A Review and Analysis of Fast Bayesian Uncertainty Estimation and Reduction of Batch Normalized Single Image Super-Resolution Network [7]

Original paper by Aupendu Kar and Prabir Kumar Biswas; Reviewed By Mathieu-Joseph Magri 260928498

Abstract

In this report, we go through a rather extensive analysis of the Fast Bayesian Uncertainty Estimation and Reduction of Batch Normalized Single Image Super-Resolution Network paper [7]. We first and foremost describe the problem setting. We then delve into the theory and mathematics of the methods used to implement the solution presented by the paper. We then move on to a presentation of both the self-generated as well as the results from the original paper. We finish off the report by describing main takeaways about the project and the paper itself.

1. Introduction

In this report, we will analyze and reproduce the Fast Bayesian Uncertainty Estimation and Reduction of Batch Normalized Single Image Super-Resolution Network by Aupendu Kar and Prabir Kumar Biswas [7]. In this paper, the authors explore the task of Single Image Super-Resolution (SISR), a rather common computer vision problem, where the goal is to upscale a singular image to enhance its spatial resolution and general quality. Super-Resolution (SR) is a very useful task in the field of computer vision, as it improves the perceptual quality of low-resolution (LR) images and creates high-resolution (HR) counterparts. Once improved, those images can be used in other computer vision tasks, such as medical imaging, where SISR can improve the quality of MRI images. Moreover, SR can also be of great help in a more real-world setting such as in the surveillance industry where high quality imagery is oftentimes lacking.

As explained above, SR and SISR can be applied to a variety of problem settings and give us valuable solutions to these problems. Indeed, many SR models have already been built, all with the same goal in mind: improving SR outputs. However, it is important to note that many of these SR models have a somewhat black-box nature. In consequence, the black-box nature of these SR models makes it difficult to understand the possible limitations of certain methodologies. This is obviously an unideal situation, as it would be careless to simply blindly follow a model and its outputs without further analysis of the methodology being applied to obtain these outputs.

Figure 1. The effects of a super-resolution model applied to a LR image [11]

To rectify this issue, uncertainty is brought into the fold. Indeed, as we've seen all through the course, uncertainty can be an extremely useful metric given that it can increase the confidence one may have with regards to the results generated by a given model. In light of the benefits of including uncertainty with generated results, both Kar and Biswas from [7] set out to implement an uncertainty measure within the SR imagery process in their paper that we will be analyzing in this report.

1.1. Importance of Uncertainty in Super-Resolution

Uncertainty is invaluable to a model for a plethora of reasons. Firstly, when training or validating a model before test use in the context of SR, a training set and validation set of images must be used. However, these sets of images may not be representative of what we may encounter in other real-world images. These images may contain complexities, textures, noise or even perturbations that may not have been encountered in the training or validation sets used for building the model. These are all factors that may heavily decrease a model's performance, especially in SISR. Given these factors, including uncertainty is of the utmost importance as it not only improves the model's transparency, but also its trustworthiness. In other words, including an uncertainty metric in a SR model would help avoid certain situations where ill-suited reconstructions are followed sense-

lessly. An example of this can be in medical imaging, where an unsuitable reconstruction of a given MRI scan can lead to an incorrect diagnosis, which is something that should definitely be avoided at all costs.

1.2. Short Literature Review

In this section, a brief overview of the existing SR models is given. In addition, we also take a look at a couple of different ways of modeling Bayesian uncertainty in computer vision tasks.

1.3. Super-Resolution Models

Many SR and SISR models have been developed over the years as researchers in the field are continuously aiming to improve performance in this computer vision domain. One of the most popular models used in recent times has been Super-Resolution Convolutional Neural Network (SRCNN) [4]. This model was one of the very first to integrate deep learning aspects in the SR space. This model has 3 phases, each phase representing its own layer, replicating the many aspects of a Convolutional Neural Network (CNN) [11]. The first phase handles patch extraction as well as convolutional filtering while the second phase handles non-linear mapping where convolutional filters add non-linearity to the model and the number of channels of a given image is changed [11]. The third phase of the model, the reconstruction phase, rebuilds the final output: a HR image [11]. Finally, this model is trained on the mean squared error (MSE) loss function [11].

