Afyon Kocatepe Üniversitesi Fen ve Mühendislik Bilimleri Dergisi

Afyon Kocatepe University Journal of Science and Engineering

AKÜ FEMÜBİD **21** (2021) 065101 (1343-1352) AKU J. Sci. Eng. **21** (2021) 065101 (1343-1352)

DOI: 10.35414/akufemubid.819319

Araştırma Makalesi/Research Article

Contribution Analysis of Optimization Methods on Super-Resolution

Yıldıray ANAGÜN[1,*], Şahin IŞIK[2]

[1,2]Eskişehir Osmangazi Üniversitesi, Mühendislik ve Mimarlık Fakültesi, Bilgisayar. Mühendisliği Bölümü, Eskişehir.

*e-posta: *Corresponding author: yanagun@ogu.edu.tr, ORCID ID: https://orcid.org/0000-0002-7743-0709*
sahini@ogu.edu.tr, ORCID ID: http://orcid.org/0000-0003-1768-7104

Geliş Tarihi: 01.11.2020 Kabul Tarihi: 31.12.2021

Keywords
Super-resolution; Deep learning; Convolutional neural network; Optimizer

Abstract

In this study, the benefits of choosing a robust optimization function with super resolution are analyzed. For this purpose, the different optimizers are included in the simple Convolutional Neural Network (CNN) architecture SRNET, to reveal the performance of the each method. Findings of this research provides that Adam and Nadam optimizers are robust when compared to (Stochastic Gradient Descent) SGD, Adagrad, Adamax and RMSprop. After experimental simulations, we have achieved the 35.91 (dB)/0.9960 and 35.97 (dB)/0.9961 accuracy rates on Set5 images from Adam and Nadam optimizers, respectively (9-1-5 network structure and filter sizes 128 and 64). These results show that selected optimization function for the CNN model plays an important role in increasing the accuracy rate in the super-resolution problem.

Optimizasyon Yöntemlerinin Süper Çözünürlük Üzerine Katkı Analizi

Anahtar kelimeler
Süper-çözünürlük; Derin öğrenme; Evrişimsel sinir ağı; Optimize edici

Öz

Bu çalışmada, süper çözünürlükte sağlam bir optimizasyon fonksiyonu seçmenin yararları analiz edilmiştir. Bu amaçla her yöntemin performansını ortaya çıkarmak için farklı optimize ediciler, basit Evrişimsel Sinir Ağı (CNN) mimarisi SRNET' e dahil edilmiştir. Bu araştırmanın bulguları, Adam ve Nadam optimize edicilerin Stokastik Gradyan İnişi (SGD), Adagrad, Adamax ve RMSprop ile karşılaştırıldığında daha kararlı olduğunu göstermektedir. Deneysel simülasyonlardan sonra, Adam ve Nadam optimize edicilerinden Set5 görüntülerinde sırasıyla 35.91 (dB)/0.9960 ve 35.97 (dB)/0.9961 doğruluk oranlarına ulaştık (9-1-5 ağ yapısı ve filtre boyutları 128 ve 64). Bu sonuçlar, CNN modeli için seçilen optimizasyon fonksiyonunun süper çözünürlük probleminde doğruluk oranını arttırmada önemli bir rol oynadığını göstermektedir.

