**Udacity Machine Learning Engineer Nanodegree**

Capstone Project Report

Svetoslav Paregov  
May 24, 2019

**Single Image Super Resolution through a CNN**

**I. Definition**

**Project Overview**

Single image super-resolution is the task of inferring a high-resolution image from a single low-resolution input. Image resolution enhancement is something that we used to see in many movies, but it is not something trivial to achieve. There are a lot of algorithms used in different software products that increase the image resolution with different degree of success.

Machine Learning has been used in many different image processing tasks with Convolutional Neural Networks. Detection of different objects, face recognition and other. In the last few years it is also used to improve the image resolution.

What it can be used for – increasing the size of pictures from old phones for printing or some other purpose; Increase the quality of old security footages; Use it as base for other Machine Learning tasks to provide better quality images;

There are a lot of researches on the topic from different universities. Inspiration for my project came from a research from Max Planck Institute for Intelligent Systems, Germany [1].

**Problem Statement**

Traditionally, the performance of algorithms for this task is measured using pixel-wise reconstruction measures such as peak signal-to-noise ratio (PSNR) which have been shown to correlate poorly with the human perception of image quality. As a result, algorithms minimizing these metrics tend to produce over-smoothed images that lack high-frequency textures and do not look natural despite yielding high PSNR values. In other words, for the computer looks good but not for the humans. A model or algorithm that is improving for human perception is needed.

I propose a CNN model that will be optimized for sharper and close as possible to the reality images. I hope to achieve this with correctly implemented loss functions. Still this is a hard problem since during the scale down a lot of information is lost and there so many variations when scaling up. So, part of the missing information should be synthesized by the CNN.

**Metrics**

Evaluation can be done visually or by feeding the enhanced images to an object recognition model. I’m going to rely mostly on visual evaluation showing the output to multiple people. If I have time will train an object recognition model for automated evaluation.

**II. Analysis**

**Data Exploration**

I have two datasets – MS COCO (Common Objects in Context) and CelebA, which will be used separately to train the model. First will start with the MS COCO. This dataset contains all kind of images – animals, people, cars, buildings and many more. I think this will be helpful so the model is more general when enhancing random image.

The second dataset is only people – mostly faces, but there are also full body pictures. In this case I want the model to be less generic. I hope this will give a better result when enhancing pictures of humans.

The low-resolution image that I’m going to feed into the model input is 32x32 pixels and the output image will be 128x128 pixels. Selected resolution increase ratio is 4 times.



Some of the images from MS COCO.



Some of the images from CelebA

**Exploratory Visualization**

Total images count for MS COCO is 409,453 which are separated in multiple folders by the provider.

MC COCO Images Count as provided per folder

Here are some statistics per folder. I am going to show the count of images per size. Since there are so many sizes I’m showing only those sizes that have more than 100 images. For the training purpose I have decided to use only images that have width and height more than 384 pixels. I call them “valid” and those that one of the dimensions is smaller than 384 pixels are called “invalid”.

We can see that per folder and in total there are much valid (338,318) than invalid (71,135).

We can see that in all of the sets the majority of the images are 640x480 pixels.

CelebA contains 202,599 images of celebrities. Mostly face and portrait images, but there are also a full body images. All of the images are provided in one folder.

Here I’m showing the image sizes that has more than 200 images, since there were too many over 100.

In conclusion. I have 338,318 images from MS COCO and 98,102 images from CelebA which I can use for training the model.

**Algorithms and Techniques**

I propose to use a Convolutional Neural Network (CNN) that learns how to synthesize the missing pixels. I'm going to use standard CNN training with calculation of the accuracy based on a validation set. The other approach I want to explorer is Generative Adversarial Network (GAN) to train my model. Training on different datasets with different loss functions will be performed and tested.

**Benchmark**

I can compare the results with different software products that increase the image resolution. I plan to train the model using different loss functions and compare the results. Since the goal is to achieve realistic images for human perception, I’m going to show the results to people and will gather the statistics for which model is enhancing best.

