Image Super-Resolution with pixel dependency

Ayushi Bansal Bhargav Sundararajan Mridula Gupta Sarmishta Burujupalli
Shivani Teegala

Department of Computer Science, University of California, Davis

Abstract

Super resolution is the process of up-scaling and improving the details of a given low-resolution image. Though there exists prior work in up-scaling a low-resolution image, most of the prior work has majorly dealt with increasing the quality by 2 folds or so. In our project, we dealt with scenarios where no visual data can be inferred by naked eye. As we like to call it, "Generating something out of nothing". Our approach is majorly based on ***Pixel Recursive Super Resolution[1]****, wherein we have replaced the regular batch normalization layer with a* ***SPatially Adaptive DEnormalization(SPADE)*** *[13] layer. Our intuition, here is that, using a spatially adaptive layer would make the model learn the spatial information, that is, the position and structure of different parts of the image better. It could be seen from the results that, the initial structure of the images has been learned faster by this approach.*

1. Introduction

Image generation has come a long way in computer vision. There exists very effective and efficient models which could generate life-like replicas of any given instance. Most of them are dependent on high-resolution images to train or pre-train the model. But many applications require zooming of a specific area of interest in the image wherein high resolution imagery is essential but not available, e.g. surveillance and satellite imaging applications. In most of these applications, the zoomed in picture quality is always degraded and close to uninterpretable, let alone have high resolution. This poses two challenges - One, Learning the visual structure of an image from low-resolutions like 8x8, where even naked eye cannot decipher any details. Two, synthesize plausible images using the learned details, which could map to numerous high resolution images. Thus, the predictive challenge shifts from recovering details to synthesizing plausible novel details.

Super resolution is the process through which low-resolution images are converted to high-resolution images without losing the visual details. Most of the known applications of super resolution deals with generating high resolution images, from a 2x or 4x zoomed images. But in this project we plan to synthesizes realistic details into images while enhancing their resolution. The images resolutions are close to 8x8, where the image doesn't hold any high level visual details. We use a probabilistic deep neural network based on the conditional generative models to implement super resolution.

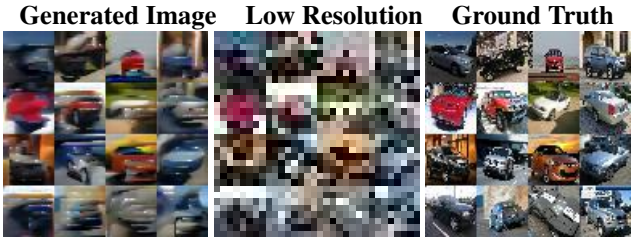

Figure 1. *Sample representation of image generation through low resolution Images*

The project has initially started as an implementation of Pixel Recursive Super Resolution with an possible option to extend it to complex datasets. But as we read on the model and started implementing it, using a spatially adaptive normalization layer instead of a regular batch normalization layer in the CNN block sounded like an better alternative for spatial learning of the features. For example the dataset used in the paper [1], CelebA [9] for faces, the spatial alignment of the eyes, nose, chin .,etc plays an important part in the image generation. Thus integrating a normalization approach like SPADE which is spatially adaptive should learn the visual structure of the image more effectively.

In these lines, our model consists of three parts. One, a Pixel CNN, Two, a conditional CNN, Three, a spatially adaptive normalization layer. The first two parts are as presented in the pixel recursive super resolution by Dahl et.al[1]. For SPADE layer, the implementation has been a direct replica of Park et.al [13], and has been replaced in the CNN blocks in-place of the regular normalization layer. Section 3 further elaborates on the architectural details of the model and the changes adapted.

2. Related Work

There is a very long list of image super-resolution papers and a detailed review of the topic is out of the scope of this section. Here, we focus on related work based on Convolutional Neural Networks (CNNs). Some of them include pre-upsampling (SR- CNN[3]) Post-upsampling (sub-pixel convolution [14]) and Progressive-upsampling [7] where CNNs are used to achieve super resolution.

The standard approach to super-resolution using CNNs is to use a fully supervised approach where a low-resolution (LR) image is processed by a network comprising convolutional and upsampling layers in order to produce a high-resolution (HR) image which is then matched against the original HR image using an appropriate loss function. This is like a paired setting as it uses pairs of LR and corresponding HR images for training.

