# FastChi – The Parallelized Fast Style Transfer toward Chi-chi video

Member: Yen Liu, Peter Chuang, Sunner Li

## 1. Abstract

By style transfer mechanism, the style of another image can be illustrated toward another target picture. However, the speed of the computation is a tough problem. In this article, we purposed an accelerating version to enhance this disadvantage. The size of render is reduced and the graphic process unit (GPU) is used to accelerate the computation. We also use multithread to speed up the process of image reading. By this improvement, the speed of reading can be accelerated up to 15 times and the speed of training can be accelerated up to 1.5 times.

## 2. Introduction

The style transfer is a very popular problem in recent year. Moreover, there're some applications which use such kind of transferring into their product. Prisma is an interesting example. The function of the application can do the style transferring toward any input image given by user with different style. By the effect of the application, it can let us to experience the daily life with different mood.

However, the speed and performance are important issues that should be conquered. In Prisma, the smart phone should spend amount of time to transfer the image. Moreover, the same shortage can be seen in the situation to transfer the style toward the whole film. Rather than dealing with single image, the program should spend more time to complete the task for the video.

In order to accelerate the style transferring procedure, we purpose the parallel

structure toward this task, and speed up the procedure of transforming. The network will split the video as two parts, and use two GPU to transform the style to the content image in parallelism. For our expectation, the video which contains chi-chi (shiba-lnu dog) will be transferred in very quick speed. Moreover, the changing can make user be more pleasure and enjoy the daily life.

## 3. Related Works

For the task of artificial style transfer, Gatys et al. [1] adopt the deep neural network to achieve the great performance, and it's the first implementation to use convolution network to complete the task. However, the speed is the bottleneck. Ulyanov el al. [2] raised another creative idea which called instance normalization to accelerate the whole transforming process. Moreover, the instance normalization can also reduce the correlation between batch training images.

It's a critical problem to design the appropriate loss function. In the previous work [1], the authors usually used pixel-to-pixel error to compute the loss. However, this method of loss computing didn't consider the spatial variance. For example, if the result which is generated by deep network shifts a little toward the structure of origin image, the value of loss computation becomes very large which isn't reasonable. To solve the correlative problem, Johnson et al. [3] purposed the combination of perceptual loss with usual forward network. The network would use feature map in higher-level to compute the loss value rather than using output of the network directly, and the designment can make the network preserve the capability which dealing with the situation of spatial difference.

For the structure issue, two kinds of structure have been purposed recently. The first one is purly convolutional neural network based (CNN-based) [1, 2]. CNN-based

structures commonly use VGG-19 as the feature extractor, and regard the mixture of style loss and content loss as objective. However, the generated result might not have flexibility to render with more creative idea. Pixel-wise least square loss limits the ability of the model to draw with higher variance. Another kind of structure is generative adversarial network base (GAN-based) [4, 5, 6, 8, 9], and this idea can be extended to solve the problem of cross domain image to image transformation. GAN-based structures adopt generator to generate the rendered image, and use discriminator to determine if it is a generative one or not. By using the adversarial idea, the generator can generate the image with more flexibility. As the result, GAN-based structures are common used and discussed recently. The paper [7] also arranges the other more papers.

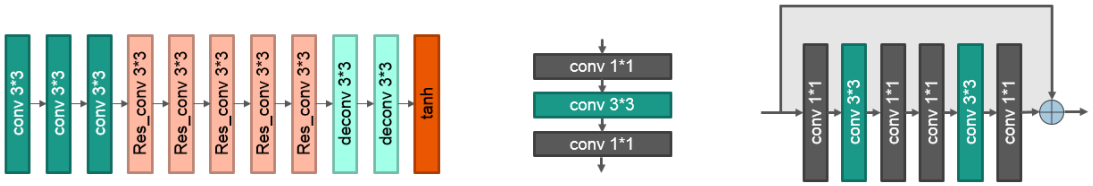
Conversely, some problem can be described. The biggest disadvantage of GAN-based structures is that the size of whole model is huge. Compared to CNN-based structure, two neural networks should be generated to do the computation. Next, these network [3, 4] try to reduce Jensen–Shannon divergence (JS divergence), but this idea is proven [10] that it's not stable to get the converge situation. Moreover, these papers [5, 8, 9] even adopt more than two networks achieve the transfer result. Since the reason which is shown above, the GAN-based structures are difficultly used in the environment which is lack of computation resources.

