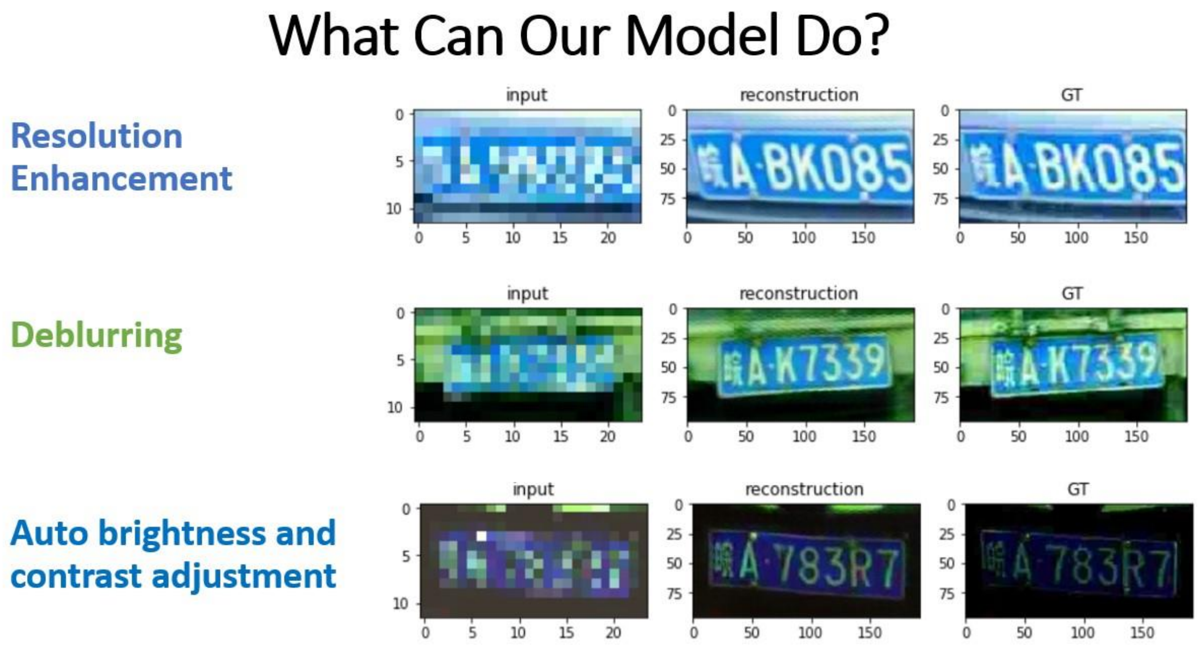
License Plate Enhancement - From TV shows to reality



What Can Our Model Do?

Resolution Enhancement

Deblurring

Auto brightness and contrast adjustment

Requirement

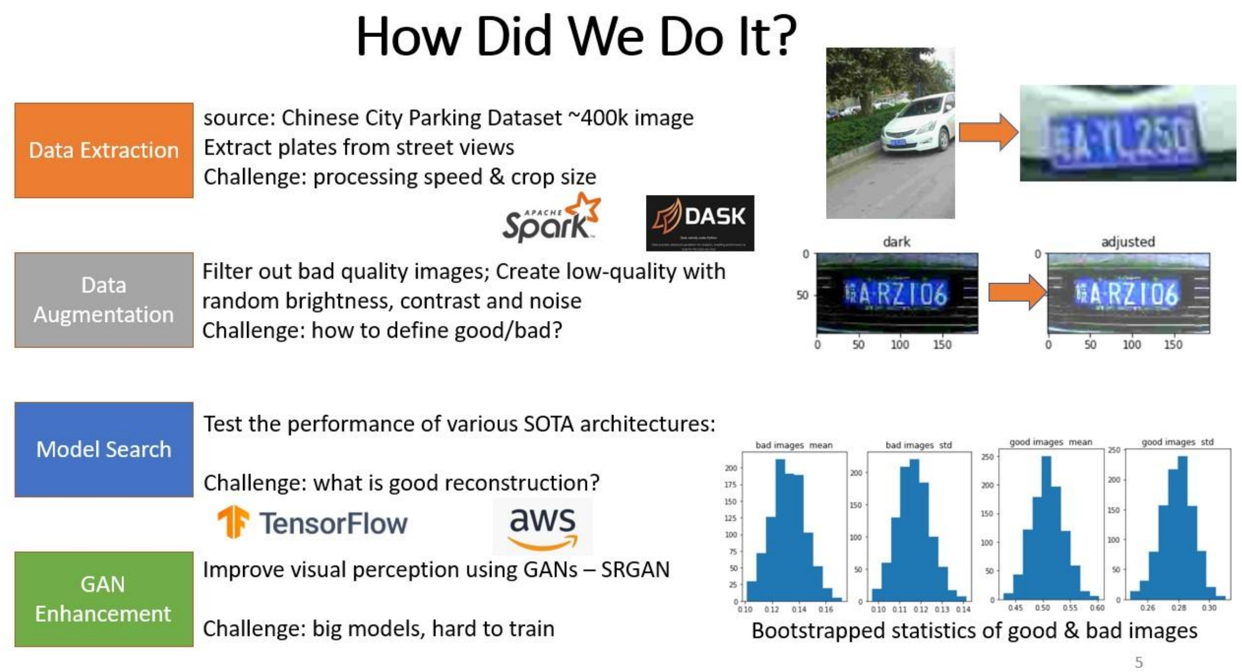
Preprocessing

- Dask >= 2.11.0
- PIL >= 6.2.2

Training & Evaluation

- tensorflow >= 2.1.0
- numpy >= 1.18.1
- matplotlib >= 3.1.3

Pipeline



How Did We Do It?

