Meeting

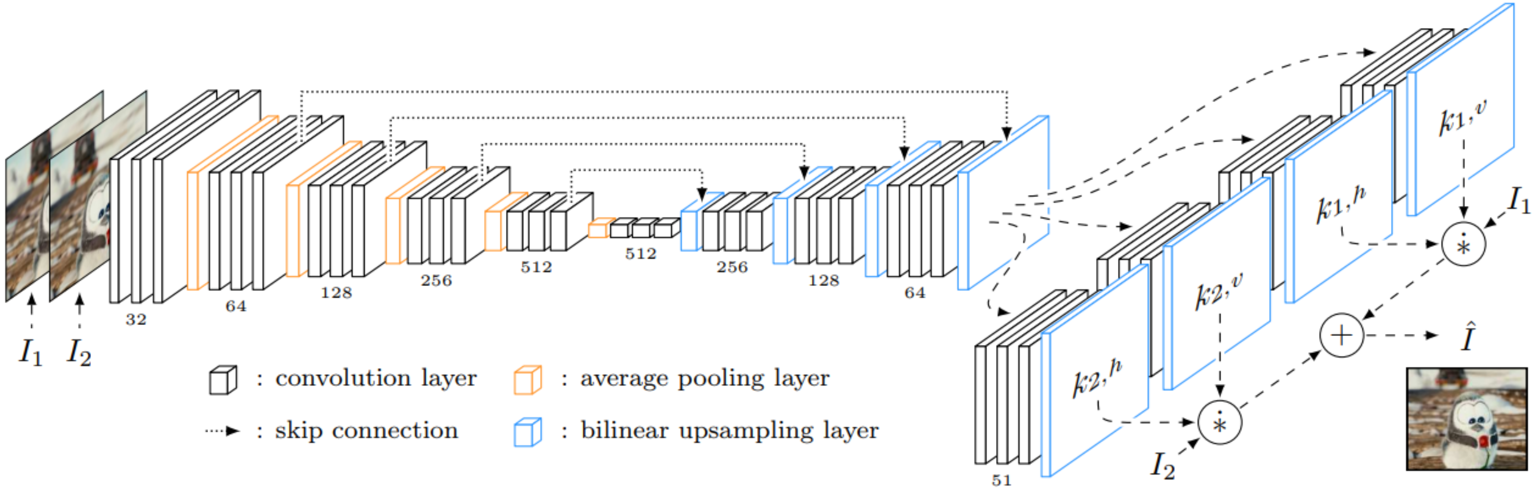
2019/07/15

Literature Search
(Paper Summaries)

- Video Frame Interpolation via Adaptive Separable Convolution(ICCV 2017)

- Traditional video frame interpolation problem approach:
  1) optical flow estimation 2) synthesize intermediate frame
- But, in this paper, authors does not estimate optical flow and only uses 1D convolution operation to generate intermediate frame.
- **Main idea:** input two frame and output two dense pixel-wise weights one intermediate frame.
- **Pros:** Reduce parameters generally by approximating 2D Conv with 2 1D Conv(n^2 => 2n). Greatly reduce computational costs. Sharp images(using CNN to simply input two frames and output intermediate frame gives blurry outputs)
- **Cons:** Supervised learning(But, it is easy to get ground truth in video frame interpolation problem.)
- **Lesson:** Reducing parameters or computational cost can be a good contribution.

Implementation Details



- Goal is to learn 1D horizontal, vertical convolution feature for each input image.
- Construct four network with same architecture which is for each of the four 1D conv olution kernels.
- Train each network with Supervised learning by L1 loss between interpolated frame and ground truth frame.
- Author also tried perceptual loss, but it degrades performance.
- However, training with L1 loss first and fine-tuning with perceptual loss made greate r performance in line with image generation, super-resolution problems.

PhaseNet for Video frame Interpolation(CVPR 2018)

- Most current approach in video interpolation problem generally do not perform well in challenging scenarios such as light changes or motion blur.
- In this paper, author proposes method which handles not only optical flow, but also phase.
- **Main idea:** Map two input images to frequency domains and interpolate frequency. Predict output image by applying inverse fourier transform .
- **Pros:**
- **Cons:** Need to calculate Fourier transform, Supervised.
- **Lesson:** Directly predicting intermediate frame often produces blurry image. To make generated image sharper, use L1 loss rather than L2.(Important)