**III. Methodology**

**Data Preprocessing**

Images are filtered so they have at least 384 pixels on the short side. Then the images are being cropped to a rectangular image using the short side for base size. Cropped images are then resized to 256x256 pixels, so they don’t lose too much details. Selected size for low-resolution image is 32x32 pixels. Since our selected increase ratio is 4, that means the high-resolution image is 128x128 pixels.

Filtered images are being merged for the MS COCO dataset and then split to train, validation and test. Since the CelebA images are already together, I just filter them and split to train, validation and test.

I have selected to use 75% for train, 20% for validation and 5% for test. 5% for test is enough since the test will be a manual process. If needed I can change the percentage. Validation is not needed for GAN training, but I use validation for some tests.

Used Jupyter notebook – *./pre\_processing/image\_filter\_and\_split.ipynb*

Since I’m not going to upload the original dataset, here is a link with the split:

TODO: Need to put the link here once the files are uploaded.

I have created also a smaller subset for train, validation and test sets for faster evaluation of the models.

**Implementation**

For start I have decided to go with regular training and validation to evaluate the performance of the model. I created few models which I have trained with different loss functions. All of the models contain only convolution layers and up sampling layers. No fully connected or pulling layers are used. This allow that the model to be used after that with images of any size.

All convolutional layers use ‘*uniform’* initializer and the kernel size I selected is 3.

TODO: Describe the loss functions used here and the created models with different loss functions.

The base model is with 10 residual blocks.



Residual block



Base model. Used mostly in the initial evaluation. **(CNN 1)**

Then I have few models with added direct connection from the input image through a bicubic interpolation and summed with the last convolutional layer to produce the output image. The idea is here is that the model has to learn less since we have the bicubic interpolation in place.



Model with added bicubic interpolation. (**CNN 2**)

The last model I evaluated is without the residual blocks and without the bicubic interpolation. This is just for compare.