Data Extraction

source: Chinese City Parking Dataset ~400k image
Extract plates from street views
Challenge: processing speed & crop size

Data Augmentation

Filter out bad quality images; Create low-quality with random brightness, contrast and noise
Challenge: how to define good/bad?

Model Search

Test the performance of various SOTA architectures:

Challenge: what is good reconstruction?

GAN Enhancement

Improve visual perception using GANs – SRGAN

Challenge: big models, hard to train

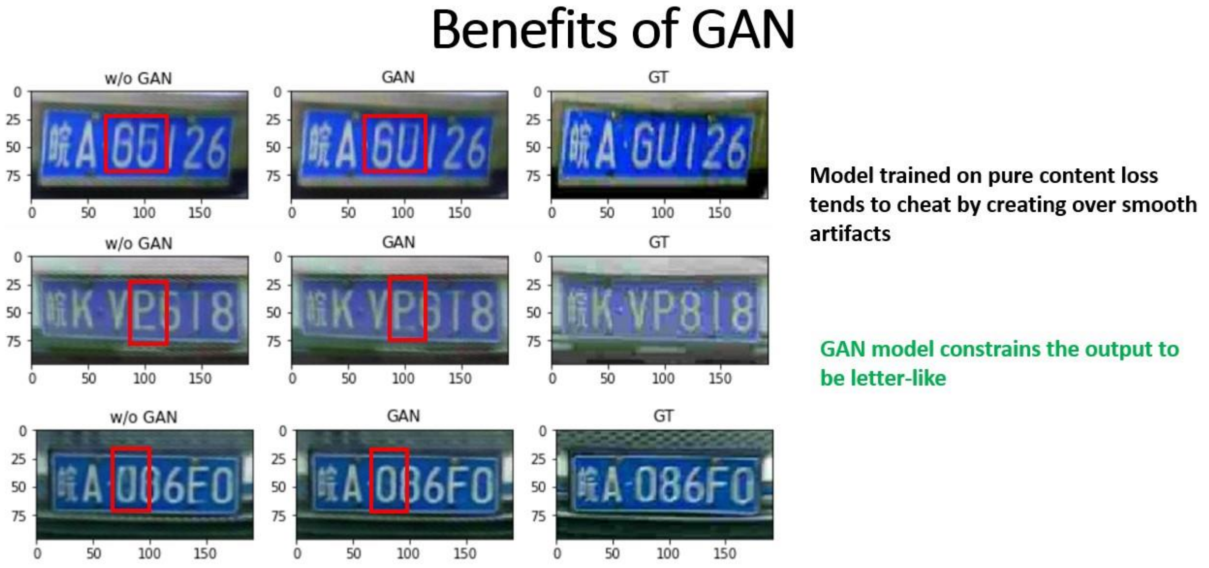
Bootstrapped statistics of good & bad images

5

Before training the model it is important to preprocess the raw dataset using the preprocess.py script

Model Architecture

Our plate enhancer model is trained in an adversarial fashion(GAN), meaning the generator is trained to create realistic reconstruction of images that can fool the discriminator, which is a binary classifier. Why GANs? Well, according to several papers, GAN network tend to create more realistic image reconstruction comparing to model solely trained in the supervised fashion. For instance, models that minimize Mean Square Error tend to have over-smoothing artifacts. Therefore, there are two models - the generator(reconstructor) and the discriminator(classifier).