Furthermore, Very Deep Super Resolution (VDSR) [8] is also another SR model, which is a direct improvement on SRCNN. As its name states, VDSR is a deeper network when compared to its predecessor, SRCNN. In VDSR, 3 by 3 convolutional filters are applied in a deeper network [11]. The VDSR network, contrary to the SRCNN network, does not learn a direct image mapping, but rather learns a residual output of a LR input image [11]. Then, both the residual image and the LR image are added together to form the final HR image [8]. Instead of using MSE, VDSR uses gradient clipping during training [11]. Both VDSR and SRCNN use a combination of deep learning and bicupic interpolation to accomplish image super-resolution [11].

In this paper, VDSR is used as the base architecture of their model. Certain adjustments are made to the base VDSR architecture to take into account the additional aspect of uncertainty, which we will get into later on in the paper.

Many other SR models exist such as Fast Super-Resolution Convolutional Neural Networks (FSRCNN) [5], Residual Channel Attention Networks (RCAN) [16], Deep Recursive Convolutional Network (DRCN) [14] and many many others, all with different advantages and disadvantages.

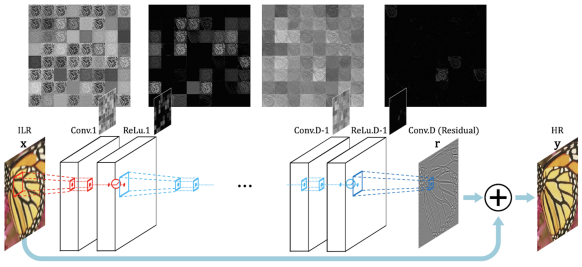

Figure 2. Original VDSR model architecture without modifications [8]

1.4. Modeling Bayesian Uncertainty

A variety of techniques and methods exist to model uncertainty. Oftentimes, in deep learning, Batch-Normalization (BN) as well as Monte-Carlo (MC) samples via dropout are applied to replicate an estimate of a posterior distribution [7]. This estimate can be used to extract Bayesian stochastic parameters like the model's mean and variance. These parameters can then help us calculate and measure the uncertainty of the given model. In other words, both MC samples and BN can be used to approximate Bayesian behaviour within a model [7]. The previously stated methods have been successfully applied to a variety of computer vision tasks such as image classification and segmentation as well as camera relocalization [7]. With regards to [7], Batch-Normalization is used to approximate Bayesian behaviour, but like with the VDSR architecture, certain modifications are made to BN which we will get into later.

1.5. Key Ideas Introduced in [7]

In [7], the authors state three key aspects that separate their method from others. Firstly, Kar and Biswas put forward a more efficient implementation of Monte-Carlo Batch-Normalization (MCBN) uncertainty, which allows them to sample at a quicker rate. Secondly, they assume to be the first to approximate uncertainty in an image reconstruction setting. Finally, Kar and Biswas also state that they establish an "uncertainty reduction method" which guards against adversarial perturbations on their SISR model and helps with out-of-distribution (OOD) LR images with and without noise. We will now look into all these key ideas in much more depth in the *Theory and Methods* section

2. Theory and Methods

In this section, we will be going through a three step process to fully comprehend the methods applied by both Kar and Biswas in [7]. To start off, a Bayesian inference summary will be presented based on Kar and Biswas' work,

followed by a description of the SISR model architecture they used. Finally, this section finishes off with a review of the three novel algorithms introduced by Kar and Biswas in their paper.

2.1. Bayesian Inference Summary

As we know, Bayesian inference is an incredibly useful technique in statistics that allows us to approximate distributions. In [7], the main goal is to estimate the following function:

$$F_W(I) : \tilde{I}_{LR} \rightarrow \tilde{I}_{HR} \quad from \;\; a \;\; training \;\; set \;\; D = \{\tilde{I}_{LR}, \tilde{I}_{HR}\} \quad (1)$$

