LP-SRGAN: Improving license plate image resolution with GAN

Gustavo Carneiro Bicalho
Aeronautics Institute of Technology
Data Science Team Itau-Unibanco
São Paulo, Brasil
Email: gustavocbicalho@gmail.com

Felipe Coelho Silva
Data Science Team Itau-Unibanco
São Paulo, Brasil
Email: felipe@lodur.com.br

Mauri Aparecido de Oliveira
Aeronautics Institute of Technology
São Paulo, Brasil
Email: mauri@ita.br

***Resumo*—Automatic license plate recognition (ALPR) and single image super-resolution (SISR) are two complex tasks that have a lot in common. Being able to automatically detect and read the characters in a license plate requires to overcome many challenges, in particular the low resolution of cameras that capture those images. In this work we propose the use of a generative adversarial network (GAN) to improve the image quality of license-plates. This method can solve one of the task in ALPR, the size of the license plate in an image. This network, LP-SRGAN, can increase the resolution of a license plate image in 8 times. In this work we used a controlled environment to generated the license plates without any variation, other than low-resolution, that could affect the ALPR process. Results show that an optical character recognition (OCR) algorithm can't differentiate a generated image from an original one.**

***Index Terms*—Super-Resolution, Computer Vision, License Plate Recognition**

I. Introduction

Automatic license plate recognition (ALPR) is a complex task that has applications in the most diverse areas, be it traffic law enforcement, parking lot access control or road traffic monitoring [1]. This problem consist of being able to recognize a license plate number from an image, be it a black and white or colored photo, or even an infrared image.

In [2], it is shown that ALPR is affected by both plate and environment variation, that result in many challenges that still have no unanimous solution.

Plate variation challenges includes:

- **Size**: plates size may vary depending on the distance and zoom factor of the camera;
- **Location**: the plate location in the image may vary;
- **Quantity**: there may have more than one plate in the image;
- **Color**: plates may have various characters and background colors due to different plate types or capturing devices;
- **Font**: different countries may have different fonts for license plates;
- **Occlusion**: some plates may have occlusion caused by unpredictable event, such as dirt;
- **Inclination**: the image may not have the best angle to capture the license plate bounding box;

And, environment variation includes:

- **Illumination**: vehicles headlights, street lights or the time of day may affect the illumination of the license plate, making it difficult to extract information;
- **Background**: the image background may contain other patterns that can be recognized as license plates by detectors.

According to [1], ALPR has two stages:

- **License plate location**: locate the license plate in the image;
- **Identifying license plate numbers**: isolate the license plate numbers and letters and interpret them.

In this work, our method will focus on solving the license plate size challenge, and check if it helps an OCR read the license plate digits. And so, we are creating a dataset that abstract all of the other plate variations and assumes that we have already detected the exact bounding box of the license plate in the image.

To accomplish our objective, we must first understand how to improve an image resolution. The task of estimating a high-resolution (HR) image from its low-resolution (LR) counterpart is called super-resolution (SR). This task is receiving more attention from the computer vision community for its various applications [3], from satellites images to multimedia industry.

When we use classic upsizing methods, the image loses details that are crucial for the complete understanding of the image. In our case, if the OCR cannot read the upsized license plate, the upsizing method is worthless for the task. When using bicubic interpolation [4], in a 15×42 pixels license plate, instead of restoring it to the Figure 1, we can only achieve 2. While a human can still read the digits, an OCR can have trouble reading it, especially recognizing ambiguous characters such as (B-8), (O-0), (I-1), (A-4), (C-G), (D-O), (K-X), and broken characters [5].

With that in mind, in [6] it is proposed a super-resolution CNN, a deep convolutional neural network to increase the

[0]Any opinions, findings, and conclusions expressed in this manuscript are those of the authors and do not necessarily reflect the views, official policy or position of the Itaú-Unibanco.

resolution of an image. This approach opened doors to many other deep learning methodologies [7], [8]. But, when [9] came up with the GAN framework, and [10] applied it to super-resolution, it raised the bar in terms of quality for super-resolution images.

In this work, we propose an application for the SRGAN, the license plate super-resolution GAN (LP-SRGAN). This network can be a valid solution to one of the challenges of ALPR, as it improves the quality of license plate images.

