Image Colorization

Karan Dhingra
MT19025

ABSTRACT

Image colorization is an ill-defined problem, i.e. for a given grayscale input, multiple color input outputs are possible. This project extends the barebone conditional autoregressive network towards faster training, evaluation, and learning to color from thermal input instead of grayscale.

1. PROJECT DESCRIPTION

In Image Colorization, the aim is to generate a colorful image given a grayscale input (as shown in fig: 3). The standard neural network techniques(discriminative) learn to predict a fixed output given an input, thus limiting the Image colorization problem's solution. There have been recent advancements in the generative network, which predicts the output in a generative fashion i.e.

$$D_o = f(I) \tag{1}$$

$$D_g^i = f(I, D_g^{i-1}) \tag{2}$$

where, I is the grayscale input, and f represents neural network.

The output of discriminative network, D_o only relies on input, and as I is constant for a grayscale input, D_o will remain constant too. In case of generative network, D_g^i also depends on D_g^{i-1}, thus by playing with D_g^{i-1} we can have different values of D_g^i, and multiple outputs given a grayscale input.

One of the popular form of neural network to learn sequences is RNNs, but in case of images where sequence size can easily be in the order of 10^4, and it would loose spatial structure too which would increase the learning time, and reduce the learning capacity of the model.

While CNNs are more efficient than RNNs when working with images, it is not possible to use them for sequence learning as output of (i, j) pixel depends on neighbouring (m, k) pixels where only 49% of those neighbouring pixels belong to earlier timestamps, and remaining are the future pixels, which network would have to predict in the future.
Link : https://github.com/kdhingra307/pixelcolor

In PixelCNN[8], the authors solved it by creating a mask function which masks all of the neighbouring pixels, which would be computed in the future.

2. ARCHITECTURE

There are three different architectures used, all of them uses FCN network in different configuration,

- PixColor[4]:
  - The original architecture, had a pretrained segmentation network to compute the feature embeddings of grayscale images, followed by 6 convolutional layers to fix the spatial size of the image consumed by the PixelCNN network.
  - PixelCNN consisted of 10 resnet Masked convolution layer, as describe above.
  - It has now been extended, and has another 4 layers after the output of PixelCNN, and the purpose is to smooth the output of pixelcnn, by conditioning on future pixels too.
- IRTranslate
  - The grayscale input in case of GR2RGB also consitutes one of the important channel, as it provides the structure to the output image, which is absent in case of IR images.
  - Thus, the problems turns out to be a Image2Image type. It consists of another pretrained network, which is trained on IR images for object detection. The output from the pretrained network is used as feature representation for the input.
  - It is followed by a FCN, which increases the spatial resolution of the feature tensor to input's image size.
- IRGenerate
  - As, it would be demonstrated in the experiments that pairwise learning does not lead

to optimal results because of thermal images having less resolution to the details, and this leads to huge bias in the network.

  - Generative adversarial networks[3] are shown to work best in these kind of situation, where our samples do not align, spatially or in terms of resolution.
  - There are two components to it, One is Conditional generator which learns to generate RGB output from thermal input, and the other one is discriminator which is forced to classify generated samples as false, and input target samples as real.

- All of the networks uses ADAM optimizer with $1e-3$ initial learning rate for first two networks, and $1e-5$ for GANs. There are three possible loss functions, logistic mixture model, softmax cross entropy, mean square error. MSE is only used as baseline for translation problem only.
- Logistic Mixture Model[7], In case of PixelCNN it was easy to compute cross entropy as it was color channels are scaled down to the size of $(32, 32)$, with 32 discrete buckets instead of 256. It was required because of slow evaluation speed of PixelCNN. The same, can not be repeated for Image translation as we have to estimate the gray-scale (structure) of the objects, which causes the OOM error. With logistic mixture model, we remodel the output of mean, variance, and mixture strength of logistic distribution which allows us to learn distribution at relatively less memory cost.

3. DATASETS

3.1 Gray2RGB

- ADA20k[9]: It is a scene parsing benchmark dataset, which has 15k training and 1.5k validation samples. The other benefit of using ADA20k is that, there is a pre-trained Resnet-101 based segmentation algorithm which helps in faster training.
- OpenImagesExtended[2] : It is a crowd source dataset, which has over 4lakhs samples. Upon training the model on subset of this dataset, it became clear that we would need to retrain the initial embeddings too, that's results from this datasets are not mentioned.

3.2 IR2RGB

FLIR Thermal[1]: It provides thermal images with high spatial resolution, and there are over 8000 samples in the dataset, but there is no alignment between thermal and rgb images.

kaist multispectral benchmark[5] : It is divided into multiple files, with each file consisting over 10000 images, but it lacks the spatial resolution as compared to FLIR, however all of the results are from this dataset.

