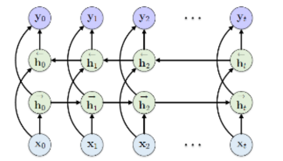
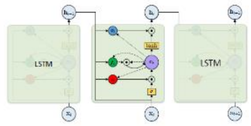
Bidirectional RNN



Long short-term memory

- Motivation
  - vanishing or exploding gradient for larger time step
  - need to "preserve" the earlier hidden node activations for prediction at current time t (Hochreiter and Schmidhuber, 1997)
    - remove and add information to the cell
    - gated mechanism



Memory cell consists of input gate i_t, forget gate f_t, cell c_t, output gate o_t and hidden state h_t which are operated as

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$
$$h_t = o_t \tanh(c_t)$$

Gated recurrent unit

Two gates and one combined state

Reset gate

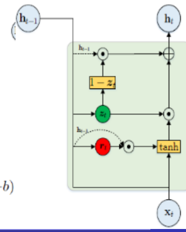
$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$

Update gate

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$

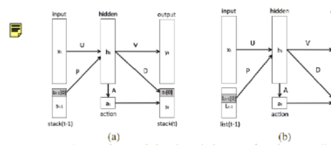
Output state

$$\tilde{h}_t = \tanh(W_x x_t + W_h(r_t \odot h_{t-1}) + b)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$



Stacked recurrent neural network (Joulin and Mikolov, 2015)

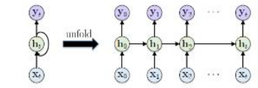
- prediction model for next symbol in a stream of discrete data
- a structured memory inspired by pushdown automaton using a stack
- learn how to operate this memory through optimization tools



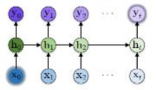
(a) neural network extended with pushdown stack and a controlling mechanism that learns what action (among PUSH, POP and NO-OP)
(b) the same model extended with a doubly-linked list with actions INSERT, LEFT, RIGHT and NO-OP

Recurrent neural network

- Unfolding in RNN



- Gradient vanishing and gradient exploding



LSTM versus GRU

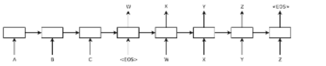
- Long short-term memory
  - input gate, forget gate and output gate
  - possess an internal memory
  - apply a nonlinearity transformation
- Gated recurrent unit
  - update gate and reset gate
  - no internal memory
  - no nonlinearity on the output
- RNNs with tanh, LSTM and GRU units (Chung et al., 2
  - GRU and LSTM outperformed tanh
  - GRU and LSTM converged faster than tanh
  - choosing GRU or LSTM depends on the dataset

Neural Turing machine versus memory network

- Most machine learning models lack an easy way to
  - read and write to part of a long-term memory component
  - combine this seamlessly with inference
- Neural Turing machine (Graves et al., 2014)
  - learns to read from and write to memory cells without explicit supervision
  - allows end-to-end training via content-based soft attention
  - emulates algorithmic mechanism in a way that allows gradient-based optimization
- End-to-end memory network (Sukhbaatar et al., 2015)
  - includes memory cells that can be accessed via an addressing mechanism
  - combines learning strategies for inference with a memory component that can be read and written to

Sequence to sequence learning

- Traditional DNN was sensibly encoded with vectors with a fixed dimensionality
- Many important problems are best expressed with sequences whose lengths are unknown a *priori*
- An input sequence "ABC" is encoded and decoded to produce "WXYZ" as the output sequence (Sutskever et al., 2014)



- LSTM architecture is applied to deal with this problem

Sequence learning

- RNN can not deal with sequential learning with input and output sequences in different lengths
- Sequence to sequence learning is performed by
  - first, map the input sequence to a fixed-sized vector using on RNN
  - second, map the vector to the target sequence using another RNN
- LSTM is used to estimate $p(y_1, \dots, y_{T'} | x_1, \dots, x_T)$ where $\{x_1, \dots, x_T\}$ is an input sequence and $\{y_1, \dots, y_{T'}\}$ is its output sequence whose length T' may differ from T
- LSTM language model is calculated by

$$p(y_1, \dots, y_{T'} | x_1, \dots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \dots, y_{t-1})$$

- LSTM computes this probability by obtaining the fixed dimensional v of $\{x_1, \dots, x_T\}$ given by the last hidden state of LSTM

CTC decoding

- Expected transcription loss is minimized

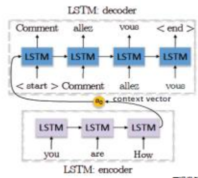
$$\mathcal{L}(x) = \sum_y \Pr(y|x)\mathcal{L}(x,y) = \sum_y \sum_{a \in \mathcal{B}^{-1}(y)} \Pr(a|x)\mathcal{L}(x,y) = \sum_a \Pr(a|x)\mathcal{L}(x, \mathcal{B}(a))$$

- RNN transducers perform decoding with beam search to yield an n-best list of candidate transcriptions
- CTC fits naturally into the existing neural network classifiers
- CTC is similar to HMM, but differs slightly in interpretation of network outputs
- CTC outperformed HMM and HMM-RNN (Graves et al., 2006)

Machine translation

(Sutskever et al., 2014) propose a sequence to sequence model

- compress all the information into a fixed length vector
- performance decreases as the input sentence increases



Evaluation for text generation

BLEU (bilingual evaluation understudy) (Papineni et al., 2002)