Figura 1. A randomly created license plate.

Figura 2. Image upsized with bicubic interpolation.

Figura 3. A low resolution license plate example.

II. Related Works

In this work, we will focus only in single image super-resolution (SISR) and convolutional neural networks (CNN) approaches.

CNN based super-resolution algorithms are showing great performance when compared to old methods to this problem. Dong et al [6], [11], propose a SRCNN to learn the mapping from a low-resolution image to a high-resolution one in an end-to-end manner. A bicubic interpolation was used to upscale an input image and trained a three layer deep fully convolutional network, this approach achieved state-of-the-art SR performance when it was published. Additionally, methods that allowed the network to learn directly the upscaling filters could further increase performance, both in accuracy and speed [12]–[14]. Many perceptual-driven methods were also proposed to improve the visual quality of SR images. Both [8] and [7] used a loss function closed to perceptual similarity to recover images more similar to real high-resolution images.

In [9], the Generative Adversarial Network framework is proposed. This framework is basically a two-player minmax game between a Discriminator D and a Generator G. Both D and G are multilayer perceptrons, but D outputs a single scalar and G captures the data distribution from input noise variables z. $D(x)$ represents the probability that x is an original data, rather than one generated by G. D is trained to maximize the probability to correctly label to both training data and samples from G. G is simultaneously trained to minimize $log - D(G(z))$. That is, we have a minmax game with a value function $V(G, D)$:

$$min_G max_D V(D, G) =$$
$$\mathrm{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))] \quad (1)$$

GANs opened a new world of possibilities in terms of SR images. In [10] and [15], GAN usage for SR is firstly introduced. The first one propose SRGAN model that used perceptual loss and adversarial loss to favor an output closer to the representation of high-resolution images. The second one used a similar approach, but explored more on the local texture matching loss. In [16], it is proposed a spatial feature transform to effectively incorporate semantic prior in an image and improve the texture matching.

After that, many methods to further improve GANs came into play, in particular the usage of residual blocks [10]. [17] propose an architecture removing batch-normalization layers in the residual blocks and expanding the model size, which achieved significant improvement.

More recently, [18] improved the network proposed in [10] by employing a more effective relativistic average GAN. It was possible to enhance the performance of SRGAN with a study to improve the three key components of SRGAN: network architecture, adversarial loss and perceptual loss.

III. Methodology

As the main goal of this work is to verify if the LP-SRGAN can improve the resolution of license plate images, we are ignoring the detection and alignment of the license plate. For that we had to create our own data set. This chapter will describe how we created the dataset, how we built a GAN to improve the image resolution of images of 42×16 pixels, up to 336×128 pixels and how we evaluated if our model can generate images with similar quality from the ones created for the dataset.

A. Dataset

For our experiments, we created our own artificial dataset. Using the Pillow package for python and a template for Brazilian license plate, we created random license plate images containing three letters followed by four numbers. Usually Brazilian license plates have the city and state where it is from, but as the number is already an unique identifier, this text has been removed.

The images created are considered the high resolution images and have 336×128 pixels and will have the letters and numbers as their label. Then a copy will be downsized to

42×16. The Figure 1 shows an example of a generated license plate and the Figure 3. The choice to divide the image size by 8, was that it was the resolution which the OCR wouldn't be able to read any character of the image.

B. LP-SRGAN

The goal of the LP-SRGAN is to train a generator model G that can predict the high resolution license plate from a low resolution one. In this work, we are using a modified version of the work proposed in [10]. Our G is a feed-forward CNN with parameters θ_g. Where $\theta_g = \{W_{1:L;b_{1:L}}\}$ is the weights an bias of a L-layer deep network and is obtained by optimizing a super-resolution specific loss function l^{SR}. In the training process, we use high-resolution images I^{HR}, with their corresponding low-resolution images I^{LR} to solve a minimization problem:

$$\hat{\theta_G} = arg\min_{\theta_G} \sum_{n=1}^{N} l^{SR}(G_{\theta_G}(I_n^{LR}), I_n^{HR}), \tag{2}$$

where $n = 1, ..., N$, and N is the total of training images.

