Code Description:

I had two datasets to compare:

- Flir Dataset(Real)
- Cvidea Dataset(Synthetic)

I used both Computer Vision and Deep Learning Algorithms to compare these two datasets. Pushed every thing on this github repo:
https://github.com/tanayag/comparing_synthetic_real_images

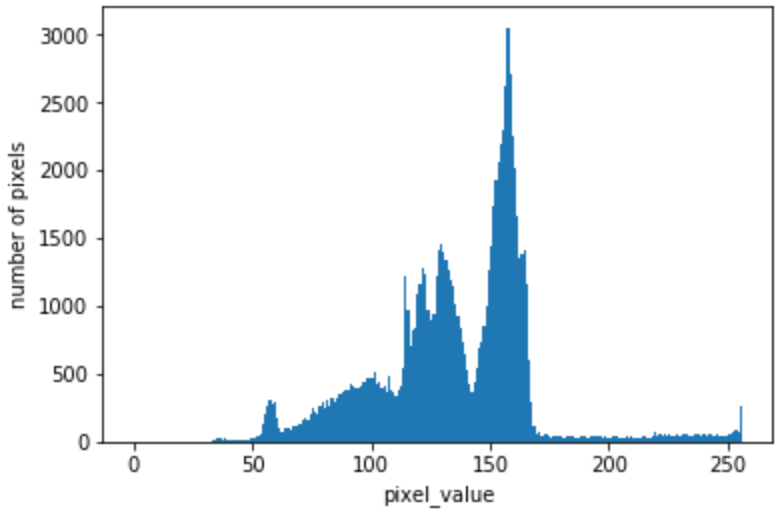
And names of the files are mentioned in the code description.

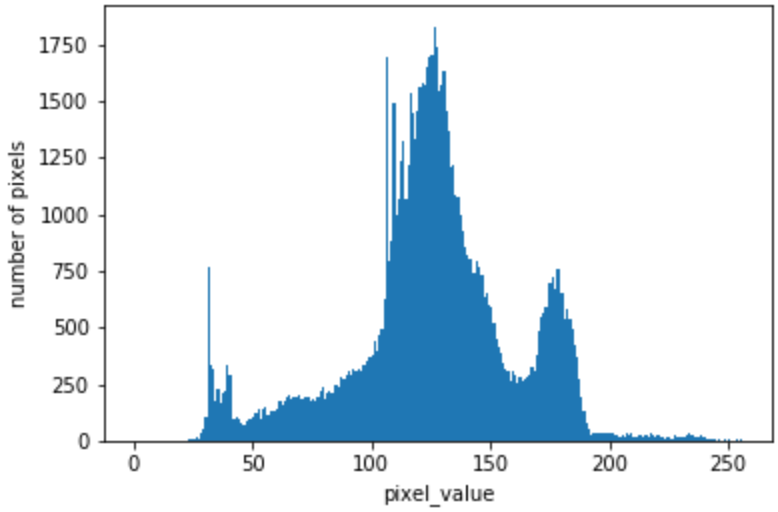
Computer Vision Algorithms:

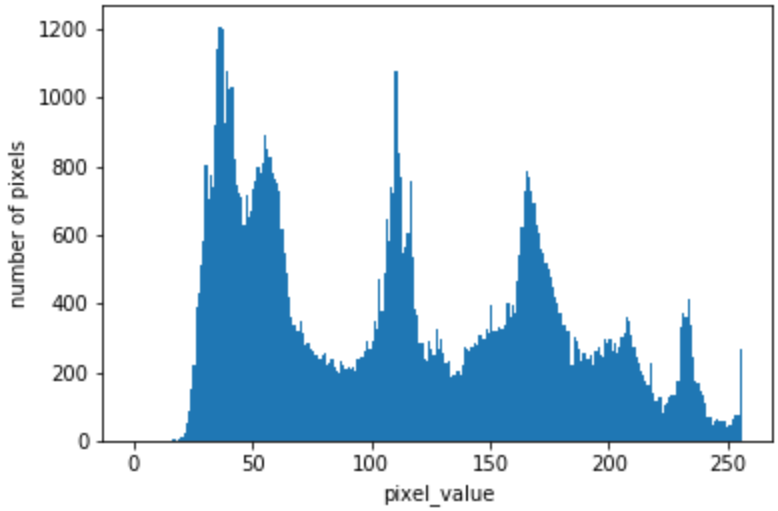
File named "vision_algos.ipynb" contains the use of computer vision algorithms,
I converted the image to grayscale to plot the histograms for pixel value vs number of pixels for those pixel values.
In the plots can be seen that the histograms of real data is more saturated as compared to synthetic data, there are extreme spikes in the histograms of synthetic data.
These are plots of some random images.