1. Introduction

lSuper-resolution (SR) reconstruction can be obtained from a set of low-resolution (LR) images, or it can be obtained from only a single image. LR images have such undesirable factors as aliasing, optical distortion, blurring, noise etc. Applications of the super-resolution technology include, but are not limited to medical imaging, military, satellite imaging, remote imaging and video surveillance. In literature, many methods have been proposed to make unsupervised, semi-supervised or supervised SR reconstruction. It can also be categorized the reconstruction of ill-posed SR problem as multi-frame and single frame. The SR reconstruction research has developed very rapidly, after it was first addressed by (Tsai and Huang 1984). They proposed an unsupervised multi-frame method that performs reconstruction in the frequency domain. Single image super-resolution (SISR) intends to the generation of a high-resolution image from a single LR observation. Although it has a key role for many image processing systems, generating a solution for the SISR is very difficult due to the ill-posed inverse problem. Moreover, the reconstructed HR image should be visually pleasing and as realistic as possible in terms of the original. In recent years, the SISR problem has been extensively researched and notable and effective algorithms have been proposed. SISR methods can be classified into three main groups as interpolation-based (Dong *et al*. 2013), (Yang *et al*. 2010), (Zhu *et al*. 2016), reconstruction-based (Mandal *et al*. 2017), (Ren *et al*. 2017), (Yan *et al*. 2015) and learning-based algorithms. The learning-based methods consist of dictionary learning-based (Kaveh and Ezzatollah 2017), (Gao *et al*. 2012), (Yang *et al*. 2008), (Yang and Wang 2013) and deep learning-based. Convolutional Neural Networks (CNNs) are used in a wide variety of signal and image processing subjects such as object and pattern recognition, segmentation and super- resolution. CNNs plays a major role for construction a deep learning-based SR model. Dong *et al*. (2015) proposed a Super-Resolution Convolutional Neural Network (SRCNN) to make a nonlinear LR-to-HR mapping function. This CNN network model provided the basis for increasing network depth (Kim *et al*. 2016) or enhance recursive layers (Tai *et al*. 2017). In Kim *et al*. (2016), a very deep convolutional network have been used inspired by VGGnet used for ImageNet classification (Simonyan and Zisserman 2015). They also demonstrated that increasing network depth provides a significant improvement in accuracy. However, VDSR cause to heavy computation time and memory consumption due to the large number of convolution layers. SRResNet, Ledig *et al*. (2017) solved the time and memory consumption problem without changing the ResNet architecture (He *et al*. 2016) too much. The high computational time cost prevents practical usage that needs real-time performance. To overcome this problem, a faster SR method is proposed in (Dong *et al*. 2016). This model used shallow network architecture, therefore it could not learn complex mappings accurately. Lai *et al*. (2018), developed a deep convolutional network within a Laplacian pyramid framework for fast and accurate image SR. In contrast to the previous works, in this study the features are extracted directly from the low-resolution input space instead of bicubic interpolation. An advanced SR algorithm is presented in Lim *et al*. (2017), which provides better results by removing unnecessary modules from the traditional ResNet architecture. They also expanded the model size in addition to the multi-scale stabilized training model to improve performance. On the other hand, more recently, Tiantong *et al*. (2019) developed a new CNN architecture to learn the SR mapping function in an image transformation domain, in particular for discrete cosine transform (DCT). CNNs have achieved impressive SR performance on video frames. Li *et al*. (2020) developed a different approach to combine motion compensation technique with CNN to estimate a high resolution video from LR counterpart. The deep unfolding architecture can be considered as a main component of a multimodal framework for image super-resolution. Marivani *et al*. (2020) proposed a multimodal deep learning method which combines sparse priorities and enables efficient integration of data from another image modality into the network architecture.

In all deep learning-based SR methods, an optimization function should be used to increase the accuracy rate and improve the quality of the output result image. In this study, the performance of the frequently used optimization functions are compared including different architectures and image sets. The rest of this paper is organized as section 2 presents the utilized method and tools, section 3 shows the information about dataset and performance evaluation and finally a conclusion is touched at the last section.

2. Method and Tools

As the key motivation behind of this study, we have investigated the contributions of optimizers when it comes to update the weights of CNN. It is believed that different optimizers would produce the different sets of weights by avoiding the under-fitting and over-fitting cases. One can say that the gradient descent is searching a local minimum by moving in the direction of steepest descent, which is represented with Figure 1.

However, minimizing a function with steepest descent method depends on the different external factors such as loss function and learning rate. These factors affect the cost of training and have a direct impact on SR performance.

By considering such facts, we need to ensure that which optimizer is best by making a tradeoff between the performance and computational time.

We have made experiments with following optimizers.

1. SGD (Rumelhart *et al*. 1986)
2. RMSprop (Tieleman and Hinton 2012)
3. Adam (Kingma and Ba 2014)
4. Adagrad (Duchi *et al*. 2011)
5. Adamax (Kingma and Ba 2014)
6. Nadam (Dozat 2016)

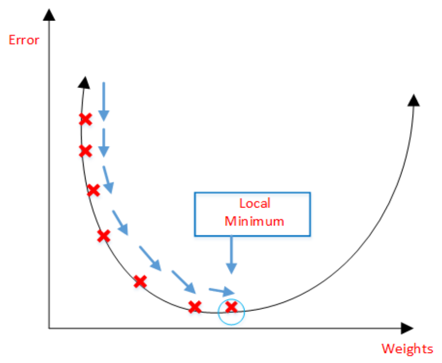

Figure 1. Gradient descent.

For all experiments, the SRCNN architecture has been chosen (Dong *et al*. 2015). Comparison of optimizers are usually a trade-off between performance and speed. To demonstrate this, we train three networks for comparison, which are 9-1-5, 9-3-5, and 9-5-5. Each network structures have been trained with respect to the three experiments: (i) first one is a shallow network with filter sizes n_1=32 and n_2=16, (ii) second one is larger network with filter sizes n_1=64 and n_2=32, (iii) third one is deeper network with filter sizes n_1=128 and n_2=64. Each convolutional layer uses a rectified linear unit (ReLU) as the activation function.