- evaluate the quality of the generated text
- calculate the corresponding scores

$$\text{BLEU-}N(C,S) = b(C,S)\exp\left(\sum_{n=1}^{N} w_n \log CP_n(C,S)\right)$$

***n*-gram precision** compares a candidate translation against multiple reference translations

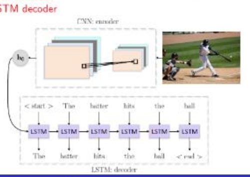
$$CP_n(C,S) = \frac{\sum_i \sum_k \min(h_k(c_i), \max_{j \in m}(h_k(s_{ij})))}{\sum_i \sum_k h_k(c_i)}$$

Brevity penalty (BP) forces a high-scoring candidate translation must match the reference translations in length

$$b(C,S) = \begin{cases} 1, & \text{if } l_c > l_r \\ e^{(1 - l_r/l_c)}, & \text{if } l_c \le l_r \end{cases}$$

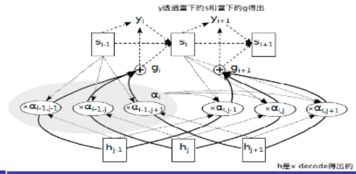
Image caption generator

- Describe the content of an image is a challenging task
  - capture the objects in images
  - express relations between objects
- (Vinyals et al., 2015) presents an end-to-end system for the problem
  - CNN encoder
  - LSTM decoder



Attention network

$$\alpha_i = \text{Attend}(s_{i-1}, \alpha_{i-1}, h) \quad g_i = \sum_{j=1}^{L} \alpha_{i,j} h_j$$
$$y_i \sim \text{Generate}(s_{i-1}, g_i) \quad s_i = \text{Recurrency}(s_{i-1}, g_i, y_i)$$



- **Content-based attention**
  - $\alpha_i = \text{Attend}(s_{i-1}, \alpha_{i-1}, h) \to \alpha_i = \text{Attend}(s_{i-1}, h)$
  - estimation of attention weight

$$e_{i,j} = \text{Score}(s_{i-1}, h_j), \quad \alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{j=1}^{L}\exp(e_{i,j})}$$

  - limitation: identical or very similar elements of h are scored equally regardless of their position in the sequence

- **Location-based attention**
  - $\alpha_i = \text{Attend}(s_{i-1}, \alpha_{i-1}, h) \to \alpha_i = \text{Attend}(s_{i-1}, \alpha_{i-1})$
  - in speech recognition, it aims to predict the distance between consequent phonemes using s_{i-1} only
  - hybrid attention mechanism is fitted to speech recognition
- Score function in content-based attention was calculated by

$$e_{i,j} = w^\top \tanh(Ws_{i-1} + Vh_j + b)$$

- Content-based attention can be location-aware through alignment produced at the previous step
  - extract k vectors $f_{i,j} \in \mathbb{R}^k$ for every position j of the previous alignment α_{i-1} by convolving it with a matrix $F \in \mathbb{R}^{k \times r}$

$$f_i = F * \alpha_{i-1}$$

  - scoring mechanism is updated by using

$$e_{i,j} = w^\top \tanh(Ws_{i-1} + Vh_j + Uf_{i,j} + b)$$

Machine translation with attention

- (Bahdanau et al., 2015) introduces attention mechanism into sequence to sequence model
  - alignment model
  - translation model

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

- Compute attention weight

$$\alpha_{ij} = \frac{\exp(e)_{ij}}{\sum_{k=1}^{T_x}\exp(e)_{ik}}$$

where $e_{ij} = a(s_{i-1}, h_j)$

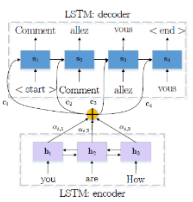
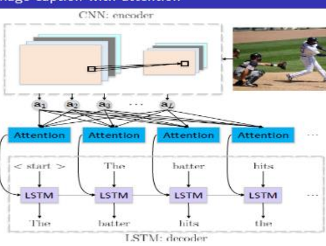
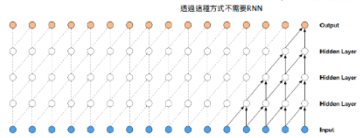


Image caption with attention

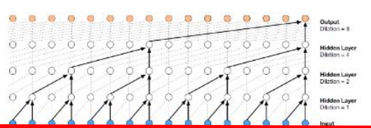


WaveNet

- Causal convolution
  - cannot depend on any of the future timesteps
  - shifting the output of a normal convolution by a few timesteps
  - CNN is faster than RNN



- Dilated convolution
  - filter is applied over an area larger than its length by skipping input values with a certain step
  - similar to pooling or strided convolutions, but the output has the same size as the input
  - dilation 1 yields the standard convolution
  - receptive field to grow exponentially with depth



Convolutional encoder

- Encoder consists of two stacked convolutional networks (Gehring et al., 2017)
  - CNN_a produces the key vector z_j

$$z_j = \text{CNN}_a(e_j)$$

  - CNN_c produces the value vector v_j

$$v_j = \text{CNN}_c(e_j)$$

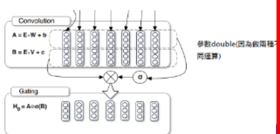
- Conditional input c_i to the decoder is

$$a_i = \text{Attention}(z_j, s_i)$$
$$c_i = \sum_{j=1}^{T} a_{ij} v_j$$

- Gated linear unit (Dauphin et al., 2017) is calculated via convolution operation $*$ for hidden layers $h_0, \dots, h_L$ as

$$h(\mathbf{E}) = (\mathbf{E} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{E} * \mathbf{V} + \mathbf{c})$$