Eg.

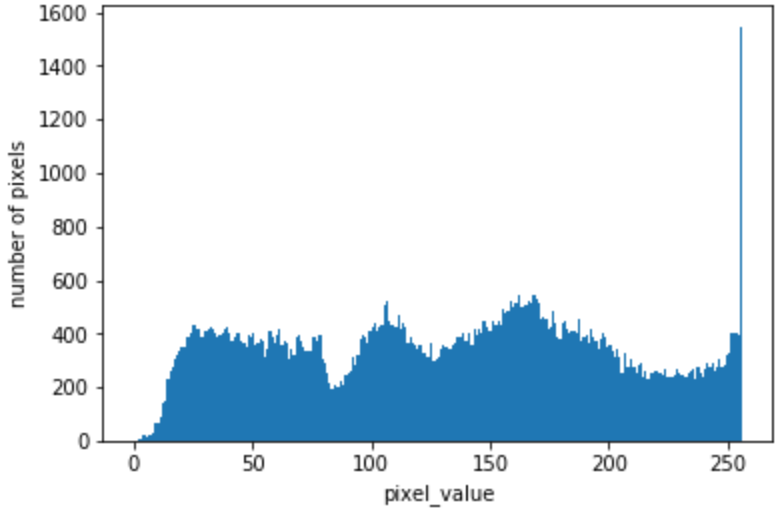
- **Synthetic data:**

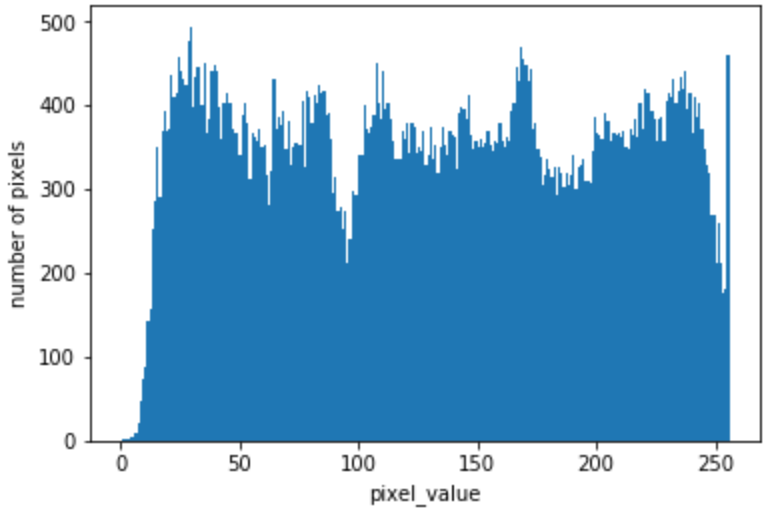


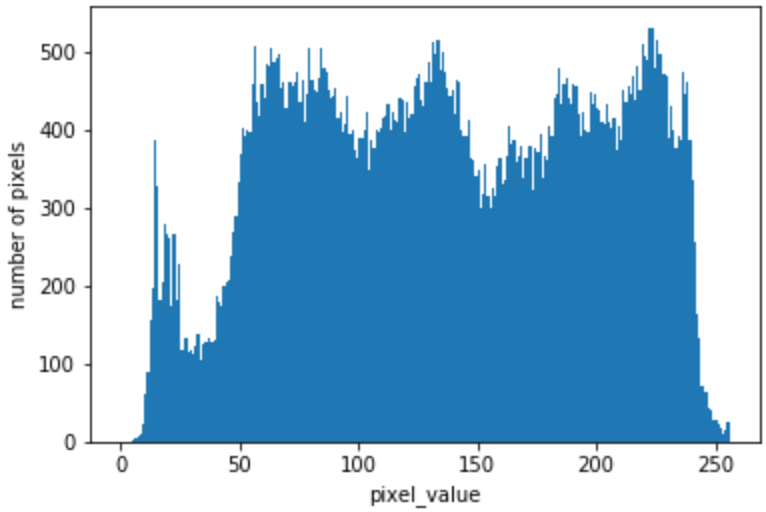


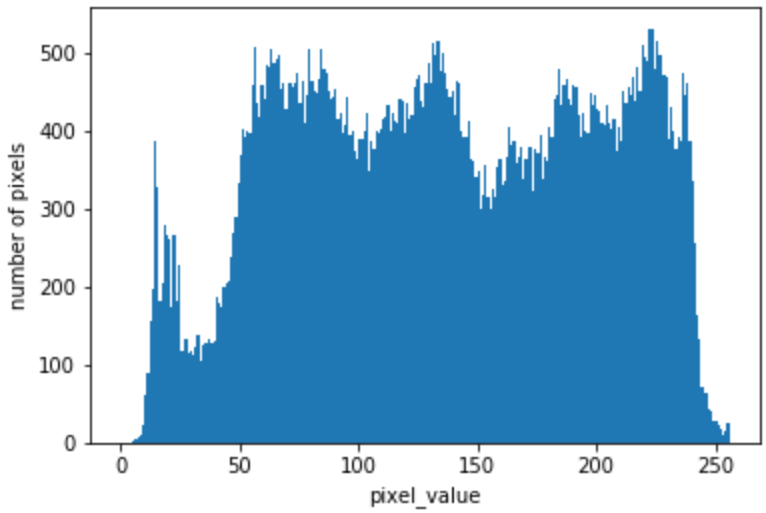

number of pixels
pixel_value
0
200
400
600
800
1000
1200
0
50
100
150
200
250

- **Real Data:**


number of pixels
pixel_value
0
200
400
600
800
1000
1200
1400
1600
0
50
100
150
200
250