The learning rate is fixed to 1e-4 for all optimizers. The loss function is selected as mean absolute error, namely L1 loss. The reason of why we used L1 loss, is explained in the thesis (Anagün 2018). In this study, the L1 loss function is faster than the other loss functions and gives effective results for solving the SR problem (Anagun *et al*. 2019).

For all optimizers, the batch size is set to 64 and is chosen for training while the rest of them are selecting for testing. Moreover, the mini-batch size is selected as 16 and epoch size is set to 20. The total parameters of SRCNN is 216,961. The Figure 2 shows the overall components of the proposed system when searching the best optimizer in CNN model. The all implementations are carried out on Python Keras library with Tensorflow backend.

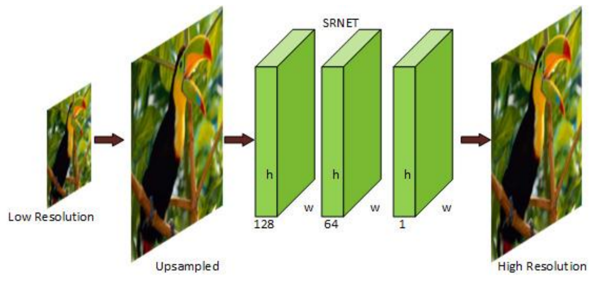

Figure 2. The utilized system for super-resolution.

We can note that the robustness of optimization function is more valuable than the required computation time. With a simple CNN structure, each optimizer produced the different weights for super-resolution.

3. Performance Evaluation

3.1. Datasets

To evaluate the system performance, we have used a common dataset, namely T91 image set (Dong *et al*. 2015). It contains 91 color images. The experiments are realized by selecting patches with stride 10 and scales 2, 3 and 4. The 30.347 samples reserved for training and 3.372 samples reserved for validation. We also compare the performances of optimizers on two standard benchmark datasets: Set5 (Bevilacqua *et al*. 2012) and Set14 (Zeyde et.al 2010).

3.2. Performance Analysis

We use the source codes of optimization methods to evaluate the runtime on the same machine with 2.5 GHz Intel i5 CPU (16GB RAM) and NVIDIA GeForce GTX 1050 with GPU (4GB Memory). For a benchmark evaluation, the performance of optimizers have been inspected in three ways; (i) subjective analysis, (ii) objective comparison and (iii) computation time analysis. The 10% of samples were chosen for validation and rest of them for training. For objective results, the Peak-Signal to Noise Ratio (PSNR) and Structural Similarity Index

Measure (SSIM) values are taken as reference. The PSNR metric is shown in Equation 1. The RMSE refers to root mean square error between predicted and actual image.

$$PSNR = 20 * \log(255/RMSE) \quad (1)$$

The subjective evaluation of proposed method is shown in Figure 3. From the visual quality analysis, we can note that the RMSprop/Adam/Nadam optimizers are robust to edge and texture preserving in super resolution output. Some blur artifacts are available on the outputs of SGD method. Also, the SGD method is not better than Bicubic. The performance declination of SGD method is explained with potential capability of utilized dataset (T91). It means that the more number of different patches are needed to make training experiments with SGD optimizer.

For the second evaluation, we have compared the overall PSNR/SSIM values by conducting experiments of Set5 and Set14 datasets. The results of the Set5 dataset according to the network structures are given in Table 1, Table 2 and Table 3, respectively. Experimental results of Set14 dataset are summarized in Table 4, Table 5 and Table 6.

Except for SGD, there is performance competition between all optimizers. When we review the top PSNR scores, it can be seen that Nadam and Adam optimizers are the best among the other methods. Although the PSNR value of RMSprop is lower than Adam and Nadam, but its SSIM value is considerably higher than the other optimizers. Comparing RMSprop and Adamax, we can note that both of them produced the similar performance. The structural similarity is not preserved for Adagrad and SGD.

To examine the effectiveness of the optimization algorithms as a third experiment evaluation, we have compared the computation time of each optimization method in case of training for all networks. It can be said that the training time depends on the network depth. In addition, it can be seen that the Nadam method is worst one and SGD method is best one in terms of required average computation time for super-resolution. It means that converging the huge weights becomes very fast with SGD method. Aside from time advantage, the visual performance of SGD is not high when observing the results giving Table 7, Table 8 and Table 9.

