiv. Retrieved from https://arxiv.org/pdf/1802.08797.pdf

[5] Piggybank. (2021, August). *Face parcing- using pytorch*. Face Parcing- Using Pytorch. Retrieved from https://spyjetson.blogspot.com/2021/08/face-parcing.html

[6]Bulat, A. (2021, April). *face-alignment: 2D and 3D face alignment library build using pytorch*. Retrieved from https://github.com/1adrianb/face-alignment

[7]Li , X. (2020). *Blind face restoration via deep multi-scale component dictionaries (ECCV 2020)*. Retrieved from https://github.com/csxmli2016/DFDNet