Dong et al.[2] [4] proposed SR- CNN to learn an end-to-end mapping from interpolated LR images to HR images. Specifically, the LR images are first upsampled to coarse HR images with the desired size using traditional methods (e.g. bicubic interpolation), then deep CNNs are applied on these images for reconstructing high- quality details. The advantage of this is that the difficult upsampling task has been done by predefined traditional algorithms, and deep CNNs only need to refine the coarse images, which significantly reduces the learning difficulty.

In post-upsampling, the low resolution images are passed to the CNNs, upsampling is performed in the last layer using learnable upsampling layers. The advantage of this method is that feature extraction is performed in the lower dimensional space (before upsampling) and hence the computational complexity is reduced. Furthermore, by using a learnable upsampling layer, the model can be trained end-to-end. In addition, these models can take interpolated images with arbitrary size and scaling factors as input, and give refined results. However the downside is that the predefined upsampling methods may amplify noise and cause blurring.[5][14]

GANs is another technique that has been increasingly used for several image based applications including Super Resolution. Parket. al [12] used a feature-level discriminator to capture more meaningful potential attributes of real High Resolution images. GAN's are the most commonly used deep learning architecture for training deep learning-based SR models.[10][8] The architecture of GAN is based on unsupervised learning.

Super-resolution GAN applies a deep network in combination with an adversary network to produce higher resolution images. During the training, A high-resolution image (HR) is downsampled to a low-resolution image (LR). A GAN generator upsamples LR images to super-resolution images (SR). They use a discriminator to distinguish the HR images and backpropagate the GAN loss to train the discriminator and the generator. One downside is that the training process of GANs is a bit difficult and unstable. Moreover, for super resolution task, other pixel-independent conditional model often results in averaging different details and hence blurry edges.

3. Technical Details

To solve the problem of Super Resolution, we have employed a probabilistic approach borrowed from the work of Dahl et al.[1]

3.1. Probabilistic super resolution

As Super Resolution is a one-many mapping problem, we formulate this task as a conditional dependency problem. Therefore, given a low resolution image, the corresponding high resolution image with the highest probability conditioned by the low resolution image is generated. The model should successfully learn the statistical dependencies between a low resolution image and a corresponding high resolution image in order to do this. Mathematically, this can be represented as (1) where x and y denote a low resolution and a high resolution image, y* represents a ground-truth high resolution image and θ represents the learnable parameters of the model.

$$O(\theta|D) = \sum_{x,y^* \in D} \log p(y^*|x) \tag{1}$$

In order to generate each pixel in the high-resolution image, we predict the probability (the logits) of each pixel for all possible 256 values. This can be done by varying the output dimension of the neural network to hold 256 values for each pixel and use a multinomial distribution as the predictive model for each pixel. This can be represented mathematically using (2) where k = 256.

$$p(y_i = k|x) = \frac{\exp\{C_{ik}(x)\}}{\sum_{v=1}^{K} \exp\{C_{iv}(x)\}} \tag{2}$$

3.2. Pixel recursive super resolution

The pixels in the high-resolution image can be generated by assuming they are conditionally independent of all the other generated pixels. This assumption can lead to very blurry images which lack finer details that is necessary in a high-resolution image. Therefore, we employ a method that generates pixel in a conditionally dependent manner to previously generated pixels.

One approach to this problem is to define the conditional distribution of the output pixels jointly as a multivariate Gaussian mixture. This will result in a highly computationally expensive model which needs to learn the statistically

dependency of each pixel with all the other pixels. Alternatively, we have decided to generate the pixel one after another and using the generated pixels as conditioning to generate the next pixels in the high-resolution image. This can be represented using (3).

$$\log p(y|x) = \sum_{i=1}^{M} \log p(y_i|x, y_{<i}) \tag{3}$$

PixelCNN is a stochastic model that does exactly this by modelling (3). The key idea in a PixelCNN is the use of cleverly masked convolutions that filters out the information of a future pixel while determining the logits of a current pixel. The reason this is advantageous is because, we can calculate the logits of all the pixels simultaneously without the need for generating them one by one for training purposes. Therefore, the training speed is increased multifold while still capturing the linear dependency of the pixels in the high-resolution image.