In order to adopt this idea widely, the CNN-based structure is the target we discuss in this paper. First, we fuse idea of perceptual loss and common feature extractor. Next, we try to use multi-thread and multi-process mechanism to accelerate the whole process. At last, we purpose a shrinking idea to reduce the number of the parameter in the render network.

# 4. Proposed Solution

In order to accelerate the whole process, three directions we try to enhance, including reduce the number of model parameter, accelerate training speed and accelerate the transformation speed. The three directions will be discussed in order.

## 4.1. Reduce the number of model parameter




(a) (b) (c)

*Figure 1.* Illustration of our purposed model. (a) The model structure which paper [2] purposed. (b) The inception convolution idea we adopt. (c) The residual block with inception convolution.

In original paper [3], the deep encoder-decoder concept has been used to be the render network. Figure 1 illustrate the original structure. In this paper, we also adopt the same structure too. However, to reach the goal of accelerating, we fuse the inception idea which has been purposed by google [11] into the render model. In each convolution operation, the image will be processed by convolution operation whose filter size is 1, and the number of channel will be shrink as $k$ times. The result can be regard as the linear combination toward the different channel. After the deed convolution operation, the same mechanism will be used again to recover the expected channel size. By the inception idea, the same computation result can be gained, but the model can have less parameters. In the residual block, the inception idea has also been used. The inception idea is also shown in Figure 1.