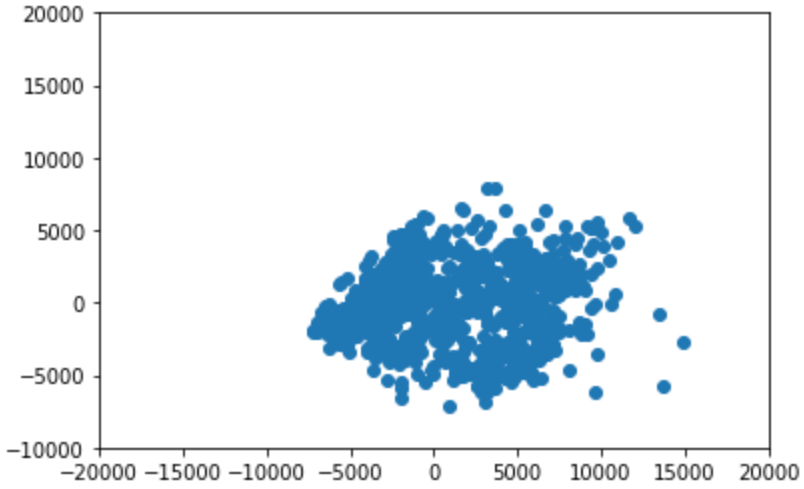

number of pixels
pixel_value
0
100
200
300
400
500
0
50
100
150
200
250


number of pixels
pixel_value
0
100
200
300
400
500
0
50
100
150
200
250

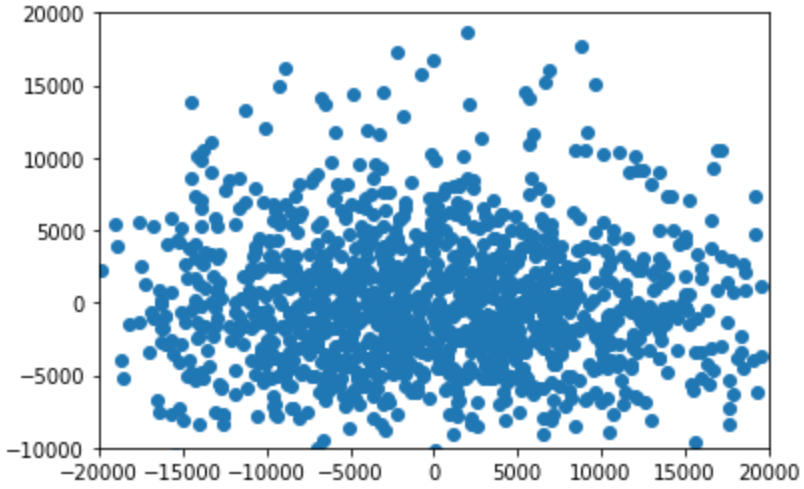

number of pixels
pixel_value
0
100
200
300
400
500
0
50
100
150
200
250

It can be seen in the plots for both synthetic and real images that, that saturation in real image is greater than that in synthetic images, extremities can be seen in synthetic data.
I also did a PCA to reduce dimensionality to 2 and plot a scatter:

- Real data



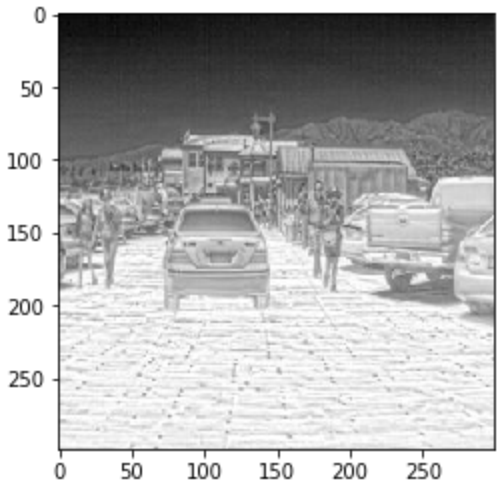
- Synthetic data



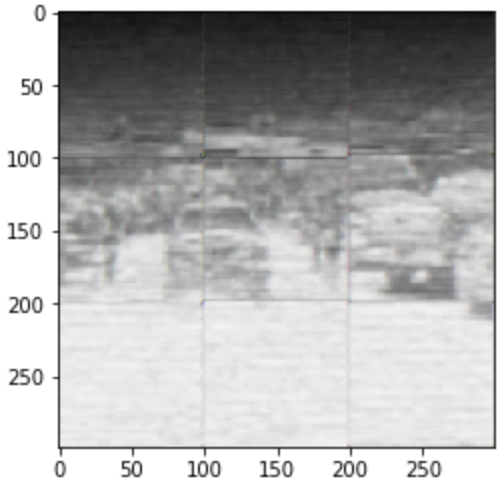
Both the graphs are plotted on same scale, and it clearly shows, real data is much more Gaussian than synthetic data and maybe this is the reason, real data performs better while training algorithms. But PCA is a statistical method, no learning involved. And from 299*299*3 to 2, a large amount of information loss. So, maybe bottleneck of autoencoders could help.