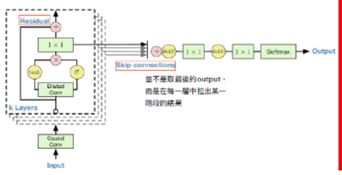
  - LSTM style with no forget and input gates required
  - only possess output gate in which information to be propagated



- Gated tanh units (GTU)
  - mimic gated mechanism

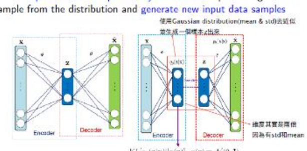
$$h(\mathbf{E}) = \tanh(\mathbf{E} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{E} * \mathbf{V} + \mathbf{c})$$

- Residual and skip connections
  - residual connection - vertical
  - skip connection - horizontal



Autoencoder vs variational autoencoder

- Autoencoder
  - learn a compressed representation of input automatically
  - compress the input and decompress it back to match the original input
- Variational autoencoder (VAE)
  - learn the parameters of a probability distribution representing the data
  - sample from the distribution and generate new input data samples



Lower bound & KL divergence

- Introduce a variational distribution $q(z)$ and adopt the Jensen's inequality for convex function $-\log(\cdot)$ to obtain

$$\log p(x|\theta) = \log \sum_z \frac{p(x,z|\theta)}{q(z)} q(z) = \log \mathbb{E}_q\left[\frac{p(x,z|\theta)}{q(z)}\right] \ge \mathbb{E}_q\left[\log \frac{p(x,z|\theta)}{q(z)}\right] \triangleq \mathcal{L}(q,\theta)$$

$$\sum_z q(z)\log p(x|\theta) \quad \mathcal{L}(q,\theta) - \sum_z q(z)\log\left\{\frac{p(z|x,\theta)}{q(z)}\right\} \triangleq \text{KL}(q\|p)$$

Evidence Decomposition

$$\log p(x|\theta) = \text{KL}(q\|p) + \mathcal{L}(q,\theta)$$

Mean field variational inference

- Variational inference aims to find a variational distribution $q(\mathbf{z}|\mathbf{x})$ that is optimally close to the original true posterior $p(\mathbf{z}|\mathbf{x})$
- $q(\mathbf{z}|\mathbf{x}) = \prod_{n=1}^{N} q(z_n|x_n)$ is assumed
- $p(\mathbf{z})$ is standard normal distribution $\mathcal{N}(0, \mathbf{I})$
- Model parameters $\Theta = \{\theta, \phi\}$ are learned by maximizing $\mathcal{L}_\Theta$

$$\log p(\mathbf{x}) \ge \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$
$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})]$$
$$\triangleq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f_\Theta(\mathbf{x},\mathbf{z})] \triangleq \mathcal{L}_\Theta$$

- Encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and decoder $p_\theta(\mathbf{x}|\mathbf{z})$ are represented by Gaussians where means and variances are calculated through multilayer

Auto-encoding variational Bayes algorithm

Initialize parameters θ, ϕ

repeat

$X^M \leftarrow$ random minibatch of M data points

$\epsilon \leftarrow$ random samples from noise distribution $p(\epsilon)$

$g \leftarrow \nabla_{\theta,\phi}\mathcal{L}_{\theta,\phi}(X^M, \epsilon)$ gradients of minibatch estimator

$\theta, \phi \leftarrow$ updating parameters using gradients g

until convergence of parameters (θ, ϕ)

return θ, ϕ

Maximum likelihood

$$\text{KL}(q\|p) = -\mathbb{E}_q[\log p(z|x,\theta)] - \mathbb{H}_q[z]$$
$$\mathcal{L}(q,\theta) = \mathbb{E}_q[\log p(x,z|\theta)] + \mathbb{H}_q[z]$$

- Maximizing $p(x|\theta)$ is equivalent to first setting $\text{KL}(q\|p) = 0$ or approximating (E-step)

$$q(z) = p(z|x, \theta^{\text{old}})$$

then maximizing the resulting lower bound (M-step)

$$\mathcal{L}(q,\theta) \triangleq Q(\theta, \theta^{\text{old}}) + \text{const}$$

where $Q(\theta, \theta^{\text{old}}) \triangleq \mathbb{E}_z[\log p(x,z|\theta)|x, \theta^{\text{old}}]$ is concave

Stochastic gradient variational Bayes

Objective: $\mathcal{L}_\Theta = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f_\Theta(\mathbf{x},\mathbf{z})]$

Gradient:

Step1: sample $\epsilon^{(l)}$ from $\mathcal{N}(0, \mathbf{I})$

Step2: $\mathbf{z}^{(l)} = \mu_z + \sigma_z \odot \epsilon^{(l)}$

Step3: $\mathcal{L}_\Theta \simeq f_\Theta(\mathbf{x}|\mathbf{z}^{(l)})$

Step4: $\nabla_\Theta \mathcal{L}_\Theta \simeq \nabla_\Theta f_\Theta(\mathbf{x}, \mathbf{z}^{(l)})$

- Reduce the variance caused by directly sampling z

Expectation-maximization algorithm

- Likelihood function for observations x in latent variable model with latent variable z

$$p(x|\theta) = \sum_z p(x, z|\theta)$$

- Expectation (E) step: calculate an auxiliary function

$$Q(\theta, \theta^{\text{old}}) = \mathbb{E}_z[\log p(x,z|\theta)|x, \theta^{\text{old}}]$$

- Maximization (M) step: find a new estimate θ^{new} via

$$\theta^{\text{new}} = \arg\max_\theta Q(\theta, \theta^{\text{old}})$$

- EM algorithm (Dempster et al., 1977) for ML can be extended for MAP

Why approximate inference?

