BMVC 2018 Papers

Anshul Gupta

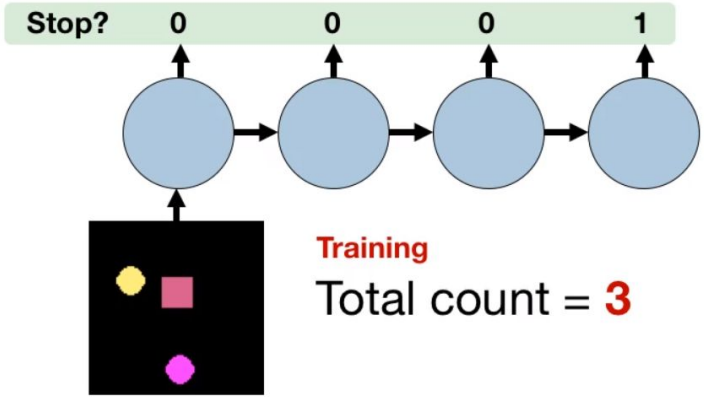
Inductive Visual Localisation: Factorised Training for Superior Generalisation

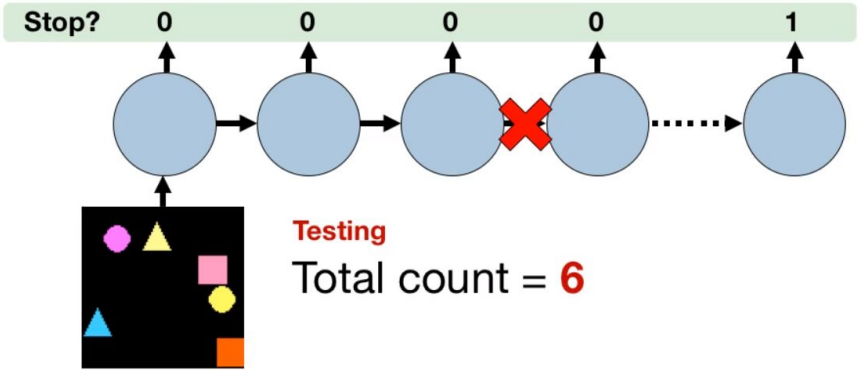
Ankush Gupta, Andrea Vedaldi and Andrew Zisserman
University of Oxford

Introduction

RNNs have poor generalization to sequence lengths beyond those in the training set

Ex. counting


Stop?
0
0
0
1
Training
Total count = 3


Stop?
0
0
0
0
1
Testing
Total count = 6

Proposed Approach

Mathematical induction: allows sequences to be analysed or generated ad infinitum

Train recurrent networks with the explicit notion of induction

Method

To generalise correctly to sequences of arbitrary lengths, an iterative algorithm must maintain a suitable invariant. Ex. list of objects visited so far

Proposed Method: Restrict the recurrent state to a spatial memory map (m_t) which keeps track of the parts of the input image which have already been explored

Method

$m \in R^{H \times W}$ is implemented as a single 2D map of the same dimensions as the image x

Focus is on sequence prediction tasks where each token in the sequence corresponds to a 2D location in the image

Δm_t is trained to encode the 2D location in the image associated with sequence label y_t

Method

$$p(y_{t+1}|y_{1:t}, c_t), \Delta m_{t+1} = \Phi(c_t, m_t)$$

$$m_{t=0} = 0^{H \times W}$$

$$m_{t+1} = m_t + \Delta m_t$$

Training and Inference

The model is trained for one-step predictions, where each training sample is a tuple — (x, y_t, m_t, m_{t+1})

Minimize loss:

$$-\log p(y_t | x_t, m_t) + \gamma ||m_t + \Delta m_t - m_{t+1}||_2^2$$

Experiments

1) Recognising multiple lines of text
2) Counting by Enumeration

Recognising Multiple Lines of Text

	# lines→	1	2	3	4	5	6	7	8	9	10
end-to-end	precision	66.69	63.97	59.23	53.70	-	-	-	-	-	-
	recall	69.27	65.50	56.52	39.14	-	-	-	-	-	-
	ED	15.91	17.81	25.08	43.78	-	-	-	-	-	-
inductive	precision	85.13	84.79	85.57	87.25	87.32	86.11	85.41	85.51	84.57	84.41
	recall	84.89	84.74	85.32	86.99	87.22	85.91	84.43	84.03	80.47	76.80
	ED	6.76	7.79	7.09	6.29	5.77	6.96	8.77	9.11	13.18	17.23