4. Conclusion

Through this study, we have analyzed the contributions of different optimization functions on super-resolution. It is interesting about this experiment is that we have found the Adam optimizer is robust in terms of numerical values, especially for PSNR and SSIM. On the other side, the Adamax method is very fast when updating the weights of a CNN structure through backpropagation stages. Overall, these results suggest that the Adam and Nadam optimizers involve various functionality for super-resolution. In future work, to show the importance of optimizer selection different CNN deep models can be included in the proposed model.

Table 1. The average PSNR (dB) and SSIM values for all scales of 9-1-5 network structure on Set5 dataset

		***n₁*=32, *n₂*=16**		***n₁*=64, *n₂*=32**		***n₁*=128, *n₂*=64**	
Optimizer Name	**Scale**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**
SGD	2	32.42	0.9890	32.54	0.9895	32.31	0.9887
	3	30,48	0.9825	30.35	0.9823	30.21	0.9813
	4	28.76	0.9740	28.61	0.9733	28.64	0.9731
RMSprop	2	35.19	0.9952	35.17	0.9954	35.18	0.9957
	3	31.77	0.9887	31.90	0.9894	32.03	0.9900
	4	29.60	0.9805	29.72	0.9814	29.79	0.9823
Adam	2	35.31	0.9949	35.63	0.9956	35.91	0.9960
	3	31.88	0.9888	32.07	0.9896	32.26	0.9903
	4	29.70	0.9813	29.89	0.9823	30.07	0.9832
Adagrad	2	31.26	0.9854	32.17	0.9881	32.97	0.9904
	3	28.77	0.9722	30.54	0.9828	30.84	0.9839
	4	27.93	0.9665	28.54	0.9724	28.87	0.9745
Adamax	2	35.31	0.9950	35.39	0.9952	35.68	0.9957
	3	31.78	0.9889	31.79	0.9883	32.96	0.9897
	4	29.59	0.9801	29.79	0.9817	29.96	0.9826
Nadam	2	35.45	0.9952	35.64	0.9956	35.97	0.9961
	3	31.92	0.9889	32.06	0.9896	32.24	0.9902
	4	29.71	0.9810	29.87	0.9820	30.85	0.9831

Table 2. The average PSNR (dB) and SSIM values for all scales of 9-3-5 network structure on Set5 dataset

		***n₁*=32, *n₂*=16**		***n₁*=64, *n₂*=32**		***n₁*=128, *n₂*=64**	
Optimizer Name	**Scale**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**
SGD	2	33.01	0.9903	32.58	0.9895	32.64	0.9896
	3	30.62	0.9831	30.41	0.9825	30.43	0.9826
	4	28.76	0.9739	28.67	0.9737	28.71	0.9739
RMSprop	2	34.99	0.9952	35.50	0.9960	35.47	0.9957
	3	31.89	0.9893	32.22	0.9905	32.37	0.9912
	4	29.80	0.9826	29.97	0.9834	29.98	0.9837
Adam	2	35.72	0.9957	35.98	0.9961	36.35	0.9965
	3	32.05	0.9896	32.29	0.9904	32.46	0.9909
	4	29.93	0.9824	30.08	0.9833	30.18	0.9838
Adagrad	2	31.36	0.9863	33.36	0.9913	33.97	0.9925
	3	30.26	0.9815	30.95	0.9844	31.17	0.9853
	4	28.63	0.9727	29.19	0.9767	29.27	0.9773
Adamax	2	35.62	0.9956	35.84	0.9959	36.00	0.9961
	3	31.88	0.9889	32.09	0.9897	32.30	0.9904
	4	29.87	0.9820	29.99	0.9827	30.14	0.9835
Nadam	2	35.78	0.9958	36.00	0.9961	36.29	0.9965
	3	32.13	0.9899	32.29	0.9904	32.48	0.9910
	4	29.82	0.9817	30.10	0.9833	30.27	0.9842

Table 3. The average PSNR (dB) and SSIM values for all scales of 9-5-5 network structure on Set5 dataset

		***n₁*=32, *n₂*=16**		***n₁*=64, *n₂*=32**		***n₁*=128, *n₂*=64**	
Optimizer Name	**Scale**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**
SGD	2	32.56	0.9895	32.28	0.9886	33.02	0.9905
	3	30.50	0.9828	30.41	0.9822	30.65	0.9832
	4	28.79	0.9743	28.74	0.9737	28.89	0.9747
RMSprop	2	35.13	0.9957	35.40	0.9960	36.00	0.9965
	3	31.92	0.9897	32.23	0.9908	32.46	0.9916
	4	29.90	0.9829	29.97	0.9839	30.02	0.9842
Adam	2	35.94	0.9961	36.23	0.9964	36.53	0.9967
	3	32.18	0.9901	32.43	0.9908	32.61	0.9914
	4	29.98	0.9827	30.18	0.9839	30.30	0.9845
Adagrad	2	33.39	0.9914	33.71	0.9920	34.83	0.9941
	3	30.96	0.9842	31.20	0.9854	31.39	0.9864
	4	29.01	0.9752	29.36	0.9780	29.61	0.9798
Adamax	2	35.67	0.9956	35.81	0.9958	36.12	0.9963
	3	31.79	0.9884	32.28	0.9902	32.45	0.9908
	4	29.87	0.9820	30.08	0.9832	30.23	0.9840
Nadam	2	35.71	0.9957	36.04	0.9962	36.51	0.9966
	3	32.28	0.9903	32.47	0.9909	32.66	0.9914
	4	30.03	0.9827	30.27	0.9843	30.36	0.9847

