FAST IMAGE STYLE TRANSFER

Aravinda Misra, Saurabh Deshmukh and Tanmay Bhatt

Deep Learning (CMPE 258)

San Jose State University

Project team members:

Aravinda Misra (011468886)
Saurabh Deshmukh (011435060)
Tanmay Bhatt (011499072)

Abstract

Deep Learning is one of the fastest growing technology and is very successful in image processing related tasks. One of the applications is Style transfer of an image chosen as style image into the content of another image i.e. painting a texture of one image into another image. Recent methods of style transfer optimize perpetual losses and style losses between style image and content image to train the feedforward convolutional neural network and generate the styled image for given style image in few seconds. We used approach proposed by Johnson, J., Alahi, A., and Fei-Fei, L. (2016) in which they have trained feed-forward image transformation network with optimization of perceptual loss. We studied the effect of changing feed-forward network and normalization method on the style transfer. We trained three different network architectures of feed-forward network to compare the visual differences of styled images. Also, we studied the visual effect of use of Batch Normalization vs Instance normalization on style transfer.

Keywords: Style transfer, Deep Learning.

Introduction

Style transfer is one of the famous implementations of image transformation task. One of the approach used by early studies is to use feed-forward network to train the network by minimizing per pixel loss function of output image and ground truth images. This approach mostly captures the low-level features of images and failed to capture perceptual differences. (Gatys, L., Ecker, A., & Bethge, M. ,2016) used another approach by minimizing the differences of high level features in the image, which will be extracted by using pre-trained convolutional neural networks. These networks had a superior performance compared to per-pixel loss since higher layers captured image features. The approach proposed by Gatys et al. designed a network that could minimize the total loss everytime given an input content image and a style image . This approach is slow for image style transfer since the network had to minimize the loss for every input pair. Johnson et al (2016), designed a Fast Style Transfer Network where they trained a feed-forward transformation network for image transformation tasks, along with the pretrained network (referred as loss network), which was used to calculate the loss. They train networks using perceptual content loss functions that depend on high-level features from a pre-trained loss network and and style loss function which is calculated using outputs from various layers of a pre-trained network. In the image transformation network mentioned in the given approach, we also used variation of layers and different loss pretrained network to build different networks. and observed the changes in the output styled image for same content image. We used Batch normalization and Instance transformation in Image transformation network to perform normalization on contents images. We trained our network using CIFAR-10 (Alex Krizhevsky,

2009) image dataset. We visually compared the output and correlated the differences with changes in image transformation layers structure as well as normalization method used.

Method

As shown in fig 1, the network consists of two parts: an image transformation network f_W and a loss network φ that is used to define loss functions for different layers. The image transformation network is a combination of deep convolutional neural network, residual network and transpose convolution network.

The approach used here is to train the Image transform network on many images such that the network learns to apply the style of the content image to the input image. Vgg-19 is used as loss network and when an image is given as input to the pre-trained network the network transforms the pixel information to feature information as the layers progresses. VGG -19 has total 19 learnable weights (that includes convolution , pooling and fully connected layers). The last fully connected layers are used for classification and since our goal is to extract feature information we do not need the fully connected layers.

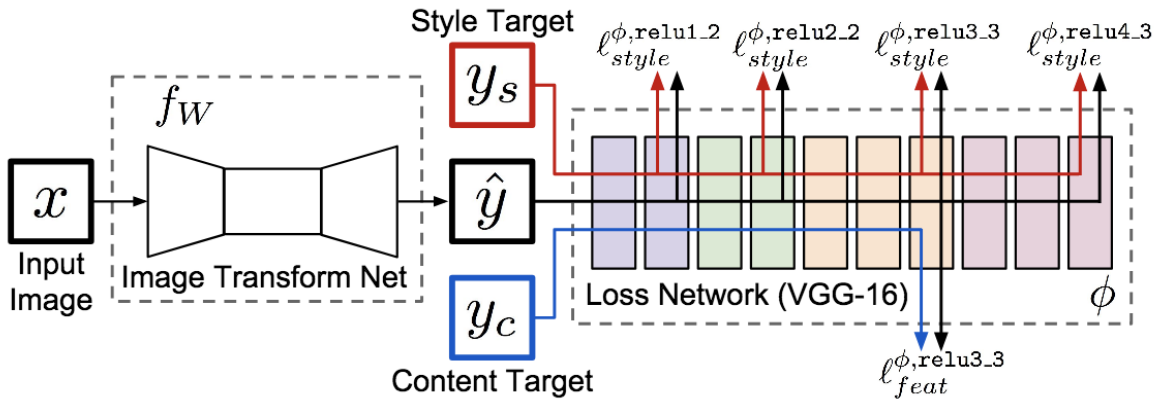

Fig 1. Image transformation and loss network (Johnson et al,2016)

