A Comparative Study of Downsampling Methods in Deep Image Super Resolution

1st Sourena Khanzadeh
dept. Computer Science
Toronto Metropolitan University
Toronto, Canada
sourena.khanzadeh@ryerson.ca

2nd Daniel Platnick
dept. Computer Science
Toronto Metropolitan University
Toronto, Canada
daniel.platnick@ryerson.ca

***Abstract*—Deep image super resolution is the process of upscaling images with deep neural networks. Downsampling is a critical step in this process, where a high-resolution image is converted to a low-resolution version in order to be processed by a deep learning model for training. The quality of the resulting super-resolution image depends heavily on the complexity and effectiveness of the downsampling method used in creating a low resolution training set. In this research, we compare several different downsampling methods in the context of deep image super resolution. The explored downsampling methods include linear interpolation, bilinear interpolation, nearest neighbor interpolation, lanczos filter, bicubic interpolation, hamming window and box filter. We first provide a qualitative comparison of each downsampling method used on images with varying upscale factors, and then provide an empirical study with quantitative results to demonstrate that linear interpolation, the hamming window, and lanczos filter work best for training super image resolution models.**

***Index Terms*—Super Resolution, Deep Learning, GAN, SRGAN, Downsampling**

I. Introduction

Deep learning has revolutionized the field of image super-resolution (ISR), enabling the creation of high-quality, high-resolution images from low-resolution inputs. Prior to the development of deep learning techniques, traditional interpolation methods such as bicubic interpolation were commonly used for ISR, but these methods often result in blurry and unrealistic images. There are several motivations for the use of ISR. ISR has direct applications to improve fields like medical imaging, synthesizing image data for deep learning models, video surveillance, satellite imaging, and generally any technical imaging field that may require enhanced zoom quality.

Downsampling, the process of reducing the resolution of an image, plays a key role in deep learning-based ISR. By training a model on low-resolution image data with their corresponding high-resolution counterparts, the model is able to learn the complex mapping between the two resolution levels. In this study, we compared model performance on different low resolution datasets created using different downsampling methods in order to see which downsampling methods work best for training deep learning ISR models. We first create a high-resolution dataset from a mixture of notable datasets. We create 7 new low resolution datasets using 7 downsampling methods: linear interpolation, bilinear interpolation, nearest neighbor interpolation, lanczos filter, bicubic interpolation, hamming window, and box filter. Then we use a VGG-16 model to predict the high resolution images. Our results showed that certain downsampling methods are more effective at producing high-quality, high-resolution images compared to others.

Image super-resolution (SR) refers to the process of increasing the resolution of an image, typically by a factor of 2 or 4, in order to improve its visual quality or enable further image processing tasks that require higher resolution. Traditional interpolation methods, such as bicubic interpolation, have been widely used for SR, but more recent techniques based on deep learning have shown promising results due to their ability to learn complex feature mappings between low-resolution and high-resolution images [1].

Single image super-resolution (SISR) refers to the problem of increasing the resolution of a single image, rather than a video or a sequence of images. SISR is a notoriously difficult problem due to the lack of clear mapping between the low-resolution and high-resolution spaces and the difficulty of building a high-dimensional mapping given large amounts of raw data [2]. However, recent deep learning-based approaches have made significant gains in both numerical and qualitative performance by leveraging the strong ability of deep learning to extract useful high-level abstractions that connect the low-resolution and high-resolution domains [3].

Convolutional neural networks (CNNs) have been widely used in SISR, with early methods such as SRCNN [1], [4], [5] demonstrating good performance. More recent approaches, such as super-resolution generative adversarial network (SRGAN) [6], have employed generative adversarial networks (GANs) [7] to achieve even better results. SRGAN uses a deep residual network (ResNet) with skip connections and down-sampling, as well as a discriminator that favors solutions that are difficult to differentiate from high-resolution reference images. The SRGAN is optimized using mean squared error divergence as the objective function, and it leverages the high-level feature mappings of the VGG network to produce high-quality results.