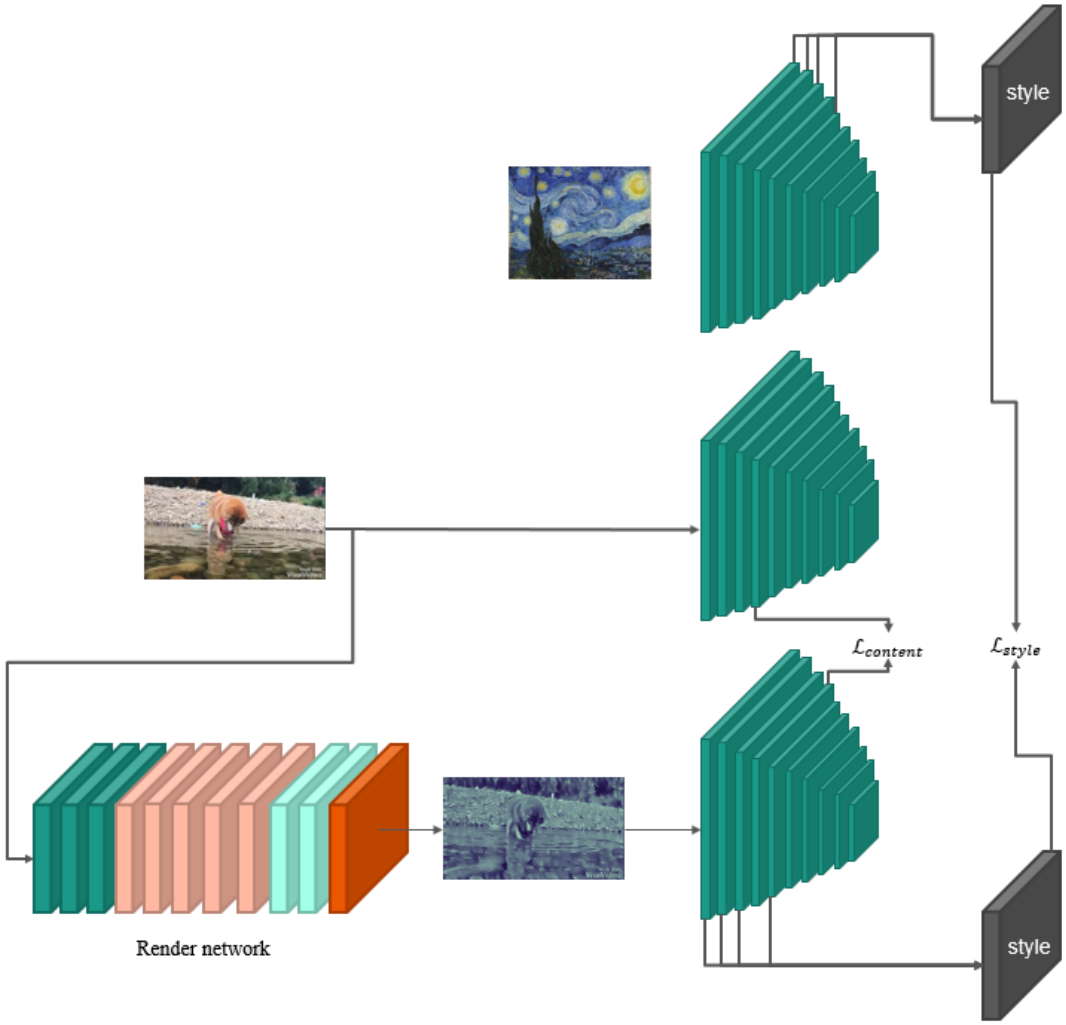



*Figure 2.* The structure of whole training model

The figure 2 shows the whole structure of whole training model. First, the style image will be sent into the feature extractor network. In the original paper[1], the pre-trained VGG-19 model is used as the feature extractor network. However, the size of VGG-19 is still very huge. To accelerate the extracting speed, we also use pre-trained tiny-yolo [12] structure to be the feature extractor network. Then we can compute the gram matrix and treat as the style.

Next, we send the training image into the render network. In each iteration, after the render image is generated, it will be sent into feature extractor network. The

program will compute the content loss by the feature map which pre-trained model generated. On the other hand, the style of rendered image will also be computed, and calculate the style loss between the gram matrix of rendered image and the gram matrix of style image.

$$\mathcal{L}_{total} = \alpha * \mathcal{L}_{content} + \beta * \mathcal{L}_{style} + \gamma * \mathcal{L}_{total\ variance} \quad (1)$$

For the loss function, we use as same as the paper purposed [3]. However, we also try to reduce the expectation of the difference rather than the summation of the difference. The whole loss function in our training stage can be shown above. The α, $\beta$ and $\gamma$ is the hyper-parameter to control the balance between the content and style.

## 4.2. Accelerate training speed

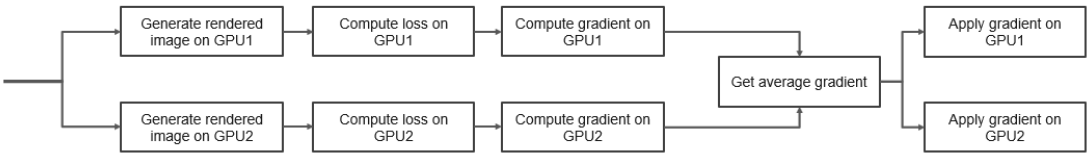



*Figure 3*. The multi-GPU training flow chart

In the training process, we use two GPU to do the acceleration. In each training iteration, the individual training batch images will be sent into each GPU. After two GPU finish gradient calculation, the average gradient will be compute. Next, the average gradient will be both apply to each GPU. By this design, the whole program can raise the batch size and consider more images in each iteration. The whole multi-GPU mechanism is illustrated in figure 3.

Data input output speed is also another bottleneck. However, there's no dependency between the reading operation and training stage. In order to reduce the training time, we also use multi-process mechanism to read the image, and store the image into a queue. By this changing, the GPU can get the training from the queue

rather than read from secondary device sequentially.

### 4.3. Accelerate transformation speed

During the video transformation process, the video will be split as image frame by frame firstly. Next, each image will be sent into render network to do the style transfer. At last, the whole result images will be encoded as the video which type is as same as original video. In order to speed up the transformation, we adopt multi-thread mechanism to drive two GPU doing transformation task. By the multi-GPU design, each GPU only needs to deal with less images. As the result, the program can use less time to finish the transformation.