Now, we have to define the discriminator network D_{θ_D} which is optimized in the min-max problem described in Equation (1), but now $x = I^{HR}$ and $z = I^{LR}$. This allows us to train the generative model G with the goal of generating a license plate that the discriminator D cannot differentiate from a real high-resolution license-plate. And the discriminator D is trained to classify if a license plate is generated from G or is from the high-resolution dataset.

Figure 4 shows architecture of the generator block of the LP-SRGAN. In our generator network G, we have 16 identical residual blocks. They are composed of two convolutional layers with 3×3 kernels and 64 filters, the first one is followed by a ReLU activation function ,as proposed in [5], and a batch-normalization [19], and the second one is followed by a batch-normalization and element-wise sum with the data that entered the residual block. Then, we have three upsampling blocks to increase the resolution of the image, each block is composed of an upsampling layer, a convolutional layer with 3×3 kernel and 256 filters, and ReLU activation function. And, as also described by [5], we use Tanh as the activation function on the output. The residual and the upsampling blocks can be seen in Figure 6.

For our discriminator network D, we use the same architecture as [10], and we used LeakyReLU with $\alpha = 0.2$ and use strided convolutions instead of pooling layers, as suggested in [5]. We have eight convolutional layers with 3×3 kernels and the number of filters being multiplied by two every two layers, from 64 to 512, and whenever we increase the amount of filters, the convolution layer has a strided convolution. The last dense layer from this block is followed by dense layer with 1024 units, another LeakyReLU and an output dense layer with sigmoid function to obtain the probability of the image being from the real high-resolution license plate dataset.

Another important part of the GAN model, is the loss function. In [10], it is proposed a loss function for the generator that is the weighted sum of a VGG19 loss and an adversarial loss, resulting in:

$$l^{SR} = l^{SR}_{VGG} + 10^{-3} l^{SR}_{Gen}, \tag{3}$$

with

$$l^{SR}_{VGG/i,j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2, \tag{4}$$

and

$$l^{SR}_{Gen} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G(I^{LR})}), \tag{5}$$

where $\phi_{i,j}$ is the feature map obtained by the j-th convolution before the i-th maxpooling layer in the VGG19 network, $W_{i,j}$ and $H_{i,j}$ are the dimensions of the respective feature map, and $D_{\theta_D}(G_{\theta_G(I^{LR})}$ is the probability that the generated license plate $G_{\theta_G(I^{LR})}$ is a real high-resolution one. This means, we want the MSE of features extracted by a VGG19 network. For a specific layer within VGG19, we want their features to be matched (minimum MSE for features).

C. Evaluation

After training the GAN model, we use the trained Generator to generate high-resolution license plates from the low-resolution ones. To evaluate our model, we are using Tesseract [20] OCR to read the characters in the license plate and then compare with the label of the original plate.

To evaluate if the generated plates are as good as the original ones, we will measure the accuracy of the OCR without any post-process techniques. If the result on the generated plate is equal to the one of the original license plate, it will be considered a hit, otherwise it will be a miss. The same procedure will apply to the license plated upsized using he bicubic interpolation method. We will calculate the mean of hits to obtain the accuracy of the methodologies. This will allow us to evaluate if the LP-SRGAN can be used to improve license plate images resolution.

IV. Results

A. Training Details

To train the LP-SRGAN, we used the following parameters:

- batch-size: 4
- epochs: 5000
- optimizer: Adam with 0.0002 learning rate and 0.5 β_1

The choice of the Adam parameters are based on studies in [5], suggested learning rate of 0.001 is too high and the momentum term β_1 of 0.9 resulted in training oscillation and instability. Our implementation used Keras [21] with Tensorflow [22] backends. It was used a Nividia GeForce GTX 1070 GPU to train the model.