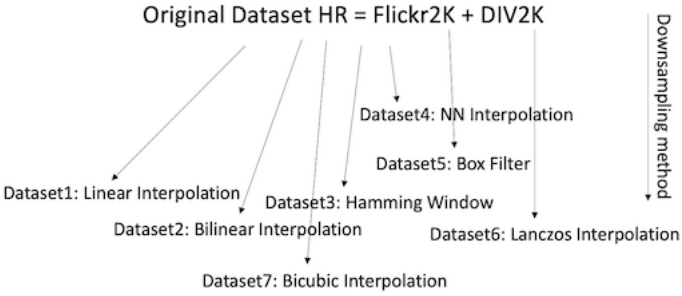

Fig. 1. Creation of 7 low resolution datasets from Flicker2K and DIV2K. Each low resolution dataset image is paired with the same corresponding high-resolution label. Each dataset was created using a different downsampling technique.

II. Literature Review

In attempts to solve SISR, classical image operator approaches were among the first used. In spite of their apparent efficacy, filtering approaches like as linear, bicubic, and Lanczos [8] filtering oversimplify the SISR issue and often provide solutions with excessively smooth textures. There are techniques that try to maintain edges [9], [10]. Stronger approaches often depend on training data and seek to construct a complex mapping between low- and high-resolution picture data. Several example-pair-based techniques utilize LR training patches with known HR equivalents. Freeman et al. conducted the first research in this topic. Similar solutions to the SR issue may be linked to the realm of compressed sensing [11]–[13]. Glasner et al. [14] empower the SR with redundancies that are independent of the size of individual picture patches. Huang et al. [15] use self-similarity as a paradigm and build self dictionaries in order to account for minor but important changes in form. Gu et al. [16] created a convolutional sparse coding method that enhances reliability by evaluating the whole picture without relying on overlapping patches.

Tai et al. [17] are able to recreate realistic texture detail while avoiding edge artifacts by combining the benefits of learning-based detail synthesis with an edge-directed SR approach based on a gradient profile prior [18]. Zhang et al. [19] propose a multi-scale lexicon to account for the repeating of seemingly similar image patches across scales. As a technique of super-resolving landmark photographs, Yue et al. [20] propose a structure-aware matching criterion for alignment and the extraction of similar HR images from the web.

Using a low-dimensional manifold to find similar LR training patches, the neighbourhood embedding technique upsamples an LR image patch by combining the associated HR patches for reconstruction [21], [22]. As overfitting is a concern with neighbourhood approaches, Kim and Kwon [23] employ kernel ridge regression to generate a more exhaustive map of example pairings. Regression may also be addressed using Gaussian process regression [24], trees [25], and Random Forests [26]. In Dai et al. [27], a large number of patch-specific regressors are learned, and the best regressors are selected for testing.

CNNs have existed for some time [28], but their popularity has recently soared [29], [30] due in large part to the network's outstanding performance in image classification. These approaches have also helped other fields of computer vision, such as object detection [31] face recognition [32], and pedestrian detection [33]. The availability of large amounts of data (such as ImageNet [34]) and the efficient implementation of training on modern powerful GPUs [30], the proposal of the Rectified Linear Unit (ReLU) [35] that makes convergence much faster while retaining good quality [30], and the availability of a large training set are significant contributors to this development (like ImageNet [30]).

The rest of this paper is organized as follows. The problem statement is discussed in Section III. Then, in Section IV, we discuss the problem of DISR. Section V presents the experiments, which is followed by Section VI containing discussion and evaluation results, and finally the paper ends with concluding remarks in Section VII.

III. Problem of Super Resolution

Deep image super resolution consists of a deep learning model learning to map a low-resolution input image to the high-resolution output image, effectively "super-resolving" the image. The result is an image with improved resolution and detail compared to the input image. We create different low resolution image training sets using various downsampling techniques. Let LR, HR, us, and ds respectively denote the low resolution image, ground truth high resolution image, upscale factor, and downsampling method used to achieve LR from HR for training observation i. Specifically, LR for training example i is given by:

$$LR_i = ds(HR_i), \tag{1}$$

where ds is one of the seven explored downsampling methods. The dimensions of LR are given by:

$$LRsize = HRsize/us \tag{2}$$

We specifically downsampled the resolution of the HR training images such that the LR image is to be equal in size to the HR image divided by the upscale factor. This is to ensure that the problem consists of upscaling a low quality image, back to the size of the original high quality test image. Doing this allows us to empirically compare the ability of VGG-16 trained on different downsampling methods, while keeping everything else constant and clearly observing the effects that different downsampling methods have on producing a high quality image from a low quality image.

