Spatially-Adaptive Pixelwise Networks for Fast Image Translation

Melaku Getahun [1] **Dawit Sefiw** [1] **Mekan Hojayev** [1] **Oluwafemi Adejumobi** [1]

Abstract

This project explores the use of a new generator architecture for an efficient high resolution image to image translation tasks. We replicated the results of ASAPNet(A Spatially Adaptive Pixelwise Network) paper by training the generator and discriminator networks from scratch. We evaluated the performance of the model on the cityscapes dataset using the inference time, Frechet Inception Distance and mean Intersection-over-union metrics and compared our results to those reported by the authors. We also investigate the possibility of using the same generator network for denoising.

Github repo: https://github.com/MelakuNe/Machine-Learning-Project-2023
Presentation file: https://github.com/MelakuNe/Machine-Learning-Project-2023/blob/main/ASAPNet.pdf

1. Introduction

Image translation from one domain to other has been a focus of research in machine learning and computer vision for years. Recent advancements in the area have resulted in the development of conditional Generative Adversarial Networks (GANs) that can learn a direct mapping between different image domains. However, these models are computationally expensive, especially when operating on high-resolution images. To address this issue, a novel architecture called ASAP-Net (A Spatially-Adaptive Pixelwise Network) (Shaham et al., 2021) that uses a new generator designed for fast image-to-image translation is proposed. The generator operates pixelwise, processing each pixel solely using a structurally simplified Multi-Layer Perceptron (MLP) that is adaptive to the input image. Generating high-quality images at very short execution times is proven to be possible by this architecture. This model is evaluated in various image to image translation domains such as segmentation maps and indoor images and comparison is made with state of the art models. In this work we would like to make sure the achievement and result mentioned in the paper is fully reproducible and direction of adapting this architecture for other image restoration purpose like denoising of an image. Image denoising is a critical step in many practical applications, as it involves recovering a clean image from a noisy observation. Even though there are high performing and low complexity models in this domain such as DnCNN, proving the applicabilty of ASAPNets generator for denoising goes a long way. One of the related issues in this area is deblurring. ASAPNet can leverage its low resolution processing technique to provide fast inference time compared to other tools that use generative models in this area.

We evaluate the model on the cityscape dataset to generate realistic city scenery from semantic segmentation maps. In addition to that, the expandibility of the proposed generator network is tested for image denoising task. In general, while achieving an equally perceptible image quality, the newly proposed architecture proves to generate images in considerably lower inference time compared to existing state-of-the-art tools.

2. Related work

Image to Image Translation:
Conditional Generative Adversarial Networks (GAN) have brought tremendous new things in generative models including image to image translation. However, the translation of images from one domain using this model has several limitations including not able to work with high-resolution image and more realistic image generation is also a challenge. (Wang et al., 2018) proposed a generator network composed of two sub-networks called global generator network and local enhance network. The former one is trained with low resolution images then the result is appended to the later network and it is trained together with high resolution images. On the other hand multi-scale discriminator with 3 sub-networks is proposed that operates at different image scales to improve the limitation of discriminator challenge of differentiating high resolution images and synthesized images. Mostly, after convolutional batch normalization

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Melaku Getahun <melaku.getahun@skoltech.ru>, Dawit Sefiw <dawit.sefiw@skoltech.ru>, Mekan Hojayev <mekan.hojayev@skoltech.ru>, Oluwafemi Adejumobi <oluwafemi.adejumobi@skoltech.ru>.

Final Projects of the Machine Learning 2023 Course, Skoltech, Moscow, Russian Federation, 2023.

layer is used and it has lack of representing the semantic information well in fact it degrades this important feature of an image. (Park et al., 2019) proposed a conditional normalization layer consisting of Batch Normalization and Adaptive Instance Normalization. When performing image to image translation task the semantic label is projected to an embedding space and convolution operation is applied to produce γ and β which has a spatial information in it unlike the common batch normalization layer with γ and β just a vector.

Denoising:
Image restoration has been also one area of interest in deep convolutional network research era, and one of the main area is removing noise from an image given noisy data. Convolutional neural network have been used widely to study the noise natural distribution in a given data, however ther is alwyas a bottlneck that these model are trained with a specific noise level which makes it not robust enough to handle not predefined noise level data. (Zhang et al., 2017) proposed a denoising CNN model, where VGG architecture is adopted in such a way that it is suitable for denoising purpose. To improve the denoising performance and reduce the training time residual learning is incorporated with batch normalization. The model is tested with any different dataset and has achieved between result than previous work, in addition to that since the model is suitable for parallel computation it has shown good inference time for an image of 512x512 with unknown noise level it took 60ms.