Steps followed to calculate the Loss :-

1. Image Transform Network is initialized with random weights ,which transforms the input image to y (hat).
2. These three images (Style Image , Content Target and y (hat)) image are passed through the Vgg-19 network and the relevant layers of each image are extracted to compute the loss.
3. For the style loss all the relu layers are extracted and style loss is computed.
4. For content loss the layer towards the end of the network is extracted as it contains feature level information .
5. The loss thus calculated is minimized and the network is trained.

Image transformation networks –

Original Image transformation network used by Johnson et al (2016) consists of 5 residual blocks and all the non-residual convolutional layers used Batch Normalization with ReLU activation functions except last convolutional output layer which used tanh activation to scale the output values in between [0,255] Input and Output image size of this network is 3x256x256 and used downsampling to reduce spatial extent of feature maps and upsampling to produce final output image with same size. For Image transfer networks we used two stride-2 convolutions to downsample the input followed by several residual blocks and then two convolutional layers with stride 1/2 to upsample. Although the input and output have the same size, there are several benefits to networks that downsample and then upsample. The first and last layer used 9x9

kernels, all other convolutional layer used 3x3 kernels. The original network architecture you can see in the Fig 2.

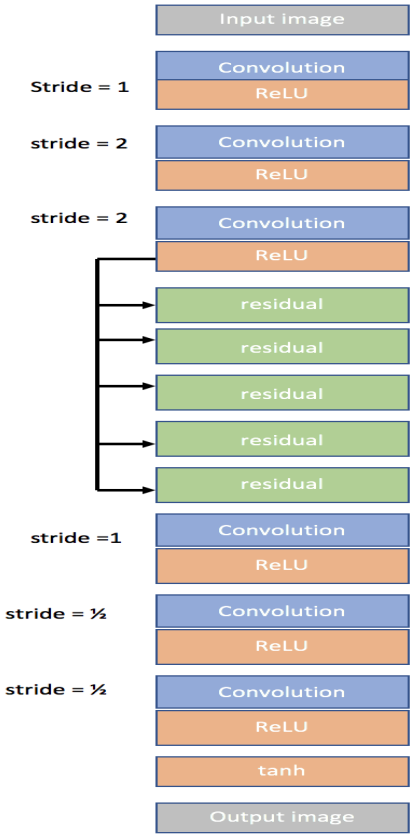

Fig 2. Image Transformation Network

Loss Network and Perceptual Loss Network –

In the original experiment, they have used VGG 16 network which pre-trained on ImageNet. The architecture and layers information of VGG 16 is given in Fig 3. (Column C). Two perceptual loss functions have defined to measure high level extracted feature differences of the images.

Content Loss – This loss function is to ensure the content of the image is preserved as the style is being applied.

The feature reconstruction loss is the (squared, normalized) Euclidean distance between feature representations given by equation -

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{i,j}(F_{ij}^{l} - P_{ij}^{l})^2$$

$\vec{p}$ The original image
$\vec{x}$ The generated image
l Layer
F_{ij}^{l} Activation of the i^{th} filter at position j in the feature representation of $\vec{x}$ in l
P_{ij}^{l} Activation of the i^{th} filter at position j in the feature representation of $\vec{p}$ in l

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig 3. VGG architecture

Style Loss – Content loss makes sure to optimize and penalize the output image $\hat{y}$ when it differs for content from content image. Style loss optimize the Style image and output image differences in color textures and common patterns to achieve this style in output image. The following Gijl represent the vector product of layer l in loss network using Gram matrix.

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}$$

$\boldsymbol{G_{ij}^{l}}$ Inner product between the vectorised feature map i and j in layer l
$\boldsymbol{F_{ik}^{l}}$ Activation of the i^{th} filter at position k in the feature representation in layer l
$\boldsymbol{F_{jk}^{l}}$ Activation of the j^{th} filter at position k in the feature representation in layer l

which calculates texture information at each layer l in loss network. Same concept of Gram matrix is used to calculate the loss of given layer l for one whole layer by using dimensions and number of distinct features available in that particular layer.