Let VGG_{ds}, LR_{ds}, and $\hat{HR}$ denote the VGG-16 neural network trained by a specific downsampling technique, the test image with the respective downsampling method, and the predicted HR image using VGG-16. After each type of LR dataset has been created, the next step is to train multiple VGG-16 networks based on different datasets for comparison:

$$VGG_{ds} = VGG \rightarrow training \rightarrow (LR_{ds}), \quad (3)$$

From there, the 7 different VGG_{ds} models are used to predict the HR image from the corresponding downsample method LR image:

$$\hat{HR} = VGG_{ds}(LR_{ds}). \quad (4)$$

$\hat{HR}$ is the predicted image of VGG_{ds}. By comparing the $\hat{HR}$ generated from each VGG_{ds} prediction of the same image, we can evaluate the performance of each network based on PSNR, SSIM, and physical appearance.

IV. Method

- Data collection: In this step, we create and collect multiple datasets of low-resolution images which correspond to one high-resolution image labels dataset used for training and evaluation. This dataset is used to train the model and then evaluate its performance with testing.
- Data preprocessing: In this step, we split the respective datasets into training, validation, and test sets and apply any necessary transformations to the images, such as cropping to reduce image size to make it easier for the VGG-16 network to process.
- Model definition: In this step, we define the model architecture that will be used for image super-resolution. This can be a traditional CNN, such as a fully convolutional network (FCN), or a more advanced architecture such as a GAN. Model training: In this step, we train the model on the training set, using the validation set to monitor the performance and tune the model hyperparameters. Model evaluation: In this step, we evaluate the performance of the model on the test set, using metrics such as peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) to measure the quality of the generated high-resolution images.

A. Data Collection

For the purposes of this study, we collected a dataset of high-resolution images from the Div2k and Flickr2K datasets. In total, our dataset consisted of 3539 images. We downloaded the dataset from Google Drive using the gdown library and created a script to automatically make folders and allocate the images into train and test sets. From this single 3539 image dataset, we create 7 different low resolution image datasets to use as input for training different VGG-16 networks.

The data was collected from DIV2K and Flickr2K which were both used in the reputable CVPR NTIRE workshop in 2016 and 2017, meaning the data is guaranteed to be of high quality. It is also widely used in image super-resolution literature and provides a diverse set of images for training and evaluation. Other datasets may also be suitable for image super-resolution depending on the specific goals of the study. The content of the image is not very important for this task, but training on higher texture photos with more information may lead to better results.

B. Data Preprocessing and Analysis

Before training VGG-16, we preprocessed and analyzed the dataset to ensure that it was suitable for the task of image super-resolution. First, we visualized the data by plotting some examples of the high-resolution images in the dataset. This gave a sense of the characteristics of the data and identify any potential issues that needed to be addressed. We removed all images less than a minimum threshold size, and cropped the remaining images to the dimensions 492x1080 to make sure the entire HR dataset is consistent in size for inputting to the neural network.

We also analyzed the data to look for patterns or trends, such as the distribution of image sizes or the type of content in the images. This helped us identify any biases or issues in the dataset that may affect the performance of the model.

Finally, after applying necessary transformations to the data, such as normalization or cropping, to prepare it for model training. We also split the dataset into three sets: a training set for model training, a validation set for monitoring the performance of the model during training, and a test set for evaluating the final model performance.

C. Process of Making the Datasets

To create 7 different LR datasets based on 7 types of downsampling, we developed a custom class called $TrainDatasetFromFolder$. This class extends the PyTorch Dataset class and takes as input a directory containing the images, a crop size, an upscale factor, and an optional interpolation method. It stores the filenames of the images in the directory and creates two transformations: one for the high-resolution (HR) images and one for the low-resolution (LR) images.

The HR transformation is created using the $train\_hr\_transform$ method from the $Helper$ class, which returns a transformation composed of a random crop and a conversion to tensor. The crop size is specified as an input to this method. The LR transformation is created using the $train\_lr\_transform$ method from the $Helper$ class, which returns a transformation composed of a conversion to a PIL image, a resize operation, and a conversion to tensor. The crop size and the upscale factor are specified as inputs to this method. The resize operation resizes the image to a smaller size by dividing the crop size by the upscale factor.