3. Spatially Adaptive Pixelwise Networks

This section discusses the design of Spatially Adaptive Pixelwise Networks. The model is designed to be input-adaptive, spatially-varying, and able to synthesize high-frequency details. The design involves using a Multi-Layer Perceptron (MLP) with ReLU activations to model:

$$f_p(x_p, p) = f(x_p, p; \varphi_p) = y_p$$

where f denotes the MLP architecture — shared by all pixelwise functions — and φ_p are the (spatially-varying) weights and biases for the MLPs. Each pixelwise function f_p takes as input the pixel's coordinates p, in addition to its color value x_p.

We found that the MLPs are with 5 layers and 64 channels per layer provide fast execution and good visual quality, the output as a spatially-varying pointwise nonlinear transformation of the high-resolution input, where each pixel is independent of the others. To preserve spatial information, each pixelwise function takes as input the pixel's coordinates, in addition to its color value. The pixelwise functions are parameterized with spatially-varying parameters that are conditioned on the input image. The parameter vectors are predicted by a convolutional network that operates

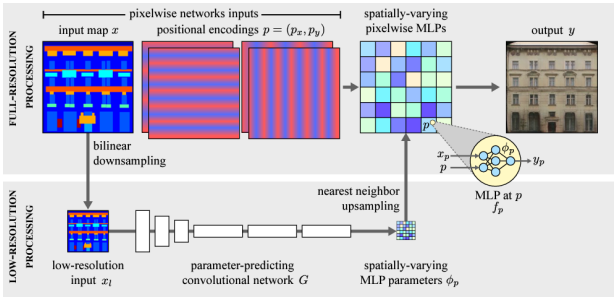

Figure 1. Architecture overview. The model first processes the input at very low-resolution, x_l, to produce a tensor of weights and biases φ_p. These are upsampled back to full-resolution, where they parameterize pixelwise, spatially-varying MLPs f_p that compute the final output y from the high-resolution input x. In practice, we found nearest neighbor upsampling to be sufficient, which means that the MLP parameters are in fact piecewise constant.

on a much lower resolution image. The model also uses positional encodings to synthesize high-frequency details. The training procedure follows best practices for the optimization procedure, including training the generator model adversarially with a multi-scale patch discriminator, using an adversarial hinge-loss, a perceptual loss, and a discriminator feature matching loss.

4. The proposed denoising CNN model

In this section, we present the proposed denoising CNN model, i.e., DnCNN, and extend it for handling several general image denoising tasks. Generally, training a deep CNN model for a specific task generally involves two steps: (i) network architecture design and (ii) model learning from training data. For model learning, we adopt the residual learning formulation, and incorporate it with batch normalization for fast training and improved denoising performance. Finally, we discuss the connection between DnCNN and TNRD, and extend DnCNN for several general image denoising tasks.

A. Network Depth
Following the principle, we set the size of convolutional filters to be 3×3 but remove all pooling layers. Therefore, the receptive field of DnCNN with depth of d should be $(2d + 1) \times (2d + 1)$. Increasing receptive field size can make use of the context information in larger image region. For better tradeoff between performance and efficiency, one important issue in architecture design is to set a proper depth for DnCNN. It has been pointed out that the receptive field size of denoising neural networks correlates with the effective patch size of denoising methods.

B. Network Architecture
The input of our DnCNN is a noisy observation $y = x + v$. Discriminative denoising models such as MLP and CSF aim to learn a mapping function $F(y) = x$ to predict the

latent clean image. For DnCNN, we adopt the residual learning formulation to train a residual mapping $R(y) \approx v$, and then we have $x = y - R(y)$. Formally, the averaged mean squared error between the desired residual images and estimated ones from noisy input:

$$L(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} ||R(y_i; \Theta) - (y_i - x_i)||_F^2$$

can be adopted as the loss function to learn the trainable parameters Θ in DnCNN. Here $(y_i, x_i)_{i=1}^N$ represents N noisy-clean training image (patch) pairs. Fig. illustrates the architecture of the proposed DnCNN for learning $R(y)$.