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^{l} - A_{ij}^{l})^2$$

$\boldsymbol{N_l}$ number of distinct feature maps
$\boldsymbol{M_l}$ the height times the width of the feature map
$\boldsymbol{G_{ij}^{l}}$ pairwise feature maps i and j in the style representation of $\vec{x}$ in l
$\boldsymbol{A_{ij}^{l}}$ pairwise feature maps i and j in the style representation of $\vec{a}$ in l

The total style loss can be calculated by using all such layers we used in Loss network as shown in Fig.1.The weighing factor is provided to take the contribution of each individual layer in final style loss. Total Style loss in loss network is given by equation:

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

$\vec{a}$ The original image
$\vec{x}$ The generated image
L Total number of all layers
w_l Weighting factors of the contribution of each layer to the total loss
E_l loss in layer l

Total Loss Function - Total Loss function to use for backpropagation is combination of Content loss and Style loss. The weighing factors α and β for content loss and Style loss to control transfer of content and style in output image.

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

α and β are the weighting factors for content and style reconstruction

Normalization methods –

The original network proposed used batch normalization which actually calculates normalization parameter for whole batch of input images and apply and tune while performing backpropagation. Batch normalization formulas for calculating mean and standard deviation use all batch images as shown below where T is size of Batch. Batch normalization perform and make the normalization more generic.

$$y_{tijk} = \frac{x_{tijk} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}, \quad \mu_i = \frac{1}{HWT} \sum_{t=1}^{T} \sum_{l=1}^{W} \sum_{m=1}^{H} x_{tilm}, \quad \sigma_i^2 = \frac{1}{HWT} \sum_{t=1}^{T} \sum_{l=1}^{W} \sum_{m=1}^{H} (x_{tilm} - mu_i)^2.$$

Instance normalization provide normalization over only one image and even after providing input images inside the batch, function will compute the mean and standard deviation for each image and perform normalization on that image.

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - mu_{ti})^2.$$

As you can see in above equation we are considering single input image of size H x W and calculating the mean and standard deviation to perform normalization.

Experiment

We used CIFAR-10 dataset of input image size 3x32x32 with 50,000 input images. The Dataset containing images for 10 classes airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck with 5,000 images per class. We reshaped input images from CIFAR-10 dataset to 3x256x256 while training the network. We used starry nights Fig 4 for four networks and Wave Fig 5 for one network.

Fig 4. Style Image 1 – Starry Night

Fig 5. Style Image 2 – Wave

Style transfer network 1 – Instance Normalization

For the first style transfer network, In Image transformation network, we used two convolution layers for downsampling with stride 1 and 2 respectively with ReLU activation function and then used three residual layers as shown in Fig 6. We used another two-convolutional layer after residuals for upsampling with stride 1 and ½ and ReLU activation function. The network is trained with VGG16 pretrained network as Loss network for 2 epochs due to limitation of computational capability with batch size equal to 4. Complete architecture of VGG16 is given in Fig 3 (Column C) . The Weighing parameters used for Total loss function is α=15 and β=100.The kernels used in convolutional layer are, for input and output non-residual convolutional layers, we used 9x9 size kernel and for rest of all convolutional layers in the network we used 3x3 kernel size. The last layer activation function was tanh to scale the data in between [0,255] .

We used Instance normalization in Image transformation network after each convolutional layer and the final Style loss we got is 1003343.94 and content loss is 5055549.0 at the end of training. We used Starry night style image for training and style transfer.