To retrieve a specific image pair from the dataset, we implemented the $\_\_getitem\_\_$ method of the $TrainDatasetFromFolder$ class. This method takes an index as input and returns the LR and HR versions of the image at that index. The LR image is obtained by applying the LR transformation to the hr image, which is read from the file using the specified filename. The $\_\_len\_\_$ method of the $TrainDatasetFromFolder$ class returns the number of image filenames in the dataset.

In addition to the $TrainDatasetFromFolder$ class, we also developed a $Helper$ class that contains several static methods providing utility functions that may be used in other

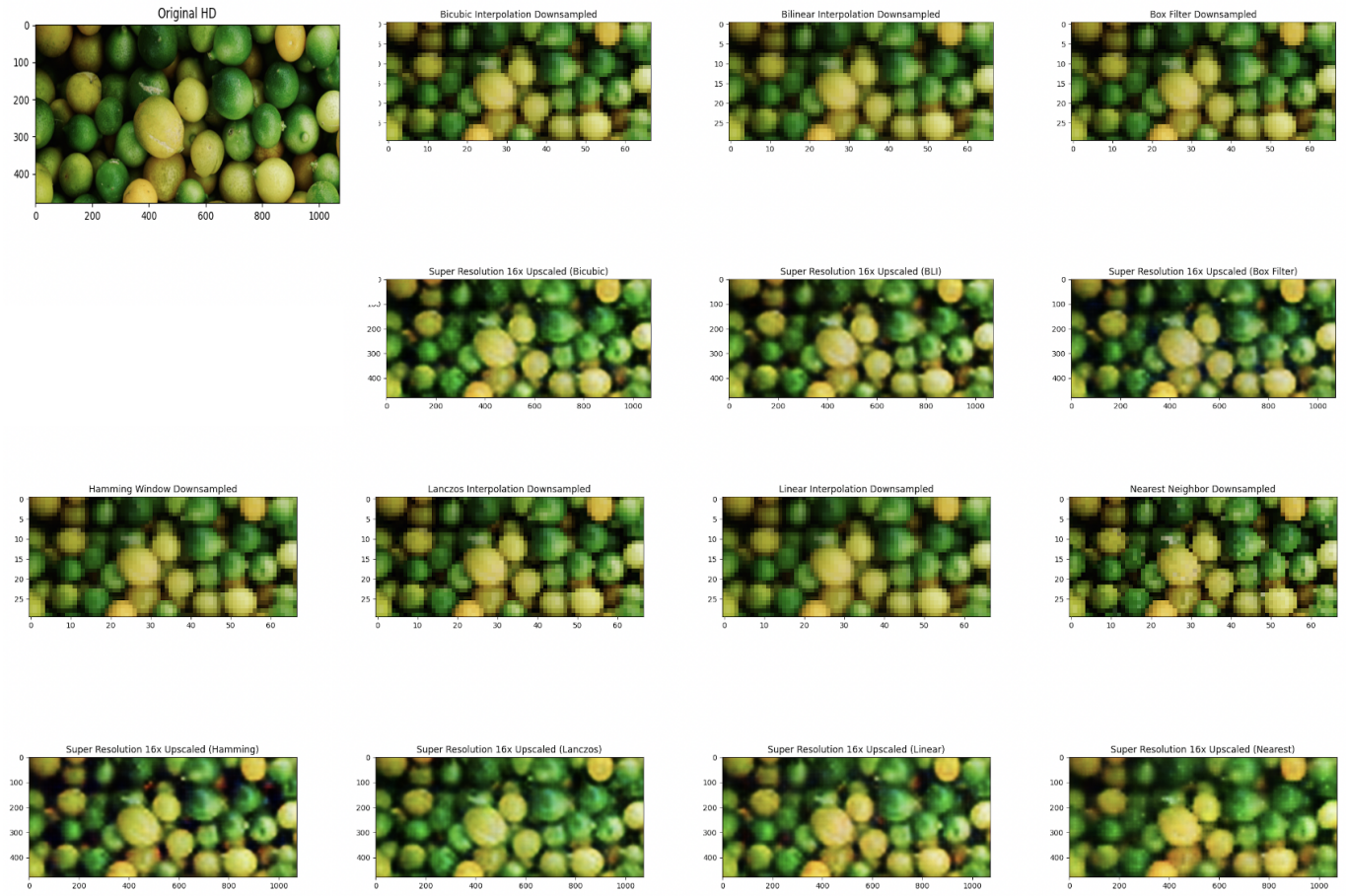

Fig. 2. Side-by-side comparison of the quality of ISR images produced by different VGG-16 models trained on different downsampling method image datasets. Top row (left to right): Ground truth HD, bicubic, bilinear, box filter. Middle row (left to right): Hamming, lanczos, linear, nearest neighbor.