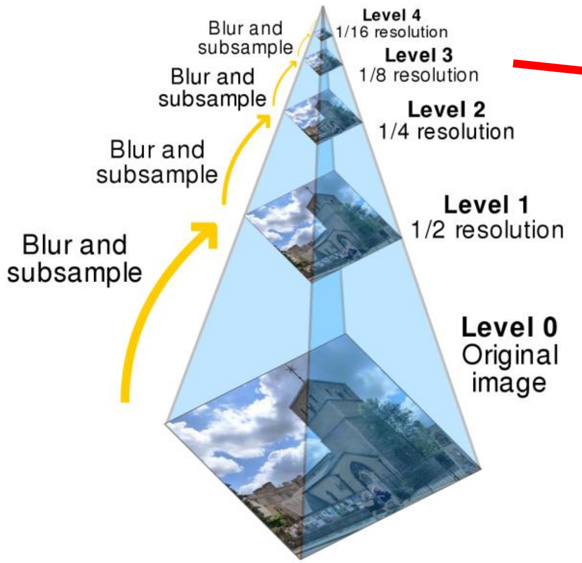
Phase and Steerable pyramid filters

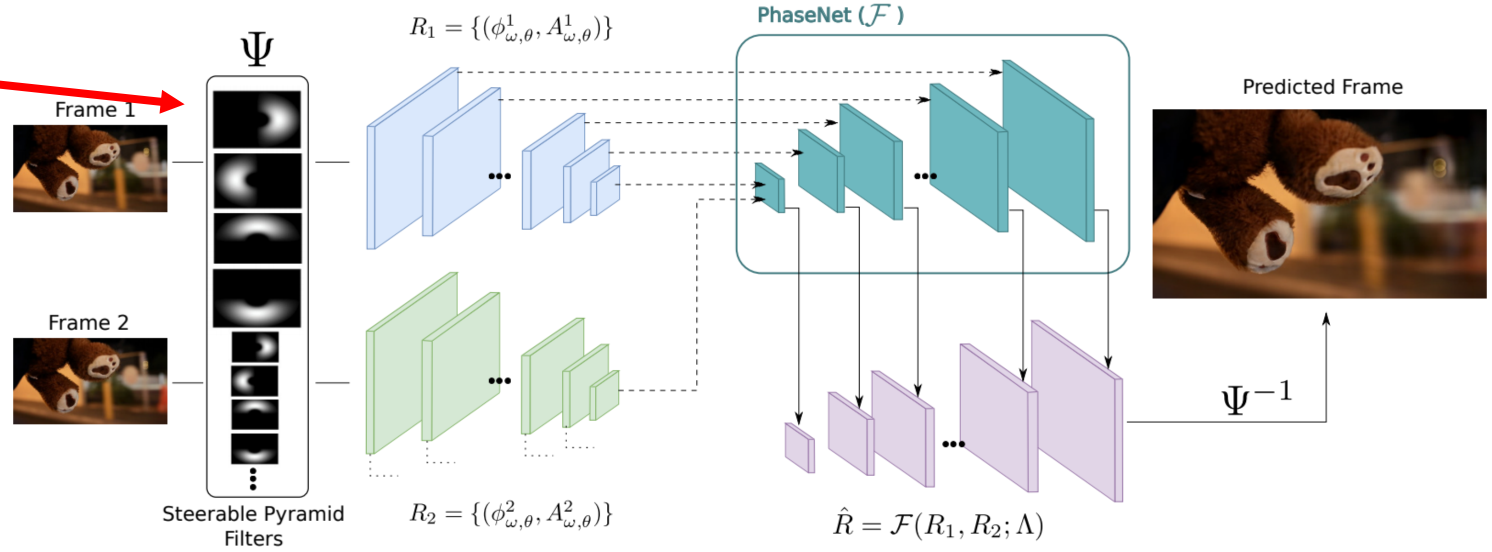
- Phase?
  - Phase calculation: Use discrete Fourier transform and decompose the two input images into a number of oriented frequency bands(map each of the two images to frequency domains)

  $\Rightarrow$ Can be interpreted as a kind of basis transform of input(think about PCA)
- Steerable pyramid filters?
  - Subsample input images recursively and make hierarchy of images with varying resolution.

Reference: Phase-Based Frame Interpolation for Video(CVPR 2015)

Implementation Details



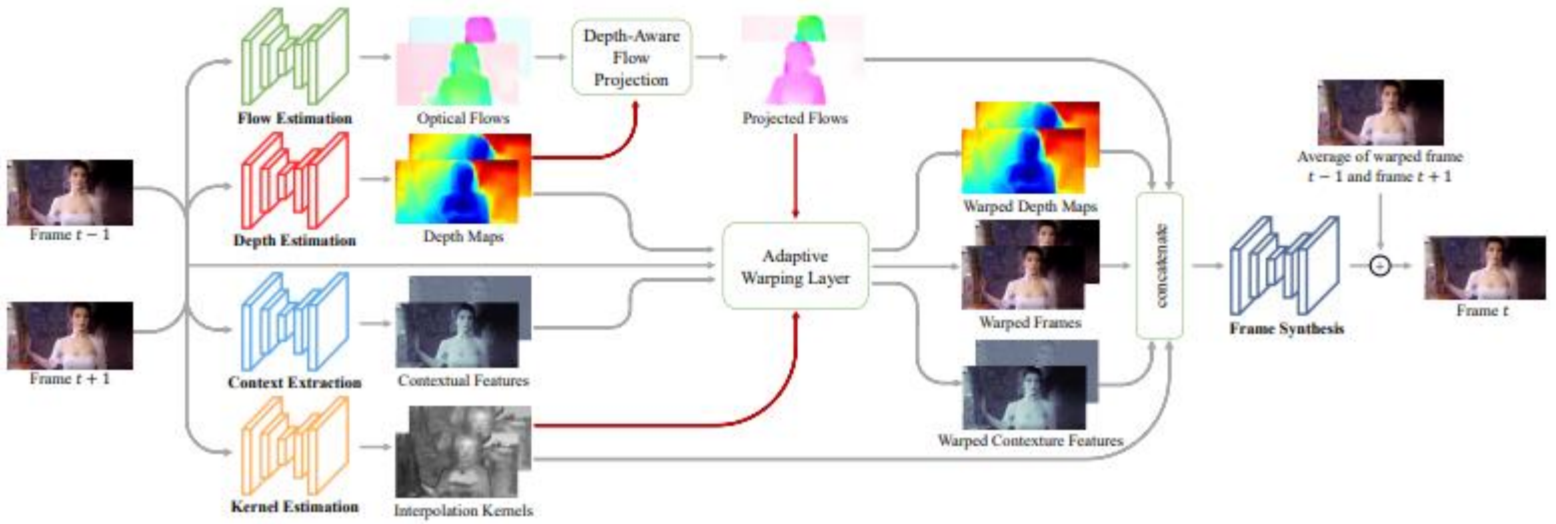


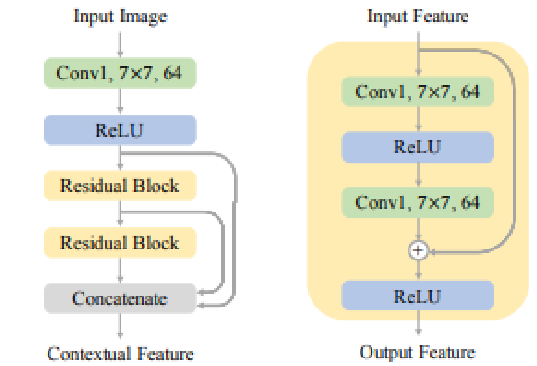
- Apply “steerable pyramid filters” to generate various resolution images.
- And decompose each of the two image to frequencies by Fourier transform.
- Construct network which input is two image in frequency domain and output is intermediate frame’s frequency.
- Finally, apply Inverse transform to interpolated frequency to output intermediate frame.