where $F_W(I)$ represents a probabilistic function that maps a low-resolution image $\tilde{I}_{LR}$ to the most appropriate HR reconstruction image $\tilde{I}_{HR}$. Here, $\tilde{I}_{LR} = \{\tilde{I}_{LR_1}, \ldots, \tilde{I}_{LRn}\}$ is the LR image set and $\tilde{I}_{HR} = \{\tilde{I}_{HR_1}, \ldots, \tilde{I}_{HRn}\}$ is the associate set of HR images. As explained in [7], the aim here is to have a good enough approximation of $F_W(I)$ to be able to map a given LR test image I'_{LR} to a HR reconstruction image I'_{HR} from a test set $D' = \{I'_{LR}, I'_{HR}\}$, where $I'_{LR} = \{I'_{LR_1}, \ldots, I'_{LRn}\}$ is the LR test image set and $I'_{HR} = \{I'_{HR_1}, \ldots, I'_{HRn}\}$ is the associate set of reconstructed HR images generated by the model from I'_{LR}.

To be able to properly approximate the probabilistic function $F_W(I)$, Kar and Biswas put forward the following approximation of $F_W(I)$:

$$F_W(I) \approx p(I'_{HR}|I'_{LR}, D) = \int (I'_{HR}|I'_{LR}, W)p(W|D)dW \quad (2)$$

where W represents the weight parameters of the function $F_W(I)$. As we've seen time and time again over the length of the course, the above integral is intractable and must be approximated further by minimizing the Kullback–Leibler (KL) divergence $KL(q_\theta(w) \;||\; p(W|D))$, which is equivalent to maximizing the likelihood function. By applying KL divergence, we get from [7]:

$$F_W(I) \approx KL(q_\theta(w) \;||\; p(W|D)) = q(I'_{HR}|I'_{LR}, D) = \int p(I'_{HR}|I'_{LR}, w)q_\theta(w)dw \quad (3)$$

where $q_\theta(w)$, the approximate joint distribution of weights and stochastic parameters, is introduced. In this case, $\theta = \{W_L, \gamma_L, \beta_L\}$ and $w = \{\mu_L, \sigma_L^2\}$. With respect to $\{W_L, \gamma_L, \beta_L\}$, these are all learnable parameters, where γ_L is the scale parameter for BN and $beta_L$ is the shift parameter for BN. On the other hand, $\{\mu_L, \sigma_L^2\}$ are both the stochastic mean and variance.

2.2. Model Architecture

As mentioned previously in the *Short Literature Review* section, the authors of [7] use a base VDSR architecture where certain modifications are made to account for uncertainty calculations and analysis. As we know by now, VDSR is a deep network which applies 3 by 3 convolutional filters to be able to form a residual image from an input LR image. Then, this residual image is added to the corresponding LR image to reconstruct the final SR image. In Kar and Biswas' paper, they make one crucial modification to this base architecture: they implement batch normalization layers after each convolutional layer in the model, except for both the first and last convolutional layers of the architecture. By doing so, Kar and Biswas are able to sample meaningful uncertainty measures with stochastic parameters $\{\mu_L, \sigma_L^2\}$ from the model's estimated posterior distribution.

The above modification in the VDSR framework allows the generation of various reconstructed HR images, produced from randomly sampled training batches that provide various values of $\{\mu_L, \sigma_L^2\}$. The authors of [7] further state that the mean of the HR images is used to create the final reconstruction of the SR image, while the standard deviation of the samples is used to create the uncertainty map of the corresponding SR image.

2.3. Algorithms Review

In this section, we will be going through the three main algorithms provided by Kar and Biswas from their paper. The first is a more efficient MC sample algorithm, the second algorithm is the Monte-Carlo Batch Normalization (MCBN) algorithm the authors used in their paper, while the third is a "Uncertainty Reduction" algorithm. All algorithms are sourced from [7], and we will be going over the key steps of each one.

2.3.1 Algorithm 1: Faster MC Sample Generation [7]

Regular MC sampling can be quite a slow and inefficient process. As stated in [7], "the main drawback of standard MCBN uncertainty estimation is that we need to process test images with different random batches to generate MC samples". This directly affects computation time, making it increase exponentially. Furthermore, in terms of qualitative effects, this MC sample process can also create patchy behaviour in finalized reconstructions.

