LEC 20

Variational Autoencoders

- vae 학습 방식은 다음과 같음

- input data를 encoder에 넣어 예측한 diagonal Gaussian 분포 p(zlx)의 평균과 공분산이 unit gaussian 분포 p(x)와 같아지도록 KL divergence term을 줄이는 방향으로 학습함

- 다음, p(zlx)에서 샘플링한 z를 decoder에 넣어 p(xlz)의 평균과 공분산을 얻고, 원래 input x의p(xlz) 분포 하에서의 가능도를 계산해 data reconstruction term을 높이도록 학습함

- 학습이 끝나면, train data에서 얻은 p(z)에서 z를 샘플링해 decoder에서 p(xlz)를 얻은 후, x를 샘플링해 새로운 데이터를 생성할 수 있음

- p(z)를 diagonal하다고 가정했기 때문에 z의 차원들이 서로 독립적이 되어, 각 차원 값을 변화시켜 특정 feature가 변형된 이미지를 샘플링할 수 있음

- vae는 이미지 생성이 빠르고, q(zlx)를 추론하기 때문에 feature representation을 얻을 수 있다는 장점이 있지만, p(x)를 직접 학습하지 않고 lower bound를 예측하기 때문에 정확하지 않으며, diagonal Gaussian을 가정하기 때문에 생성된 이미지가 흐릿해지는 단점도 있음

- vector-quantized variational autoencoder는 autoregressive 모델과 variational 모델의 장점을 합쳐 성능을 높인 방식임

- vae와 유사한 모델로 latent vector 대신 feature vector의 latent grid를 학습하고, raw pixel이 아닌 latent code space에서 PixelCNN을 이용해 data를 생성함

Generative Adversarial Networks

- gan은 p(x)를 모델링하지 않으면서도 p(x)로부터 샘플링 가능하며, 가장 높은 성과를 보임

- p(z)에서 샘플링한 z를 생성자 G에 넣어 샘플을 생성하고, 판별자 D에서 생성된 샘플과 실제 데이터를 분류함

- 생성자는 fake data에 대해 D(G(z))=1이 되도록 학습하고, 판별자는 D(x)=1, D(G(z))=0이 되도록 학습하며, G와 D의 gradient를 번갈아 가며 학습함

- 이 때 생성자는 log(1-D(G(z)))를 최소화하면 vanishing gradient 문제가 생기므로, -log(D(G(z)))로 바꿔 학습함

- objective 함수 식을 정리해보면, 판별자가 Pdata(x)/(Pdata(x)+Pg(x)) 값을 가지며, 생성자가 Pg(x) = Pdata(x)을 만족할 때 global minimum이 됨을 알 수 있음

- 실제 학습 때는 G와 D를 고정된 architecture의 neural network를 사용해 표현하기 때문에, 한정된 space의 함수만 표현 가능하므로, optimal 값에 수렴하지 못할 수 있음

Conditional GANs

- conditional gan은 생성자와 판별자의 input에 label y를 추가적으로 넣어, 특정 클래스의 data를 생성할 수 있음

- y에 label 대신 text를 넣는 text-to-image, 저해상도 이미지를 넣어 고해상도 이미지를 출력하는 super-resolution 등 다양하게 변형 가능함

LEC 21

Last Layer

- first layer의 filter를 시각화하면 edge나 color 정보를 갖는 것을 알 수 있지만, higher layer는 시각화한 결과를 직관적으로 해석하기 어려움

- 그 대신 last layer의 feature vector로 test image에 nearest neighbor와 차원 축소를 적용했을 때, low level feature가 아닌 클래스에 대한 semantic feature를 잘 담고 있다는 것을 알 수 있음

Visualizing Activations / Maximally Activating Patches

- 중간 layer의 filter weight 대신 activation map을 시각화 해 각 filter가 어떤 feature를 학습했는지 알 수 있음

- 또는 뉴런의 activation을 최대화시키는 image patch들을 시각화해 각 neuron이 찾는 feature를 알아보는 방법도 있음

Saliency

- image의 특정 부분을 가렸을 때의 예측 확률 변화로 얻은 saliency map을 통해 어떤 부분이 모델의 판단에 중요한지 알 수 있는 occlusion 방법이 있음

- 계산량을 줄인 backprop 방법은 각 image 픽셀에 대해 class score의 gradient를 계산해 saliency map을 얻음

- 중간 layer 뉴런의 gradient를 통해서도 픽셀 중요도를 알 수 있고, 이 경우에는 ReLU를 역전파할 때, 추가적으로 gradient가 음수일 때도 전파하지 않는 guided backprop을 이용함