## 5. Experiment & Result

To evaluate the performance of our revision, we demonstrate the processing speed and render phenomenon in order.

### 5.1. Processing speed

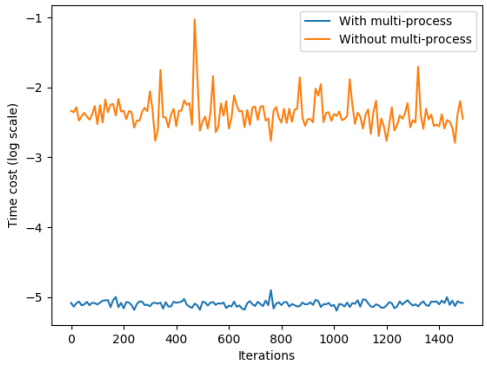



Figure 4. The time cost curve of data reading.

For the training stage of deep learning, besides the program should spend a lot of time to deal with matrix computation, it's also a critical problem that the process of data reading is time consuming. As the result, it's still a great enhancement to accelerate the process of data reading. The Figure 4 shows the time cost during image reading in

training stage. By the multi-process mechanism to deal with image reading, the program speeds up for about 15 times than the original implementation.

On the other hand, adopting multi-GPU can also reduce the processing time. By our experiment, the process of training can be speed up about 1.5 times with the same iteration. For the transformation task, the program can also speed up about 1.55 times with the same video. To sum up the whole results, the acceleration we purposed reaches the goal.

## 5.2. Render performance

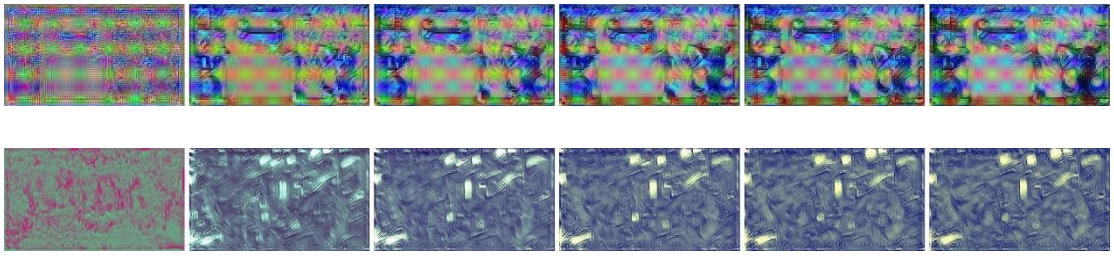


*Figure 5.* The render result that the network only minimizes style loss

In our experiment, the value of $k$ is set as 2. To evaluate the render ability of the network, we split as three parts. In the first part, the model only minimizes the style loss, and the result is shown in figure 5. The figure shows the result in the previous 1500 iterations, and we record the result for each 300 iterations. The upper six images is the result that the original network generates, and the lower six images is the result that the inception version network generate.

By the figure 5, the style render performance is a little less than original model. Even though the inception model cannot learn the style and structure completely, it still learns the basic idea of painting and distorted lines. The model can learn the rough structure of style image firstly, and learn the color distribution secondly. These results show that the inception model still has ability to learn the style.

*Figure 6.* The render result that the network only minimizes content loss

In the second parts, the model is only trained to minimize the content loss, and the result is shown in Figure 6. The picture also shows the previous 1500 iterations with being captured for each 300 iterations. In the upper side, the original model can learn the intensity with full color information. On the other hand, even though the inception model cannot catch the color intensity completely, the contour and rough information still be gained by the model.