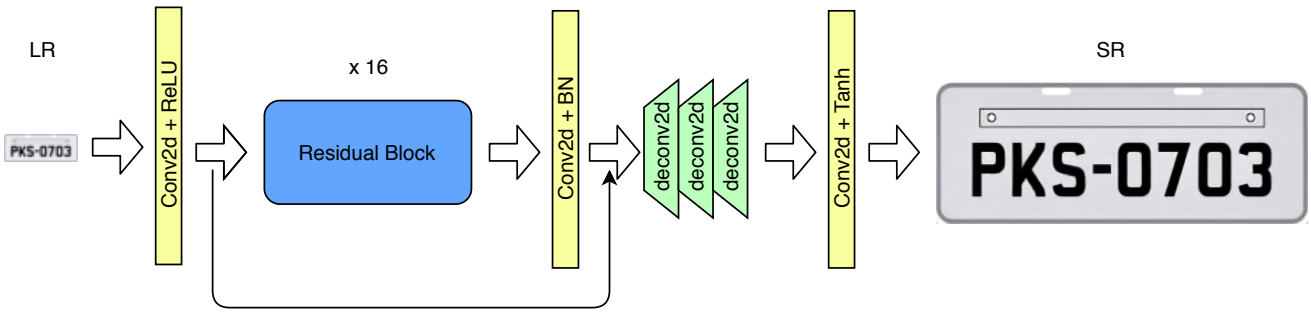

Figura 4. Generator architecture with 16 residual blocks and 3 upsize layers.

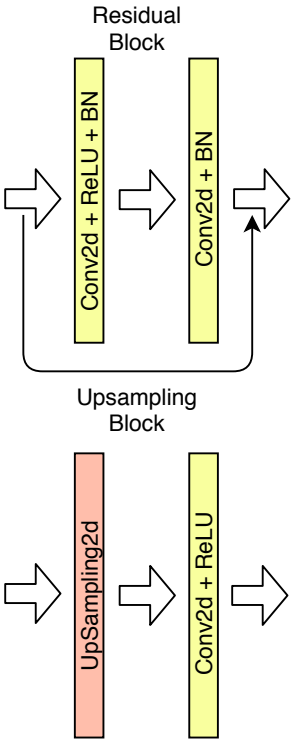

Figura 5. Residual and upsampling block.

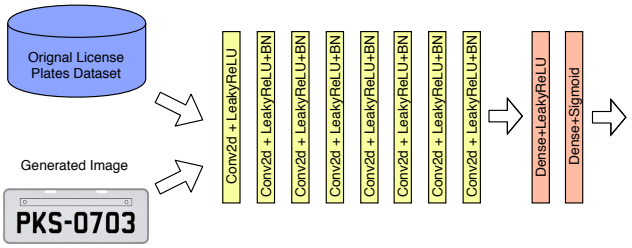

Figura 6. Discriminator architecture.

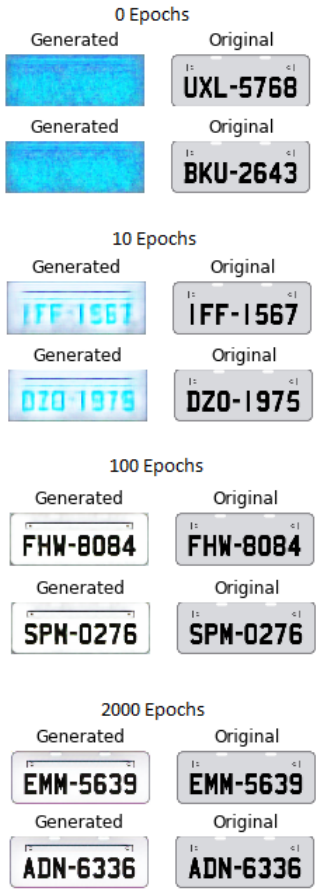

Figura 7. Generated license plates over the training. In the first epochs, the model is still trying to learn the colors of the license plate, and later it will learn the representation of each number.

B. LP-SRGAN performance

The models were saved every 250 epochs, and images were sampled every 10 epochs. Figure 7 shows the evolution of the LP-SRGAN. It is interesting to notice how the network has to learn both the color of the license plate and the numbers representation. After the initial 50 epochs, the color is almost settled, and has very few changes. Although, after 3000 epochs, the Adam optimizer shows instability, and the images lose all information and can't recover, this can be seen in Figure 8. The graph shows a huge drop on the Generator loss as the model learns the numbers and digits representation, and the Discriminator loss increases as the generated license plates get more similar with the real ones.