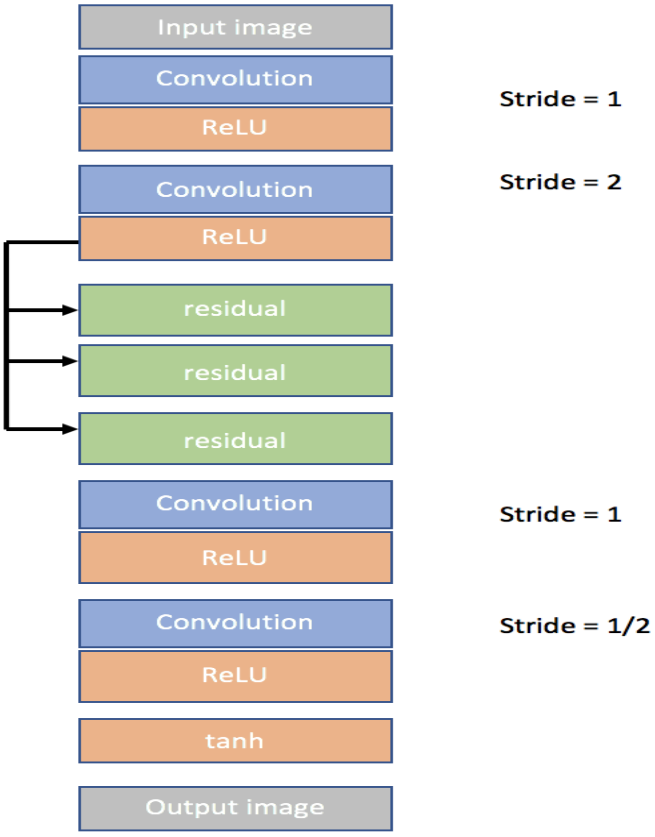

Fig 6. Style transfer network 1

Style transfer network 2 – Instance Normalization

For the second style transfer network, In Image transformation network shown in Fig 7, we used four convolution layers for downsampling with stride 1,2,2 and 2 respectively with ReLU activation function and then used five residual layers as shown in Fig 5. We used another four-convolutional layer for upsampling with stride 1,1/2,1/2 and 1/2 respectively with again ReLU activation function. The network is trained with VGG19 pretrained network as Loss network for 2 epochs due to limitation of computational capability with batch size equal to 4. Complete architecture of VGG19 is given in Fig 3. The Weighing parameters used for Total loss

function is $\alpha=15$ and $\beta=100$.The kernels used in convolutional layer are, for input and output non-residual convolutional layers, we used 9x9 size kernel and for rest of all convolutional layers in the network we used 3x3 kernel size. This was one of the largest network we trained and took more time than the first network to train.

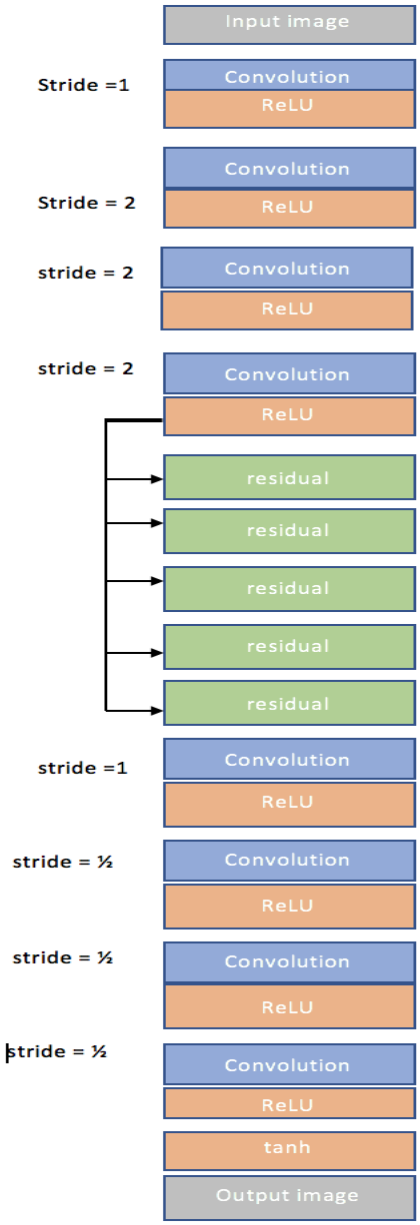

Fig 7. Style transfer Network 2

We used Instance normalization in Image transformation network after each convolutional layer and the final Style loss we got is 700043.94 and content loss is 4802049.0 at the end of training. We used Starry Night style image for training and style transfer.

Style transfer network 3 – Batch Normalization

For image transformation network, we used the original architecture shown in Fig 2 and used CIFAR-10 dataset as the original approach was tested on COCO dataset (Lin et al., 2015) and VGG19 as loss network. We used Batch Normalization layer after each convolutional layer with α=15 and β=100.The final content loss after training for 2 epochs with batch size equal to 4, was 1025437.2 and style loss was 5308114.5.

Style transfer network 4 and 5 – Instance Normalization

We used original architecture as shown in Fig 2 with Instance normalization on CIFAR-10 dataset and trained two networks for both Starry Nights Fig 4 and Wave Style Fig 5 images. The network with Wave as Style image trained up to 4 epochs with batch size equal to 4 and α=15 and β=100 the final Style loss reduces till 390543.51 and Content loss reduces till 4132841.0.

The Network with Starry Night image as Style image trained only for 2 epochs with other parameters same as above network and Final Style loss 700328.9 and content loss 4900148.0. We trained this network to compare the original network style transfer output with the different image transformation network we used for training.

Observations and Discussion

The primary intention of this experiment and project was to visually compare the effect of changing image transformation network and normalization methods on final output image. We used few content input image to transfer style from trained network. Fig 8 and Fig 9 given below used as content image for further discussion.