Table 4. The average PSNR (dB) and SSIM values for all scales of 9-1-5 network structure on Set14 dataset

		***n*₁=32, *n*₂=16**		***n*₁=64, *n*₂=32**		***n*₁=128, *n*₂=64**	
Optimizer Name	**Scale**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**
SGD	2	29.07	0.9752	29.18	0.9757	28.95	0.9745
	3	27.48	0.9649	27.39	0.9642	27.28	0.9633
	4	26.16	0.9531	26.06	0.9522	26.08	0.9524
RMSprop	2	31.45	0.9854	31.49	0.9856	31.51	0.9859
	3	28.58	0.9721	28.64	0.9727	28.76	0.9731
	4	26.81	0.9592	26.89	0.9597	26.95	0.9604
Adam	2	31.45	0.9853	31.72	0.9858	31.92	0.9863
	3	28.63	0.9722	28.75	0.9729	28.88	0.9735
	4	26.87	0.9599	27.02	0.9608	27.13	0.9618
Adagrad	2	28.21	0.9703	28.86	0.9740	29.50	0.9774
	3	26.26	0.9534	27.55	0.9654	27.77	0.9670
	4	25.57	0.9463	25.99	0.9513	26.24	0.9539
Adamax	2	31.46	0.9851	31.55	0.9854	31.77	0.9859
	3	28.54	0.9719	28.56	0.9720	28.79	0.9730
	4	26.79	0.9592	26.93	0.9602	27.04	0.9611
Nadam	2	31.58	0.9855	31.72	0.9858	31.96	0.9864
	3	28.62	0.9722	28.76	0.9729	28.87	0.9734
	4	26.86	0.9597	26.99	0.9605	27.13	0.9617

Table 5. The average PSNR (dB) and SSIM values for all scales of 9-3-5 network structure on Set14 dataset

		***n*₁=32, *n*₂=16**		***n*₁=64, *n*₂=32**		***n*₁=128, *n*₂=64**	
Optimizer Name	**Scale**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**
SGD	2	29.54	0.9775	29.24	0.9761	29.23	0.9759
	3	27.60	0.9657	27.47	0.9648	27.44	0.9645
	4	26.17	0.9533	26.10	0.9526	26.12	0.9528
RMSprop	2	31.35	0.9851	31.73	0.9863	31.70	0.9858
	3	28.67	0.9724	28.89	0.9734	28.94	0.9740
	4	26.93	0.9613	27.05	0.9614	27.11	0.9617
Adam	2	31.78	0.9859	31.95	0.9863	32.18	0.9868
	3	28.74	0.9728	28.91	0.9735	29.01	0.9740
	4	27.04	0.9609	27.15	0.9618	27.23	0.9623
Adagrad	2	28.20	0.9700	29.87	0.9792	30.34	0.9812
	3	27.31	0.9636	27.85	0.9675	28.04	0.9688
	4	26.07	0.9522	26.47	0.9562	26.53	0.9567
Adamax	2	31.75	0.9858	31.87	0.9862	31.97	0.9864
	3	28.61	0.9723	28.77	0.9730	28.91	0.9736
	4	26.97	0.9605	27.05	0.9610	27.15	0.9617
Nadam	2	31.84	0.9861	32.00	0.9864	32.16	0.9867
	3	28.78	0.9731	28.90	0.9735	29.07	0.9744
	4	26.94	0.9603	27.14	0.9616	27.32	0.9628

Table 6. The average PSNR (dB) and SSIM values for all scales of 9-5-5 network structure on Set14 dataset