Gradient Ascent

- gradient ascent 방식은 입력 image에 의존하지 않고, activation을 최대화하는 image를 직접 생성함

- zero image에서 시작해 neuron value의 gradient를 각 pixel에 대해 역전파해 image를 업데이트하는 과정을 반복적으로 진행하고, 생성된 image가 자연스럽도록 regularization 항도 추가함

- 실제 image에서 시작해 다른 클래스에 대해 gradient ascent를 진행해 adversarial example을 만들 수도 있음

Feature Inversion

- feature inversion은 특정 이미지의 feature vector와 생성 이미지의 feature vector가 같아지도록 학습하며, layer가 깊어질수록 중요한 정보만 남게 되는 모습을 보여줌

- deepdream은 gradient를 layer의 activation 값으로 설정해, 특정 layer의 activation을 증폭시킨 이미지를 생성함

Neural Texture Synthesis

- gram matrix는 각 픽셀의 C차원 vector간 외적을 평균한 값으로, 공간 정보는 버리고 filter끼리의 co-occurrence, correlation 정보를 나타냄

- neural texture synthesis는 random noise로 초기화된 생성 이미지와 input texture 이미지의 각 layer에서의 gram matrix가 서로 같아지도록 학습하는 방식으로 진행됨

- texture synthesis는 texture 정보를 보존하고, feature reconstruction은 공간 정보를 보존하므로, style image와 content image을 각각의 input으로 넣어 neural style transfer을 할 수 있음

LEC 22

Self-Supervised Learning

- self-supervised learning은 label이 아닌 raw data에서 얻은 signal을 예측하는 pretext task를 통한 학습을 의미하며, 학습을 거친 모델을 downstream task에 적용하는 데 사용됨

- pretext task는 input을 복원하는 generative task, input에 관한 정보를 바탕으로 예측하는 discriminative task, 여러 signal을 이용하는 multimodal task로 나눌 수 있음

Generative

- sparse autoencoder는 기존의 autoencoder에 sparse activation 제약을 추가했고, denoising autoencoder는 input에 noise를 추가했음

- grayscale 이미지를 rgb 이미지로 복원하는 colorization, 가려진 부분을 예측하는 inpainting 방법도 있음

- masked autoencoder는 vision transformer를 이용해 image 패치 중 일부를 mask 처리한 후, 남은 패치들을 encoder에 넣어주고, decoder에서 mask patch들을 예측하는 방법임

Discriminative

- generative task는 downstram task에서 중요하지 않은 사소한 부분에 집중하는 단점을 discriminative task로 해결 가능함

- context prediction은 한 image의 두 개의 패치를 각각 파라미터를 공유하는 cnn에 넣어, 두 패치의 상대적 위치를 맞추는 task임

- deep clustering은 cnn으로 구한 image feature를 k-means로 분류하고, 그 결과로 다시 cnn을 학습하는 과정을 반복함

- exemplar cnn은 augmented patch를 입력해 실제 image를 예측하는 classification task임

- contrastive learning은 유사한 image는 서로 feature도 유사하도록, 다른 image는 feature도 다르도록 학습하고, augmentation으로 positive, negative pair를 얻음

Multimodel Self-Supervised Learning

- 언어 data는 의미 정보를 담는 데 유리하고, 어떤 개념이든 표현 가능하며, 수집하기 쉽기 때문에 image와 caption pair의 multimodal로 자주 쓰임

- VirTex 연구는 image를 cnn에 입력해 얻은 feature와 caption을 transformer에 넣어 language modeling을 진행함. 이후 학습된 cnn을 downstream task에 사용 가능함

- CLIP 연구는 image caption pair를 contrastive learning으로 학습함. 학습이 끝나면 label text와 image 유사도를 계산해 image classification을 zero-shot으로 수행할 수 있음

LEC 23

Depth Map

- depth map은 각 픽셀의 카메라와의 거리를 나타내며, 1\*H\*W 크기를 가지고, 3d 센서를 통해 직접 얻을 수 있음

- rgb 이미지를 fully convolutional network에 입력해 per-pixel loss로 depth map을 학습할 수 있음. 이 때 거리에 따른 object 크기가 달라지는 문제는 scale invariant loss로 보정할 수 있음

- surface normal은 object surface의 xyz 방향을 나타내며, 3\*H\*W 크기를 가지고, 각 픽셀의 각도를 비교해 loss를 계산함

- depth map은 가려진 부분은 예측할 수 없는 단점이 있음

Voxel Grid

- voxel grid는 물체의 occupancy를 3d grid로 표현함

- voxel을 input으로 받은 후, 3d conv 연산을 해 classification을 할 수 있음

- 2d image로부터 voxel을 생성할 경우, 2d cnn에서 얻은 3차원 feature를 fc layer를 거친 후 reshape해 4차원 feature를 만든 후 3d conv을 하는 과정으로 진행됨