parts of the code. These methods include is_image_file, which takes a filename as input and returns True if the file extension is one of a set of image formats, and $calculate\_valid\_crop\_size$, which takes a crop size and an upscale factor as inputs and returns the crop size adjusted so that it is divisible by the upscale factor.

We used the $TrainDatasetFromFolder$ class and the $Helper$ class to create our training dataset of image pairs, which we then fed into our deep learning model to train it for the image super resolution task. The model was able to learn to increase the resolution of the low-resolution images and produce high-resolution versions with increased detail and improved visual quality.

D. Model

In this study, we fine-tuned a VGG-16 deep learning model for the task of image super-resolution, which involves increasing the resolution of a low-resolution image to produce a high-resolution version that is more detailed, contains more information, and is visually appealing. Our model consists of two main components: a generator and a discriminator.

The generator is responsible for producing the high-resolution version of the input image. It is implemented as a convolutional neural network (CNN) and consists of a series of convolutional, downsampling, and fully-connected layers. The input to the generator is a low-resolution image and the output is the corresponding high-resolution version. The discriminator is responsible for helping the generator produce realistic results, directly giving feedback to the generator with a separate neural network.

To construct the generator, we defined a series of blocks of layers. The first block consists of a single convolutional layer with a kernel size of 9 and a padding of 4, followed by a parametric ReLU activation function. The next five blocks are each instances of a $ResidualBlock$ class, which itself consists of two convolutional layers with a kernel size of 3, batch normalization layers, and a parametric ReLU activation function. This design allows the generator to learn the residual mapping between the low-resolution and high-resolution images, which can improve the quality of the generated images.

The seventh block of the generator consists of a single convolutional layer with a kernel size of 3 and a padding of 1, followed by a batch normalization layer. The final block consists of a sequence of $upsample\_block\_num$ instances of an

$UpsampleBlock$ class, which itself consists of a convo-

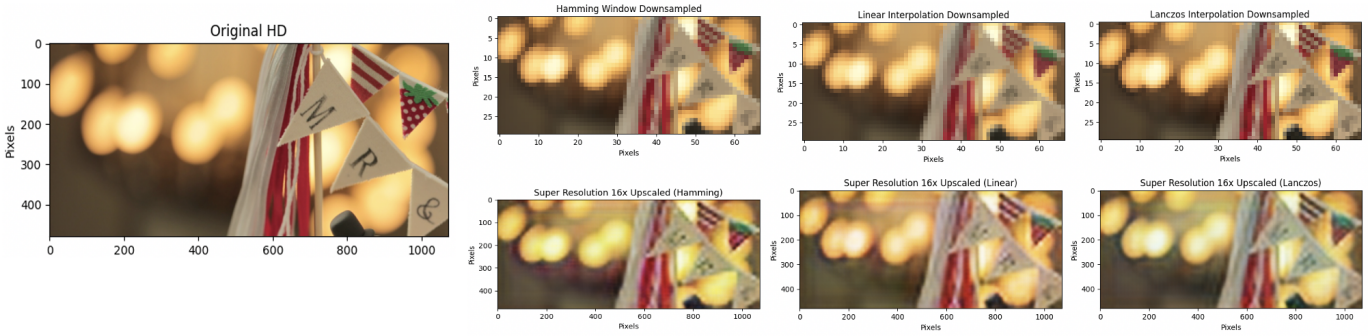

Fig. 3. Side-by-side comparison of the quality of ISR images produced by different VGG-16 models using the best 3 found methods of downsampling in terms of PSNR and SSIM for 16x ISR.

lutional layer, a pixel shuffle layer, and a parametric ReLU activation.

V. Experiments

In this section, we illustrate the qualitative and quantitative performances of the various fine-tuned VGG_{ds} networks. We measure the performance of these networks by using common image quality metrics such as PSNR [36] and SSIM [37].