4. EXPERIMENTS

4.0.1 Experiment - 1

In this experiment, IRTranslate network is trained with MSE loss, as it acted as an replacement of cross entropy loss. As, it can be observed in the figure 1, network is unable to output any thing, and whatever we see corresponds to blurry average output, and initially it looked like it was because of MSE loss function, but later when the same experiment was performed with logistic mixture based loss function, similar (better than MSE, in fig:3) result highlighted the issue towards input image, and target image.

Figure 1: IRTranslate-Mean : Top(Output), Bottom(Target)

As it can be observe in top image of fig:3, (i) the alignment between thermal image, and of rgb is not accurate, and also thermal image lacks in spatial resolution which results in neural network biasing towards blurry out-

puts. If we observe the middle image of fig:3, it can be seen that network is not able to learn fine features, while it is able to color red (in the right) based on coarse features. In order to verify that there is no training error, the same network was trained again but this time it takes grayscale image as the input, instead of IR, and results can be seen in fig:2.

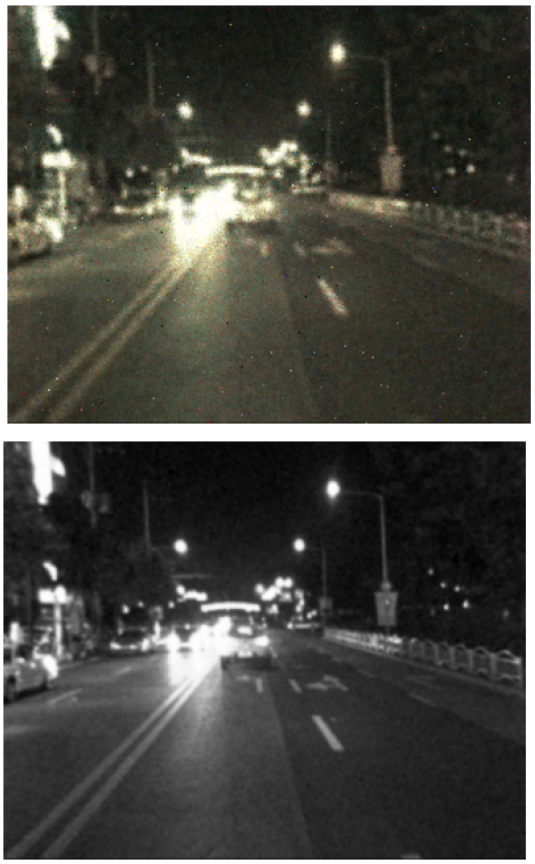

Figure 2: GrayTranslate-Mean : Top(Output), Bottom(Input)

Figure 3: IRTranslate-Logistic :Top(Input), Middle(Output), Bottom(Target)

The flir dataset, had better resolution than the kaist dataset but upon analysis of the dataset, it turned out that there are two critical alignment issues, (i) they have used multiple rgb cameras for shooting in different angles but there is only one thermal angle. Using homography techniques, a close enough pair between rgb and thermal was found for the resolution of (1800, 1200) rgb images but in further analysis, it turned out that both the camera is out of sync.

Thus, in order to further improve the performance of problem, translation network was swapped with a GAN based generative network as GANs does not need aligned images, and also generator wont penalize for less spatial resolution because instead generator tries to increase the resolution in order to fool the discriminator, as seen in fig:4. The results from GANs were not complete, as I lost my trained model, which had better result than translation model but I lost my trained model,

and while retraining I could not achieve the same performance as the initial trained network was not stable.

Figure 4: IRGenerative : Top(Output), Bottom(Target)

4.1 Experiment - 2

The other component in this project was to improve the performance of pixelcnn model, and for that two changes based on other work related to pixelcnn have been analysed.

4.1.1 FCN after pixelcnn output

One of the reason, output of pixelcnn leads to artifact generation is because it is not prohibited from accessing future neighbour pixels, and also at generation time outputs are repressively computed which during training, a teacher enforcement is present which leads to error propagating easily during generation time. In order to mitigate both, a 5 layer FCN is present which is also minimized based on same loss function, and thus its function is to filter out any junk pixelcnn might get during regression.

Figure 5: PixelCNN(Cross entropy) :Top (Before final FCN), Bottom(After)

4.1.2 Logistic Loss function

It was proposed that, network has to first learn the color distribution because we are using softmax, and it easily leads to biasness over the color present in the training set. In order to mitigate both network is trained to learn logistic mixture model of the distribution. It allows unseen color to have effect in the output too.