Given those issues, Kar and Biswas propose a different approach to increase the efficiency of the MC sampling process. Indeed, after the model has been trained, parameters $\{\mu_L, \sigma_L^2\}$ are approximated using distinct random batches. The parameters are provided by "each BN layer for [each] batch" [7] and this allows us to have multiple batches with a diverse set of mean and variance pairs. These mean and

variance pairs are then used to sample MC samples at a one to one ratio during the testing phase of implementation. In other words, sampled training stochastic parameter sets are used during testing.

Algorithm 1: Layer-wise batch mean and variance estimation for single shot MC samples generation

Input: training image set I
Output: layer-wise batch mean and variance set $\widehat{w}_L^T = \{\mu_L^T, {\sigma^2}_L^T\}$ of trained network. Where L is layer no and T is maximum number of MC samples required.
Training SR Network:
for total iterations **do**
  forall batches B of image set I **do**
    train();
  end
end
$\widehat{w}_L^T$ ***estimation:***
for T batches of B **do**
  forward pass;
  estimate $\widehat{w}_L^T$;
end

Figure 3. Algorithm 1 [7]

2.3.2 Algorithm 2: MCBN [7]

Continuing on from where we left off with regards to Algorithm 1, the generated MC samples give us a set of stochastic parameters. These parameters are then used during the testing phase, where "we concatenate the same test image based on the required number of MC samples", as seen in Algorithm 2. Furthermore, these images, in the BN phase, are then normalized using the parameters generated by Algorithm 1. In consequence, we are able to reconstruct a plethora of HR images, based upon the "posterior distribution learned from the dataset" [7].

Essentially, Algorithm 2 uses Algorithm 1's generated stochastic parameters to enhance a LR image into a variety of HR images. These HR images are then averaged out and form a single HR image, our final output. Then the variance between all HR images is used as an uncertainty measure to create uncertainty maps for the generated SR/HR images.

2.3.3 Algorithm 3: Uncertainty Reduction [7]

Algorithm 3 focuses heavily on uncertainty reduction. Uncertainty reduction is extremely useful in this context as it can prevent failure when the model is attempting to handle

Algorithm 2: Our MCBN algorithm for SISR

Input: test image I_{LR}, number of MC samples N, batch mean and variance of layer L $\widehat{w}_L^N = \{\mu_L^N, {\sigma^2}_L^N\}$
Output: image mean prediction $\hat{I}_{SR}$, predictive image uncertainty σ
concat I_{LR} for T times (${I^T}_{LR}$);
${I^N}_{SR} = F_W({I^N}_{LR}, \widehat{w}_L^N)$;
$\hat{I}_{SR} = mean({I^N}_{SR})$;
$\hat{\sigma} = std({I^N}_{SR})$;

Figure 4. Algorithm 2 [7]

OOD images. Uncertainty reduction can also be extremely useful when defending against noise and perturbations. In light of this, the authors propose Algorithm 3, an "inverse perturbation mechanism", divided into two key sections: determination of average gradient directions and perturbation level selection [7].

In the gradient direction calculation part of Algorithm 3, two mean-variance pairs are selected arbitrarily and used on a LR test image to form two different SR/HR reconstructions. Pixel-wise mean squared loss is then minimized between the two HR images. Finally, the "negative sign gradient at each pixel [...] is the update direction to reduce uncertainty" [7].

In the perturbation level selection section of Algorithm 3, we purposely add noise and perturbations to a given LR test image. Indeed, the perturbation parameter β is tuned on a case by case basis. As a matter of fact, this β value increases every iteration as long as uncertainty is still decreasing. Once uncertainty starts to rise again, perturbation increase is stopped and β remains fixed.

3. Results

In this section of the report, we will be taking a look at the experimental environment used in [7] and comparing it to my own. We will also take a look at the various datasets used in Kar and Biswas' paper and the ones used in my experiments. Finally, we will then take a look at a mix of quantitative and qualitative results from both [7] and my experimentation.

3.1. Datasets

Kar and Biswas used a total of 6 datasets: DIV2K [1,12], Set5 [3], Set14 [15], BSD100 [9], Urban100 [6], Manga109 [10]. DIV2K was used to train the model, while all other datasets were used for testing the model's capabilities and performance.