3.3. Model Architecture

The Model Architecture is depicted in the Figure 2. The two main components of the model are the Prior Network and the Conditioning Network. Both these components output a 256 sized vector for each pixel in the high-resolution image representing the logits for each possible pixel value. A late fusion mechanism is applied by combining the outputs of both these components to generate the final high-resolution image.

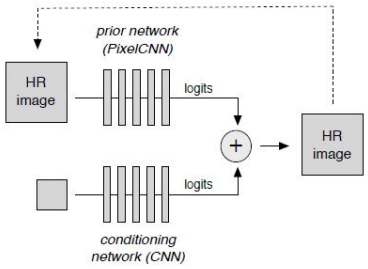

Figure 2. Baseline Model

The Prior Network is the PixelCNN model which is responsible for capturing the serial dependencies between each pixel in the high-resolution image. It takes in a high-resolution image and a series of gated masked convolutions are performed to generate the logits.

The Conditioning Network is responsible for capturing the global structure of the low-resolution image. A generic convolutional neural network used in previous implementations of super resolution is used in this case. The output dimensions of both the Prior and the Conditioning networks are the same.

As the Conditioning Network needs to capture the spatial dependency between each pixel in the low-resolution image, we have modified the residual block of the network to perform Spatially-Adaptive Denormalization (SPADE) in order to improve the Conditioning Network's ability to capture spatial information.

3.4. SPADE

As an extension in the existing model, we added a SPADE (Spatially Adaptive Denormalization)[13] Resnet Block after each convolution layer in the conditioning network of our baseline architecture.

SPADE block takes semantic map of image as input layout for modulating the activations in normalization layers through a spatially adaptive, learned transformation. In our case we are using low resolution image as the input to the SPADE block. As visible from fig. SPADE Resnet block consists of combination of SPADE block, Relu and Convolution layer, of which SPADE block is the key part as it takes the semantic mask as input to learn the affine parameters of sync batch normalization.

Finally, we employ a Softmax Cross-Entropy loss between the generated high-resolution image and the ground truth image in order to train the model.

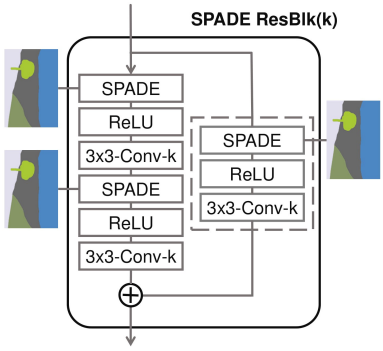

Figure 3. SPADE ResBlock

The main motivation behind replacing the normal batch normalization in conditioning network with SPADE, is that SPADE captures the semantic information of the input image (low resolution). This allows the model to better learn about the global structure of low resolution image. Same size LR image is fed to the SPADE Resnet block with dimension matching the dimension of the output image of previous convolution layer. This ensures that the effective predicted vector of logit values of of each convolution layer+ SPADE block, captures the semantic of original LR image and also the learned features of previous convolution layer with updated size of image (image enlarged at each convolution step in conditioning network).

4. Experimentation Setup

Figure 4. Sample Data sets.

4.1. Datasets

We have tried to collect the data sets pertaining to surveillance application. On those lines, below are the four datasets we have worked on.

- *Celeb Faces Dataset* - 200,000 images
- *Cars Dataset* - 16,200 images
- *House Numbers Dataset*- 200,000 images
- *License Plate Dataset* - 100,000 images

Figure 4. shows few samples of the datasets been used. The datasets are selected to have diversity in complexity from simple number images to visually detailed images like faces and cars. Out of the four datasets, we could only fully test on the first two datasets of cars and faces owing to limited GPU power. The later data sets have very limited epoch runs.

The collected data sets are first preprocessed to 256x256 images. Which are then resized to 32x32 Images. The faces data set has been used in the paper by Dahl et al.[5] and thus we have used it as an performance metric to test our SPADE extension model.