- There are a number of latent variables in model-based methods
  - semantic topics in topic model
  - hidden units in layered network
- Latent variables are coupled in their posteriors
- Posterior distribution of multiple latent variables should be factorizable to find analytical solution to inference algorithm
- Exact inference does not exist in a complicated system
- Evolution of inference algorithms
  - maximum likelihood
  - maximum *a posteriori*
  - variational inference
  - Gibbs sampling

VAE versus CVAE

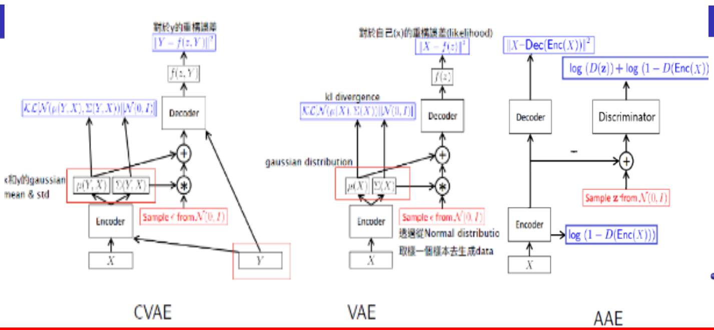
- Variational autoencoder VAE只能生成一個跟training data很像而已，無法控制生成什麼
  - generation of data is not controllable
  - labels of data are unnecessary
  - lower bound and objective function $\mathcal{L}(x,\theta,\phi)$

$$\log p_\theta(\mathrm{x}) \geq -\mathrm{KL}[q_\phi(z|\mathrm{x})||p(z)] + \mathbb{E}_{q_\phi(z|\mathrm{x})}[\log p_\theta(\mathrm{x}|z)] \triangleq \mathcal{L}(x,\theta,\phi)$$

- Conditional variational autoencoder conditional有label 透過conditional likelihood控制生成data
  - generation of data is controllable
  - labels of data are required
  - lower bound and objective function $\mathcal{L}(x,y,\theta,\phi)$

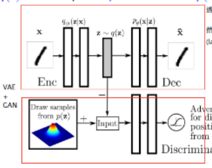
$$\log p_\theta(\mathrm{y}|\mathrm{x}) \geq -\mathrm{KL}[q_\phi(z|\mathrm{x},\mathrm{y})||p(z)] + \mathbb{E}_{q_\phi(z|\mathrm{x})}[\log p_\theta(\mathrm{y}|\mathrm{x},z)]$$

$$\triangleq \mathcal{L}(x,y,\theta,\phi)$$



Adversarial autoencoder

- Discriminator is trained to distinguish samples of latent distribution $q(z)$ from samples of prior distribution $p(z)$



- Objective function
  - $\mathcal{L}_{\theta_{dis}} = -\mathbb{E}_{z\sim p(z)}[\log(D(z))] - \mathbb{E}_{x\sim p_{enc}}[\log(1-D(\mathrm{Enc}(x)))]$
  - $\mathcal{L}_{\theta_{enc}} = \mathbb{E}_{x\sim p_{enc}}[\log(1-D(\mathrm{Enc}(x)))]$
  - $\mathcal{L}_{\theta_{enc,dec}} = \mathbb{E}_{x\sim p_{enc}}[||x-\mathrm{Dec}(\mathrm{Enc}(x))||^2]$

Algorithm

Initialize parameters $\theta_{enc}, \theta_{dis}, \theta_{dec}$

repeat

$X \leftarrow$ random minibatch of M data points

$z \leftarrow$ sample from prior $p(z)$

$g_{\theta_{enc,dec}} \leftarrow \nabla_{\theta_{enc,dec}}\mathcal{L}_{\theta_{enc,dec}}(X)$ gradients of updating encoder and decoder to reconstruct data

$g_{\theta_{dis}} \leftarrow \nabla_{\theta_{dis}}\mathcal{L}_{\theta_{dis}}(X,z)$ gradient of updating discriminator to distinguishes real data from fake data

$g_{\theta_{enc}} \leftarrow \nabla_{\theta_{enc}}\mathcal{L}_{\theta_{enc}}(X)$ gradients of updating encoder to confuse discriminator

$\theta_{enc}, \theta_{dec}, \theta_{dis} \leftarrow$ updating parameters using $g_{\theta_{enc,dec}}, g_{\theta_{dis}}, g_{\theta_{enc}}$

until convergence

VAE versus CVAE

- Variational autoencoder VAE只能生成一個跟training data很像而已，無法控制生成什麼
  - generation of data is not controllable
  - labels of data are unnecessary
  - lower bound and objective function $\mathcal{L}(x,\theta,\phi)$

$$\log p_\theta(\mathrm{x}) \geq -\mathrm{KL}[q_\phi(z|\mathrm{x})||p(z)] + \mathbb{E}_{q_\phi(z|\mathrm{x})}[\log p_\theta(\mathrm{x}|z)] \triangleq \mathcal{L}(x,\theta,\phi)$$