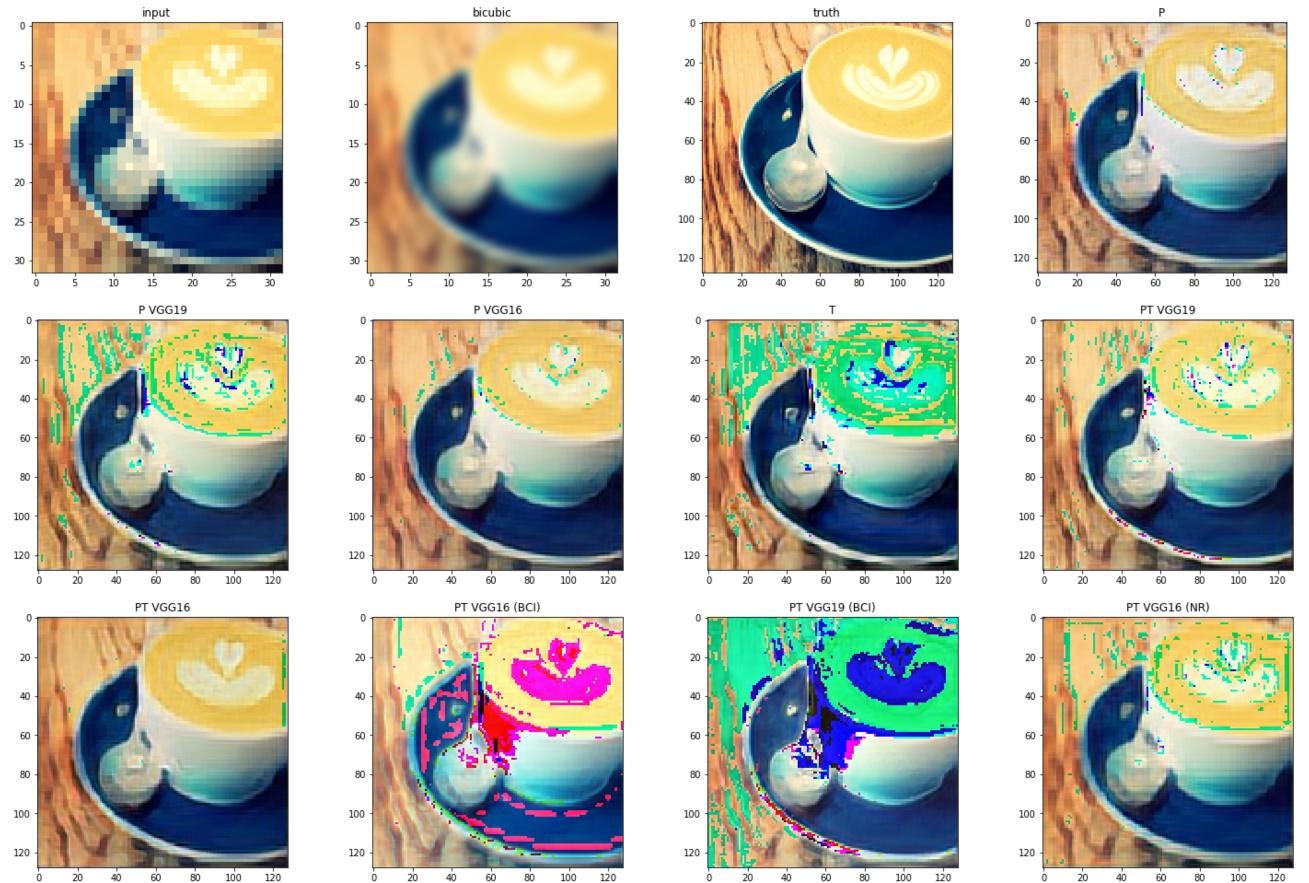


Model without residual blocks and without bicubic interpolation. (**CNN 3**)

I performed an initial evaluation of the models with the small datasets (600 train images, 80 validation images and 20 test images). This was done on MS COCO and CelebA datasets. I used the standard train function in Keras, passing the validation data. Optimizer is Adam with the default learning rate of 0.001. Since I did the training on my laptop the batch size was set to 8 and after running this multiple time it was clear that 50 epochs are enough.

Used Jupyter notebook: ***image\_super\_resolution\_initial.ipynb*** with these parameters:  
use\_small\_dataset = True  
use\_dataset\_celeba = False  
disable\_training = False  
load\_weights\_before\_training = False  
All of the models have to be enabled.  
train\_epochs = 100  
train\_batch\_size = 32  
test\_image\_index\_to\_show = range(20)  
optimizer = Adam(lr=0.001)

Here are some of the test results for MS COCO and CelebA.