Depth-aware Video Frame Interpolation(CVPR 2019)

- Close objects from camera should be synthesized more in interpolated frame.
- Using depth estimation, estimating optical flow is more clear.
- The reason why generic CNN outputs blurry interpolated frame is that it cannot model multimodal distribution of images.(I couldn’t understand it)
- **Main Idea:** Depth estimation + optical flow estimation + Context extraction + kernel estimation interpolation.
- **Pros:** Total combination of current SOTA video frame interpolation methods. Can generate intermediate frame at arbitrary time between two frames
- **Cons:** I think that title of the paper is quite different with its contents. Need various ground truth(depth, flow)
- **Lesson:** Naming is important too.

Implementation Details





(a) Context extraction network (b) Residual block

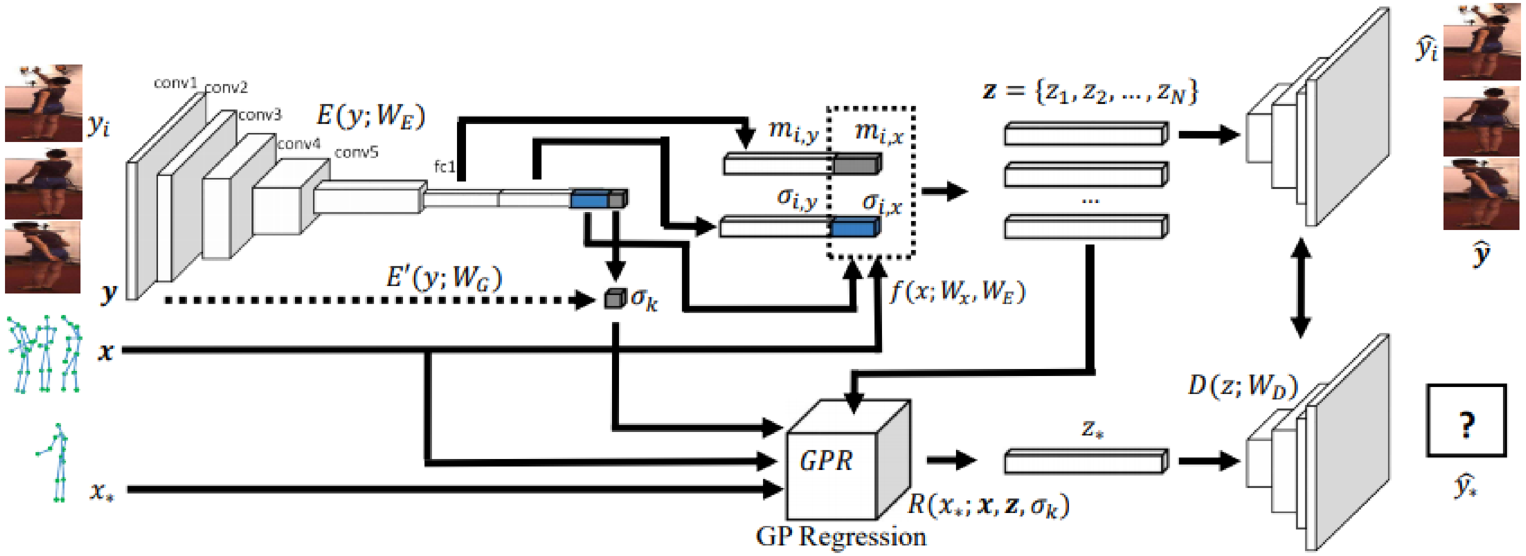
Figure 4. **Structure of the context extraction network.** Instead of using the weights of a pre-trained classification network [23], we train our context extraction network from scratch and learn hierarchical features for video frame interpolation.

- Estimate bidirectional optical flow from frame 1 to 2 and from frame 2 to 1 using pretrained “PWC-net”. And estimate depth by pretrained depth estimation network.
- Using flow information and depth information, calculate optical flow more accurately.
- Extract context and estimate local interpolation kernels.
- By considering estimated denth-aware optical flow, estimate intermediate depth, frame, contextual features.

Variational Autoencoded Regression: High Dimensional Regression of Visual Data on Complex Manifold(CVPR 2019)

- Using VAE, we can model latent variable which captures impotant variations among data points.
- However, doing regression with latent variable as input seems reasonable, it does not perform well generally. Because standard VAE does not trained to do regression well.
- **Main Idea:** Do regression in latent space by combining Gaussian Process Regression and VAE and jointly train them.
- **Pros:** Smooth Interpolation.
- **Cons:** Need large data set to accurately perform high dimensional regression, but time/space complexity is quite high.
- **Lesson:** Gaussian Process model is good for smooth interpolation. VAE latent features can be jointly trained to do other tasks.

Implementation Details



- First, construct standard VAE architecture to reconstruct given input images.
- By training VAE, Variational encoder can output latent variables {means, variances} given input images.
- And fit sampled latent variables with another observed explanatory variables to fit GP regr ession model.
- Use fitted GP regression model to predict new latent variable given new observed explanatory variables.