Counting by Enumeration

Dataset	Model	Number of Objects							
		3	4	5	6	7	8	9	10
Coloured Shapes	end-to-end	99.53	99.53	98.93	0	0	0	0	0
	inductive	100	99.89	99.52	98.93	97.18	98.47	95.48	95.45
DOTA	end-to-end	82.00	70.50	74.80	0	0	0	0	0
	inductive	82.50	79.00	75.50	72.50	69.00	43.81	32.21	29.20

VSE++: Improving Visual-Semantic Embeddings with Hard Negatives

Fartash Faghri, David J. Fleet, Jamie Ryan Kiros and Sanja Fidler
University of Toronto, Google Brain

Introduction

Focus is on visual-semantic embeddings for cross-modal retrieval; i.e. the retrieval of images given captions, or of captions for a query image

Performance is measured by R@K, i.e., recall at K – the fraction of queries for which the correct item is retrieved in the closest K points to the query in the embedding space

The correct target(s) should be closer to the query than other items in the corpus

Introduction

New technique for learning visual-semantic embeddings for cross-modal retrieval

Uses the concept of hard negatives in common loss functions

Visual Semantic Embedding

Features for image i: $\varphi(i;\theta_\varphi)$

Features for caption c: $\psi(c;\theta_\psi)$

Projection into joint embedding space:

$$f(i;W_f,\theta_\phi) = W_f^T\,\phi(i;\theta_\phi)$$
$$g(c;W_g,\theta_\psi) = W_g^T\,\psi(c;\theta_\psi)$$

Visual Semantic Embedding

Similarity function:

$$s(i,c) = f(i;W_f,\theta_\phi)\cdot g(c;W_g,\theta_\psi)$$

We minimize cumulative loss:

$$e(\theta,S) = \frac{1}{N}\sum_{n=1}^{N}\ell(i_n,c_n)$$

Loss Function

Traditional hinge based triplet loss function:

$$\ell_{SH}(i,c) = \sum_{\hat{c}}[\alpha - s(i,c) + s(i,\hat{c})]_+ + \sum_{\hat{i}}[\alpha - s(i,c) + s(\hat{i},c)]_+$$

Proposed modification:

$$\ell_{MH}(i,c) = \max_{c'}\left[\alpha + s(i,c') - s(i,c)\right]_+ + \max_{i'}\left[\alpha + s(i',c) - s(i,c)\right]_+$$

Loss Function

Puts emphasis on hard negatives, i.e. the negatives closest to each training query