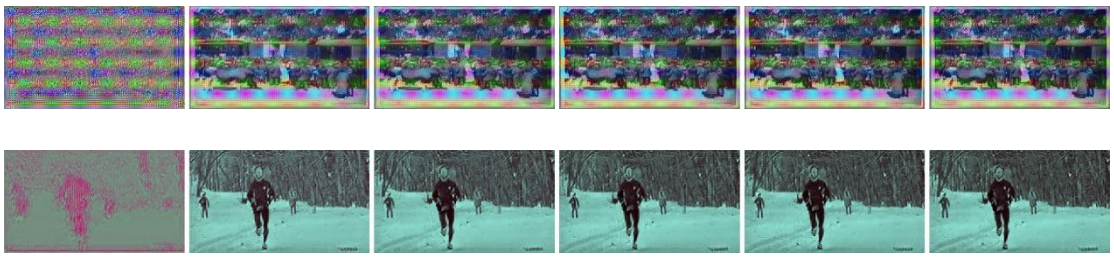


*Figure 7.* The render result that the network minimizes total loss

In the last part, we make the model reduce the total loss, and the respective results are shown in Figure 7. The upper six pictures are the image that original network generates. However, the model can render with more clear content information but less style information. Moreover, the images still have the noise dots which is the disadvantage of the original render network. On the other hand, the lower six pictures are the image that our inception network generates. Even though the image detail cannot be captured, the rough contour and the painting style can still be rendered into the result.

Moreover, the images are much smoother than the result of original render network. To sum up for whole result shown above, the inception version of render network can also learn the style and render very well.

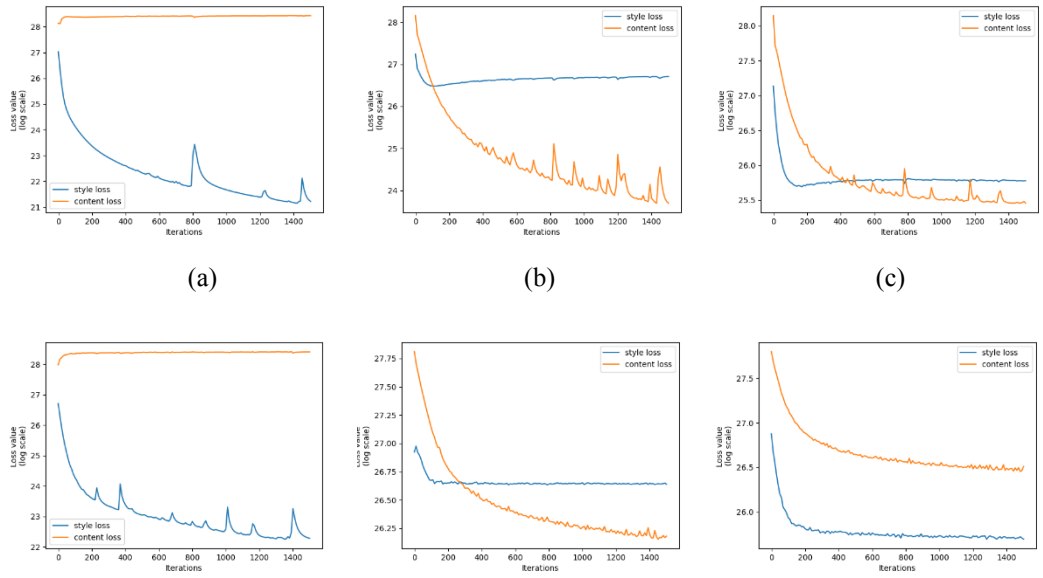



*Figure 8.* The loss curve of our experiment. (a) The loss curve of the original model that the model only train toward style loss. (b) The loss curve of the original model that the model only train toward content loss. (c) The loss curve of the original model that the model train toward total loss. (d) The loss curve of the inception model that the model only train toward style loss. (e) The loss curve of the inception model that the model only train toward content loss. (f) The loss curve of the inception model that the model train toward total loss.

In Figure 8, the trend of loss dropping are also shown. The blue curve is the style loss, and the orange curve is the content loss. The upper three plots is the result of original model, and the lower three plots is the result of inception model. Both model have convergence phenomenon in three training strategies. The more training iteration it is, the lower value of the loss it is. On the other hand, the Figure 9 shows our transformation result.


