Didactic Supervised Image Super resolution EE 610 Assignment 2

R Shrinivas
Roll No. 150070036

***Abstract*—This assignment studies a particular problem of increasing the resolution of a given image.In this we try a supervised learning approach using convolutional neural networks and some advances made in the same.This assignment explores the objective of trying to be more didactic in training the convolutional neural network by allowing the model to learn the appropriate sub-super resolutions too.The implementation utilizes a variation of loss functions as the training of the model progresses.We also look into the use of a Dense convolutional network and sub-pixel convolution to allow efficient feature extraction and gradient flow.**

I. Introduction

Image super resolution means we take an image of a given resolution and increase its resolution way beyond its original resolution.This is achieved traditionally using some clever interpolation techniques.Some of them are linear interpolation,Bi cubic or by using a fixed convolutional kernel.But all of these methods have a major drawback and that is none of these methods are data driven and have qualitative upper bounds on how good a result they can produce.Convolutional neural nets have the unique advantage of not only being able to mimic these methods but out perform them given a good enough metric of similarity to super resolved image target in a supervised learning paradigm and also an exhaustive data set.

When we take an image it is basically a discrete 2 dimensional sample of a 2 dimensional analog waveform. To perfectly capture the entire analog image we need to sample at a very high rate and that means we end up with an image of a very large resolution. The major disadvantages are that this process is extremely computationally and memory intensive.An alternative is that we take an image limited by our hardware at a particular resolution and leverage the fact that we have a probabilistic model of being able to guess a particular texture in great detail given its low resolution image or technically called a sub sampled image.In the setting of natural images there are many common textures such as vertical edges in cities and the intricate texture of leaves and grasses and also animal and human hair which are commonly observed in both amateur and professional photography. Given such a sub sampled texture in a particular patch a well trained CNN can potentially give a very good guess of the super resolved image.

II. Background

A. Structural Similarity Index [1]

The most common metric of comparing two images is by using the SSIM which stands for Structural Similarity Index .It is a scalar quantity which ranges from 0-1 with 0 being no similarity at all and 1 being perfectly similar.SSIM is basically the product of similarity on 3 major metrics: contrast(s),luminosity(l) and correlation(s). These metrics are computed on a local window of a user specified dimension and averaged out for the entire image. The luminosity term is defined mathematically for two images whose corresponding windows are denoted by x,y as:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

$$\hat{l}(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{2}$$

The contrast term is defined mathematically for two images whose corresponding windows are denoted by x,y as:

$$\sigma_x = (\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2)^{\frac{1}{2}} \tag{3}$$

$$\hat{c}(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{4}$$

The correlation term is defined mathematically for two images whose corresponding windows are denoted by x,y as:

$$\hat{s}(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{5}$$

And finally once we have l(x,y) c(x,y) and s(x,y)

$$SSIM = \hat{s}(x,y)^\alpha \hat{c}(x,y)^\beta \hat{l}(x,y)^\gamma \tag{6}$$

The exponents α, β & $\gamma$are tuned as per requirement usually to emphasis or de-emphasis a term in SSIM.$C_1, C_2, C_3$are very small constants.The SSIM is a very good in identifying the similarity in structure between two images but it was not helpful in learning the exact color of the patch in the reproduced image and we shall discuss about the same in later sections.

B. L1 loss

One of the most common metrics of losses is L1 loss and the equation describing L1 loss between reconstructed x and target y is as follows:

$$Loss(x, y) = \frac{\sum_{i=1}^{N}(|x_i - y_i|)}{N} \tag{7}$$

This is a good metric for our model to learn the colour scheme of the target image to be reproduced.Unfortunately it does little help in telling us about the structure of the image unlike SSIM.We shall use a combination of these losses to achieve our goal.

C. Convolutional neural network

A convolutional neural network is a family of neural networks that aims at learning the weights of a/multiple kernel given an optimization objective. Usually the optimization objective is to minimize a certain loss or dissimilarity metric of output image (output of convolution) and target image in a supervised learning paradigm.Below mentioned is a single convolution to generate a single filter from a single filter input. In most state of the art convolutional neural network architecture we extract multiple features from multiple input features with each output feature being the output of convolution of a 3-d kernel with all the input features while convolving along 2 axis (width and height).