- Conditional variational autoencoder conditional有label 透過conditional likelihood控制生成data
  - generation of data is controllable
  - labels of data are required
  - lower bound and objective function $\mathcal{L}(x,y,\theta,\phi)$

$$\log p_\theta(\mathrm{y}|\mathrm{x}) \geq -\mathrm{KL}[q_\phi(z|\mathrm{x},\mathrm{y})||p(z)] + \mathbb{E}_{q_\phi(z|\mathrm{x})}[\log p_\theta(\mathrm{y}|\mathrm{x},z)]$$

$$\triangleq \mathcal{L}(x,y,\theta,\phi)$$

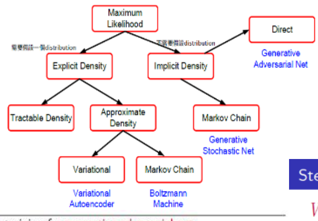
Generative model

- Conventional generative model
  - supposes training examples are obtained by $\mathrm{x} \sim p_{\mathrm{data}}(\mathrm{x})$
  - builds a model that can draw samples $\mathrm{x} \sim p_{\mathrm{model}}(\mathrm{x})$
  - assumes $p_{\mathrm{model}}(\mathrm{x}) = p_{\mathrm{data}}(\mathrm{x})$
  - maximum likelihood estimate of model parameters θ is obtained by

$$\theta_{\mathrm{ML}} = \arg\max_\theta \mathbb{E}_{\mathrm{x}\sim p_{\mathrm{data}}}[\log p_{\mathrm{model}}(\mathrm{x}|\theta)]$$

- Generative adversarial network (GAN) (Goodfellow et al., 2014)
  - general idea to learn a sampling mechanism
  - use latent code
  - asymptotically consistent
  - often regarded as producing the best samples
  - no Markov chains involved so no issue of mixing

Taxonomy of generative models



Step 1: finding optimal discriminator

$$V(D,G) = \mathbb{E}_{\mathrm{x}\sim p_{\mathrm{data}}(\mathrm{x})}[\log D(\mathrm{x})] + \mathbb{E}_{\mathrm{z}\sim p_{\mathrm{z}}(\mathrm{z})}[\log(1-D(G(\mathrm{z})))]$$

把期望值寫成積分

$$= \int_{\mathrm{x}} p_{\mathrm{data}}(\mathrm{x})\log D(\mathrm{x})d\mathrm{x} + \int_{\mathrm{z}} p_{\mathrm{z}}(\mathrm{z})\log(1-D(G(\mathrm{z})))d\mathrm{z}$$

把z用x換掉 Pg(x)代表generator的分佈x 第二項的D(x)的x是由P_g(x)產生

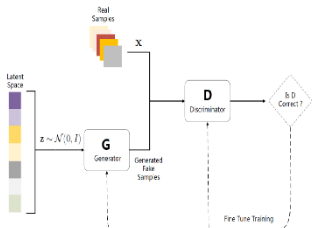
$$= \int_{\mathrm{x}} \left\{ p_{\mathrm{data}}(\mathrm{x})\log(D(\mathrm{x})) + p_g(\mathrm{x})\log(1-D(\mathrm{x})) \right\} d\mathrm{x}$$

- For the fixed G, the optimal discriminator D, maximizing $V(D,G)$, is obtained by

偏微分，令等於0 可得D(x)的最佳解

$$D^*_G(\mathrm{x}) = \frac{p_{\mathrm{data}}(\mathrm{x})}{p_{\mathrm{data}}(\mathrm{x}) + p_g(\mathrm{x})}$$

Generative adversarial nets



Minimax game

- Minimax two-player game over G and D C只看第二項式子 要對C(z)做max 可是式子是1-D(G(x)) 所以變成min
  - two models trainable simultaneously through back-propagation over value function $V(D,G)$ using neural network

$$\min_G \max_D \left\{ \mathbb{E}_{\mathrm{x}\sim p_{\mathrm{data}}(\mathrm{x})}[\log D(\mathrm{x})] + \mathbb{E}_{\mathrm{z}\sim p_{\mathrm{z}}(\mathrm{z})}[\log(1-D(G(\mathrm{z})))] \right\}$$

x從p_data取出來，是真實的data z從standard gaussian sample出來的

- Generative model G D(x)代表屬於第一類真的data的機率 透過G去生成fake data 1-D(G(z))代表屬於第二類假的的機率
  - captures the data distribution
- Discriminative model D
  - classifies a sample coming from the training data rather than G
- Train a model G via maximizing the probability of D making a mistake
- Generator G minimizes the classification accuracy of D

Algorithm 1 SGD training for generative adversarial net

for number of training iterations **do**

for k steps **do**

- sample minibatch of m noise samples $\{\mathrm{z}^{(1)},\ldots,\mathrm{z}^{(m)}\}$ from prior $p_{\mathrm{z}}(\mathrm{z})$
- sample minibatch of m examples $\{\mathrm{x}^{(1)},\ldots,\mathrm{x}^{(m)}\}$ from data generating distribution $p_{\mathrm{data}}(\mathrm{x})$
- update the discriminator by ascending its stochastic gradient

$$\nabla_{\theta_d} \frac{1}{m}\sum_{i=1}^m \left[\log D(\mathrm{x}^{(i)}) + \log(1-D(G(\mathrm{z}^{(i)})))\right]$$

end for

- sample minibatch of m noise samples $\{\mathrm{z}^{(1)},\ldots,\mathrm{z}^{(m)}\}$ from prior $p_{\mathrm{z}}(\mathrm{z})$
- update the generator by descending its stochastic gradient