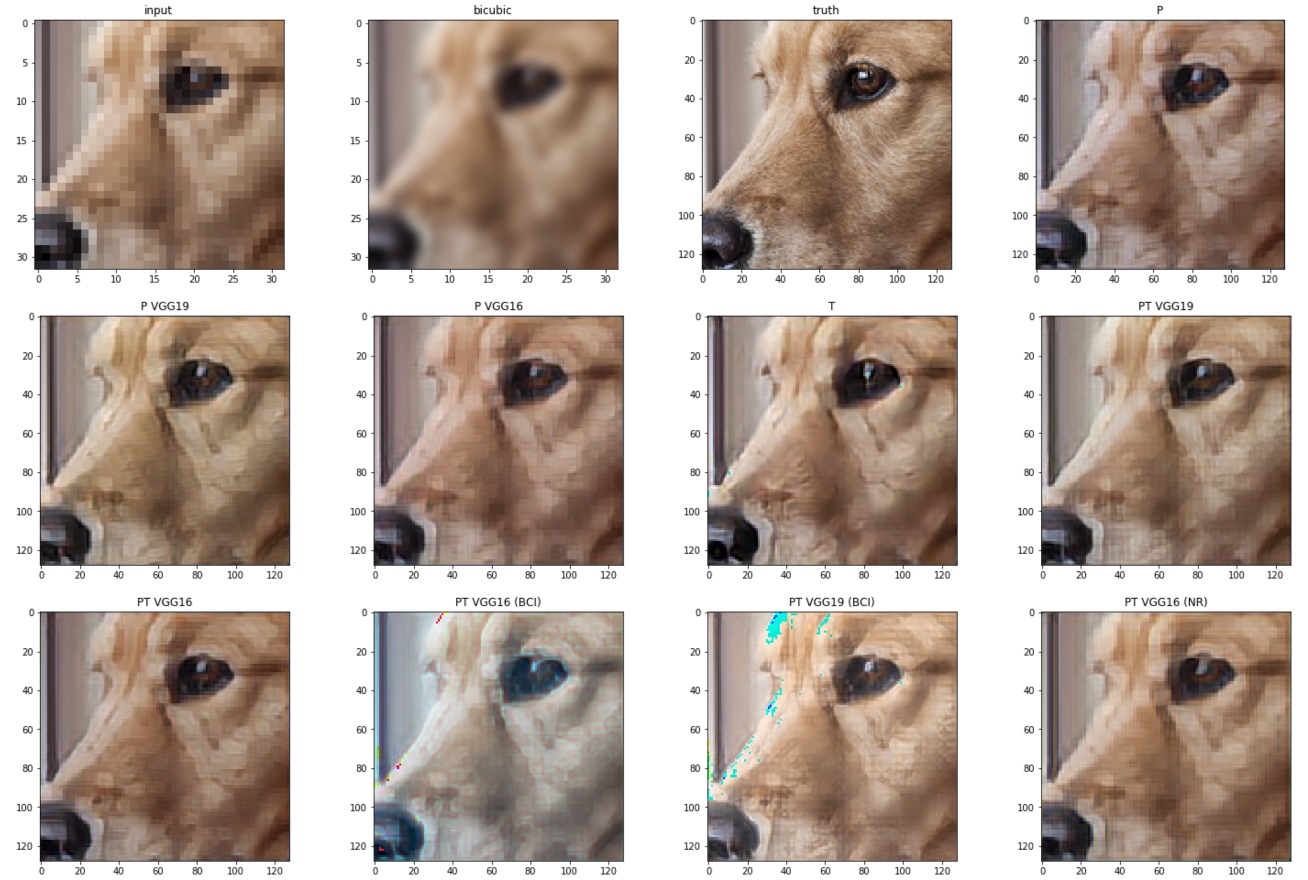
MS COCO initial evaluation. Image 1.



MS COCO initial evaluation. Image 3.