On my end, I used a total of 3 datasets: DIV2K, Set5,

Algorithm 3: Uncertainty reduction for an image

Input: test image I_{LR}, number of MC samples N, batch mean and variance of layer L $\widehat{w}_L^N = \{\mu_L^N, {\sigma^2}_L^N\}$
Output: perturbed test image I'_{LR}
for total updates on I_{LR} **do**
  Gradient directions for I_{LR}***:***
  for total iterations, T **do**
    Random Select t_1 and t_2 ;
    $I^{t_1}{}_{SR} = F_W(I_{LR}, \widehat{w}_L^{t_1})$;
    $I^{t_2}{}_{SR} = F_W(I_{LR}, \widehat{w}_L^{t_2})$;
    $\mathfrak{L} = \frac{1}{CWH}\sum(I^{t_1}{}_{SR} - I^{t_1}{}_{SR})^2$;
    $\mathfrak{N} = \mathfrak{N} + sign(\nabla\mathfrak{L})$
  end
  Perturbation Level Selection
  for different levels of perturbation, β **do**
    $I'_{LR} = I_{LR} - \beta.\frac{\mathfrak{N}}{T}$;
    Calculate Uncertainty for I'_{LR}, $U(\beta)$;
    if $U(\beta) > U(\beta - 1)$ **then**
      $I'_{LR} = I_{LR} - (\beta - 1).\frac{\mathfrak{N}}{T}$;
      break ;
    end
  end
end

Figure 5. Algorithm 3 [7]

and BSD100. DIV2K was used to train the model while Set5 and BSD100 were used for experimentation.

All datasets were publicly available except Manga109. The DIV2K dataset is pre-divided into a training set of 800 HR images with their LR counterparts, while the validation and test sets contain 100 images each. A variety of downscalling factors were also available (2, 3 and 4). The Set5 dataset is a very small dataset of a total of 5 images while BSD100 contains a total of 100 images.

3.2. Environment

In Kar and Biswas' case, they trained on the entire DIV2K dataset after having performed a variety of preprocessing tasks. This information can be found in [7]. On my end, I tried to follow everything as closely as possible. However, two of the major differences between my training process and the one found in [7] were the amount of DIV2K images I used to train on my implementation as well as the amount of epochs the model was able to get through. In Kar and Biswas' case, they used the full DIV2K dataset and ran the model for 1000 epochs, while I, due to computational and memory constraints, ran half of the DIV2K dataset on my implementation for 250 epochs. The GPU I used was the one provided by the free plan of Google Colab, while Kar and Biswas do not mention any GPU.

When evaluating the model's quantitative performance, the authors of [7] used Peak signal-to-noise ratio [13] (PSNR). PSNR measures the reconstruction quality for images or videos and the higher the PSNR value, the better.

3.3. Experimental Results and Comparisons

The first results we will be looking at is the time saved by using the faster implementation of MC sample generation. As we can see from figure 6, whether we sample for 5 MC samples, 10 or 15, the time saved is absolutely non-negligible. For 5 samples, 14.28 less execution time is needed, while for 10 it is 33.48 less execution time and for 15 it is 52.67 when compared to the baseline of 5 MC samples using the new MC sampling approach. These results are provided by the authors of the paper and are not mine. I did not try to recreate this experiment either, as it did not involve uncertainty in any way shape or form.

MC Samples	MCBN	Our approach
5	14.28	1.0
10	33.48	1.97
15	52.67	2.96

Figure 6. Monte-Carlo Sample generation on the Set14 dataset. Results and table directly pulled from [7]

Moving on, in Kar and Biswas' paper, they include a variety of experiments which explore the relationship of uncertainty with many variables such as noise, perturbation levels, scaling factors and the PSNR value of a given reconstructed image. I decided to sample 20 random images from my implementation of the Bayesian VDSR model in [7] and use their associated PSNR values and uncertainty values and see if my results follow the same behaviour as [7]. As we can see, as a PSNR value increases, uncertainty decreases, which is logical, given that PSNR is a measure of reconstruction quality. Thus, the higher the PSNR, the more accurate our reconstruction and the more sure we are of our results.