		***n*₁=32, *n*₂=16**		***n*₁=64, *n*₂=32**		***n*₁=128, *n*₂=64**	
Optimizer Name	**Scale**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**
SGD	2	29.18	0.9758	28.95	0.9745	29.52	0.9775
	3	27.50	0.9650	27.43	0.9644	27.58	0.9656
	4	26.17	0.9531	26.13	0.9528	26.22	0.9535
RMSprop	2	31.50	0.9858	31.64	0.9861	31.98	0.9868
	3	28.64	0.9723	28.86	0.9738	28.97	0.9745
	4	27.03	0.9611	27.08	0.9617	27.16	0.9622
Adam	2	31.96	0.9863	32.12	0.9867	32.29	0.9871
	3	28.81	0.9731	29.00	0.9738	29.10	0.9743
	4	27.06	0.9611	27.21	0.9622	27.33	0.9630
Adagrad	2	29.83	0.9789	30.15	0.9804	31.06	0.9839
	3	27.87	0.9677	28.05	0.9688	28.20	0.9699
	4	26.32	0.9546	26.58	0.9571	26.77	0.9589
Adamax	2	31.72	0.9857	31.87	0.9861	32.07	0.9866
	3	28.54	0.9718	28.88	0.9734	29.02	0.9740
	4	26.97	0.9605	27.13	0.9615	27.23	0.9624
Nadam	2	31.79	0.9859	32.04	0.9865	32.30	0.9870
	3	28.89	0.9734	29.04	0.9739	29.13	0.9744
	4	27.09	0.9612	27.29	0.9626	27.35	0.9630

Table 7. Computational time analyses of 9-1-5 network structure on train dataset

		***n_1*=32, *n_2*=16**	***n_1*=64, *n_2*=32**	***n_1*=128, *n_2*=64**	
Optimizer Name	**Scale**	**Time (s)**	**Time (s)**	**Time (s)**	**Average Time (s)**
SGD	2	145.90	230.02	401.69	**259.01**
	3	150.08	229.76	397.82	
	4	151.82	228.19	395.84	
RMSprop	2	148.92	231.85	401.93	**260.90**
	3	152.79	227.58	402.42	
	4	155.79	227.60	399.24	
Adam	2	149.86	231.59	403.26	**262.22**
	3	156.93	228.20	401.80	
	4	157.66	230.37	400.27	
Adagrad	2	150.76	230.96	397.86	**262.41**
	3	155.63	227.81	407.12	
	4	157.81	230.12	403.58	
Adamax	2	153.94	229.11	397.48	**263.01**
	3	153.62	229.51	407.28	
	4	159.20	229.26	407.72	
Nadam	2	154.33	232.70	407.87	**266.81**
	3	155.07	233.79	412.10	
	4	163.35	233.28	408.82	

Table 8. Computational time analyses of 9-3-5 network structure on train dataset

		***n_1*=32, *n_2*=16**	***n_1*=64, *n_2*=32**	***n_1*=128, *n_2*=64**	
Optimizer Name	**Scale**	**Time (s)**	**Time (s)**	**Time (s)**	**Average Time (s)**
SGD	2	176.99	294.05	586.00	**351,56**
	3	182.59	295.97	583.27	
	4	183.38	287.34	574.47	
RMSprop	2	177.64	294.53	584.75	**352,35**
	3	184.22	295.20	583.53	
	4	184.81	290.65	575.85	
Adam	2	179.43	293.12	586.75	**353,38**
	3	186.27	296.13	583.23	
	4	186.66	292.70	576.15	
Adagrad	2	181.49	293.69	588.56	**354,17**
	3	186.23	298.55	587.32	
	4	187.40	290.83	573.44	
Adamax	2	180.30	296.39	586.33	**352,89**
	3	185.16	295.12	584.79	
	4	185.86	289.88	572.14	
Nadam	2	185.79	299.52	594.23	**358,60**
	3	189.40	302.69	587.99	
	4	195.34	294.89	577.55	

Table 9. Computational time analyses of 9-5-5 network structure on train dataset

		***n_1*=32, *n_2*=16**	***n_1*=64, *n_2*=32**	***n_1*=128, *n_2*=64**	
Optimizer Name	**Scale**	**Time (s)**	**Time (s)**	**Time (s)**	**Average Time (s)**
SGD	2	243.04	383.45	656.42	**425.25**
	3	221.25	390.04	667.86	
	4	223.58	388.23	653.40	
RMSprop	2	219.46	387.74	666.06	**425.78**
	3	221.76	390.07	666.41	
	4	227.44	392.73	660.34	
Adam	2	225.17	391.21	668.70	**428.76**
	3	225.90	392.56	668.12	
	4	227.93	398.00	661.24	
Adagrad	2	219.66	387.49	670.07	**427.34**
	3	226.43	393.28	662.14	
	4	226.07	393.65	667.25	
Adamax	2	219.94	389.31	669.22	**428.75**
	3	226.61	396.66	663.42	
	4	229.79	398.00	665.81	
Nadam	2	224.41	394.87	673.96	**432.43**
	3	229.23	396.51	666.68	
	4	232.72	402.01	671.44	