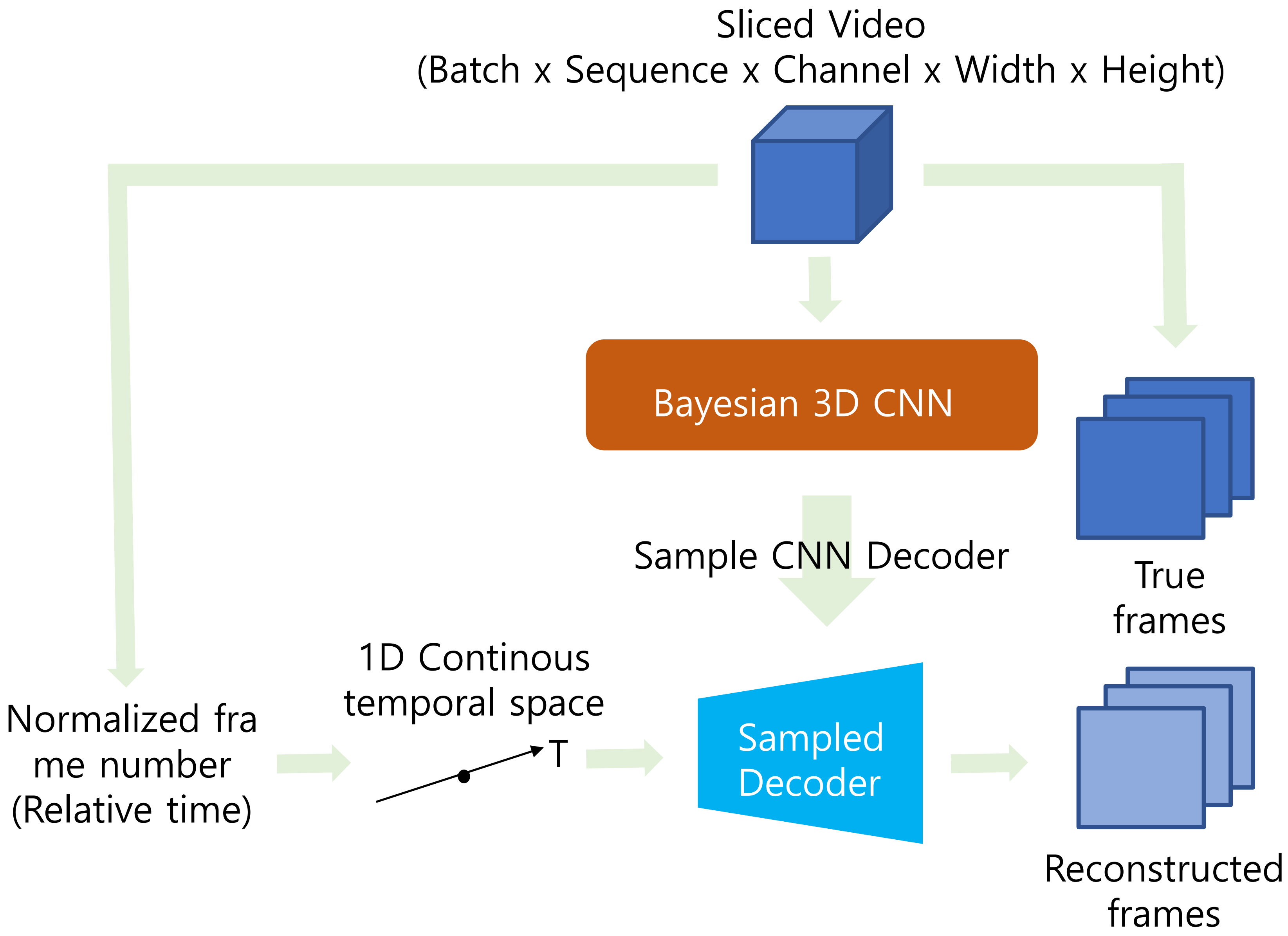
Experiments

My approach for problem

- Ultimate goal was to learning to learn a simple, smooth mapping function that maps frame time to Image given video.
- Also, visualizing “temporal space walking” was my wish.

- Problem formulation
  - Supervised learning by estimating image conditioned on time.
  - Loss = Reconstruction loss + smoothness penalty
  - Smoothness penalty can be gradient penalty of time gradient with respect to reconstruction loss.(I am not sure it is reasonable)