4.2. Implementation Details

We initially implemented the model using Tensorflow library for various datasets. To recreate the results of Dahl et al.[1], we ran the model for CelebA dataset and obtained similar results as in original paper. Then, we used the same base model with some data pre-processing changes to get the result for other datasets which require semantic knowledge of the background, as we are incorporating pixel dependent features of the images for training.

Then, we added SPADE Resnet block to conditioning network of the baseline-network, along with other changes to support the effective usage of SPADE. Normal batch normalization is exchanged with sync batch norm of SPADE. For each resnet block of SPADE, low resolution image of dimension matching with the dimension of the input image from previous convolution layer of the conditioning network is used to learn the semantic features of the image. This matching is done by up-scaling the original low resolution image. Also, each convolution layer within SPADE resnet block is similar with kernel size=4x4 and stride=2 and appropriate padding depending on the input image. Spectral normalization is used in each SPADE resblock.

Table below lists all value of hyperparameters used for training. They are tuned to get the best results for all datasets.

Hyperparameter	Value
Learning Rate	4e-4
Batch size	32
Optimiser	RMSProp (decay=0.95, momentum=0.9, epsilon=1e-8)

Table 1. Hyperparameters used for training the model

Results are pretty convincing, but the training of the model is very slow, even on google cloud GPUs. This can be attributed to the tensorflow library usage. Thus, motivating us to convert the whole code in Pytorch implementation.

4.3. Evaluation metrics

For our quantitative evaluations of the model, we use two of the widely used metrics - *Peak Signal-to-Noise Ratio (PSNR)* and *Structural Similarity Index (SSIM)*. Both are used for evaluating the quality of the generated image from the trained model with respect to the ground truth image.

PSNR computes the peak signal to noise ratio. It is used for reconstructed image quality comparison, where noise is the error introduced by compression in the original signal image. PSNR is an approximation to human perception of reconstruction quality. SSIM is the opposite of Mean Square Error and look for similarities within pixels; i.e. if the pixels in the two images line up and or have similar pixel density values. Higher values in both represents higher similarity between the compared images.

As crucial for any generative model we have done the perceptual evaluation of the samples, among our self. This helped us in checking and evaluating the model performance at every stage.

5. Results

5.1. Cars Dataset

For cars dataset [6], We could run the model for 26000 epochs. The samples have been collected at every 1000th epoch to track the progress. Figures 5-8 show couple of the samples that have been collected. We have observed that the

Figure 5. Epoch: 2000

Figure 6. Epoch: 8000

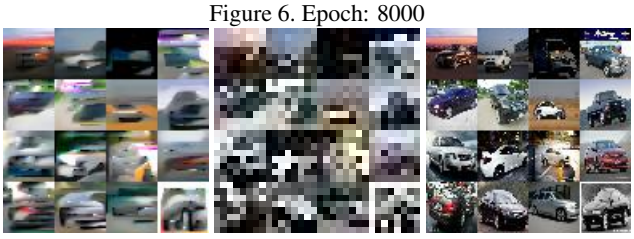

Figure 7. Epoch: 16000

Figure 8. Epoch: 24000

Generated images on Cars Dataset

model has partially learned the structure of the car around the 8000 epoch, as seen in Figure 6. And by the end of the 24000 epoch, Figure 8, most of the color of the cars have been learned and generated well by the model. Though the synthesized image is not sharp enough yet, we could see from table 1 the SSIM value fluctuating around 0.4 and we believe more training and computation power is could lead to generation more plausible images.

PSNR and SSIM metric has been calculated for every sample. The table 1 below shows the metrics for the cars dataset. We have used the generated image and the ground truth high resolution image for the metric comparisions.

5.2. Celeb Dataset

Coming to the celeb faces dataset, we have used the dataset to run on both the models. The original Dahl implementation and our own implementation of integrated spade model. Figures 9-13 show the samples generated by the integrated spade model Whereas Figure 14-15 have been attached to compare the Non-SPADE vs SPADE Extension model. We

Iterations	PSNR	SSIM
4000	28.082	0.272
8000	28.374	0.440
12000	28.521	0.417
16000	28.604	0.414
20000	28.506	0.391
24000	28.337	0.377

Table 2. Quantitative results for Cars Dataset

could run the model for 21000 epochs.