Deep Learning Methods:

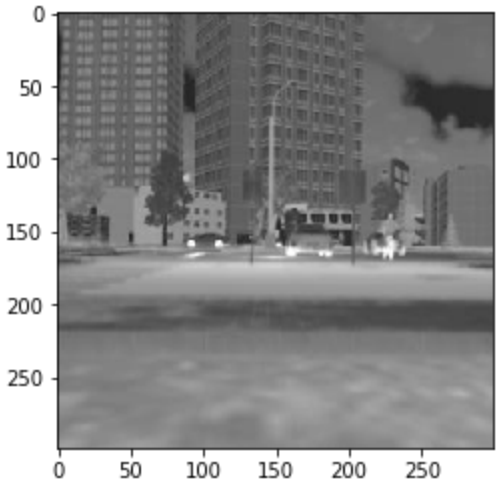
- **AutoEncoders**
  I trained two convolutional autoencoders for both real and synthetic data in file ”autoencoder_training.ipynb”, and used the bottleneck to reduce the features.
  Now, I wanted to converge the bottleneck to two units, but I was testing other model which converges to 8*37*37, so applied PCA, which is much better then applying PCA to 3*299*299.
  So due to computational constraints, I didn’t trained another model with bottle neck of 2 units. But the autoencoders were trained pretty good.



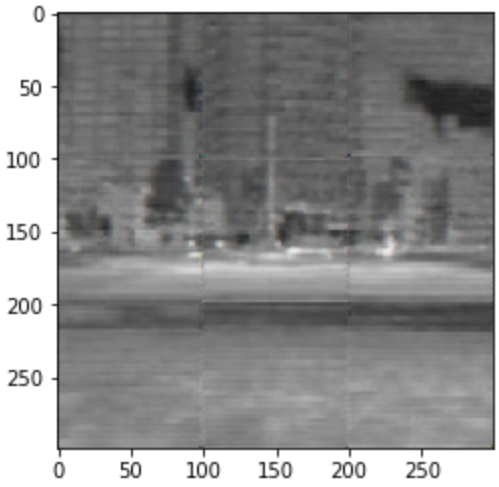
Real Image



Real Image AutoEncoder output



Synthetic Image



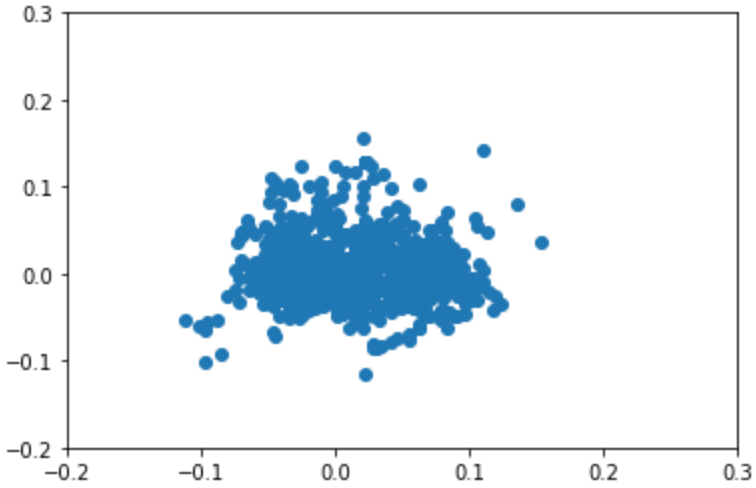
Synthetic Image AutoEncoder Output

Again I could have trained them from more epochs but computational constraints.