A. Image Quality Metrics

All experiments are compared on the basis of known quantitative metrics like PSNR and SSIM values [36]. Let $\hat{HR}_{ds}$ and HR denote the predicted image using VGG-16 trained on a specific downsampling method ds, and the ground truth high quality image. Given $\hat{HR}_{ds}$ and HR, the PSNR(HR, $\hat{HR}_{ds}$) of an MxN image is defined as:

$$\text{PSNR}(HR, \hat{HR}_{ds}) = 10\log_{10}(255^2/\text{MSE}(HR, \hat{HR}_{ds}) \quad (5)$$

where,

$$\text{MSE}(HR, \hat{HR}_{ds}) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(HR_{ij} - \hat{HR}_{dsij})^2 \quad (6)$$

As the MSE approaches zero, the PSNR value approaches infinity. Consequentially, a higher PSNR provides higher image quality in general. Another metric used is SSIM, which was originally developed by Wang et al. [38] and is able to measure the similarity between two images in a different sense. SSIM models the images distortion as a combination of three factors: loss of correlation, contrast distortion, and luminance distortion [37]. SSIM is often more accurate in terms of human visual perception.

B. Data and Training

The newly combined dataset of Flickr2K and DIV2K images consist of 3539 HR images. Both datasets come from reputable CPVR NTIRE workshops. The DIV2K dataset is a diverse HR image dataset consisting of 1000 HR images gathered from crawling google images while paying special attention to qualities that would make the dataset valuable for image super resolution, such as being atleast 2000 pixels in one dimension [39]. The DIV2K dataset is 3.7GB in size.

Flickr2K is similar to DIV2K in that it contains many high-resolution images well suitable for training deep learning models on DISR [40]. Flicker2K dataset is 10.8GB in size. By combining both datasets, we are capable of increasing the number of datapoints for our model to learn from, helping the model learn to generalize to more data. After combining both datasets we find the smallest images and remove them past a certain size threshold (492x1080 pixels). We then crop all remaining images to have consistent dimensions of 492x1080 for the VGG-16 model to process.

Seven different pre-trained VGG-16 models were fine-tuned on different downsampling methods for the purposes of DISR. Each pre-trained model was fine-tuned using the remaining 3490 cropped images for learning. An 80%, 10%, 10% train/validation/test split was used for training the VGG-16 model. After the split there remains 2792 train, 349 validation and 349 images saved for testing. There were 14 VGG-16 models trained in total. The first 7 models are trained on each downsampling method for the task of 8x super resolution. The next 7 models are trained on each downsampling method for the problem of 16x super resolution. For each downsampling method explored, we trained an 8x and a 16x ISR VGG-16 model to predict the respective upscaled image.

Each VGG-16 model (i.e. VGG_{linear}, $VGG_{lanczos}$, etc.) was trained for 35 epochs with a batch size of 64. All 14 models were trained using an Nvidia Quadro RTX 8000 GPU to speed up the training process. The training process for 1 VGG-16 model takes about 40 minutes.

VI. Discussion

The overarching purpose of this experiment is to examine how well a powerful neural network is capable of learning to approximate the inverse function of the downsampling function used in the specific training set. Our results depict that a complicated deep learning model such as VGG-16 has an easier time upscaling images 16x that were preprocessed with linear interpolation, hamming window, or lanczos downsampling techniques. Figure 4 quantitatively describes that when VGG-16 is trained and tested on linear interpolated data,

VGG-16	8x Super Resolution		16x Super Resolution	
Training Method	PSNR	SSIM	PSNR	SSIM
Linear	8.890	0.0658	**8.892**	**0.0696**
Bilinear	8.889	0.0666	8.891	0.0679
Box	**8.890**	**0.0668**	8.889	0.0685
Nearest	8.890	0.0664	8.891	0.0684
Bicubic	8.889	0.0666	8.891	0.0675
Lanczos	8.890	0.0659	8.891	0.0685
Hamming	**8.889**	**0.0670**	**8.891**	**0.0689**

Fig. 4. Empirical results of each model trained with different downsampling methods for 8x and 16x super resolution based on PSNR and SSIM values. The best performing models for 8x super-resolution are box filter and hamming window. For 16x super-resolution, the top performing models are VGG-16 trained on linear interpolated downsampling, as well as hamming window, with lanczos coming as a close third best.