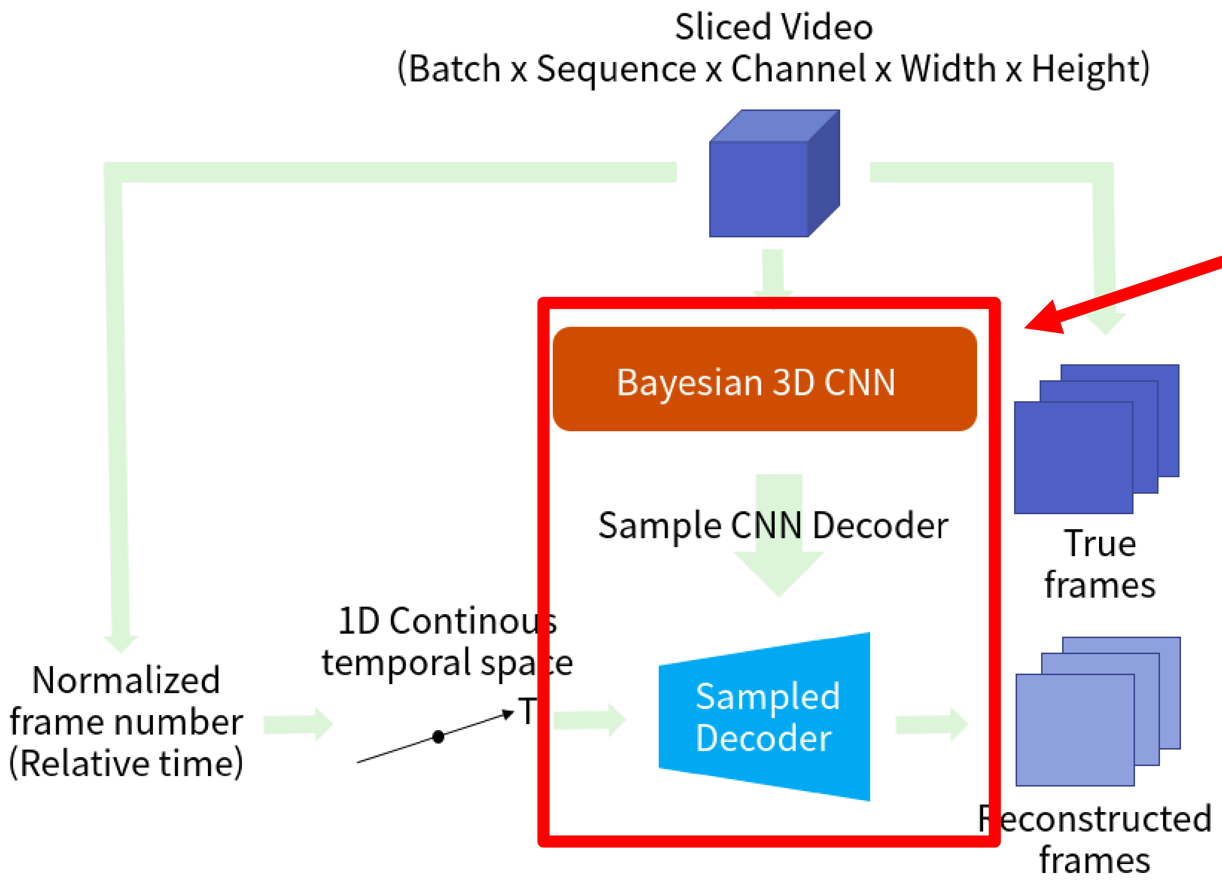
First Idea: Training



Loss = Recontruction loss + Gradient penalty at temporal space

Training goal is to learn a model that:

1) Gain robustness against context changes for different video
2) Get better generalization.



But Sang-heon pointed that this part seems impossible.
I thought that his opinion make sense, too.

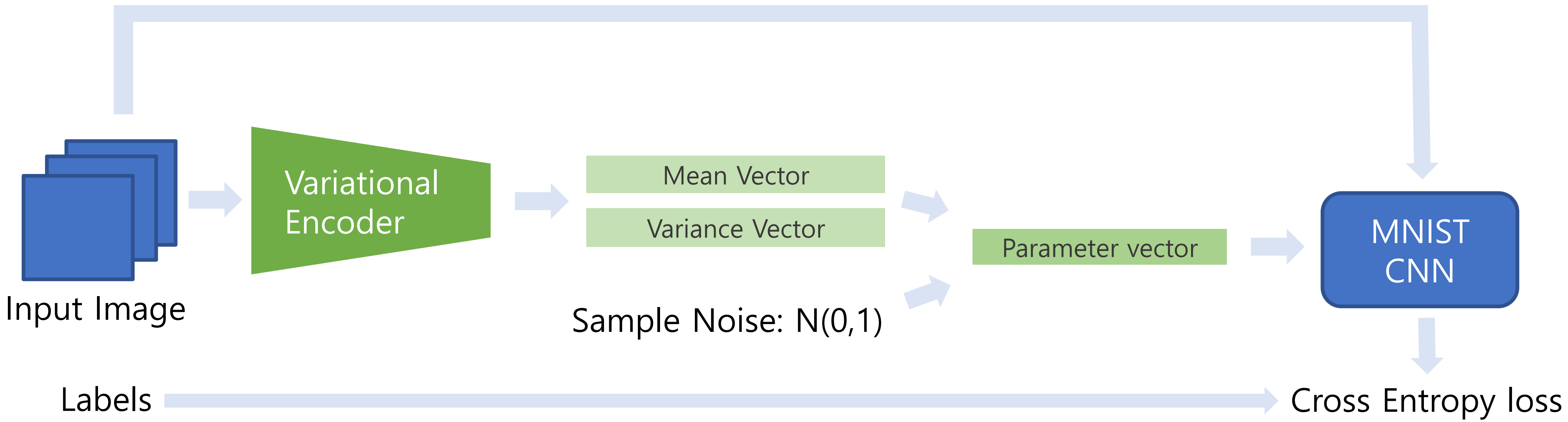
Another idea: Variational Encoder?

Variational Autoencoded Regression: High Dimensional Regression of Visual Data On Complex Manifold(CVPR 2019)

- Constuct variational encoder
- For a given image, variational encoder map each image to latent space by outputting mean vector and variance vector.
- And model a smooth function by gaussian process regression.(Can be interpreted as solving regression problem by learnable distance function(kernel function) between every fitted data points(Generally Convex optimization problem)

$\Rightarrow$ What if I change Gaussian Process regression to CNN? (Since standard GP regression has time complexity of O(n^3), space complexity of O(n^2), it is not practical for large data. In this case, not practical for many frame sequences)

Question: Can Variational Encoder learn the distribution of parameters?



Experimental Setup:

1) Construct simple CNN model which has been known to have high test accuracy after training.
2) Construct another CNN model which outputs mean and variance of parameters in CNN.
3) Backpropagate Cross-Entropy loss to variational encoder and optimize variational encoder only.