When looking at the quantitative results of my implementation of Bayesian VDSR in comparison with a self-implementation of standard VDSR and the results presented in [7], we see that I was unable to obtain the same sort of success. This can be due to a variety of factors that we will discuss in the *Discussion* section of the report.

When looking at the qualitative results of my implementation, we can notice much of the same. The uncertainty map I developed, which is in black and white, seems to perform much more poorly than the Bayesian VDSR implementation directly from both the paper and the open-source code [2] the authors of [7] left available. In all implementations, it is possible to notice high values of uncer-

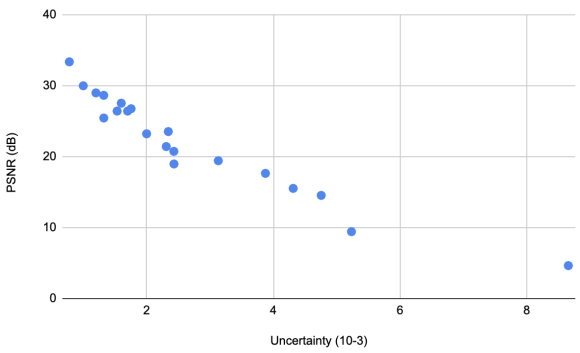

Figure 7. Relationship between PSNR and uncertainty. As we can see, As the PSNR increases, uncertainty decreases.

	Super-Resolution Models Applied to a Scale Factor of 2		
	BayesianVDSR by Kar and Biswas	Standard VDSR (self-implementation)	BayesianVDSR (self-implementation)
PSNR (dB)	32.02	27.63	25.43
Uncertainty	0.0011	NA	0.00298

Figure 8. Model PSNR and uncertainty performance. The results from the first column are directly taken from [7], while the last two columns are self-implemented.

tainty where we encounter edges, which is logical given that edges are areas in an image that are subject to high gradient changes. However, in my implementation, there seems to be some sort of noisy effect in the map that I did not manage to fix unfortunately, which causes the map to be filled with white spots. This is a repeated issue in a variety of test images. (Note: These qualitative results will be included at the end of the report)

Finally, the authors in the paper tested their defense mechanism (Algorithm 3) on different types of noises, perturbation attack and noise levels, and scaling factors. They concluded that their defense mechanism does a good job at defending against all these different factors, and include quantitative results backing up their claims. I did not reproduce this aspect of the paper given that I am unfamiliar with this sort of experimentation and did not have enough time to come up with an experimental plan, but their results are present in [7].

4. Discussion

In this section, we will state the main takeaways from conducting this project. I will also suggest possible avenues for improvement and future work that can be done.

4.1. Strengths

The strengths of the developed method in [7] are plenty. It seems to be, first and foremost, a very efficient and viable alternative to SISR. Furthermore, this approach seems to work quite well with out-of-distribution samples. This is extremely useful as this infers that the Bayesian VDSR model developed by Kar and Biswas is able to perform on images that are not always part of the model's posterior distribution. Furthermore, although I did not investigate this aspect of the method in this report, the paper [7] states very positive results when defending against adversarial attacks. This is also a very valuable characteristic for your model to have, given that it is able to perform, not only on OOD images, but also images that have been perturbed.

4.2. Assumptions

The model also makes a few assumptions that must be taken into account as well. First of all, the model assumes, in a sense, that the BN layers can be effectively leveraged to generate MC samples that are stochastic enough to reflect an approximate distribution. Additionally, this model also assumes that our stochastic parameters, our mean and variance, are an accurate measure of uncertainty. Although not specified in the paper [7], I believe this was done to reflect the epistemic uncertainty of the model, but in my implementation, this may have done more harm than good given that I trained on half the dataset used in the original paper. Moreover, the uncertainty maps generated from the various variance samples are assumed to be indicative of regions of where the model truly struggles. However, the paper doesn't really specify why we are allowed to make this sort of assumption. In my opinion, given that we are sampling batch statistics from an image training set that we use to train the model, we assume the variance, and thus the uncertainty, from our batches are representative of our entire dataset. Instead, we could have perhaps explored the possibility of including the HR images from the training set and use them as a basis of comparison to create our uncertainty maps in a more accurate manner.