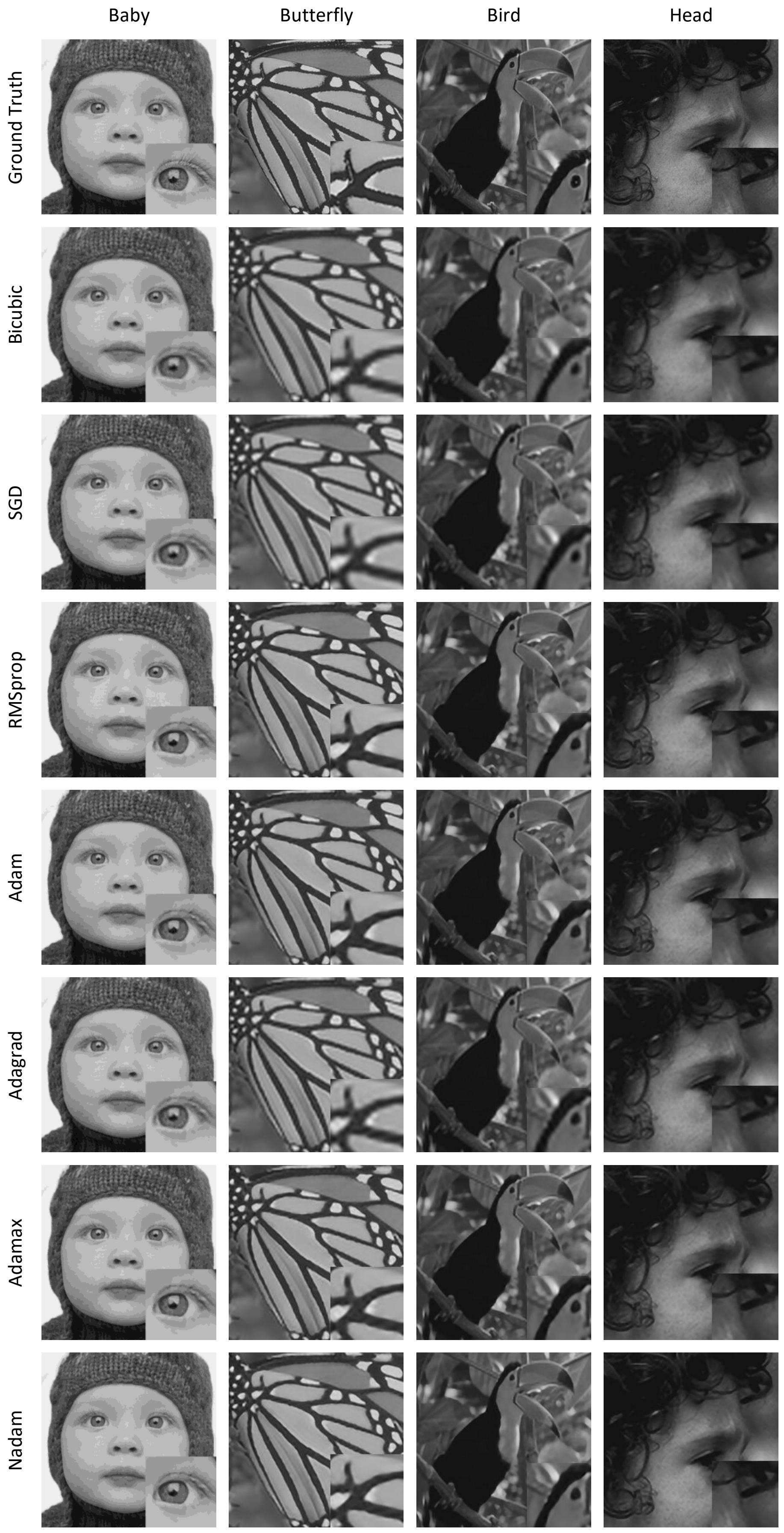

Figure 3. Visual results of 9-5-5 (n_1=128, n_2=64) for scale 3 on Set5.

5. References

Anagun, Y., Isik, S. and Seke, E., (2019). SRLibrary: Comparing different loss functions for super-resolution over various convolutional architectures. *Journal of Visual Communication and Image Representation*, **61**, 178-187.

Anagün, Y., (2018). Düşük çözünürlüklü video sahnelerinden yüksek çözünürlüklü video sahnelerinin elde edilmesi. Doktora Tezi, Fen Bilimleri Enstitüsü Elektrik ve Elektronik Mühendisliği Anabilim Dalı, Eskişehir, 75.