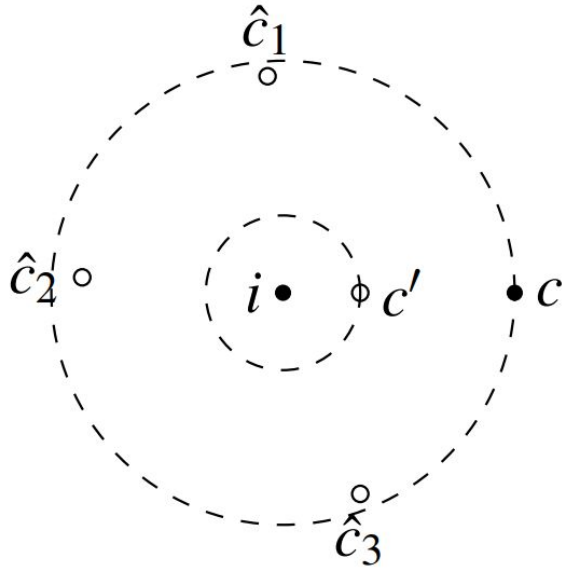
Hardest negatives are given by

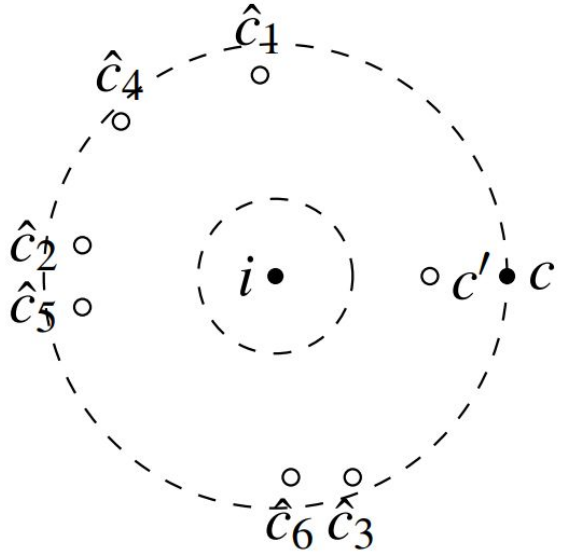
$i' = \text{argmax}_{j!=i}\, s(j, c)$

And

$c' = \text{argmax}_{d!=c}\, s(i,d)$

Loss Function





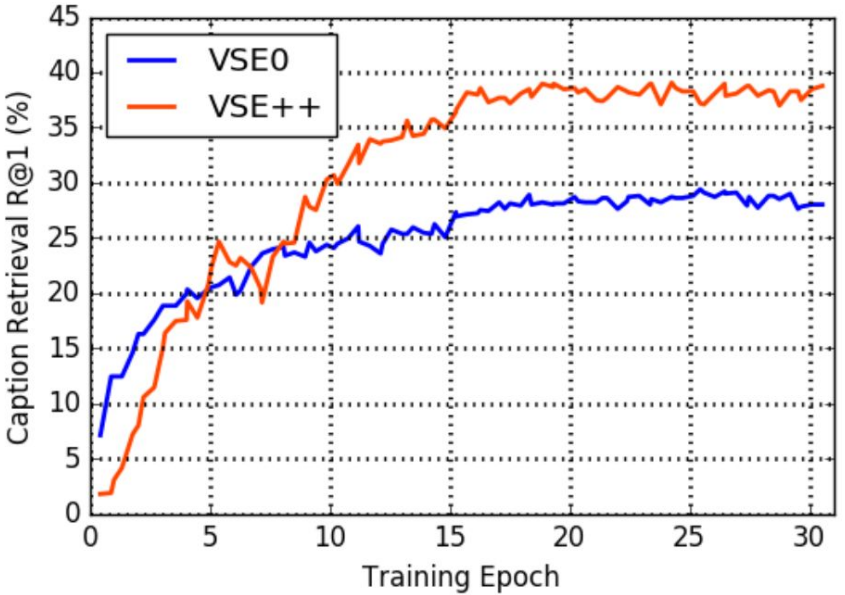
Experiments

#	Model	Trainset	Caption Retrieval				Image Retrieval			
			R@1	**R@5**	**R@10**	**Med r**	**R@1**	**R@5**	**R@10**	**Med r**
		1K Test Images								
1.1	*UVS* ([15], GitHub)	*1C* (1 fold)	43.4	75.7	85.8	2	31.0	66.7	79.9	3
1.2	*Order* ([31])	*10C+rV*	46.7	-	88.9	2.0	37.9	-	85.9	2.0
1.3	Embedding Net ([32])	*10C+rV*	50.4	79.3	69.4	-	39.8	75.3	86.6	-
1.4	sm-LSTM ([12])	?	53.2	83.1	91.5	**1**	40.7	75.8	87.4	2
1.5	2WayNet ([8])	*10C+rV*	55.8	75.2	-	-	39.7	63.3	-	-
1.6	*VSE++*	*1C* (1 fold)	43.6	74.8	84.6	2.0	33.7	68.8	81.0	3.0
1.7	*VSE++*	*RC*	49.0	79.8	88.4	1.8	37.1	72.2	83.8	2.0
1.8	*VSE++*	*RC+rV*	51.9	81.5	90.4	**1.0**	39.5	74.1	85.6	2.0
1.9	*VSE++ (FT)*	*RC+rV*	57.2	86.0	93.3	**1.0**	45.9	79.4	89.1	2.0
1.10	*VSE++ (ResNet)*	*RC+rV*	58.3	86.1	93.3	**1.0**	43.6	77.6	87.8	2.0
1.11	*VSE++ (ResNet, FT)*	*RC+rV*	**64.6**	**90.0**	**95.7**	**1.0**	**52.0**	**84.3**	**92.0**	**1.0**
		5K Test Images								
1.12	*Order* ([31])	*10C+rV*	23.3	-	65.0	5.0	18.0	-	57.6	7.0
1.13	*VSE++ (FT)*	*RC+rV*	32.9	61.7	74.7	3.0	24.1	52.8	66.2	5.0
1.14	*VSE++ (ResNet, FT)*	*RC+rV*	**41.3**	**71.1**	**81.2**	**2.0**	**30.3**	**59.4**	**72.4**	**4.0**