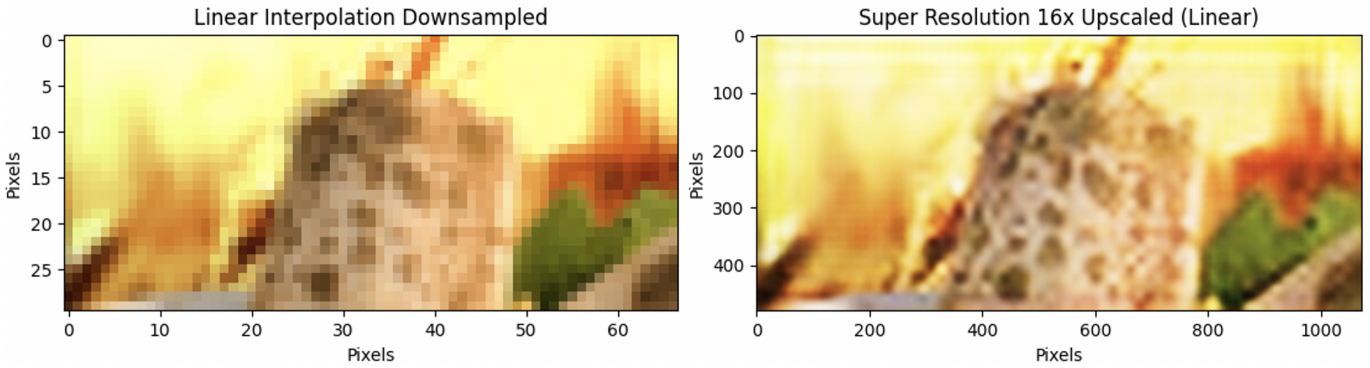

Fig. 5. Qualitative results on a test image of a close up burrito. The linear interpolation downsampling image is on the left, and the 16x resolved upscale version is on the right. The VGG-16 model fine-tuned for linear interpolation downsampling performs best when compared to all other models. It does an excellent job of predicting the low level details of burrito skin texture in the image.

it has the potential to greatly increase the image resolution, resulting in finer textures and details compared to the low resolution input image. Linear interpolation for 16x super resolution achieves a PSNR value of 8.892 and an SSIM value of 0.0696. Additionally the hamming window achieves a high PSNR value of 8.891 and a SSIM value of 0.0689, less than linear interpolation but still comparable. Our experiments show that linear interpolation and the hamming window are the most effective downsampling techniques and VGG-16 is more capable of approximating the inverse function of the respective input image when using these techniques.

Interestingly, the results slightly differ when performing the task of 8x super resolution. When predicting an 8x super resolution of a LR image, VGG-16 is more successful when trained and tested using the box filter and hamming window downsampling techniques. This is described in figure 4 which shows that VGG-16 achieves a PSNR of 8.890 and an SSIM of 0.0670 when using the hamming window method. The box filter training method achieves the second best performance when being compared to the other methods, scoring a PSNR of 8.890 and SSIM of 0.0668. These results show that for varying levels of super-resolution prediction, the ideal downsampling method likely changes.

This study was conducted on a strict timeline, and lots of additional experimentation is left for future work. For instance, in this work each model is tested on the same type of downsampling method it was trained on. It would be beneficial to have a test dataset consisting of a mix of different downsample methods. This way we could see how training on a singular downsampling method helps the model generalize to many different downsampling methods. One other issue, is that the training took a long time, and there are ways to further optimize the data-training pipeline and increase the efficiency.

Additionally, this study only explored one DNN

architecture. We would like to include more types of DNN architectures in a future study for more comparison examples. Exploring different types of NN architectures such as CNNs, RNNs, GRUs, and transformers would give us a more informative idea of how these downsampling methods effect different architectures and their performances. Additionally, there is is room to explore different training strategies such as a training set which contains an even distribution of different downsample methods. It is possible that having a training set full of a diverse range of downsampling methods would help these models generalize and super-resolve images more consistently and with greater detail. The code for the paper can be found on Github at: https://github.com/DanielPlatnick/downsampling_in_deep_ISR.