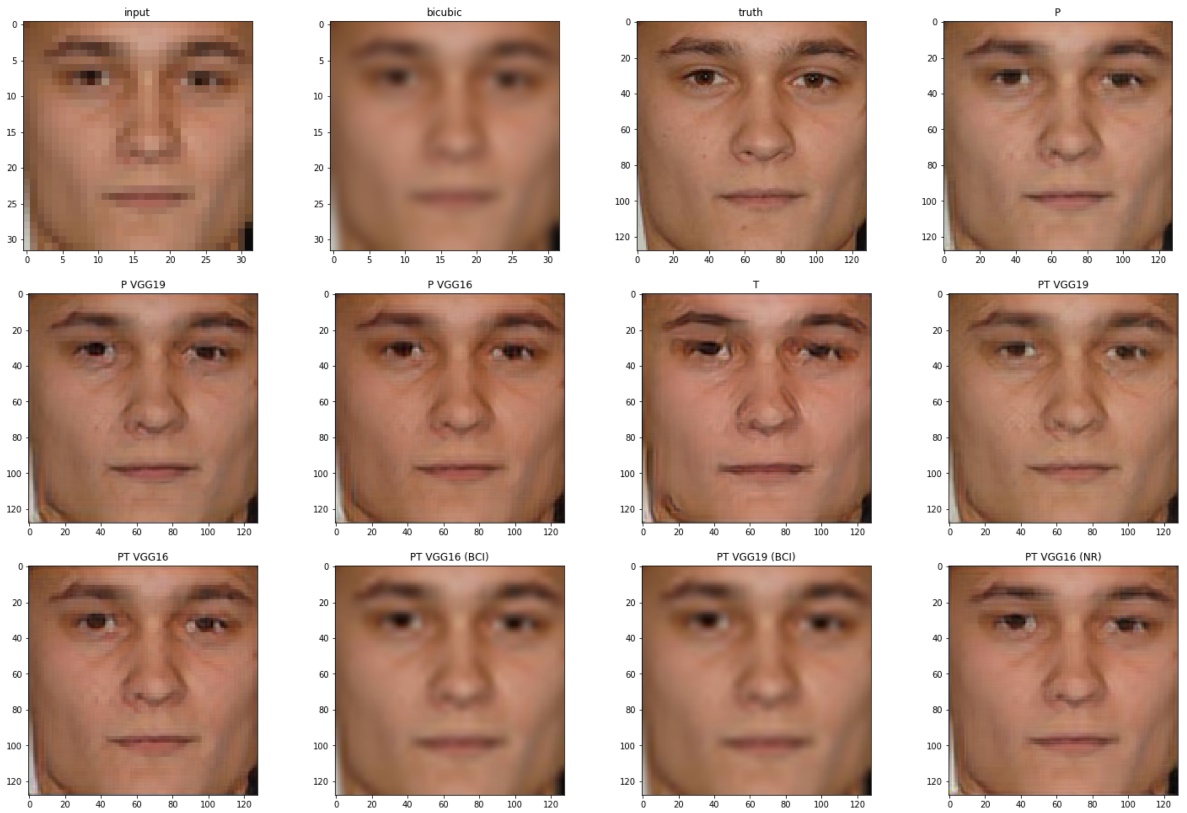
MS COCO initial evaluation. Image 5.



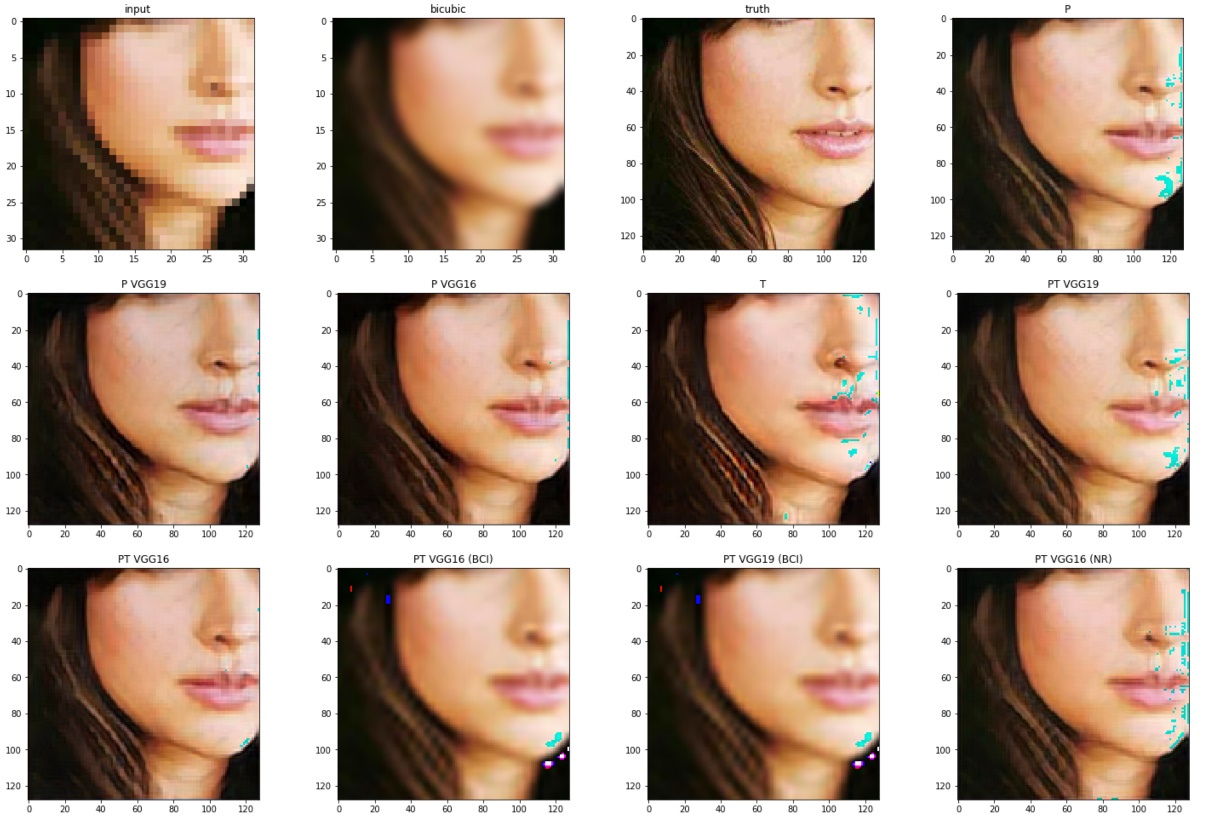
MS COCO initial evaluation. Image 13.