Result

```
epoch : 346,train loss = 5.12285, test loss = 4.67339, running test loss = 4.72886, time: 124.59 sec
epoch : 347,train loss = 5.40293, test loss = 4.97005, running test loss = 4.72955, time: 124.32 sec
epoch : 348,train loss = 4.89405, test loss = 4.32834, running test loss = 4.72840, time: 123.94 sec
epoch : 349,train loss = 5.06811, test loss = 5.02731, running test loss = 4.72926, time: 123.96 sec
epoch : 350,train loss = 5.10727, test loss = 4.35988, running test loss = 4.72821, time: 124.00 sec
epoch : 351,train loss = 4.84524, test loss = 4.53541, running test loss = 4.72766, time: 146.95 sec
epoch : 352,train loss = 5.26486, test loss = 4.82169, running test loss = 4.72792, time: 146.88 sec
epoch : 353,train loss = 4.99693, test loss = 5.12295, running test loss = 4.72904, time: 179.70 sec
epoch : 354,train loss = 4.87336, test loss = 4.81076, running test loss = 4.72927, time: 179.95 sec
epoch : 355,train loss = 5.07117, test loss = 4.36777, running test loss = 4.72825, time: 180.74 sec
epoch : 356,train loss = 5.05512, test loss = 4.54885, running test loss = 4.72775, time: 179.39 sec
epoch : 357,train loss = 5.13996, test loss = 4.63156, running test loss = 4.72748, time: 179.93 sec
epoch : 358,train loss = 5.09878, test loss = 4.10300, running test loss = 4.72574, time: 179.38 sec
epoch : 359,train loss = 4.82127, test loss = 4.94845, running test loss = 4.72636, time: 181.51 sec
epoch : 360,train loss = 5.10913, test loss = 4.52768, running test loss = 4.72581, time: 180.24 sec
epoch : 361,train loss = 5.19734, test loss = 4.56392, running test loss = 4.72537, time: 179.10 sec
epoch : 362,train loss = 5.32150, test loss = 4.94781, running test loss = 4.72598, time: 165.95 sec
epoch : 363,train loss = 5.33477, test loss = 4.59838, running test loss = 4.72563, time: 163.75 sec
epoch : 364,train loss = 4.86725, test loss = 5.08715, running test loss = 4.72662, time: 176.95 sec
epoch : 365,train loss = 4.78916, test loss = 4.79366, running test loss = 4.72680, time: 178.80 sec
epoch : 366,train loss = 5.10289, test loss = 4.90142, running test loss = 4.72728, time: 179.34 sec
epoch : 367,train loss = 5.01897, test loss = 3.99928, running test loss = 4.72530, time: 177.45 sec
epoch : 368,train loss = 4.94977, test loss = 4.36831, running test loss = 4.72433, time: 173.82 sec
epoch : 369,train loss = 4.97316, test loss = 5.08215, running test loss = 4.72530, time: 176.64 sec
epoch : 370,train loss = 5.16887, test loss = 4.27206, running test loss = 4.72408, time: 176.00 sec
epoch : 371,train loss = 4.93571, test loss = 5.19073, running test loss = 4.72533, time: 177.58 sec
epoch : 372,train loss = 5.05728, test loss = 4.49682, running test loss = 4.72472, time: 176.20 sec
epoch : 373,train loss = 4.86736, test loss = 4.70898, running test loss = 4.72468, time: 176.76 sec
epoch : 374,train loss = 5.23768, test loss = 4.56737, running test loss = 4.72426, time: 128.58 sec
epoch : 375,train loss = 5.05398, test loss = 4.39143, running test loss = 4.72337, time: 122.21 sec
epoch : 376,train loss = 5.14426, test loss = 4.68398, running test loss = 4.72327, time: 122.20 sec
epoch : 377,train loss = 5.15025, test loss = 4.72425, running test loss = 4.72327, time: 122.00 sec
epoch : 378,train loss = 5.09115, test loss = 4.26079, running test loss = 4.72205, time: 122.08 sec
epoch : 379,train loss = 4.98177, test loss = 4.06096, running test loss = 4.72031, time: 122.37 sec
epoch : 380,train loss = 4.82636, test loss = 4.93960, running test loss = 4.72089, time: 122.07 sec
epoch : 381,train loss = 4.99136, test loss = 5.27597, running test loss = 4.72234, time: 121.97 sec
epoch : 382,train loss = 4.94865, test loss = 4.64626, running test loss = 4.72214, time: 122.49 sec
epoch : 383,train loss = 5.01252, test loss = 4.67929, running test loss = 4.72203, time: 121.92 sec
epoch : 384,train loss = 5.18126, test loss = 4.41131, running test loss = 4.72122, time: 121.80 sec
epoch : 385,train loss = 5.14948, test loss = 4.63606, running test loss = 4.72100, time: 122.37 sec
epoch : 386,train loss = 5.02946, test loss = 4.79328, running test loss = 4.72119, time: 122.00 sec
epoch : 387,train loss = 5.15448, test loss = 4.74847, running test loss = 4.72126, time: 121.70 sec
epoch : 388,train loss = 5.06833, test loss = 4.92270, running test loss = 4.72178, time: 122.19 sec
epoch : 389,train loss = 4.82415, test loss = 4.49622, running test loss = 4.72120, time: 121.96 sec
epoch : 390,train loss = 5.26494, test loss = 3.93171, running test loss = 4.71918, time: 122.27 sec
epoch : 391,train loss = 5.20952, test loss = 4.58693, running test loss = 4.71884, time: 121.97 sec
epoch : 392,train loss = 5.06202, test loss = 4.47471, running test loss = 4.71822, time: 121.77 sec
epoch : 393,train loss = 5.30149, test loss = 4.27940, running test loss = 4.71711, time: 122.46 sec
epoch : 394,train loss = 5.02714, test loss = 4.81618, running test loss = 4.71736, time: 121.97 sec
epoch : 395,train loss = 5.13257, test loss = 3.91510, running test loss = 4.71533, time: 121.71 sec
epoch : 396,train loss = 5.19331, test loss = 4.17552, running test loss = 4.71397, time: 121.58 sec
epoch : 397,train loss = 5.15059, test loss = 4.18126, running test loss = 4.71263, time: 121.60 sec
epoch : 398,train loss = 4.88619, test loss = 5.09501, running test loss = 4.71359, time: 121.51 sec
```

However, it seems that loss never decreases…
(Checked that there is no other problems such as code errors and non-reasonable scales of hyperparameters)

-> Methodology design was completely wrong.