Experiments

#	Model	Trainset	Caption Retrieval				Image Retrieval			
			R@1	**R@5**	**R@10**	**Med r**	**R@1**	**R@5**	**R@10**	**Med r**
3.1	*UVS* ([15])	*1C*	23.0	50.7	62.9	5	16.8	42.0	56.5	8
3.2	*UVS* (GitHub)	*1C*	29.8	58.4	70.5	4	22.0	47.9	59.3	6
3.3	Embedding Net ([33])	*10C*	40.7	69.7	79.2	-	29.2	59.6	71.7	-
3.4	DAN ([24])	?	41.4	73.5	82.5	2	31.8	61.7	72.5	3
3.5	sm-LSTM ([12])	?	42.5	71.9	81.5	2	30.2	60.4	72.3	3
3.6	2WayNet ([8])	*10C*	49.8	67.5	-	-	36.0	55.6	-	-
3.7	DAN (ResNet) ([24])	?	**55.0**	**81.8**	**89.0**	**1**	**39.4**	**69.2**	**79.1**	**2**
3.8	*VSE0*	*1C*	29.8	59.8	71.9	3.0	23.0	48.8	61.0	6.0
3.9	*VSE0*	*RC*	31.6	59.3	71.7	4.0	21.6	50.7	63.8	5.0
3.10	*VSE++*	*1C*	31.9	58.4	68.0	4.0	23.1	49.2	60.7	6.0
3.11	*VSE++*	*RC*	38.6	64.6	74.6	2.0	26.8	54.9	66.8	4.0
3.12	*VSE0 (FT)*	*RC*	37.4	65.4	77.2	3.0	26.8	57.6	69.5	4.0
3.13	*VSE++ (FT)*	*RC*	41.3	69.1	77.9	2.0	31.4	60.0	71.2	3.0
3.14	*VSE0 (ResNet)*	*RC*	36.6	67.3	78.4	3.0	23.3	52.6	66.0	5.0
3.15	*VSE++ (ResNet)*	*RC*	43.7	71.9	82.1	2.0	32.3	60.9	72.1	3.0
3.16	*VSE0 (ResNet, FT)*	*RC*	42.1	73.2	84.0	2.0	31.8	62.6	74.1	3.0
3.17	*VSE++ (ResNet, FT)*	*RC*	52.9	80.5	87.2	1.0	**39.6**	**70.1**	**79.5**	**2.0**

Experiments



License Plate Recognition and Super-resolution from Low-Resolution Videos by Convolutional Neural Networks