In the following, we explain the architecture of DnCNN and the strategy for reducing boundary artifacts.

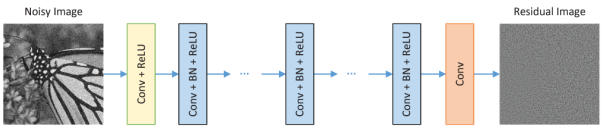

Figure 2. The architecture of the proposed DnCNN network

5. Experiments

We validate the performance of the proposed method for image segmentation and denoising task and compare inference time and performance metrics achieved by this model with their corresponding baseline models.

Baselines. In image segmentation, the performance of ASAPNet model is put in comparison with pix2pixHD, a model which takes a residual learning approach that enables high-resolution image translation.

Datasets. For the segmentation task, We used the Cityscapes dataset, which contains images of urban scenes and their semantic label maps, split into 3000/500 training/validation images. We construct 256 × 512 versions of this dataset. Denoising experiment is performed using the Berkeley Segmentation Dataset(BSD) splitted into 432/68 training/validation images. For the training data, a synthetic noise with different standard deviations of noise level ranging from 10 to 55 is added and validation data is also influenced by a synthetic noise with standard deviation of noise level 25.

Speed: The inference time of all models is benchmarked on NVIDIA GeForce GTX 1080ti. The ASAPNet paper generalizes that their model's inference time is 2 - 6 × faster than pix2pixHD. For the experiment we did with cityscapes dataset of size 256×512, we observed that ASAPNet takes 18ms to generate an image while it is 51ms for pix2pixHD, which approximates to a 3× faster inference time.

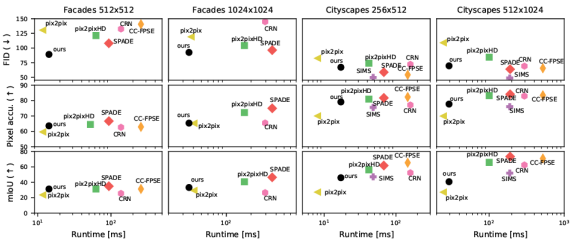

Figure 3. Comparison of the performance of ASAPNet to the pix2pixHD baseline on image segmentation using inference time, FID(lower is better) which measures the similarity between the distributions of real and generated images, and mean intersection over union(higher is better).

Qualitative comparisons: Examples of translated images are shown in figure . It can be deduced that images generated using ASAPNet have a visual quality at least comparable to the baseline.

Quantitative comparisons: Even though a specific numerical result of FID and mean intersection over union for the specified datset and size is not mentioned on the paper, results they provided in graphical form is demonstarated for comparison in figure. The authors have visualized that ASAPNet achieves a better FID score that pix2pixHD, while our experiment shows an FID score of 70.6 and 65, respectively. Based on the experiment, the mean intersection over union of these models is 30 and 29, for ASAPNet and pix2pixHD, respectively, while figure illustrates the latter is better.