4.3. Limitations

With regards to the limitations that are present in this approach, we really quite heavily on BN. Given this fact, methods that completely ignore BN or rely on different normalization techniques may not be able to benefit from this method. Additionally, the approach in [7] is very domain specific to both SISR and CNN architecture. Therefore, this approach may not generalize well to other computer vision tasks or to other architectures. Lastly, as mentioned in the *Assumptions* section, in cases where BN does not capture full stochasticity, the model will not be able to reflect true uncertainty either.

4.4. Future Potential Experiments

In the future, it would be interesting to see this technique applied to different sorts of CNN based super-resolution

models. This would allow us to see the potential generalizability of the methods developed by Kar and Biswas as well as seeing its performance in different architectures and settings. In addition, in this report of Kar and Biswas' paper, the focus was put on both algorithm 1 and algorithm 2, as those were the key algorithms related to uncertainty estimation in the VDSR SR model. Algorithm 3, although very interesting, was not reproduced or analyzed further and was taken as is when implementing the model in practice. Ideally, I would have had both the time and computational power to be able to do a full implementation of all algorithms, however this can be left for future work.

4.5. Critique of Paper and Personal Uncertainties

All in all, the paper was quite interesting to read through and reproduce. However, there were still a few issues present in the paper. First of all, the authors never include an updated visual representation of the modified "Bayesian" VDSR model they implement. This made it difficult for me to fully comprehend the model's architecture. Furthermore, the code repository present at [2] wasn't a great help given that the code is not accompanied by great documentation. This made things much harder to understand and may be one of the reasons that my self-implemented results do not perform as well as the results in the paper. Additionally, the paper didn't always explain algorithms clearly which lead to educated guess work at times. Moreover, generating the uncertainty maps proved to be more difficult than I first thought but that may be do in large part to my inexperience in developing models and solutions in this field of study.

Furthermore, when evaluating the quality of their reconstructed images, they only use PSNR, but perhaps the model can be extended to be evaluated on more metrics such as structural similarity index measure (SSIM).

5. Conclusion

All in all, this project was enjoyable, as it allowed me to apply certain aspects of the theory we learned in class to an actual practical problem. This project also allowed me to explore a subset of computer vision I was not familiar with and allowed me to grow my knowledge within the field.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 126–135, 2017. 4

[2] aupendu. sr-uncertainty. https://github.com/aupendu/sr-uncertainty, 2024. [Online; accessed 29-November-2024]. 5, 7

[3] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012. 4

[4] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015. 2

[5] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 391–407. Springer, 2016. 2

[6] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5197–5206, 2015. 4

[7] Aupendu Kar and Prabir Kumar Biswas. Fast bayesian uncertainty estimation and reduction of batch normalized single image super-resolution network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4957–4966, 2021. 1, 2, 3, 4, 5, 6

[8] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016. 2

[9] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001. 4

[10] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia tools and applications*, 76:21811–21838, 2017. 4

[11] Shaoni Mukherjee. Image super-resolution: A comprehensive review — DigitalOcean, the simplest cloud that scales with you. https://www.digitalocean.com/community/tutorials/image-super-resolution, 2024. [Online; accessed 27-November-2024]. 1, 2

[12] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, Bee Lim, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. 4

[13] Wikipedia contributors. Peak signal-to-noise ratio — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Peak_signal-to-noise_ratio&oldid=1247121104, 2024. [Online; accessed 5-December-2024]. 5

[14] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, and Qingmin Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121, 2019. 2

[15] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces: 7th International Conference, Avignon, France,*

June 24-30, 2010, Revised Selected Papers 7, pages 711–730. Springer, 2012. 4

[16] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 286–301, 2018. 2

Appendix

Figure 1: LR and HR pair of images from the BSD100 dataset

Figure 2: Uncertainty Map Generated by Kar and Biswas

Figure 3: Uncertainty Map Generated by running the code from the publicly available GitHub Repository

Figure 4: Uncertainty Map Generated by my Implementation

Note: The map was changed to white instead of red to easily differentiate it from figure 2 and 3. Furthermore, white spots may be easier to spot in a dark setting.