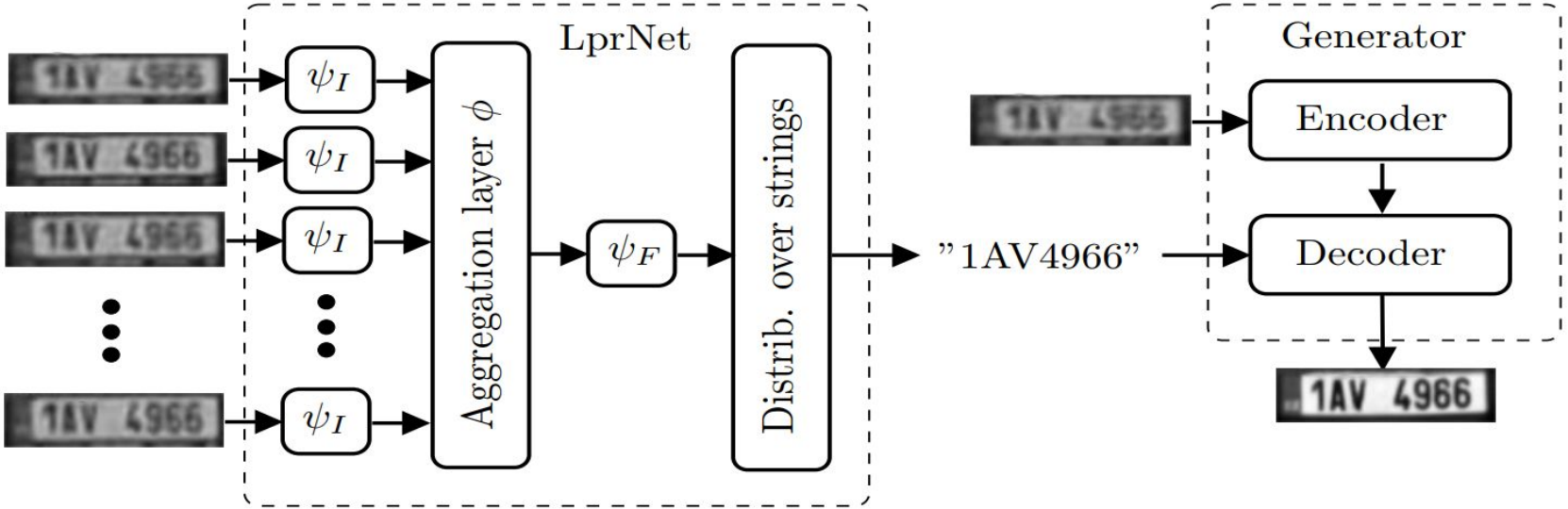
Vojtech Vašek, Vojtech Franc and Martin Urban
Eyedea Recognition, Czech Technical University in Prague

Introduction

CNN for License Plate Recognition (LPR) from low-resolution videos

CNN based super-resolution generator of LP images

Architecture



Architecture

$\psi_I : X \rightarrow R^K$ and $\psi_F : R^K \rightarrow R^D$ are CNNs with a chain architecture composed of convolution, max-pooling, fully-connected and ReLU layers

$\varphi : R^{K \times N} \rightarrow R^K$ is an aggregation layer converting a sequence of N K-dimensional vectors to a single K dimensional vector;

φ_{avg} and φ_{max} considered

Architecture

Parameters θ learned by maximizing the log-likelihood

$$L(\theta) = \sum_{j=1}^{m} \left(\log p(L^j \mid \bar{x}^j; \theta) + \sum_{i=1}^{L^j} \log p_i(c_i^j \mid \bar{x}^j; \theta) \right)$$

Architecture

Generator:

$$F(\omega_d, \omega_e, \omega_l) = \frac{1}{m}\sum_{j=1}^{m}\Big(\|\hat{x}^j - d(e(x^j;\omega_e),\overline{c}^j;\omega_d)\|_1 + \log(1-\ell(d(e(x^j;\omega_e),\overline{c}^j;\omega_d),\overline{c}^j;\omega_l)) + \log\ell(\hat{x}^j,\overline{c}^j;\omega_l)\Big)$$

$$(\omega_d^*, \omega_e^*) \leftarrow \min_{\omega_d,\omega_e}\max_{\omega_l} F(\omega_d, \omega_e, \omega_l)$$

Data: CNN LP recognition

- Video tracks: 31K sequences, avg 72 frames
- Still images: 1.4M high res images
- Synthetic images: Synthetic images

Still images and synthetic images used as first frame. For consecutive frames, distortion transformation applied