$$\nabla_{\theta_g} \frac{1}{m}\sum_{i=1}^m \log(1-D(G(\mathrm{z}^{(i)})))$$

end for

Gradient-based updates with momentum can be used

Step 2: finding optimal generator

$$C(G) = \max_D V(G,D)$$

$$= \max_D \int_{\mathrm{x}} p_{\mathrm{data}}(\mathrm{x})\log(D(\mathrm{x})) + p_g(\mathrm{x})\log(1-D(\mathrm{x}))d\mathrm{x}$$

$$= \int_{\mathrm{x}} p_{\mathrm{data}}(\mathrm{x})\log(D^*_G(\mathrm{x})) + p_g(\mathrm{x})\log(1-D^*_G(\mathrm{x}))d\mathrm{x}$$

$$= \int_{\mathrm{x}} p_{\mathrm{data}}(\mathrm{x})\log\left(\frac{p_{\mathrm{data}}(\mathrm{x})}{p_{\mathrm{data}}(\mathrm{x})+p_g(\mathrm{x})}\right) + p_g(\mathrm{x})\log\left(\frac{p_g(\mathrm{x})}{p_{\mathrm{data}}(\mathrm{x})+p_g(\mathrm{x})}\right)d\mathrm{x}$$

trick 讓分子各乘以2 最後在減回來

$$= \int_{\mathrm{x}} p_{\mathrm{data}}(\mathrm{x})\log\left(\frac{2p_{\mathrm{data}}(\mathrm{x})}{p_{\mathrm{data}}(\mathrm{x})+p_g(\mathrm{x})}\right) + p_g(\mathrm{x})\log\left(\frac{2p_g(\mathrm{x})}{p_{\mathrm{data}}(\mathrm{x})+p_g(\mathrm{x})}\right)d\mathrm{x} - \log 4$$

$$= \mathrm{KL}\left(p_{\mathrm{data}}(\mathrm{x}) \middle\| \frac{p_{\mathrm{data}}(\mathrm{x})+p_g(\mathrm{x})}{2}\right) + \mathrm{KL}\left(p_g(\mathrm{x}) \middle\| \frac{p_{\mathrm{data}}(\mathrm{x})+p_g(\mathrm{x})}{2}\right) - \log 4$$

就可以換成KL divergence的公式

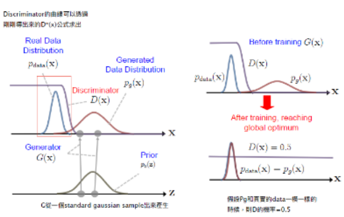
$$= 2\,\mathrm{JS}(p_{\mathrm{data}}(\mathrm{x})\|p_g(\mathrm{x})) - \log 4$$

可以換成另一個距離的公式

當D到達最佳解後 對於C來說的最佳解就是當Pdata=Pg 也就是距離=0時，此時C的最小值=-log4

- Global minimum of $C(G)$ is achieved if and only if $p_{\mathrm{data}}(\mathrm{x}) = p_g(\mathrm{x})$
- $C(G)$ achieves the value $-\log 4$

Global optimum at $p_{\mathrm{data}} = p_g$



Gradient vanish

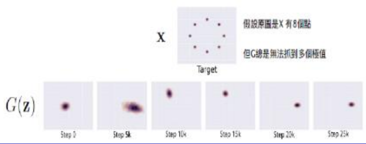
- GAN is difficult to optimize and training is unstable
- Generator and discriminator must be carefully maintained for converging 兩個model的訓練速度要相當的 才能一起converge 不然有一個太強另一個會無法訓練
  - balance the capacity between G and D
- Discriminator easily takes samples of generator as fake
  - $D(G(\mathrm{z})) = 0$ 當C和D不balance時，比方說C太強或是D都分不出來
  - $\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^m \log(1-D(G(\mathrm{z}^{(i)}))) = 0$ 此時公式第二項的值就會是0 G就會有gradient vanish問題
    - gradient vanish in G
- To deal with it, reformulate the optimization objective for G as a minimization of

$$\mathbb{E}_{\mathrm{z}\sim p_{\mathrm{z}}(\mathrm{z})}[-\log D(G(\mathrm{z}))]$$

為了避免這個問題 直接把1拿掉，把負號放外面 會比較好一點比較容易產生gradie

Mode collapse

- Lack of diversity in generated samples
  - G learns to map different inputs z to the same output point $G(\mathrm{z})$
  - G is likely to generate sample that D believes to be real rather than fake



Conditional GAN

- Original GAN objective function

$$V(D,G) = \mathbb{E}_{x\sim p_{\mathrm{data}}(\mathrm{x})}[\log D(\mathrm{x})] + \mathbb{E}_{\mathrm{z}\sim p(\mathrm{z})}[\log(1-D(G(\mathrm{z})))]$$

- Conditional GAN objective function

$$V(D,G) = \mathbb{E}_{\mathrm{x}\sim p_{\mathrm{data}}(\mathrm{x})}[\log D(\mathrm{x}|\mathrm{y})] + \mathbb{E}_{\mathrm{z}\sim p(\mathrm{z})}[\log(1-D(G(\mathrm{z}|\mathrm{y}$$

- G and D are conditioned on some extra information y
- Combine prior input noise $p(z)$ and y in joint hidden representat

Information GAN

多加一個regularization term 希望產生的可以充分反映類別的資訊

- InfoGAN objective function (Chen et al., 2016)

$$\min_G \max_D V(D,G) - \boxed{\lambda I(\mathrm{c}, G(\mathrm{z},\mathrm{c}))}$$

