Deep Blind Video Decaptioning - fact sheet

August 1, 2018

Deep Blind Video Decaptioning with 3D-2D Gated Convolutions

1 Team details

- **Team name**
  KAIST-RCV
- **Team leader name**
  Sanghyun Woo
- **Team leader address**
  Room 212, N1 building, 291 Daehak-ro, Yuseong, Daejeon (34141), Korea
- **Team leader address**
  +82-42-350-5465
- **Team leader email**
  shwoo93@kaist.ac.kr
- **Rest of the team members**
  Dahun Kim, Joon-Young Lee, In So Kweon(Corresponding Author)
- **Team website URL (if any)**
  www.rcv.kaist.ac.kr
- **Affiliation**
  Korea Advanced Institute of Science and Technology (KAIST), Adobe Research

2 Contribution details

- **Title of the contribution**
  Deep Blind Video Decaptioning with 3D-2D Gated Convolutions
- **Final score**
  1st place on final test set
  MSE:0.0011 PNSR:33.3527 DSSIM:0.404

- **General method description**

We propose a general video decaptioning framework that can be applied to remove text overlays of any natural videos. The proposed framework has three important properties. First, it is able to perform contextual spatio-temporal reasoning based on 3D convolutional layers. Second, it does not require any input mask which locates places of a video where decaptioning is needed. Instead, we place a big skip-connection between input and output to enforce whole network to approximate text removal function, learning residual, effectively. Third, the proposed framework adopts gated convolution, providing a learnable attention mechanism across all layers.

Preprocess
We adopt horizontal flipping, vertical flipping, and color jittering for data augmentation. We normalized the input data by dividing the pixel values by 255.(0-1 normalization)

Architecture

The model architecture is based on U-net[1,2]. In order to handle video input, we inflate the encoder part as 3D convolution. To explicitly enable spatio-temporal reasoning, the encoder takes multiple frames(9) with temporal strides(3) in between. Here, the goal is to remove text overlays of the middle frame which is the 5th frame out of the 9 input frames. The encoder reduces the temporal dimension to 1, so that following bottleneck layers and a decoder can operate with 2D convolutions. The bottleneck layers consist of several dilated convolutions to enlarge the receptive field size. This helps capturing wider context in spatial dimensions, which is important in our target task. We also use gated convolutions[3] to model attention mechanism which emphasizes important pixels(features) and suppresses irrelevant ones. We place a skip-connection between input and output, so that the whole network is trained to predict the residual image: text overlays. Our final activation function is sigmoid which constraints the output values in the range of 0-1.

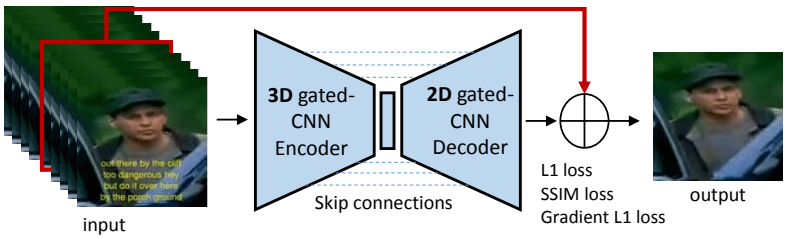
Optimizer & Loss function
We take Adam as the optimization method with $\beta = (0.9, 0.999)$ and introduce three loss functions: L1 loss, SSIM loss, and Gradient L1 loss.

Post-process
We copy input pixel value if the absolute difference between the input(5th frame out of 9 frames) and output of that pixel is less than 0.01.(0-1 scale)

- **References**

[1] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2536-2544).
[2] Iizuka, S., Simo-Serra, E., Ishikawa, H. (2017). Globally and locally consistent image completion. ACM Transactions on Graphics (TOG), 36(4), 107.
[3] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T. S. (2018). Free-Form Image Inpainting with Gated Convolution. arXiv preprint arXiv:1806.03589.

- **Representative image / diagram of the method**



- **Describe data preprocessing techniques applied (if any)**
  We convert mp4 video files into png image files(128x128). Augmentation: Horizontal Flipping, Vertical Flipping, Color jittering
  Normalization: Division from 0-255 to 0-1 scale.

3 Method description

3.1 Features / Data representation

Our final model does not rely on any external features or data representation.

3.1.1 Dimensionality reduction

An encoder-decoder architecture (UNet) with bottleneck (stacked dilated convolutional layers) is adopted[2].

3.1.2 Compositional model

Our final model is a single model.

3.1.3 Learning strategy

Adam optimizer with $\beta = (0.9, 0.999)$ is used. Learning rate is set to 0.001.

3.1.4 Other techniques

We introduce three loss functions for training: L1 loss, SSIM loss, and Gradient L1 loss.

3.1.5 Method complexity

Estimated method complexity

- Number of convolutional layers in our proposed network: 23 layers
- The total number of parameters of Generator is about 10.5M

3.2 Data Fusion Strategies

Our final model is simple feed-forward model without any data fusion strategies.

3.3 Global Method Description

- **Which pre-trained or external methods have been used (for any stage, if any)**
  None (training from scratch)
- **Which additional data has been used in addition to the provided ChaLearn training and validation data (at any stage, if any)**
  None
- **Qualitative advantages of the proposed solution**
  The inference takes 1-2 seconds for each clip.(125 frames)
- **Results of the comparison to other approaches (if any)**
  As observed in the leader board, our proposed method shows the best performance in all the metrics (user ID : 'Sanghyun Woo') compared to other approaches.
- **Novelty degree of the solution and if is has been previously published**
  Novel. To the best of our knowledge, the network of performing blind decaptioning(inpainting) task in video domain has not been introduced so far. Our method is simple yet effective to generate clean videos given any text overlays.

4 Other details

- **Language and implementation details (including platform, memory, parallelization requirements)**
  - Tools: Python 3.6.4, Pytorch 0.3 (Ubuntu version), CUDA v8.0, CuDNN 6
  - Spec: Intel Xeon @2.10GHz, 132GB RAM, Nvidia GTX 1080 ti(4 gpus)
- **Human effort required for implementation, training and validation?**
  - Implementation: One needs to implement model architecture and loss functions in pytorch.
  - Training/validation: Finding the best hyper-parameters (e.g. # of input frames, temporal strides of input, learning rate, weight-coefficients of loss terms, number of layers, image up/downscale rate etc.) may result in better performance. However, we found our method is quite robust to different hyper-parameter settings.
- **Training/testing expended time?**
  Training: 3days, Testing: 1-2 sec per video
- **General comments and impressions of the challenge? what do you expect from a new challenge in face and looking at people analysis**?
  There were some difficulties during the challenge, but it was solved well thanks to the help of organizers. Overall, it was a very good experience in our research career.