- 3d conv은 계산량이 많기 때문에, 2d conv만을 이용한 후 마지막 layer에서 v개의 채널을 voxel의 z 차원으로 해석하는 voxel tube 방식을 사용하기도 함

Implicit Surface

- 3d object의 좌표를 입력으로 받아, 각 position이 shape의 안과 바깥 중 어디에 위치하는지를 출력하는 implicit function을 neural network로 학습할 수 있음

- 그 후, 3d shape의 일부를 sampling해 implicit function을 거쳐 explicit representation을 얻는 과정을 반복해, 전체 object의 explicit shape를 얻을 수 있음

Point Cloud

- point cloud는 3d shape을 P개의 point들의 집합으로 표현함

- point의 밀도를 조절해 부분적인 디테일을 나타낼 수 있지만, explicit한 표현이 아니므로 후처리가 필요하다는 단점도 있음

- PointNet은 pointcloud를 입력 받아 class score를 출력함. 각 point에 대해 독립적으로 mlp를 거친 후, max-pooling하는 과정으로 진행되기 때문에 point 순서는 신경 쓰지 않음

- pointcloud 간 비교를 할 때는 loss function으로 chamfer distance를 이용함. Pointcloud set의 각 point마다 가장 가까운 point와의 L2 거리의 합으로 계산함

Triangle Mesh

- triangle mesh는 3d shape을 v개의 point 집합인 vertices와 point들을 꼭짓점으로 하는 삼각형들의 집합인 faces로 표현함

- face 크기를 조절해 detail을 나타낼 수 있고, vertex에 feature를 추가해 interpolation을 통해 색상, 텍스쳐 등의 정보도 표현할 수 있음

- Pixel2Mesh는 여러 아이디어를 통해 2d image를 받아 triangle mesh를 출력함

- 처음엔 ellipsoid mesh로 시작해, network를 거쳐 반복적으로 refine하는 과정으로 진행됨

- graph convolution은 receptive field에 있는 vertex들의 feature vector와 weight를 곱하는 과정을 통해 각 vertex의 새로운 feature vector를 계산함

- mesh의 vertex를 평면에 projection시킨 후, interpolation을 이용해 2d image feature를 가져오는 과정을 통해 image와 mesh를 연결시킬 수 있음

- 예측한 mesh와 ground truth mesh 간의 loss는 point들을 샘플링한 후 chamfer distance를 계산해 구할 수 있음

Metrics / Cameras

- outlier에 민감한 chamfer distance 대신 threshold를 기준으로 계산한 f1 score를 loss function으로 사용 가능함

- 3d shape을 예측할 때 표준좌표계를 사용하는 canonical coordinate보다, input과 같은 방향을 이용하는 view coordinate가 새로운 shape이나 클래스에 더 일반화가 잘 됨

Mesh R-CNN

- mesh r-cnn은 mask r-cnn에 mesh head를 추가해 object마다 3d shape을 생성함

- mesh deformation을 이용해 3d shape을 생성하지만, 기존의 방법은 ellipsoid 형태에 고정돼 생성 가능한 형태에 한계가 있기 때문에 initial mesh로 voxel prediction을 사용함

LEC 24

Video Classification

- video data는 image data에 시간 차원을 추가한 4차원 tensor로 나타낼 수 있음

- video classification은 action을 인식하는 task임

- video data는 용량이 크기 때문에, subsampling으로 fps를 낮춘 짧은 clip을 train할 때 사용하고, test할 때는 전체 video를 여러 clip으로 나눈 후 출력된 예측값을 평균해 최종 예측을 구함

Single-Frame CNN

- single-frame cnn 모델은 video의 각 프레임을 독립적으로 2d cnn으로 학습하며, 단순하지만 강력한 baseline임

Late Fusion

- fc layer를 사용한 late fusion 모델은 2d cnn에서 출력된 각 frame의 feature들을 concatenate하고, flatten한 후 mlp에 넣어 class score를 출력함. 그러나 fc layer의 파라미터가 많이 필요하고, overfitting이 발생할 수 있음. 이 문제는 frame feature를 global average pooling을 해 개선할 수 있음

Early Fusion

- late fusion 모델은 frame 간 low-level motion 차이를 비교하기 어렵기 때문에, 초반부터 frame을 비교하는 early fusion 모델이 제안됨

- early fusion 모델은 여러 프레임의 4차원 input을 3차원으로 reshape한 후, 첫 conv layer에서 temporal information을 붕괴시켜 2d cnn만을 진행함

3D CNN

- early fusion과 late fusion 둘 다 서서히 증가하는 spatial receptive field와 달리 temporal receptive field는 한 번에 커지기 때문에, layer 하나에서만 temporal information을 다루기엔 부족함