C就是類別，希望generator出來的樣本 和樣本C的mutual info越大越好(越相關)

- $I(\mathrm{c}; G(\mathrm{z},\mathrm{c}))$ is the mutual information between generated samples and the latent code

$$I(\mathrm{c}, G(\mathrm{z},\mathrm{c}))$$

$$= H(\mathrm{c} - H(\mathrm{c}|G(\mathrm{z},\mathrm{c})))$$

$$= \mathbb{E}_{\boxed{\mathrm{x}}\sim G(\mathrm{z},\mathrm{c})}[\mathbb{E}_{\boxed{\mathrm{c}'\sim p(\mathrm{c}|\mathrm{x})}}[\log p(\mathrm{c}'|\mathrm{x})]] + H(\mathrm{c})$$

$$= \mathbb{E}_{\mathrm{x}\sim G(\mathrm{z},\mathrm{c})}[\underbrace{D_{KL}(p(.|\mathrm{x})\|q(.|\mathrm{x}))}_{\geq 0} + \mathbb{E}_{\mathrm{c}'\sim p(\mathrm{c}|\mathrm{x})}[\log q(\mathrm{c}'|\mathrm{x})]] + H(\mathrm{c})$$

KL-divergence是非負的

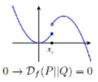
$$\geq \mathbb{E}_{\mathrm{x}\sim G(\mathrm{z},\mathrm{c})}[\mathbb{E}_{\mathrm{c}'\sim p(\mathrm{c}|\mathrm{x})}[\log q(\mathrm{c}'|\mathrm{x})]] + H(\mathrm{c})$$

f-divergence GAN

- f-GAN (Nowozin et al., 2016)
  - generalize Jensen-Shannon divergence to f-divergence
- f-divergence is defined

$$\mathcal{D}_f(P\|Q) = \int_{\mathrm{x}} q(\mathrm{x}) f\left(\frac{p(\mathrm{x})}{q(\mathrm{x})}\right)d\mathrm{x}$$

- two distributions P and Q
- absolutely continuous density functions $p(\mathrm{x})$ and $q(\mathrm{x})$
- f is convex and lower-semicontinuous f function有無限多種，只要符合條件即可



- $p(\mathrm{x}) = q(\mathrm{x})$ and $f(1) = 0 \rightarrow \mathcal{D}_f(P\|Q) = 0$

Name	$\mathcal{D}_f(P\|Q)$
Kullback-Leibler	$\int p(\mathrm{x})\log\frac{p(\mathrm{x})}{q(\mathrm{x})}d\mathrm{x}$
Reverse KL	$\int q(\mathrm{x})\log\frac{q(\mathrm{x})}{p(\mathrm{x})}d\mathrm{x}$
Pearson $\mathcal{X}^2$	$\int\frac{(q(\mathrm{x})-p(\mathrm{x}))^2}{p(\mathrm{x})}$
Squared Hellinger	$\int\left(\sqrt{p(\mathrm{x})}-\sqrt{q(\mathrm{x})}\right)^2d\mathrm{x}$
Jensen-Shannon	$\frac{1}{2}\int p(\mathrm{x})\log\frac{2p(\mathrm{x})}{p(\mathrm{x})+q(\mathrm{x})} + q(\mathrm{x})\log\frac{2q(\mathrm{x})}{p(\mathrm{x})+q(\mathrm{x})}d\mathrm{x}$
GAN	$\int p(\mathrm{x})\log\frac{2p(\mathrm{x})}{p(\mathrm{x})+q(\mathrm{x})} + q(\mathrm{x})\log\frac{2q(\mathrm{x})}{p(\mathrm{x})+q(\mathrm{x})}d\mathrm{x} - \log(4)$

- f-GAN objective function

算出真實的likelihood 算出fake的likelihood

$$V(G,D) = \mathbb{E}_{\mathrm{x}\sim p_{\mathrm{data}}(\mathrm{x})}[D(\mathrm{x})] - \mathbb{E}_{\mathrm{z}\sim p(\mathrm{z})}[f^*(D(G(\mathrm{z}))]$$

使這兩個機率越接近越好

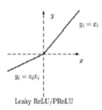
- GAN objective function

$$V(G,D) = \mathbb{E}_{\mathrm{x}\sim p_{\mathrm{data}}(\mathrm{x})}[\log D(\mathrm{x})] + \mathbb{E}_{\mathrm{z}\sim p(\mathrm{z})}[\log(1-D(G(\mathrm{z}))]$$

- f-GAN is reduced to GAN, when
  - $f^*(t) = -\log(1-e^t)$ GAN是一個f-GAN的special case
  - output activation function of D
    - f-GAN $\rightarrow -\log(1+e^{-v(\mathrm{x})})$
    - GAN $\rightarrow \frac{1}{1+e^{-v(\mathrm{x})}}$
    - $v(\mathrm{x})$ is the output before activation function
- Consider measuring with difference divergences 在上述條件下 計算4種不同方法的距離 等於時距離為0很正常 不過不等於時不同計算的方式會有不同
  - $\mathrm{W}(p_0\|p_\theta) = |\theta|$
  - $\mathrm{JS}(p_0\|p_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$
  - $\mathrm{KL}(p_\theta\|p_0) = \begin{cases} +\infty & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$
  - $\delta(p_0\|p_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$