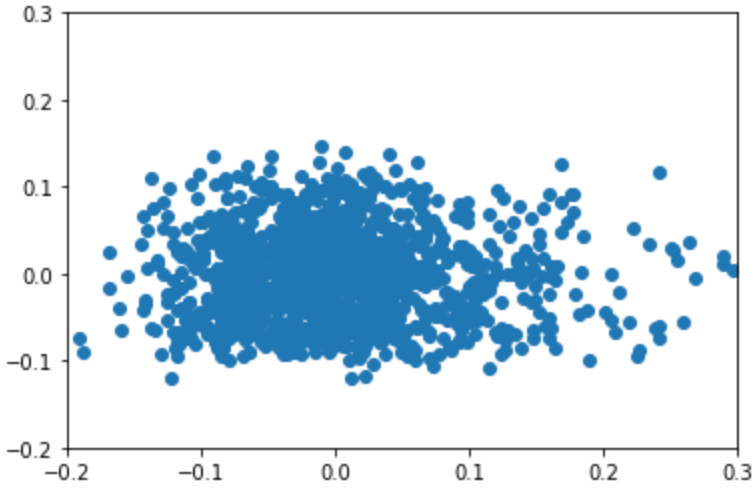
In file “saving_output_autoencoder_bottleneck.ipynb” I have loaded the models and saved the tensor output of bottleneck to pickle files: “real_output_encoder.pkl” and “synthetic_output_encoder.pkl”.

So, I used the bottleneck and plotted these two graphs in this file "using_autoencoder_output.ipynb". Plots are as follows

- Real Data



- Synthetic Data



And again we see the same pattern for scatter plotted on same scale.

- **Binary Classifier for Real and Synthetic images**

  I trained a binary classifier on VGG16 network(used weights of imagenet for weight initialization), in order to visualise the feature maps, for real and synthetic images and get more insight.
  Code can be found in file "training_vgg_classifier.ipynb"

  Feature Maps for ReLU Layers at position, (3,8,15,22)

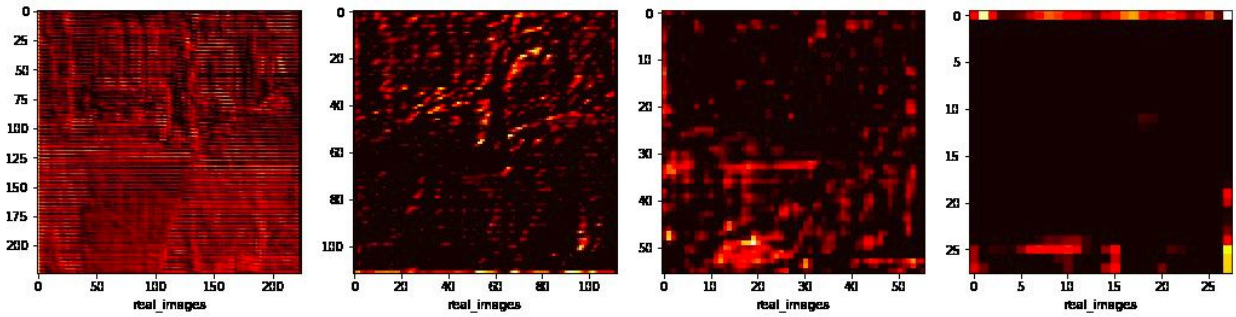


Fig.1 - Real Image

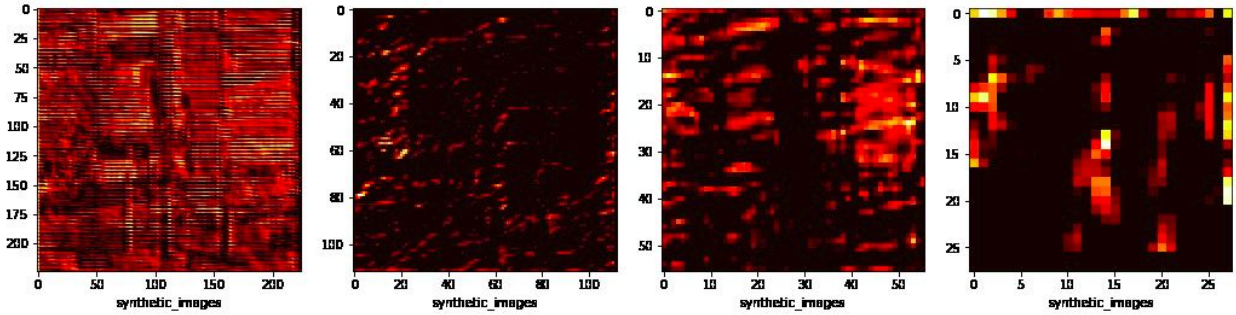


Fig.2 - Synthetic Image

Feature Maps for Conv Layers at position, (2,7,14,21)