5 images per sequence

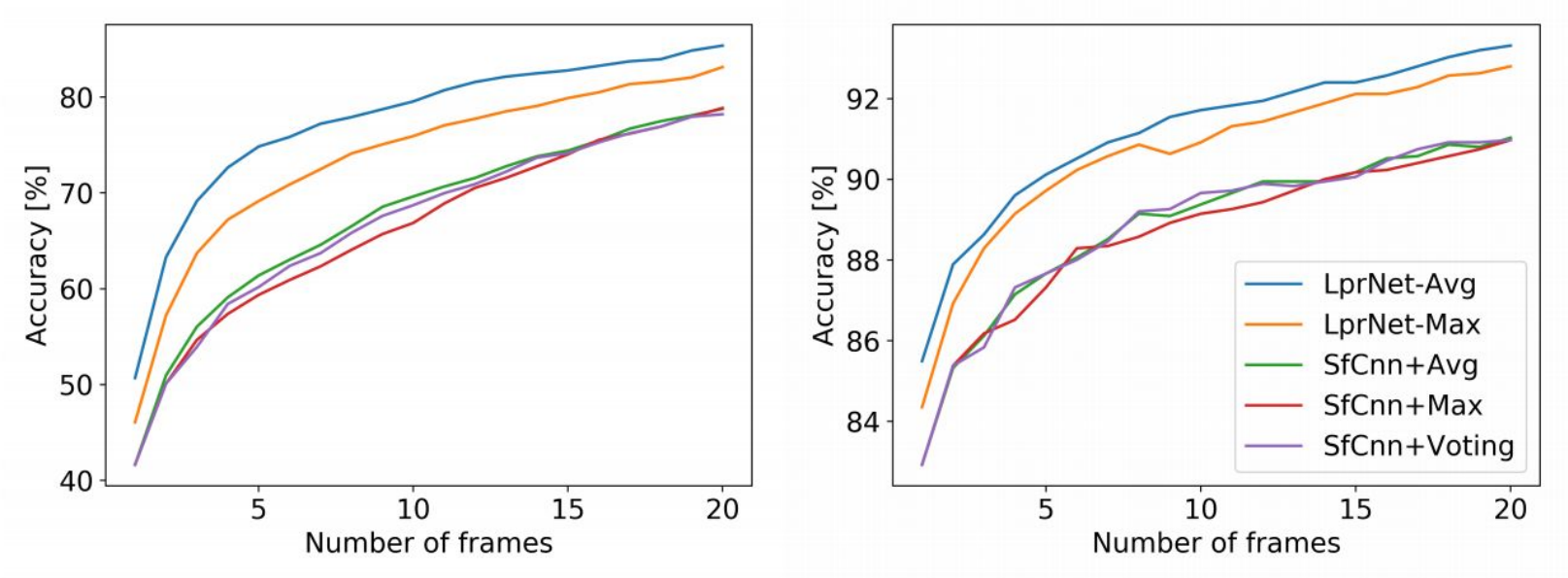
Data: Super-resolution Generator

- Still + synthetic images only

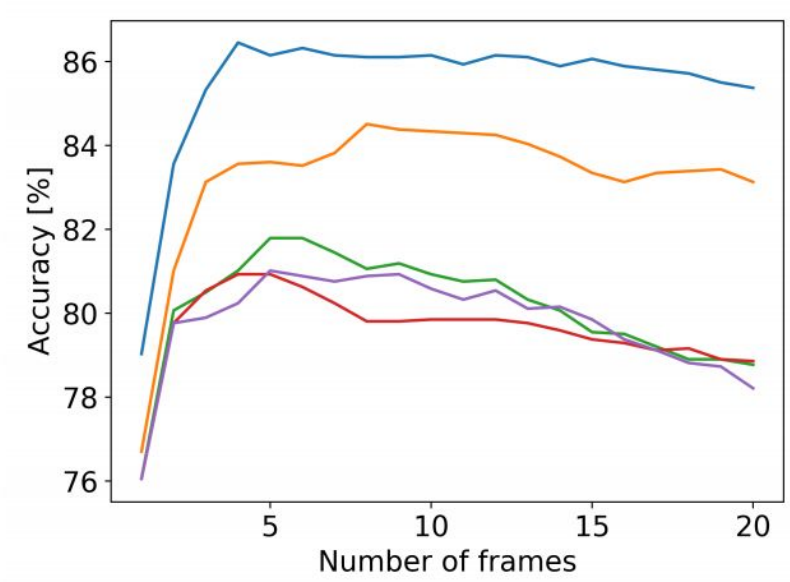
Image distorted by a random affine transform was used as the desired generator’s output