For the initial evaluation of CelebA data set I used the same parameters as for the MS COCO, except for this one:

use\_dataset\_celeba = True



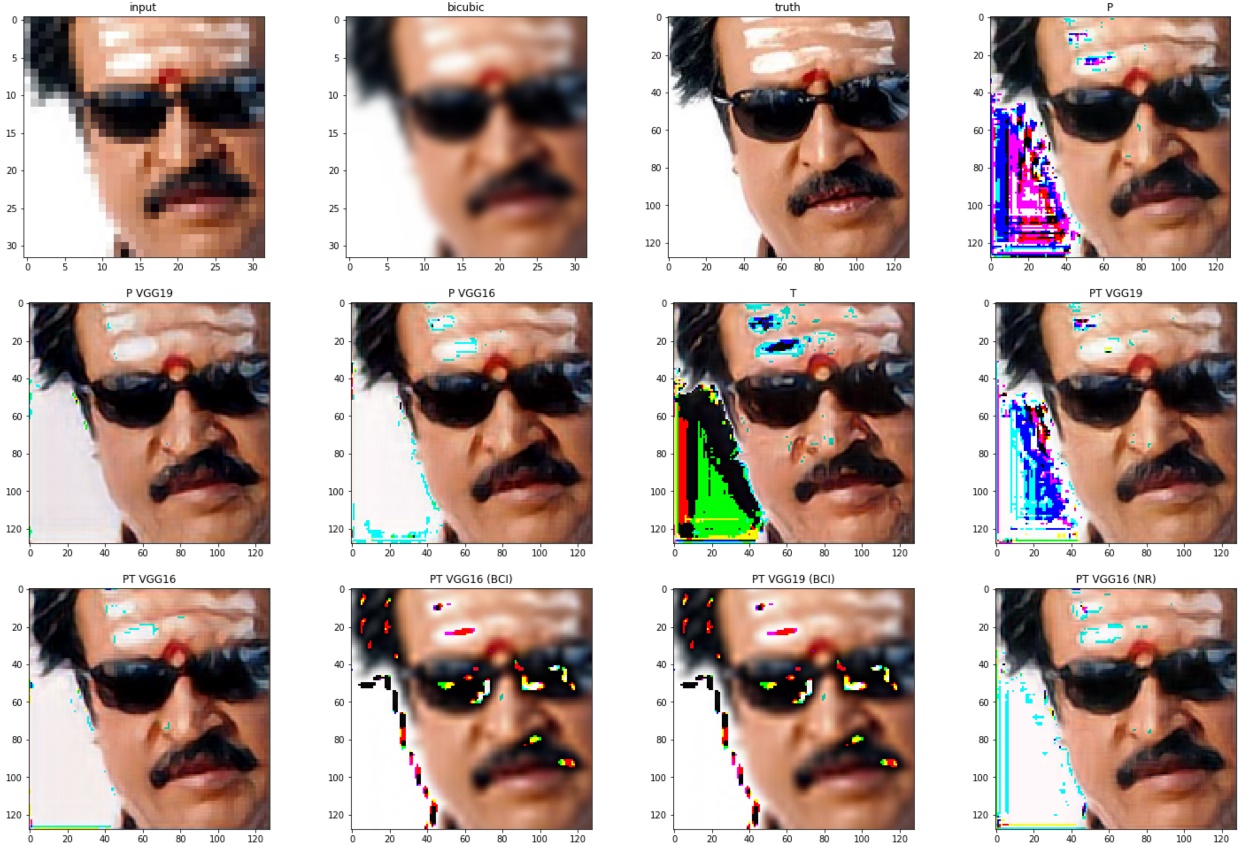
CelebA initial evaluation. Image 3.