Upon analysis6, it turned out that FCN have higher impact on the model which is trained to learn logistic mixture, and model with cross entropy has color bias

Figure 6: PixelCNN(Logistic) :Top (Before final FCN), Bottom(After)

towards yellow, but the model trained with cross entropy performs better in visual test as compared to logistic mixture, and it comes down to the fact that the model with cross entropy learns relatively faster because of pretrained feature embeddings, and because the distribution to learn was small already (of 32).

5. CONCLUSION

The generative output from thermal images, enable us to reverse map from thermal to RGB but because of being generative in nature, we cannot guarantee complete representation of the input thermal image. In pixelcnn, there is a technique to increase the evaluation speed by a factor of 183x[6] but this speed upon study is only achieve-able when we have batch sizes of 256-512, which is not possible under current problem formulation, and compute resources.

6. REFERENCES

[1] Free flir thermal dataset for algorithm training.

[2] Open images extended - crowdsourced.

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[4] S. Guadarrama, R. Dahl, D. Bieber, M. Norouzi, J. Shlens, and K. Murphy. Pixcolor: Pixel recursive colorization. *arXiv preprint arXiv:1705.07208*, 2017.

[5] S. Hwang, J. Park, N. Kim, Y. Choi, and I. So Kweon. Multispectral pedestrian detection: Benchmark dataset and baseline. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1037–1045, 2015.

[6] P. Ramachandran, T. L. Paine, P. Khorrami, M. Babaeizadeh, S. Chang, Y. Zhang, M. A. Hasegawa-Johnson, R. H. Campbell, and T. S. Huang. Fast generation for convolutional autoregressive models. *arXiv preprint arXiv:1704.06001*, 2017.

[7] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture. *ICLR*.

[8] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29:4790–4798, 2016.

[9] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

APPENDIX

A. LOSS FUNCTIONS

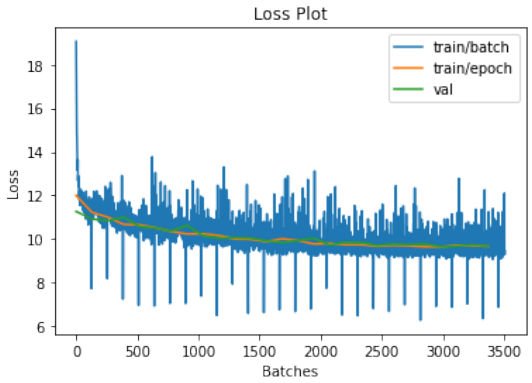

Figure 7: MEAN GRAYTranslate

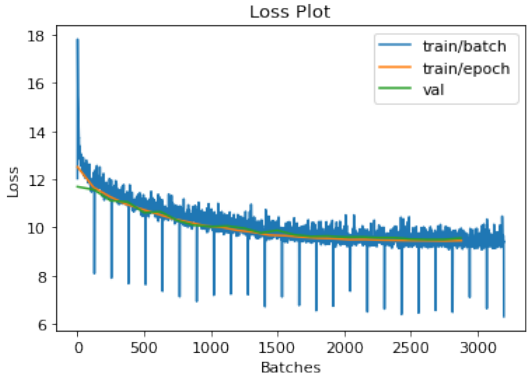

Figure 8: Logistic Translate

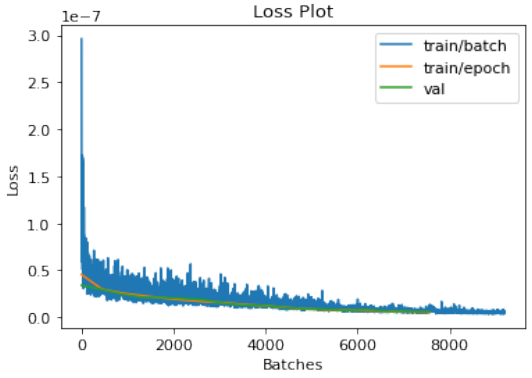

Figure 9: MEAN IRtranslate

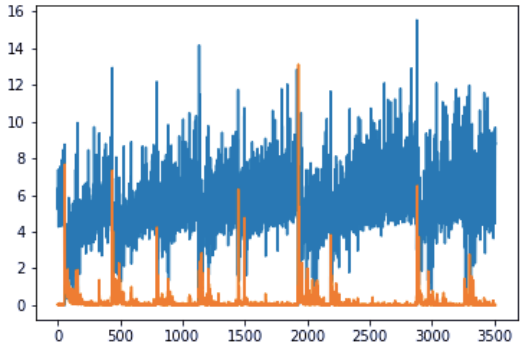

Figure 10: GAN Network