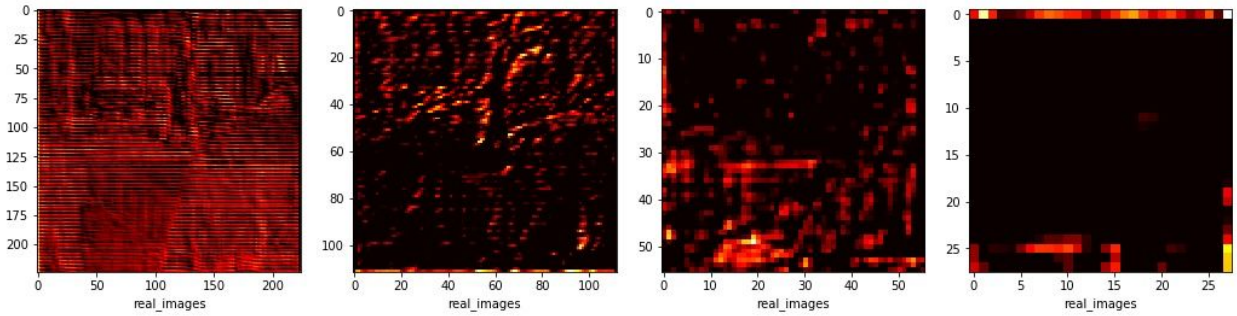


Fig.3 - Real Image

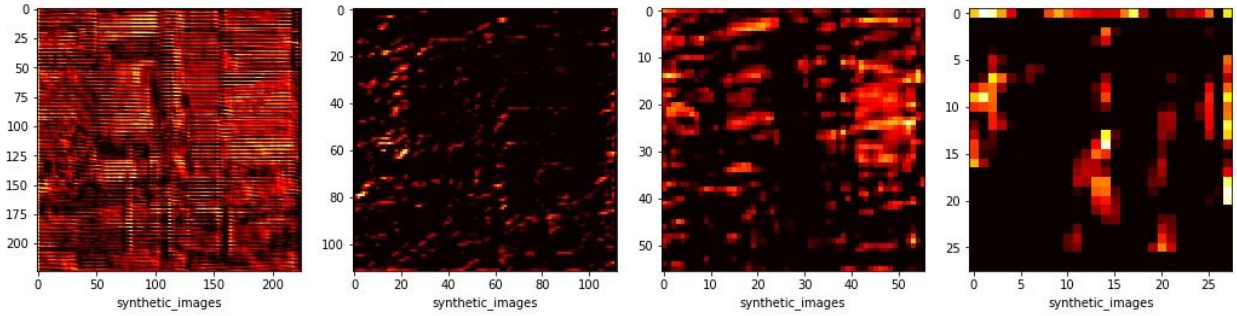


Fig.4 - Synthetic Image

Feature Map in Fig.1 and Fig.3 are on same Image, similarly Fig.2 and Fig.4 are conv and relu feature maps on same synthetic Image.

Conclusion:

In the process, we can see how Real and Synthetic data differs and of course its a difficult task to use synthetic dataset in training and use models in real life scenario.
But the problem is solvable .

Many techniques can be used to solve this problem,
Neural Style Transfer(https://arxiv.org/pdf/1508.06576.pdf) otherwise one of the approachs would be GANs, we can train a conditional GANs Network subjected to synthetic data, but that would require a lot of labeled data.
A better approach is training the CycleGANs(https://arxiv.org/pdf/1703.10593.pdf), where we can train on unpaired networks.