CelebA initial evaluation. Image 4.



CelebA initial evaluation. Image 5.



CelebA initial evaluation. Image 15.



CelebA initial evaluation. Image 18.

After the initial evaluation I decided to focus on only few models and train them on the full datasets. When I tried to train my model on ~250,000 images (MS COCO) I was not able because of not enough memory on my computer. So far, I was trying to load all of the images in the memory and then start the training. This is faster for training but it doesn’t work very well on large datasets and with not enough resources. I was using the ***fit()*** function in Keras. They have a ***fit\_generator()*** function, which is useful for large datasets. I implemented my generator (***DataGenerator***) which loads the data on batches. The drawback of using the generator is the GPU is not as well utilized as when all of the images are in memory.

Use the python script - image\_super\_resolution\_generator.py

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

* *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?*
* *Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?*
* *Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

**Refinement**

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

* *Has an initial solution been found and clearly reported?*
* *Is the process of improvement clearly documented, such as what techniques were used?*
* *Are intermediate and final solutions clearly reported as the process is improved?*

**IV. Results**

*(approx. 2-3 pages)*

**Model Evaluation and Validation**

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model’s solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

* *Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?*
* *Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?*
* *Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?*
* *Can results found from the model be trusted?*

**Justification**

In this section, your model’s final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

* *Are the final results found stronger than the benchmark result reported earlier?*
* *Have you thoroughly analyzed and discussed the final solution?*
* *Is the final solution significant enough to have solved the problem?*

**V. Conclusion**

**Free-Form Visualization**

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

* *Have you visualized a relevant or important quality about the problem, dataset, input data, or results?*
* *Is the visualization thoroughly analyzed and discussed?*
* *If a plot is provided, are the axes, title, and datum clearly defined?*

**Reflection**

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

* *Have you thoroughly summarized the entire process you used for this project?*
* *Were there any interesting aspects of the project?*
* *Were there any difficult aspects of the project?*
* *Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?*

**Improvement**

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

* *Are there further improvements that could be made on the algorithms or techniques you used in this project?*
* *Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?*
* *If you used your final solution as the new benchmark, do you think an even better solution exists?*

**Before submitting, ask yourself. . .**

* Does the project report you’ve written follow a well-organized structure similar to that of the project template?
* Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
* Would the intended audience of your project be able to understand your analysis, methods, and results?
* Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
* Are all the resources used for this project correctly cited and referenced?
* Is the code that implements your solution easily readable and properly commented?
* Does the code execute without error and produce results similar to those reported?

**References**

1. <https://arxiv.org/abs/1612.07919>