Bevilacqua, M., Cunningham, A., Guillemot, C., et al., (2012). Low-complexity single-image super-resolution based on nonnegative neighbor embedding.

Dong, C., Loy, C. C., He, K., et al., (2015). Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **38** (2), 295-307.

Dong, C., Loy, C. C. and Tang, X. (2016). Accelerating the Super-Resolution Convolutional Neural Network, *European Conference on Computer Vision (ECCV)*.

Dong, W., Zhang, L., Lukac, R., et al., (2013). Sparse representation based image interpolation with nonlocal autoregressive modeling. *IEEE Trans. Image Process*, **22** (4), 1382-1394.

Dozat, T. (2016). Incorporating nesterov momentum into adam, ICLR Workshop, (1):2013–2016.

Duchi, J., Hazan, E. and Singer, Y., (2011). Adaptive subgradient methods for online learning and stochastic optimization. **12** (7).

Gao, X., Zhang, K. and Tao, D., (2012). Image Super-Resolution With Sparse Neighbor Embedding. *IEEE Trans. Image Process*, **21** (7), 3194-3205.

He, K., Zhang, X., Ren, S., et al. (2016). Deep residual learning for image recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA.

Kaveh, A. and Ezzatollah, S., (2017). Single-image super resolution using evolutionary sparse coding technique’. *IET Image Process.*, **11** (1), 13-21.

Kim, J., Lee, J. K. and Lee, K. M. (2016). Accurate image super-resolution using very deep convolutional networks, *in IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA.

Kingma, D. P. and Ba, J., (2014). Adam: A method for stochastic optimization.

Lai, W. S., Huang, J. B., Ahuja, N., et al., (2018). Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **41** (11), 2599-2613.

Ledig, C., Theis, L., Husz´ar, F., et al. (2017). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, F., Bai, H. and Zhao, Y., (2020). Learning a Deep Dual Attention Network for Video Super-Resolution. *IEEE transactions on image processing*, **29**, 4474-4488.

Lim, B., Son, S., Kim, H., et al. (2017). Enhanced Deep Residual Networks for Single Image Super-Resolution, *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Honolulu, HI.

Mandal, S., Bhavsar, A. and Sao, A. K., (2017). Noise adaptive super-resolution from single image via non-local mean and sparse representation. *Signal Processing*, **132**, 134-149.

Marivani, I., Tsiligianni, E., Cornelis, B., et al., (2020). Multimodal Deep Unfolding for Guided Image Super-Resolution. *in IEEE Transactions on Image Processing*, **29**, 8443-8456.

Ren, C., He, X. and Nguyen, T. Q., (2017). Single image super-resolution via adaptive high dimensional non-local total variation and adaptive geometric feature. *IEEE Trans. Image Process.*, **26** (1), 90-106.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J., (1986). Learning representations by back-propagating errors. *Nature*, **323** (6088), 533-536.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition, *International Conference on Learning Representations 2015 (ICLR 2015)*.

Tai, Y., Yang, J. and Liu, X. (2017). Image super-resolution via deep recursive residual network, *IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA.

Tiantong, G., Hojjat, S. M. and Vishal, M., (2019). Adaptive Transform Domain Image Super-Resolution via Orthogonally Regularized Deep Networks. *IEEE transactions on image processing*, **28** (9), 4685-4700.

Tieleman, T. and Hinton, G., (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, **4** (2), 26-31.

Tsai, R. Y. and Huang, T. S., (1984). Multiframe image restoration and registration. *Advances in Computer Vision and Image Processing. JAI Press Inc.*, **1** (1), 317-339.

Yan, Q., Xu, Y., Yang, X., et al., (2015). Single image superresolution based on gradient profile sharpness. *IEEE Trans. Image Process.*, **24** (10), 3187-3202.

Yang, J., Wright, J., Huang, T., et al. (2008). Image super-resolution as sparse representation of raw image patches, *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA.

Yang, J., Wright, J., Huang, T. S., et al., (2010). Image super-resolution via sparse representation. *IEEE transactions on image processing*, **19** (11), 2861-2873.

Yang, M. C. and Wang, Y. C. F., (2013). A self-learning approach to single image super-reso- lution. *IEEE Trans. Multimed.*, **15** (3), 498-508.

Zeyde R., Elad M., and Protter M., (2010). On single image scale-up using sparse-representations. In Proceedings of the International Conference on Curves and Surfaces.

Zhu, S., Zeng, B., Zeng, L., et al., (2016). Image interpolation based on non-local geometric similarities and directional gradients. *IEEE Trans. Multimed.*, **18** (9), 1707-1719.