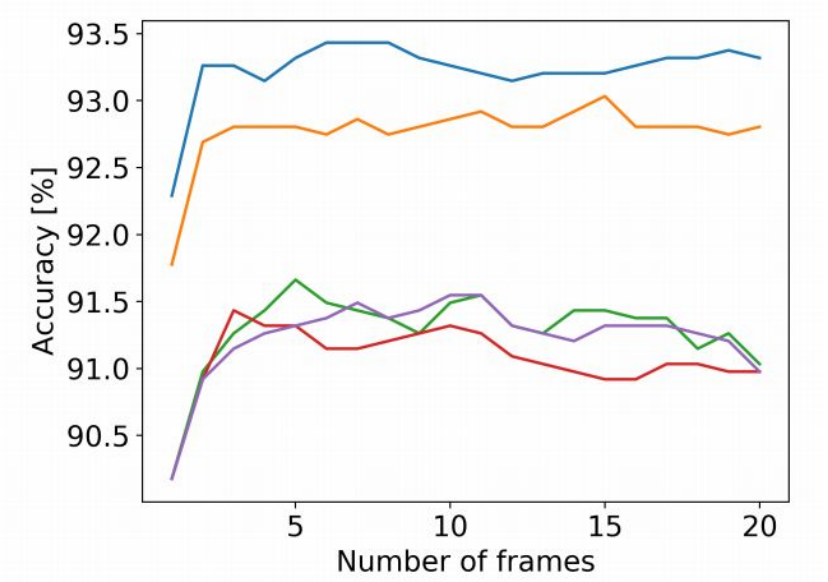
Input image obtained by same distortion transform

Experiments



Experiments





Experiments

Test accuracy w.r.t. the number of image frames in the sequence shown for the proposed LprCnn-Avg/Max and the baselines SfCnn-Avg/Max/Voting.

The left column shows results on low-resolution sequences and the right column on higher-resolution ones. The top row is for sequences with increasing image resolution and the bottom for the decreasing

Gated Fusion Network for Joint Image Deblurring and Super-Resolution

Xinyi Zhang, Hang Dong, Zhe Hu, Wei-Sheng Lai, Fei Wang, Ming-Hsuan Yang
Xi'an Jiaotong University, Hikvision Research, University of California, Merced, Google Cloud

Introduction

Existing super-resolution algorithms cannot reduce motion blur well

State of the art deblurring algorithms generate clear images but cannot restore fine details and enlarge the spatial resolution

Introduction

Solution 1:

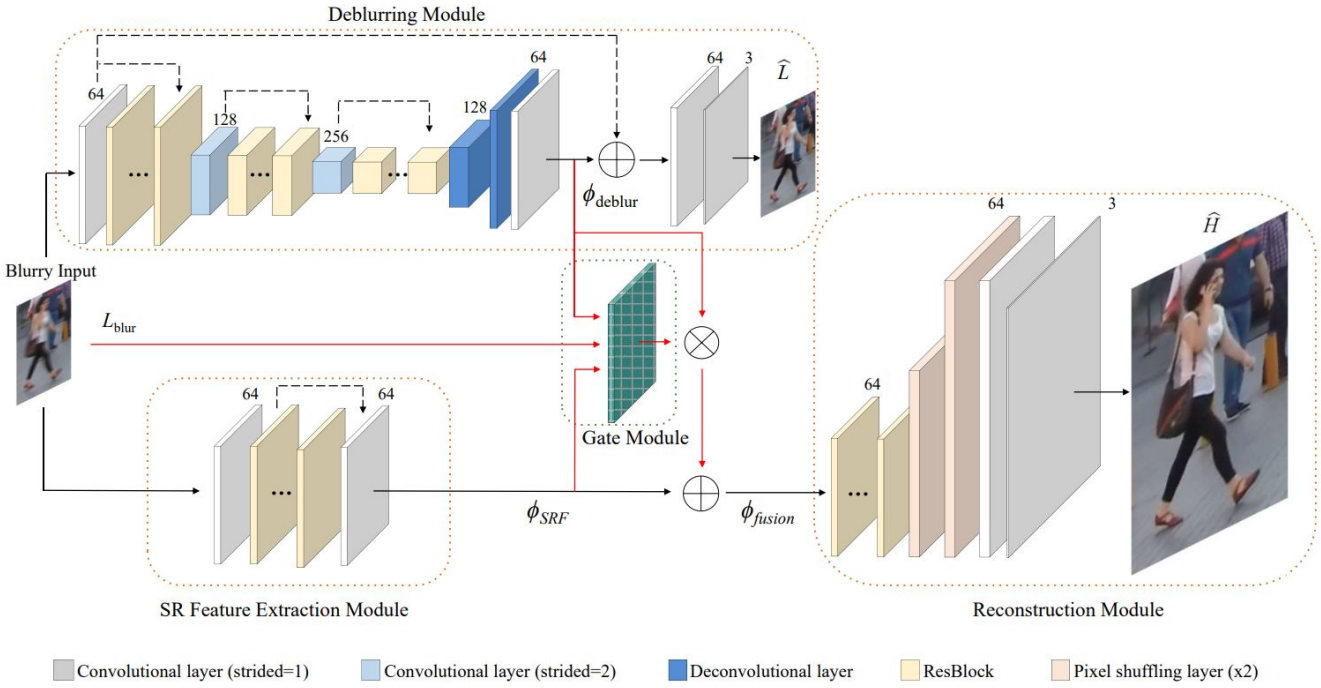
Solve the two problems sequentially, i.e., performing image deblurring followed by super-resolution, or vice versa