Fig 8. Content Image 1

Fig 9. Content Image 2

Style transfer Network 1 – Instance Normalization

In style transfer network 1, we have used three residual layers and two convolutional layers to perform downsampling and upsampling each. As you can compare between content images and Styled Images (Fig 8 & Fig 10, Fig 9 & Fig 11)

Fig 10. Styled Image 1 – Network 1

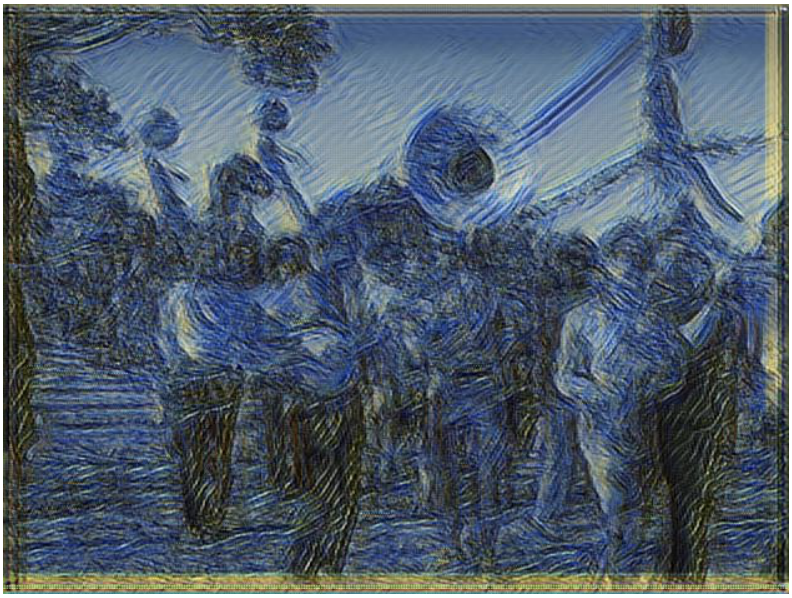

Fig 11. Styled Image 2 – Network 1

Due to Loss network(VGG16), trained network not able extracted all feature of content image also residual layers are known for retaining original features and adding new features and in this network, residual layers are less that's why trained network losing few of the fine contents of input images.

Style transfer network 2 – Instance Normalization

In style transfer network 2, we have added one more convolutional layer for performing downsampling and upsampling each and also number of residual layers in this network is five.

Fig 12. Styled Image 1 – Network 2

You can compare the styled images Fig 12 and Fig 13 with its original content images Fig 8 and Fig 9 and check that most of the content of original images are retained in styled image and also with less (epoch =2) training of the network, it is transferring style very well. If you compare this

two images with the styled images generated by original network Fig 16 and Fig 17 then you will see, due to extra downsampling and then upsampling the edges and fine details are somewhat blurred.

Fig 13. Style Image 2- Network 2

Style transfer network 3 – Batch Normalization

In this image transformation network, we used Batch Normalization over the batch size of 4 and rest of the network was same as the original network shown in Fig 2.

Fig 14. Style Image 1- Network 3

Fig 15. Style Image 2 – Network 3

As you can compare the batch normalized images in Fig 14 and Fig 15 with above images which are trained on Instance normalization, we are missing fine details, style color and patterns in Content images and content of images are not fully getting transferred into styled images.

Style transfer network 4 & 5 – Instance Normalization

These both networks are trained with same parameters but different style images, with Starry Night and Wave. The network with Wave as style image trained for 4 epochs and the network with Starry Nights trained for 2 epochs.

Fig 16 and Fig 17 are styled images which are converted using image transformation network with five residual layers and three convolutional layers for downsampling and upsampling each. This Network retained all the contents of input image and added style but due to the network is not fully trained,the final output image is not completely containing style and contents as expected.

Fig 16. Style Image 1 – Network 4

Fig 17. Style Image 2 – Network 4

Fig 18 and Fig 19 showing the styled images of network trained with Wave Style image for 4 epochs. The overall Style loss and content loss are reduced due to more training of network.

Fig 18. Style Image 1 – Network 5

Fig 19. Style Image 2 – Network 5

The above two styled images are looking good as they used the fully trained style transfer network with less content and style loss and these images are closer to original style image and their respective content image and containing almost all content from content image and style from style image.

References

Gatys, L., Ecker, A., & Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on,* 2414-2423.

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution.

Alex Krizhevsky (2009). Learning Multiple Layers of Features from Tiny Images.

Tsung-Yi., Lin, Michael., Maire, Serge., Belongie, Lubomir., Bourdev, Ross., Girshick, James., Hays...Piotr., Dollár (2015). Microsoft COCO: Common Objects in Context