*Figure 9.* The transformation results

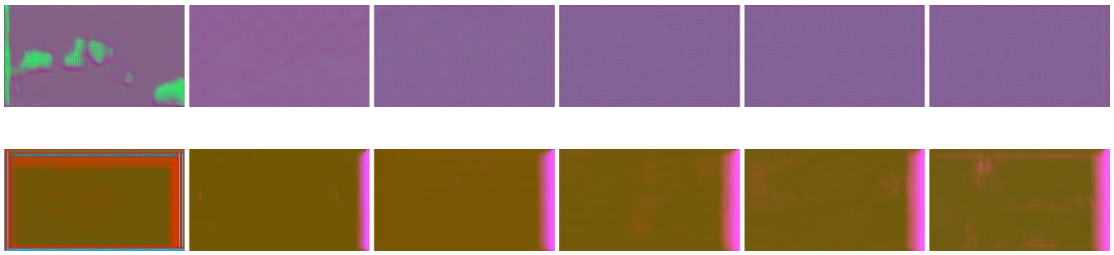


*Figure 10.* The training result by using tiny-yolo

At last, the Figure 10 shows the training result by using pre-trained tiny-yolo as the feature extractor network. Both results are trained toward total loss. However, the upper six image shows that the render cannot learn the style. The lower six images is the result that change to the expectation of the loss. Even though some texture can be shown, the printing style and rough structure cannot be obtained. This result shows that it's very important to use pre-trained model that the size is enough to represent the correlation between different pixels.

## 6. Conclusions

For the image transfer task, we purposed an inception version of render network whose number of parameters is less than original network. Furthermore, the multi-thread mechanism is adopted to accelerate the image reading process. Multi-GPU design can make the program do more computation in the same period, and can be used to speed up the transformation speed too. At last, we tried another smaller network as the feature extractor network and shows the importance to use the meaningful network.

There are some directions can be explored in the future. This paper [13] shows a creative idea to use random weight VGG-19 to extract the feature, and the performance is also well. By training for more iterations and using more complex loss function can help the render network learn more better, and get the detail capture of the style.

# 7. References


[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, "Image Style Transfer Using Convolutional Neural Networks," In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada, USA, 27-30 June, 2016, pp. 2414-2423.

[2] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky, "Instance Normalization: The Missing Ingredient for Fast Stylization," arXiv: 1607.08022v2 [cs.CV], Sep. 2016.

[3] Justin Juhnson, Alexandre Alahi, and Li Dei-Dei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," arXiv: 1603.08155v1 [cs.CV], March 2016.

[4] Yaniv Taigman, Adam Polyak, and Lior Wolf, "Unsupervised Cross-Domain Image Generation," In *International Conference on Learning Representations (ICLR)*, Toulon, France, 24-26 April, 2017, pp. 1-15.

[5] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," In *International Conference on Computer Vision (ICCV)*, Venice, Italy, 22-29, October, 2017, pp. 2223-2232.

[6] Hang Zhang, and Kristin Dana, "Multi-style Generative Network for Real-time Transfer," arXiv: 1703.06953v2 [cs.CV], November 2017.

[7] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, and Mingli Song, "Neural Style Transfer: A Review," arXiv: 1705.04058v1 [cs.CV], May 2017.

[8] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong, "DualGAN: Unsupervised Dual Learning for Image-to-Image Translation," In *International Conference on Computer Vision (ICCV)*, Venice, Italy, 22-29, October, 2017, pp. 2849-2857.

[9] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim, “Learning to Discover Cross-Domain Relations with Generative Adversarial Networks,” In International Conference on Machine Learning (ICML), Sydney, Australia, 6-11, August, 2017, pp. 1857-1865.

[10] Martin Arjovsky, and Leon Bottou, “Towards Principled Methods for Training Generative Adversarial Networks,” In *International Conference on Learning Representations (ICLR)*, Toulon, France, 24-26 April, 2017, pp. 1-17.

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, “Going Deeper with Convolutions,” In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, USA, 8-10, June, 2015, pp. 1-9.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, 2016, pp. 779-788.

[13] Kun He, Yan Wang, and John Hopcroft, “A Powerful Generative Model Using Random Weights for the Deep Image Representation,” In Electronic Proceedings of the Neural Information Processing Systems Conference (NIPS), Barcelona, Spain, 5-10, December, 2016, pp. 1-9.