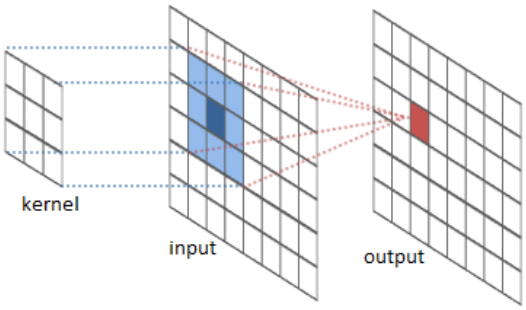


Fig. 1: 2-d convolution for a single filter output

Once a convolution output/filter has been generated the values are passed through an activation function a(x) mapping values from real numbers to real numbers.Some common activations are sigmoid,relu,tanh etc. The activation which we are using is relu activation and the reason is that it allows gradients to flow efficiently through the network and the output of this activation lies in the positive real numbers co-domain and that shall help us in reconstructing the super resolved target image.

D. Dense Convolution Network[2]

A DCN also called as Dense Convolution Network is one of the advances made in convolutional neural networks.Each output filter in the computation graph in the DCN block is concatenated with the previous convolution outputs in the DCN block and fed forward to a convolution. This recursive process is done as many times as required. Say in a DCN-9 block does this process 9 timed recursively. A major advantage of DCN is that because it houses so many skip connections inherently in its architecture it allows the gradients to flow very efficiently and also critical features required for reconstruction are forward propagated successfully.

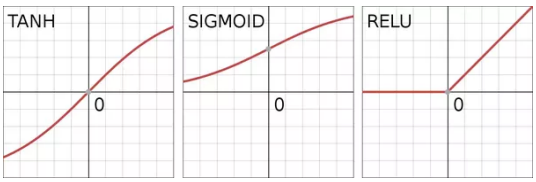


Fig. 2: some common activation functions

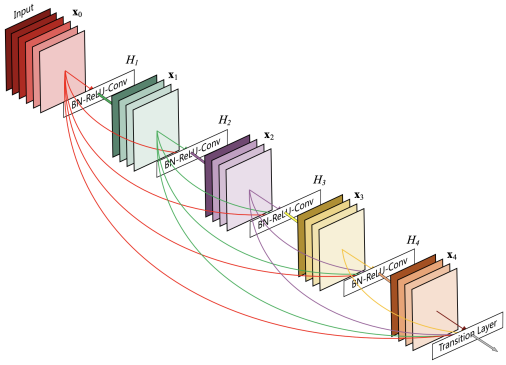


Fig. 3: A DCN 5 block

A major drawback is that DCN blocks are computationally expensive to add and introduce too many kernel weights to be learned that the user must make sure that there are sufficiently many data-points to prevent the model from over fitting.

E. Sub pixel convolution

Sub pixel convolution [4] is another major advance in the field of convolutional neural nets.The concept is fairly simple.Given a filter output of size $Height, Width, Kr^2$. We shuffle the pixels along the the depth axis in a fixed order to generate a Height·r,Width·r,K sized output.

This becomes more clear when we refer to Fig-6 which shows the schematics of this process.

III. Implementation

This assignment was implemented using python and the library used for the same is pytorch.This was my first time using pytorch.All the weights are learnt by back propagation by minimizing a loss metric which was SSIM for half the training and L1 loss for the other half.The model was trained on dual Nvidia GTX 1080-ti in sli and an Intel i-9 cpu with 64GB RAM.Further details are in the description below.

A. Architecture

The model is a Didactic Supervised Super Resolution Algorithm.The architecture is based on the paper published [3].The computation graph as seen in Fig-7 consists of a