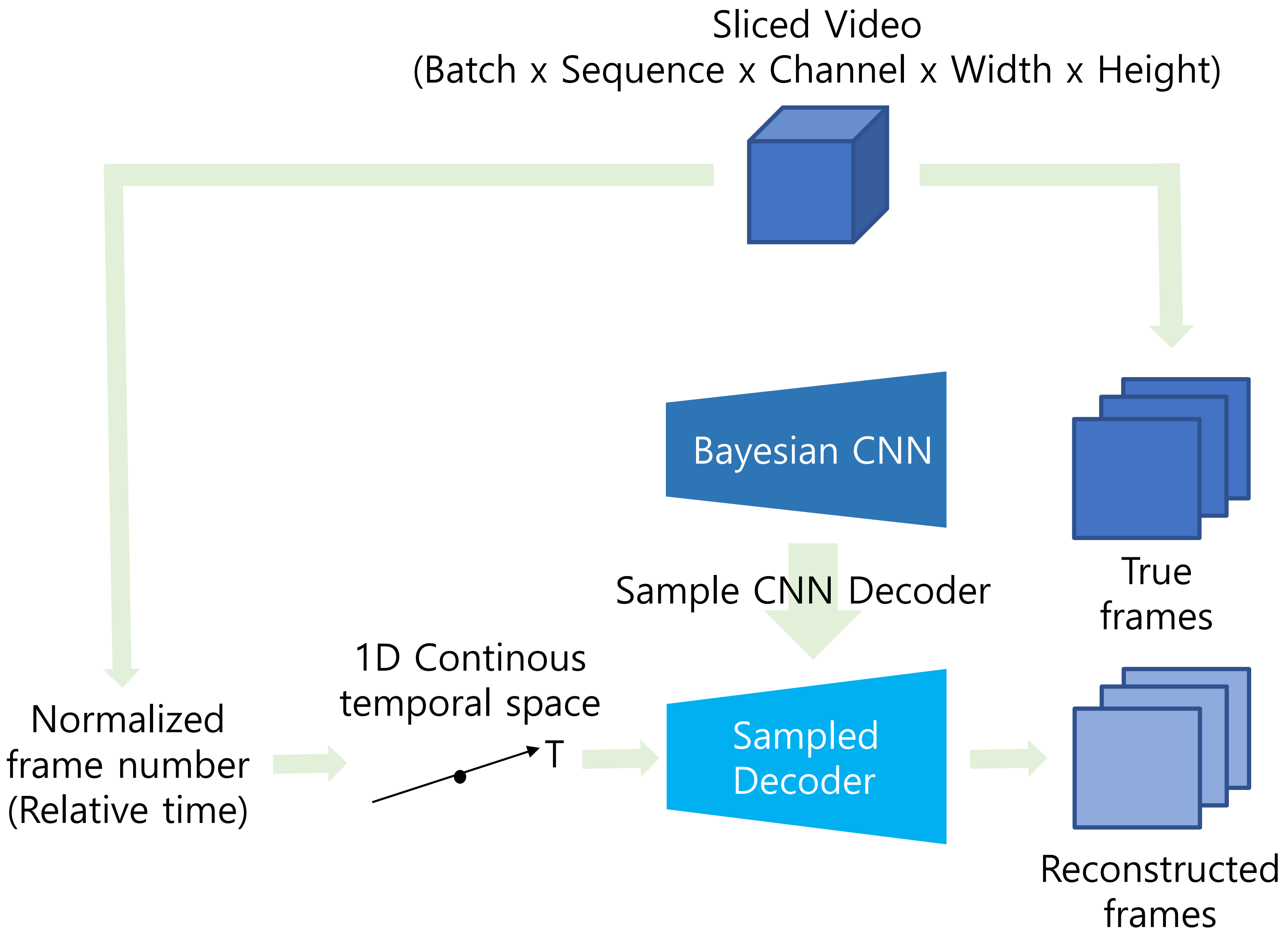
I think that since variational encoder outputs factorized (independent) gaussian parameter, it cannot correctly model the joint distributions of CNN parameters.

To Do

- More research considerations:
  - Learning a function that maps time to image given video is possible?
  - How can we make use of data specific information(context, depth, phase…) differently from currently published research?

- Video frame interpolation problem has a large intersection with video summarization and video compression. How about changing problem slightly?

Next planned experiment



⇒ Just test training Bayesian CNN model directly first, rather than trying complicated way.
(Test simple supervised learning fi rst for checking scalability)

+ Implement original Variatio nal auto-encoded regressio n paper

Other Consideration

Main obstacle in lab life

1) Keep focusing in ML models itself rather than developing practical and creative ideas.

2) Maybe Computer Vision domain knowledge is not enough to start application.

3) Doesn’t even got a clue how to write a paper…

Thinking about future?

1) Change in military systems(reducing alternative military service vacancy)
2) Ability to Implement paper from scratch? And Coding test?

NAVER Clova

상시채용

소개

• Clova AI Research는 클로바에 필요한 선행 AI기술 연구개발 위한 글로벌 팀입니다. 클로바를 글로벌 수준의 AI 플랫폼으로 만들기 위해 우수한 역량을 보유한 AI 연구자 및 엔지니어분들을 찾고 있습니다.