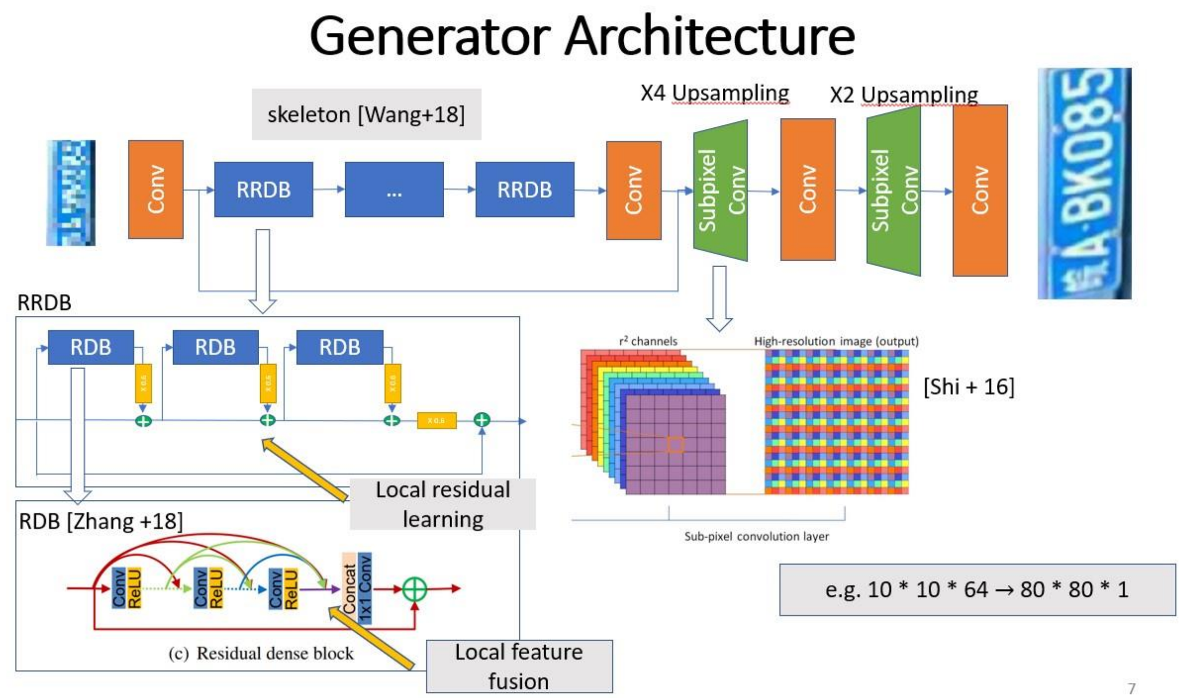


Benefits of GAN

Model trained on pure content loss tends to cheat by creating over smooth artifacts

GAN model constrains the output to be letter-like

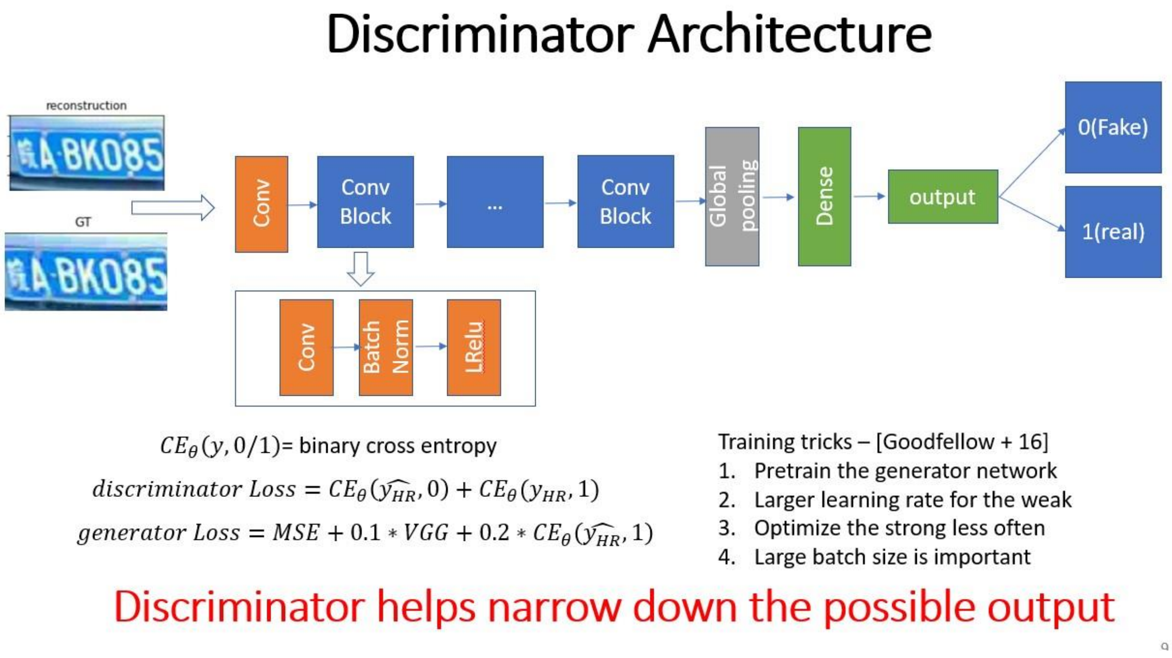
Generator



Generator Architecture

The generator is trained to minimize a novel hybrid loss function, namely the perceptual loss defined in the SRGAN paper

Discriminator



Discriminator Architecture

$CE_{\theta}(y, 0/1)$= binary cross entropy

$$discriminator\ Loss = CE_{\theta}(\widehat{y_{HR}}, 0) + CE_{\theta}(y_{HR}, 1)$$

$$generator\ Loss = MSE + 0.1 * VGG + 0.2 * CE_{\theta}(\widehat{y_{HR}}, 1)$$

Training tricks – [Goodfellow + 16]

1. Pretrain the generator network
2. Larger learning rate for the weak
3. Optimize the strong less often
4. Large batch size is important

Discriminator helps narrow down the possible output