As can be seen from Figure.14, In the comparison between SPADE extension and the regular pixel recursive model, SPADE model has learned the structure of the face faster than the model proposed by Dahl et.al. Thus proves the intuition of using a spatially adaptive normalization layer. But in contrast to that, the consecutive runs have not shown on par results. And the model has not learned any face details even till the end of 20000 epoch. Where as by the end of 20000 epoch, Figure 15, Dahl.et.al model could generate plausible images.

The evaluation metric has been same as described in cars dataset. With the samples generated so far, we are hard pressed to make any comments on the final end performance of the SPADE extension. We could see that the model has learned the features faster, but as can be seen from Figure 13, the samples have degraded.

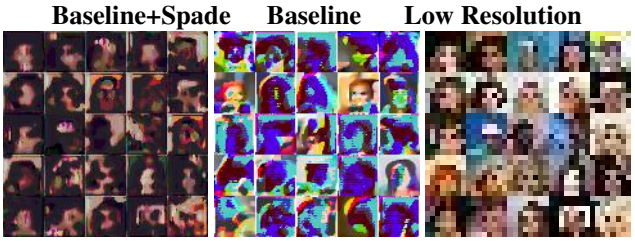

Figure 14. Epoch: 2000

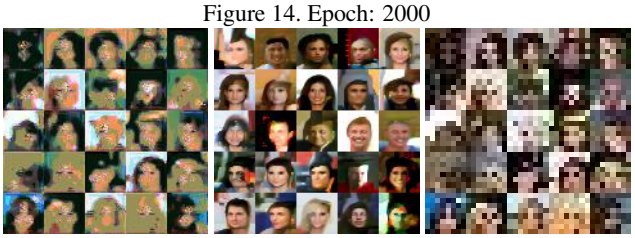

Figure 15. Epoch: 19000

Comparing generated images of SPADE-integrated and baseline model

5.3. Additional Datasets

Figures 16-17 shows the samples generated for the street view house numbers dataset [11], that has been extracted from stanford database. Though initially we thought the model should work well for the numbers, since there are only few

Figure 9. Epoch: 2000

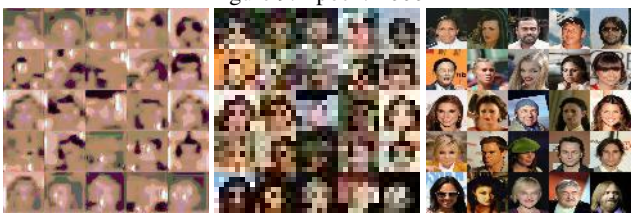

Figure 10. Epoch: 4000

Figure 11. Epoch: 8000

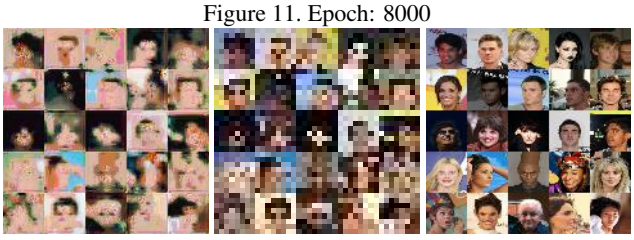

Figure 12. Epoch: 16000

Figure 13. Epoch: 19000

Samples collected from spade extension model.

Iterations	PSNR		SSIM	
	w/o SPADE	SPADE	w/o SPADE	SPADE
2000	27.922	27.915	**0.108**	**0.256**
4000	28.058	27.907	0.263	0.388
8000	28.318	28.233	0.470	0.568
16000	28.550	28.078	0.543	0.481
19000	28.779	28.235	0.613	0.527

Table 3. Quantitative results for CelebA Faces Dataset

digits and patterns. We realised few iterations into the dataset that, the model is majorly focusing on the background of the numbers dataset and not the color. Thus we have tried collecting a new license plate dataset from Kaggle which only comprises of black digits on white background. Unfortunately we didn't end up with great samples for that dataset, since the images needed different pre-processing step entirely to be used for the new model. Thus we have skipped adding the license plate dataset.