VII. Conclusion

This study explores different methods of downsampling to be used in the process of DISR. Using transfer learning, a pre-trained VGG-16 model is fine-tuned on 7 different datasets created based on 7 methods of downsampling, and then tested on low quality images of each respective downsampling technique. The VGG-16 model is then tasked with predicting the inverse function of the respective downsampling training method, increasing the resolution, sharpness, and detail of the image. Based on quantitative metrics such as PSNR and SSIM, as well as qualitative metrics such as concrete image examples, we show that training VGG-16 on linear interpolation will result in the best performance for 16x super-resolution. Additionally, we extend our study to 8x super-resolution and demonstrate that downsampling with the box filter or the hamming window will yield the best predictions. We conclude when performing DISR with a VGG-16 model for 8x and 16x image super-resolution, the best downsampling methods to use are box filter, hamming window, and linear interpolation.

References

[1] 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*, vol. 38, no. 2, pp. 295–307, 2015.

[2] C.-Y. Yang, C. Ma, and M.-H. Yang, "Single-image super-resolution: A benchmark," in *European conference on computer vision*. Springer, 2014, pp. 372–386.

[3] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao, "Deep learning for single image super-resolution: A brief review," *IEEE Transactions on Multimedia*, vol. 21, no. 12, pp. 3106–3121, 2019.

[4] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*. Springer, 2014, pp. 184–199.

[5] Z. Wang, J. Chen, and S. C. Hoi, "Deep learning for image super-resolution: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3365–3387, 2020.

[6] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[8] C. E. Duchon, "Lanczos filtering in one and two dimensions," *Journal of Applied Meteorology and Climatology*, vol. 18, no. 8, pp. 1016–1022, 1979.

[9] J. Allebach and P. W. Wong, "Edge-directed interpolation," in *Proceedings of 3rd IEEE International Conference on Image Processing*, vol. 3. IEEE, 1996, pp. 707–710.

[10] X. Li and M. T. Orchard, "New edge-directed interpolation," *IEEE transactions on image processing*, vol. 10, no. 10, pp. 1521–1527, 2001.

[11] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *2008 IEEE conference on computer vision and pattern recognition*. IEEE, 2008, pp. 1–8.

[12] W. Dong, L. Zhang, G. Shi, and X. Wu, "Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization," *IEEE Transactions on image processing*, vol. 20, no. 7, pp. 1838–1857, 2011.

[13] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *International conference on curves and surfaces*. Springer, 2010, pp. 711–730.

[14] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *2009 IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 349–356.

[15] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5197–5206.

[16] S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang, "Convolutional sparse coding for image super-resolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1823–1831.

[17] Y.-W. Tai, S. Liu, M. S. Brown, and S. Lin, "Super resolution using edge prior and single image detail synthesis," in *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 2400–2407.

[18] J. Sun, Z. Xu, and H.-Y. Shum, "Image super-resolution using gradient profile prior," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[19] K. Zhang, X. Gao, D. Tao, and X. Li, "Multi-scale dictionary for single image super-resolution," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 1114–1121.

[20] H. Yue, X. Sun, J. Yang, and F. Wu, "Landmark image super-resolution by retrieving web images," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4865–4878, 2013.

[21] R. Timofte, V. De Smet, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1920–1927.

[22] ——, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Asian conference on computer vision*. Springer, 2015, pp. 111–126.

[23] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 6, pp. 1127–1133, 2010.

[24] H. He and W.-C. Siu, "Single image super-resolution using gaussian process regression," in *CVPR 2011*. IEEE, 2011, pp. 449–456.

[25] J. Salvador and E. Perez-Pellitero, "Naive bayes super-resolution forest," in *Proceedings of the IEEE International conference on computer vision*, 2015, pp. 325–333.

[26] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3791–3799.

[27] D. Dai, R. Timofte, and L. Van Gool, "Jointly optimized regressors for image super-resolution," in *Computer Graphics Forum*, vol. 34, no. 2. Wiley Online Library, 2015, pp. 95–104.

[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[31] W. Ouyang, P. Luo, X. Zeng, S. Qiu, Y. Tian, H. Li, S. Yang, Z. Wang, Y. Xiong, C. Qian *et al.*, "Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection," *arXiv preprint arXiv:1409.3505*, 2014.

[32] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," *Advances in neural information processing systems*, vol. 27, 2014.
[33] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2056–2063.
[34] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," *Advances in neural information processing systems*, vol. 19, 2006.
[35] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.
[36] A. Horé and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 2366–2369.
[37] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
[38] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
[39] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1122–1131, 2017.
[40] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.