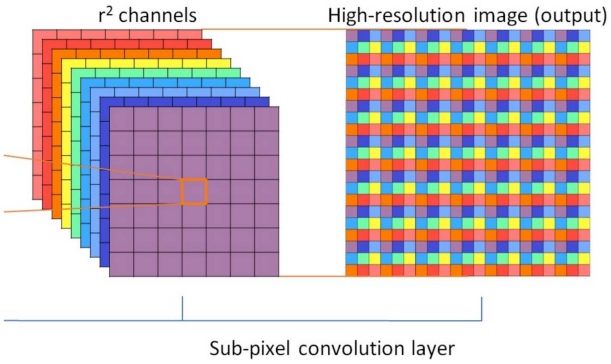


Fig. 4: Sub pixel convolution

backbone CNN and 2X,4X and 8X super resolved image branches branching out of it. The CNN backbone consists of some preliminary convolutions to generate a total of 160 filter outputs. All the convolutions are reflection padded and have a relu activation as mentioned before.once the preliminary convolutions are done we then feed our filter outputs to the DCN9 block which is a 9 block Dense convolutional neural network as explained[].

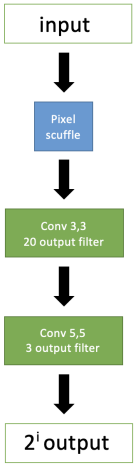


Fig. 5: Architecture for u0,u1 and u3

The outputs are propagated further in the CNN backbone and a copy of the same is propagated further in the 2X super resolution block.this copy of the DCN-9 block output are fed further to a pixel shuffle(2) block which up scales the height and width by 2 at the cost of reducing the number of filters by 4 and convolve this further to generate a 3 channel 2X super resolved image.

The outputs of DCN9 block forward propagated in the backbone of CNN are fed further to a pixel shuffle(2) block which up scales the height and width by 2 at the cost of reducing the number of filters by 4.this is then convolved with

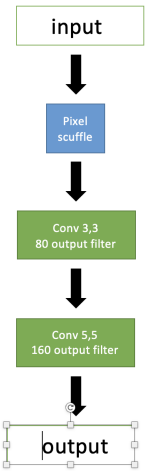


Fig. 6: Architecture for u2

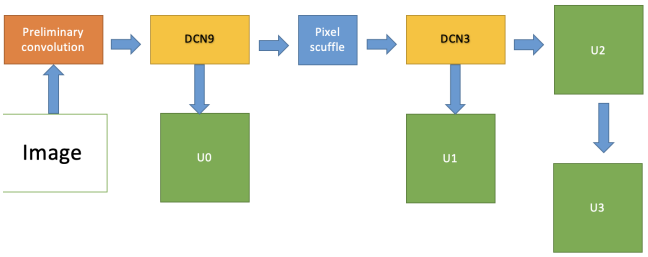


Fig. 7: Architecture

a 5,5 window kernel to generate 160 filters which are fed into DCN3 block.

Just like before the output of DCN9 , the output of DCN3 are fed to a 4X super resolution block and also into the CNN backbone further.

The DCN3 copy in the 4X resolution block is passed via a pixel shuffle(2) and a series of convolutions to generate the 4X resolved image.

The DCN3 copy propagated further into the CNN backbone is fed to a pixel shuffle(2) and an 8X resolution block which generated the 8X resolved image.

B. Data set

The data set that was used in training was DIV2K [**?**] training set and the input and the 3 targets were generated by sub sampling at 8X,4X,2X and 1X times a 600 by 600 area of the original high resolution images. This was done due to GPU memory constraints. The data set themselves have around 900 images for training and a total of 9 crops were generated from each image bringing us 8100 unique cropped segments of the data set used in our training.

C. Training

The optimizer used for the purpose of training is Adam optimizer with β_1 and β_2 as 0.9 and 0.99 and a learning rate of 0.001 and an epsilon of 0.00001 The targets once generated were used for computing the loss and back propagating through the network to minimise the same. Initially the loss metric was

$$Loss(x, y) = -\sum_{i=1}^{3}(SSIM(X_{2^1}, Y_{2^1})) \tag{8}$$

and this was doe for nearly 20 epochs. the model was able to train and learn the general structure of the image but not the colour scheme.The outputs were sepia filter like images and to overcome this the model was trained for 20 more epochs on L1 loss:

$$Loss(x, y) = \sum_{i=1,j=1}^{i=3,j=N}(L1(Xj_{2^1}, Yj_{2^1})) \tag{9}$$

this enabled the model to learn the colour palette and also to gain an average SSIM of nearly 0.81 at the end of training

IV. Results

The results that have been displayed are the 2X 4X and 8X super resolved images re sized to fit a width of 4 cm along with their ground truth in Image table A. There have been 3 major cases considered in this report an animal with fur, a monument with a geometric construction and a natural image with low lighting conditions.The average SSIM across all images is close to 0.81 with a window size of 5,10 and 15 for the 2X 4X and 8X resolved images with α, β, γ =1.One common observation is that the model is unable to match the exact color temperature of the Ground truth but this is not a particularly difficult problem as these can be anyway be corrected in post production.