Introduction

Problem?

- error accumulation, i.e., the estimated error of the first model will be propagated and magnified in the second model
- the two-step network does not fully exploit the correlation between the two tasks

Architecture



Architecture

- Deblurring module: asymmetric residual encoder-decoder architecture
- Super-resolution feature extraction module: eight resblocks for high dimensional feature extraction
- Gate module: Fuses features from first two modules
- Reconstruction module: fused features φ_{fusion} are fed into eight ResBlocks and two pixel-shuffling layers to enlarge the spatial resolution by 4X

Architecture

Optimize loss function

$$\min \ \mathcal{L}_{SR}(\widehat{H}, H) + \alpha \mathcal{L}_{deblur}(\widehat{L}, L)$$

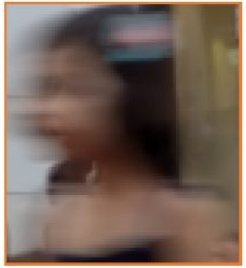
Pixel wise MSE loss function for both L_{SR} and L_{deblur}

Experiments

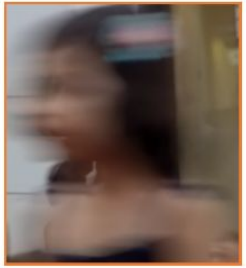
Method	#Params	LR-GOPRO 4× PSNR / SSIM / Time (s)	LR-Köhler 4× PSNR / SSIM / Time (s)
SCGAN [40]	1.1M	22.74 / 0.783 / 0.66	23.19 / 0.763 / 0.45
SRResNet [18]	1.5M	24.40 / 0.827 / 0.07	24.81 / 0.781 / 0.05
EDSR [20]	43M	24.52 / 0.836 / 2.10	24.86 / 0.782 / 1.43
SCGAN* [40]	1.1M	24.88 / 0.836 / 0.66	24.82 / 0.795 / 0.45
SRResNet* [18]	1.5M	26.20 / 0.818 / 0.07	25.36 / 0.803 / 0.05
ED-DSRN* [45]	25M	26.44 / 0.873 / 0.10	25.17 / 0.799 / 0.08
DB [21] + SR [18]	13M	24.99 / 0.827 / 0.66	25.12 / 0.800 / 0.55
SR [18] + DB [21]	13M	25.93 / 0.850 / 6.06	25.15 / 0.792 / 4.18
DB [16] + SR [18]	13M	21.71 / 0.686 / 0.14	21.10 / 0.628 / 0.12
SR [18] + DB [16]	13M	24.44 / 0.807 / 0.91	24.92 / 0.778 / 0.54
DB [16] + SR [20]	54M	21.53 / 0.682 / 2.18	20.74 / 0.625 / 1.57
SR [20] + DB [16]	54M	24.66 / 0.827 / 2.95	25.00 / 0.784 / 1.92
DB [21] + SR [20]	54M	25.09 / 0.834 / 2.70	25.16 / 0.801 / 2.04
SR [20] + DB [21]	54M	26.35 / 0.869 / 8.10	25.24 / 0.795 / 5.81
GFN (ours)	12M	27.74 / 0.896 / 0.07	25.72 / 0.813 / 0.05

Experiments

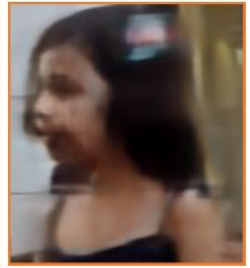
(a) Blurry low-resolution input



(b) Input patch



(c) EDSR [20]



(d) Ours

(e) Input patch

(f) DeepDeblur [21]

(g) Ours

Thank you