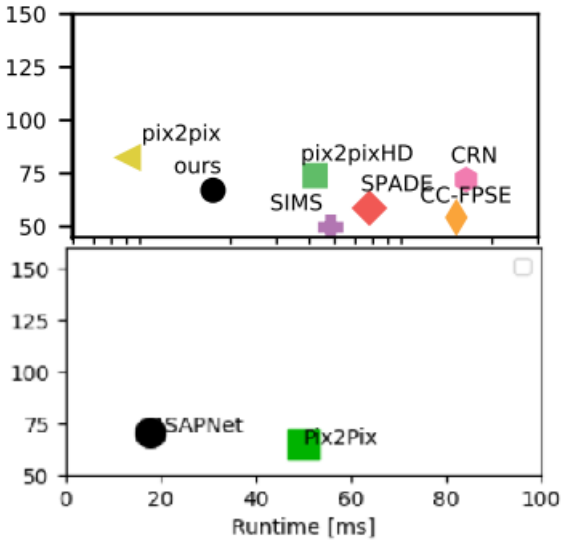

Figure 4. FID comparison of paper and replication work

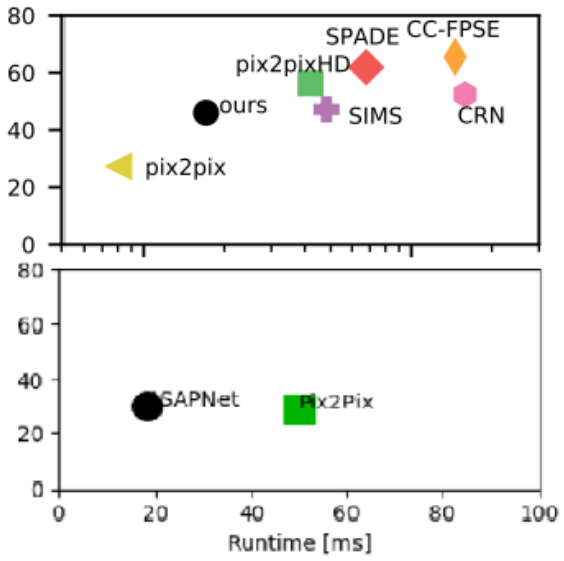

Figure 5. mean IoU of paper and replication work

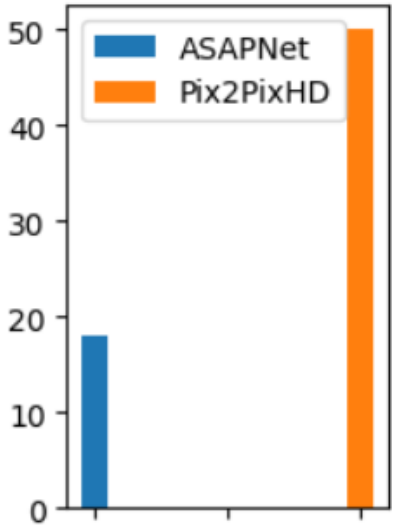

Figure 6. Inference Time

6. Conclusion

The replication tasks involved downloading and preparing the Cityscapes dataset, training the proposed ASAPNet model for segmentation and comparing the results with the values obtained by the authors in the paper. The results were compared with the pix2pix model, and inference time and GPU specifications were reported. The ASAPNet model was found to be an order of magnitude faster than previous work, while maintaining high visual quality.

The research tasks involved using the proposed generator network for non-blind denoising, training a DnCNN model as a baseline for the same data, and comparing the results with the ASAPNet model. The results were reported in terms of PSNR and SSIM metrics for the validation data, as well as inference time. The generator architecture of ASAPNet was found to be effective for denoising tasks, and outperformed the DnCNN baseline in terms of both performance and speed. The use of a pixelwise network was found to be a good idea for this task. This project replicates a novel neural architecture for generative adversarial image-to-image translation problems that is faster than previous works, yet maintains the same high level visual quality. We have also experimented on the applicabilty of the generator architecture on other image to image translation tasks such as denoising and deblurring.

Figure 7. For training, a synthetic noisy image is created with $y = x + \sigma n$, $n \sim N(0, I)$, with different σ of noise level ranging from 10 to 55. And for validation, the same type of synthetic image is created with σ of 15, 25, and 50. Peak signal to noise ratio (PSNR) of 28.55 is achieved.

References

Park, T., Liu, M.-Y., Wang, T.-C., and Zhu, J.-Y. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2337–2346, 2019.

Shaham, T. R., Gharbi, M., Zhang, R., Shechtman, E., and Michaeli, T. Spatially-adaptive pixelwise networks for fast image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14882–14891, 2021.

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., and Catanzaro, B. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8798–8807, 2018.

Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Melaku Getahun (25% of work)

- Initially put forward the idea
- Training the replication part
- Experimenting with model parameters
- Preparing the GitHub Repo
- Preparing the Section 2 of this report
- Creating presentation

Dawit Sefiw (25% of work)

- Training the replication part
- Experimenting with model parameters
- Preparing the GitHub Repo
- Preparing the Section 3 of this report

Mekan Hojayev (25% of work)

- Training the research part
- Redacting presentation
- Preparing the GitHub Repo
- Preparing the Sections 4 of this report

Oluwafemi Adejumobi (25% of work)

- Training the research part
- Experimenting with model parameters
- Preparing the GitHub Repo
- Preparing the Section 5 of this report

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   General comment: If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   Students' comment: None

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☐ Yes.
   ☐ No.
   ☑ Not applicable.

   Students' comment: because we were asked to use the paper parameters, so no selection of hyperparameters.

9. The exact number of evaluation runs is included.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.

- [x] Yes.
- [ ] No.
- [ ] Not applicable.

Students' comment: None