• Clova AI Research는 클로바 내부 뿐 아니라 네이버 내의 타 조직들과 매우 강력하고 열린 협업체계를 구축하고 있습니다.

• 글로벌 팀이므로 공용언어는 영어를 사용합니다.

지원자격

• Clova의 성능향상을 위한 AI 선행 기술 연구가 가능한 인력
(자연어처리, 컴퓨터비전, 추천 등을 포함한 기계학습기반 AI기술 전 분야)

선행기술 연구 전문가 (Research Scientist)

- 최신 AI 연구 기술에 대한 이해 , 우수한 연구 역량 및 실적
- 프로젝트를 리딩하고 연구방향을 제시할 수 있는 강력한 연구개발 리더십

AI SW 엔지니어

- Tensorflow, PyTorch, MXNet, Caffe2 등 오픈소스 프레임워크 기반 개발역량
- 멀티 GPU 및 고성능 컴퓨팅관련 업무 경험
- 머신러닝/딥러닝 모델 설계 및 구축 경험
- 최신 AI 논문에 대한 빠르고 정확한 구현 능력

우대

선행기술 연구 전문가

- AI관련분야(기계학습, 자연어처리, 컴퓨터비전, 응용수학 등) 박사학위 소지자 혹은 이에 준하는 역량/경력 소유자
- AI관련 분야 Top-tier 학회 논문의 저자(NIPS, ICML, ICLR, CVPR, ACL, EMNLP, ICCV, AAAI, IJCAI, KDD 등)
- 우수한 영어작문 및 커뮤니케이션 역량 보유자

AI SW 엔지니어

- CUDA기반 GPU 프로그래밍 혹은 Hadoop /Spark 등 분산처리 프로그래밍 경험자
- 우수한 시각화 및 Front-end 개발 경험 보유자

ICCV 2019 fee(not official)

ICCV 2019 Registration Fee

PASSPORT REGISTRATION	Early	Standard	On-site
	Till Sat 31st August (2z3:59 Korea time)	Sun 1st November – Tue 15th October (23:59 Korea time)	Sun 27th October – Sat 2nd November
IEEE or CVF member passport	KRW 730,000	KRW 950,000	KRW 1,070,000
Non-member passport	KRW 880,000	KRW 1,160,000	KRW 1,350,000
Student IEEE or CVF member passport*	KRW 250,000	KRW 350,000	KRW 400,000
Non-member student*	KRW 300,000	KRW 420,000	KRW 480,000
Life/Retired members passport	KRW 450,000	KRW 700,000	KRW 850,000
ONE DAY WORKSHOP/TUTORIAL PASS*			
IEEE or CVF Member One-day Workshop/Tutorial pass	KRW 340,000	KRW 380,000	KRW 400,000
Non-member One-day Workshop/Tutorial pass	KRW 410,000	KRW 450,000	KRW 480,000
Student IEEE or CVF member One-day	KRW 200,000	KRW 250,000	KRW 300,000
Non-member Student One-day	KRW 240,000	KRW 300,000	KRW 360,000
Exhibition & Posters Only One-day	KRW 400,000	KRW 400,000	KRW 400,000
* Student fees and Daily fees DO NOT INCLUDE attendance to the Banquet. Registrant who wish to attend the Banquet can purchase an "Extra Banquet ticket" while registering.			
Others			
Extra Banquet ticket for accompanying person** **Limited seats availability. Attendance will be granted on a first-come first-served basis.	Closed		
Poster Printing	KRW 150,000	KRW 150,000	KRW 150,000
Additional Expo *** Required the Expo Code	KRW 400,000	KRW 400,000	KRW 400,000
Press Pass *** Required the Press Code			
Sponsor *** Required the Sponsor Code			

Can be changed and not confirmed

I think I should not attend it…!

Good News

[학석사연계과정 안내 및 유의사항]

1. 학석사연계과정은 학사과정 재학생이 대학원 진학을 예약하는 제도로 대학원 진학 시 별도의 전형절차(원서접수, 시험, 면접 등) 없이 합격됩니다.
2. 학사과정 졸업평가 통과가 졸업조건인 대학에서는 졸업평가 통과의무를 면제합니다.

3-1. 학사과정 수강학점 범위 내에서 최대 6학점까지(전문대학원 학술연구과정은 9학점까지) 대학원 과정 석박공통과목을 수강하여 학사과정 전공학점(전공일반)으로 취득할 수 있으며(단, 소속학과와 진학 예정 대학원이 동일계열이 아닌 경우 학사과정은 선택학점으로 인정), 이 학점은 대학원 입학 후 대학원 전공학점으로도 중복 인정됩니다.
3-2. 학사과정 수강학점 범위 내에서 팀연구프로젝트에 참여해 최대 6학점까지 학사과정 전공학점(전공일반)으로 취득할 수 있으며, 이 학점은 대학원 입학 후 대학원 전공학점으로 중복 인정됩니다.
3-3. 대학원과정 석박공통과목을 수강하여 취득한 학점과 팀연구프로젝트에 참여해 취득한 팀연구학점을 합하여 총 6학점(전문대학원 학술연구과정은 9학점)을 초과할 수 없습니다.

4. 학사과정 재학 중 취득하여 대학원 입학 후 중복인정 받은 학점이 일반대학원 6학점 이상, 전문대학원 학술연구과정 9학점 이상일 경우 대학원 과정 수업연한을 1학기 단축 가능합니다.

☐ **위의 유의사항을 확인하였습니다.**

- 신청내용 입력 후 제출 버튼을 누르면 제출이 완료됩니다. 학과장 결재 후 신청결과가 '신청완료'로 표시되며, 발표예정일자에 선정결과가 '선정'으로 되어야 최종 선발됩니다.

신청결과	결재완료	선정결과	선정	발표예정일자	2019.07.10. (수)