- 3d cnn 모델은 temporal information을 여러 layer에 걸쳐 서서히 처리할 수 있음

- 3d conv은 같은 motion이 다른 time step에서 발생하더라도, 하나의 filter만으로 학습 가능한 temporal shift-invariance 특성을 가짐

Recognizing Actions from Motion

- 연속된 두 프레임을 이용해 각 pixel의 움직임을 계산하는 optical flow를 통해 local motion 정보를 얻을 수 있음

- two-stream networks는 frame image로부터 공간 정보를 계산하는 cnn과, optical flow로부터 시간 정보를 계산하는 cnn을 병렬로 놓고, 두 output을 합쳐 class score를 출력함

Modeling long-term temporal structure

- cnn으로 짧은 video에 대한 feature vector를 얻고, 그 feature들을 rnn에 입력하는 방식으로 long-term video를 처리할 수 있음

- recurrent cnn은 두 가지 방식을 합쳐, multi-layer rnn 구조에 cnn을 대신 사용함

- 이전 layer의 같은 timestep에서의 feature와 같은 layer의 이전 timestep에서의 feature를 input으로 받음

Spatio-Temporal Self-Attention

- rnn은 긴 sequence를 다루기엔 느리기 때문에, self-attention을 적용해 볼 수 있음

- 4차원 feature로 self-attention을 수행하는 nonlocal block을 3d cnn 사이에 넣을 수 있음

- 3d cnn 모델은 기존 2d cnn 모델 구조를 재사용하고, filter weight를 시간 차원 수만큼 복사해 사용할 수 있음

SlowFast Networks

- slowfast 모델은 optical flow 없이 raw pixel 만을 input으로 이용한 two-stream network임

- slow branch는 low frame data를 입력해 높은 channel dimension과 낮은 time dimension을 가지고, fast branch는 반대로 high frame date를 입력함

- 따라서 각각 공간 정보와 시간 정보를 학습하게 됨

LEC 25

LEC 26

Reinforcement Learning

- 강화학습은 agent가 환경으로부터 state를 받아 action을 수행해 reward를 받는 과정을 반복하는 문제를 의미함

- 이 때 학습을 통해 최대의 보상을 받는 것을 목표로 함

Reinforcement Learning vs Supervised Learning

- 지도학습의 model이 dataset으로부터 input을 받아 prediction을 전달한 후 최종 loss를 받는 과정은 강화학습 과정과도 유사하지만, 몇 가지 차이점이 있음

- 강화학습의 reward와 state transition은 랜덤한 값이 될 수 있음

- reward가 직전의 action만으로 결정되지는 않음

- reward의 gradient를 구할 수 없음

- state가 action에 따라 변할 수 있고, agent가 학습함에 따라서도 변하므로 non-stationary함

- 따라서 일반적으로 강화학습 문제가 지도학습보다 풀기 어려움

Markov Decision Process

- markov decision process는 possible state set, possible action set, (state,action) pair에 따른 reward 분포와 next state 분포, discount factor의 5개 요소로 구성된 tuple로 강화학습 문제를 정의함

- reward와 next state가 오직 현재 state에만 의존하는 markov property를 가짐

- 누적 reward를 최대로 얻도록 하는 optimal policy를 찾는 것이 목적임

Value Function and Q Function

- 특정 state의 가치함수는 그 state에서 policy를 따를 때 기대되는 reward를 의미함

- q 함수는 (state,action) pair의 기대 reward를 의미함

Q-Learning

- optimal q-function Q\*는 optimal policy를 따를 때의 q-function을 뜻함

- bellman equation으로 Q\*의 recurrence relation을 알 수 있음

- 이를 통해 Q함수를 bellman equation에 따라 업데이트해 Q\*에 수렴시키는 value iteration 방식을 이용 가능함

- 그러나 모든 (state,action) pair에 대한 Q함수를 계산해야 하므로, 따로 계산하기보단 neural network를 이용해 Q함수를 구하는 Deep Q-learning 방식을 사용 할 수 있음

- 이 방법은 현재 weight에 따라 target인 Q함수 값이 계속 변하는 non-stationary 문제와 batch 샘플링 방식에 관한 문제가 있음

Policy Gradients

- q-learning은 Q함수를 학습하기 어렵거나, policy를 학습하는 게 더 쉬운 경우도 있기 때문에, 이 경우 policy gradient 방식을 이용함

- policy gradient의 목표는 특정 policy를 따를 때의 기대 reward를 최대화하는 것이지만, reward의 gradient를 계산할 수 없어 바로 gradient ascent 방법을 쓸 수 없음

- objective 함수를 경로 x에 대해 적분한 후 계산하면, 샘플링한 경로 x에서의 reward f(x)와, 모델 weight에 대한 action score의 gradient 값을 통해 전체 함수의 gradient를 구할 수 있음