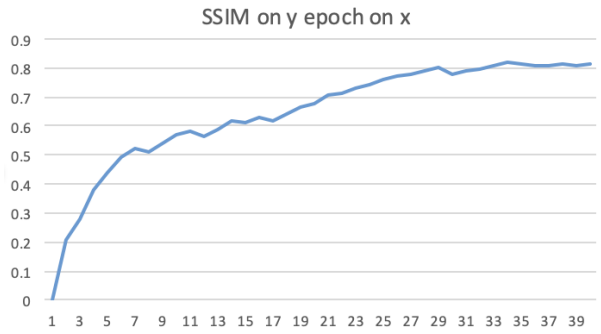
V. Observations

A. Training observations

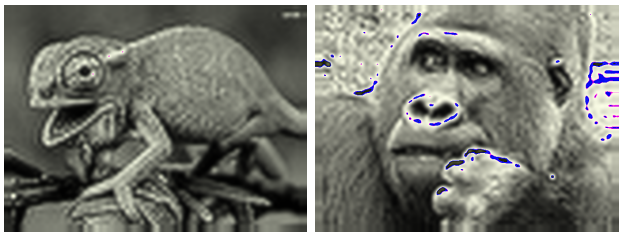
During training it was observed that there was a very critical need for changing the loss metric form SSIM to L1 loss as the images that were generated were able to come close to the structure of the target image but were very inaccurate in their colour and contrast.This was achieved by letting the model to continue training on an L1 loss and after 20 epochs we were able to learn the colour scheme and contrasts accurately.

B. Hyper-parameter optimization

A number of permutations of loss functions mainly weighted SSIM averages were tried out but all in vain, the reason being that emphasizing one super resolution was happening at a great cost of the other super resolution. Say the SSIM loss for 2X was weighted with 16 ,4 for 4X and 1 for 8X this caused the model to not be able to learn the 8X super resolved image at all and hence the optimal weights were the one which were equally prioritizing each sub-super



(a) SSIM over epochs



(a) chamaleon 8X (b) gorilla 8X

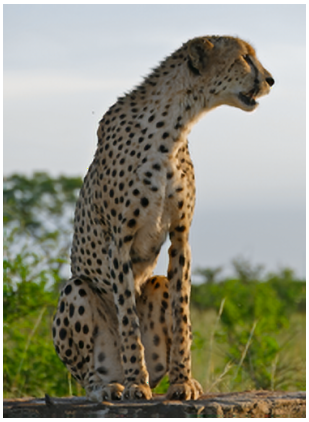
resolution. This was critical because the features used by the 2X resolved output was an input in the computation graph to calculate the 4X and 8X images,emphasizing SSIM of lower resolution up scaled image adversarialy affects us by not learning generalized features which can be used by all the 2X,4X and 8X super resolving blocks.

VI. Challenges remaining

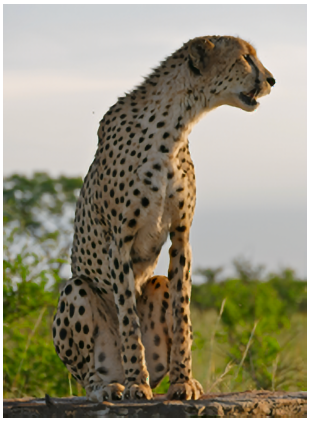
One of the major challenge in the field of super resolution is the question that what if the sub-sampling rate is so low that we are unable to sample the high frequency information even partially? In such cases we may need to generate those features looking at the patch an correlating it to the most probable texture.such false texture generation can be accurately done by a Generative Adversarial Network. This is a major issue when there are very fine textures like hair or fur or window sills of a building very far away from the camera.