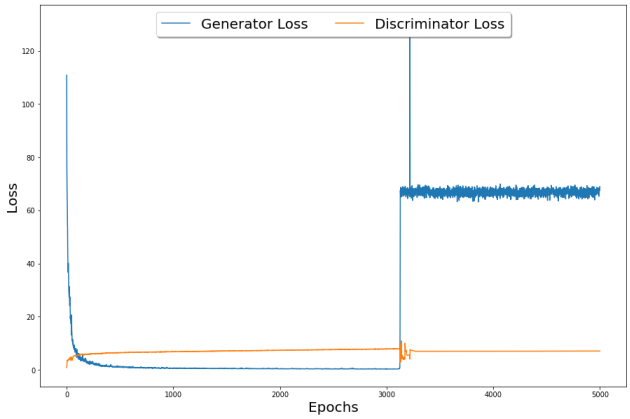

Figura 8. GAN training loss.

	Accuracy
LP-SRGAN	0.738
Bicubic Interpolation	0.399

Tabela I
ACCURACY OF THE METHODS WHEN COMPARING OCR RESULT ON THE ORIGINAL LICENSE PLATE AND THE UPSIZED IMAGE.

After training for 5000 epochs, the model with the lowest generator loss was chosen as the best. It was used to improve the quality of 79,819 low resolution license plates. The model chosen for this prediction was after 2250 epochs, with 0.1898951 generator loss. Figure 9 shows the result obtained with this model. When the resolution is increased with the bicubic interpolation, we can see that the numbers and letter are blurry, and the license plate has square edges. But when we use LP-SRGAN to estimate a high-resolution image, we get an image almost identical to the original high-resolution one, we can see a small difference in the gray background that doesn't affect the ability to read the numbers and digits.

For a more analytical analysis of the generated licenses plates, we used the Tesseract OCR on the licenses plates and when compared to the original image, the OCR gets the same results on 0.738% of images. While the bicubic interpolation has an accuracy of 0.399%, mainly missing the license plates with ambiguous symbols, such as X and K or Q and O. Both methods have some mistakes that could be fixed with some post processing (such as space, impossible characters, letter when we can only have number and vice versa), but as we want the generated plate to be as close as possible to the original one, we stick to not changing any of the string obtained in the OCR. Table I shows the result of the OCR comparison of both methods to the original image.

V. CONCLUSION

In this work, we proposed a LP-SRGAN that is capable of turning a 42×16 pixels license plate into a 336×128 maintaining the information necessary to do an ALPR. This method is a great improvement when compared to the classic bicubic interpolation.

While the network performs extremely well on our created dataset, it does not reflects the performance on a real world experiment. To test that scenario, we would need a dataset with license plates in the real world, being affected by variations in illumination, position and templates. However, having a dataset with a large size of examples, and enough computer power, there is nothing that prevents LP-SRGAN to have a good performance in the real world.

REFERÊNCIAS

[1] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, "Automatic license plate recognition," *IEEE transactions on intelligent transportation systems*, vol. 5, no. 1, pp. 42–53, 2004.

[2] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (alpr): A state-of-the-art review," *IEEE Transactions on circuits and systems for video technology*, vol. 23, no. 2, pp. 311–325, 2012.

[3] A. Singh and J. S. Sidhu, "Super resolution applications in modern digital image processing," *International Journal of Computer Applications*, vol. 150, no. 2, 2016.

[4] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[5] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[6] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*. Springer, 2014, pp. 184–199.

[7] J. Bruna, P. Sprechmann, and Y. LeCun, "Super-resolution with deep convolutional sufficient statistics," *arXiv preprint arXiv:1511.05666*, 2015.

[8] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[10] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[11] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[12] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European conference on computer vision*. Springer, 2016, pp. 391–407.

[13] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.

[14] Y. Wang, L. Wang, H. Wang, and P. Li, "End-to-end image super-resolution via deep and shallow convolutional networks," *IEEE Access*, vol. 7, pp. 31 959–31 970, 2019.

[15] M. S. Sajjadi, B. Scholkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4491–4500.

[16] X. Wang, K. Yu, C. Dong, and C. Change Loy, "Recovering realistic texture in image super-resolution by deep spatial feature transform," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 606–615.

[17] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 136–144.

Figura 9. Low-Resolution license plate (42×16), results obtained using bicubic interpolation and LP-SRGAN (336×128), and the original image (336×128).

[18] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[20] A. Kay, "Tesseract: An open-source optical character recognition engine," *Linux J.*, vol. 2007, no. 159, pp. 2–, Jul. 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1288165.1288167

[21] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/