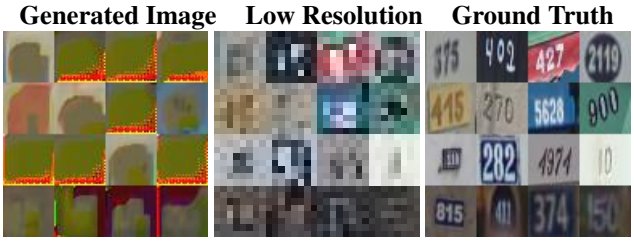

Figure 16. Epoch: 2000

Figure 17. Epoch: 4000

Generated images on House Numbers Dataset

6. Conclusion

Main aim of our project was to generate images of higher resolution from low resolution image, with the constraint that the LR images are not decipherable by humans. Thus, 'Generating something out of nothing'. The datasets are chosen keeping in mind the application of the project in the field of security and camera surveillance.

We used Pixel Recursive Super Resolution as baseline of our approach. First the results are obtained using the Dahl et al.[1] model. Then, we replaced regular batch normalization layer of base architecture with the SPatially Adaptive DEnormalization layer (SPADE), with the hope that the model will learn the spatial features of the low resolution image during training and thus will give better results. As evident from the results, the model did perform better than base Pixel Recursive Super Resolution model, giving better generated images and higher PSNR and SSIM values at low iteration steps. But as we move on to higher iterations, the results are not convincing. It can be concluded from this that further changes are needed in the model to fully utilize the benefits of SPADE. Possible solution can be to alter loss function of the model, to ensure that the in initial steps of training, spatial information of LR image is given more importance and as the training progresses, more weight age be given to pixel dependent parameters.

7. Discussion

7.1. Challenges

The key challenge that we faced is related to slow training of the model. The samples generated during training after each 1000 iterations requires a lot of time as for each batch, images are generated pixel by pixel, that is, 32x32x3 = 3072 runs for each batch. This further slows the training down the model. As the model integration and fine-tuning is an iterative process, we needed to generate decent samples sizes to gauge the model performance. Thus resulting in exhausting all our GPU credits before we could come up with good model iterations.

Adding SPADE to model, further slows the training, as our model now has more parameters to learn as part of SPADE Resblock. This can attributed to the fact that the source code of the model is written in Tensorflow, which is slow in implementation. Thus we tried implementing the model in Pytorch, but due to time constraint, we faced many debugging problems.

7.2. Future Work

As evident from the results, SPADE integration to the baseline architecture of probabilistic super resolution, improved the results initially but failed to give promising generated images for more iterations. This can be resolved by adding more fine-tuning parameters in the model and adjusting the loss function, so that at higher level of iterations, the model learns better, depending less on the spatial information of low resolution image and more on the pixel-wise information of image. Implementing the entire model in Pytorch can also bump up the training process and allowing to train for large number of iterations to get better results.

References

[1] R. Dahl, M. Norouzi, and J. Shlens. Pixel recursive super resolution. *CoRR*, abs/1702.00783, 2017. 1, 2, 4, 6

[2] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 184–199, Cham, 2014. Springer International Publishing. 2

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *CoRR*, abs/1501.00092, 2015. 2

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, Feb 2016. 2

[5] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. *CoRR*, abs/1608.00367, 2016. 2

[6] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 4

[7] W. Lai, J. Huang, N. Ahuja, and M. Yang. Fast and accurate image super-resolution with deep laplacian pyramid networks. *CoRR*, abs/1710.01992, 2017. 2

[8] C. Ledig, L. Theis, F. Huszár, J. A. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2016. 2

[9] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. 1

[10] K. Nasrollahi and T. B. Moeslund. Super-resolution: a comprehensive survey. *Machine Vision and Applications*, 25:1423–1468, 2014. 2

[11] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011. 5

[12] S.-J. Park, H. Son, S. Cho, K.-S. Hong, and S. Lee. *SRFeat: Single Image Super-Resolution with Feature Discrimination: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI*, pages 455–471. 09 2018. 2

[13] T. Park, M. Liu, T. Wang, and J. Zhu. Semantic image synthesis with spatially-adaptive normalization. *CoRR*, abs/1903.07291, 2019. 1, 3

[14] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *CoRR*, abs/1609.05158, 2016. 2