References

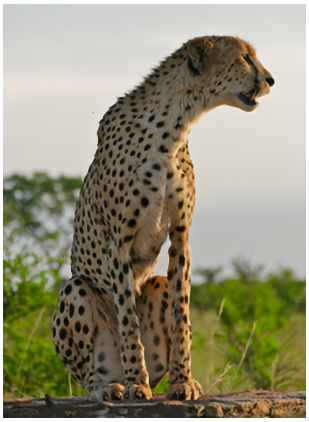
[1] Structural Similarity Index the original paper defining the same. http://www.cns.nyu.edu/pub/lcv/wang03-reprint.pdf
[2] This is the original paper implementation of DCN block. https://arxiv.org/abs/1608.06993
[3] This is the original paper implementation of a didadic supervised super resolution algorithm. https://fperazzi.github.io/files/publications/prosr.pdf
[4] This is the original paper implementation of subpixel convolution. https://arxiv.org/pdf/1609.05158.pdf



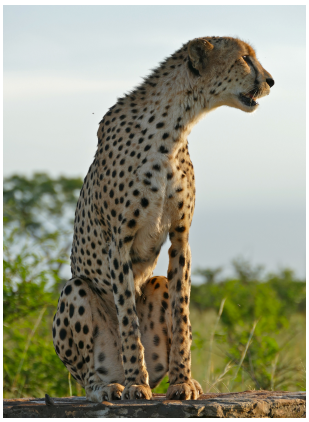
(a) 2X chetah



(b) 4x chetah



(c) 8X chetah



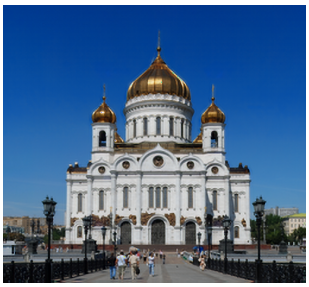
(d) GT chetah

(e) 2X chetah

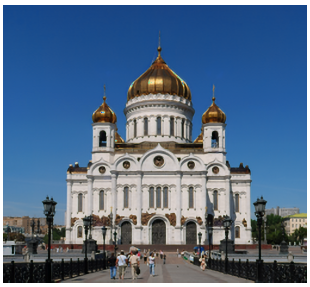
(f) 4x chetah

(g) 8X chetah

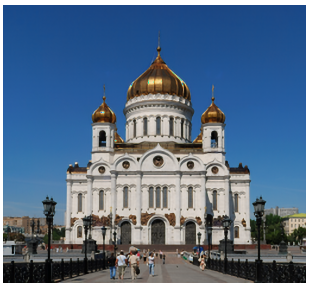
(h) GT chetah



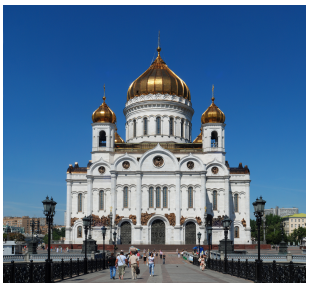
(i) 2X taj



(j) 4x taj



(k) 8X taj



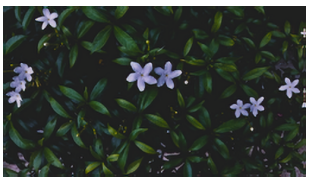
(l) GT taj

(m) 2X taj

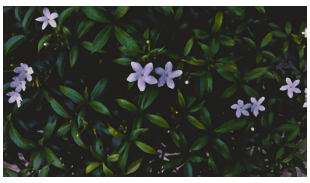
(n) 4x taj

(o) 8X taj

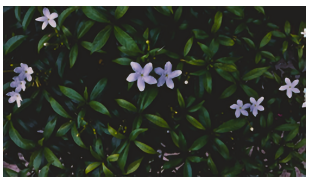
(p) GT taj



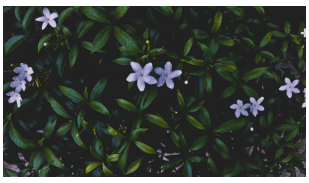
(q) 2X flower



(r) 4x flower



(s) 8X flower



(t) GT flower

(u) 2X flower

(v) 4x flower

(w) 8X flower

(x) GT flower