Deep convolutional GAN

- Deep convolutional GAN (DCGAN) (Radford et al., 2015)
  - convolutional neural networks + adversarial learning
- Architecture guidelines for stable DCGAN
  - replace any pooling layers with convolutions in D
  - batchnormlization in both G and D
  - remove fully connected hidden layers
  - use LeakyReLU for all layers in D



Wasserstein GAN

Wassertein GAN (WGAN)(Arjovsky et al., 2017) is optimized by the Wasserstein distance to avoid the problem in original GAN

- gradient vanish
- mode collapse

Use the Earth-Mover (EM) distance or Wasserstein-1 distance

$$W(p_{\mathrm{data}}\|p_{\mathrm{gen}}) = \inf_{\gamma\in\Pi(p_{\mathrm{data}},p_{\mathrm{gen}})} \mathbb{E}_{(x,G(z))\sim\gamma}[\|x-G(z)\|]$$

從gramma sample出x和z 一個一個去算距離 而不是算整體樣本的距離

$\Omega(p_{\mathrm{data}}, p_{\mathrm{gen}})$ denotes the set of all joint distributions $\gamma(x, G(z))$ whose marginals are respectively p_{data} and p_{gen}

Total Variance (TV) 直接算2個distribution的距離

$$\delta(p_{\mathrm{data}}\|p_{gen}) = \sup_{A\in\Sigma}|p_{\mathrm{data}}(A) - p_{gen}(A)|$$

Kullback-Leibler (KL) 直接算2個distribution的差量

$$\mathrm{KL}(p_{\mathrm{data}}\|p_{gen}) = \int \log\left(\frac{p_{data}(x)}{p_{gen}(x)}\right)P_r(x)d\mu(x)$$

Jensen-Shannon (JS) KL不對稱，透過兩個相加使得變對稱

$$\mathrm{JS}(p_{data}\|p_{gen}) = \mathrm{KL}(p_{data}\|p_{gen}) + \mathrm{KL}(p_{gen}\|p_{data})$$

Earth-Mover (EM) or called **Wasserstein-1**

$$\mathrm{W}(p_{data}\|p_{gen}) = \sup_{\gamma\in\Pi(p_{data},p_{gen})} \mathbb{E}_{(x,y)\sim\gamma}[\,||x-y||\,]$$

Image generation

- Super resolution GAN (SRGAN) (Ledig et al., 2016)

SRGAN objective function

$$E_{\mathrm{x}^{\mathrm{HR}}\sim p_{\mathrm{data}(\mathrm{x})}}[\log D(\mathrm{x}^{\mathrm{HR}})] + E_{\mathrm{x}^{\mathrm{LR}}\sim p_g(\mathrm{x})}[\log(1-D(G(\mathrm{x}^{\mathrm{LR}})))]$$

讓低解析度的經過generator後 跟高解析度的分不出來

- input a low-resolution image x^{LR} to G instead of latent code z

$\mathcal{L}^{\mathrm{SR}}$ is weighted combination of content loss and adversarial loss

- $\mathcal{L}^{\mathrm{SR}} = \mathcal{L}_{\mathrm{content}} + \lambda\mathcal{L}_{\mathrm{adv}}$

Objective function

$$\mathcal{L}_{\mathrm{content}/i,j} = \overbrace{\frac{1}{W_{i,j}H_{i,j}}\sum_{n=1}^{W_{i,j}}\sum_{m=1}^{H_{i,j}}\left([\phi_{i,j}(\mathrm{x}^{\mathrm{HR}})]_{n,m} - [\phi_{i,j}(G(\mathrm{x}^{\mathrm{LR}}))]_{n,m}\right)^2}^{\mathcal{L}^{\mathrm{SR}}(G(\mathrm{x}^{\mathrm{LR}}),\mathrm{x}^{\mathrm{HR}})}$$

$\phi_{i,j}(\cdot)$: feature map of j^{th} convolution before i^{th} maxpooling

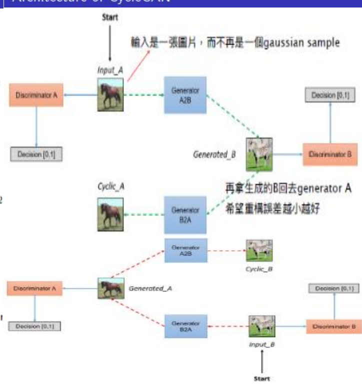
$W_{i,j}$ and $H_{i,j}$: dimensions of feature map

Adversarial loss

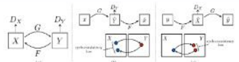
- encourages network to favour images that reside in manifold of natur images

$$\mathcal{L}_{\mathrm{adv}} = -\log D(G(\mathrm{x}^{\mathrm{LR}}))$$

Architecture of CycleGAN



CycleGAN



- Adversarial Loss 下面只有列出從x->y的(負的cross entropy越大越好) 還有從y->x的

$$\mathcal{L}_{\mathrm{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y\sim p_{\mathrm{data}}}[\log D_Y(y)] + \mathbb{E}_{x\sim p_{\mathrm{data}}(x)}[\log(1-D_Y(G(x)))]$$

就是重構誤差

- Cycle Consistency Loss

$$\mathcal{L}_{\mathrm{cyc}}(G,F) = \mathbb{E}_{x\sim p_{\mathrm{data}}(x)}[||F(G(x))-x||_1] + \mathbb{E}_{y\sim p_{\mathrm{data}}(y)}[||G(F(y))-y||_1]$$

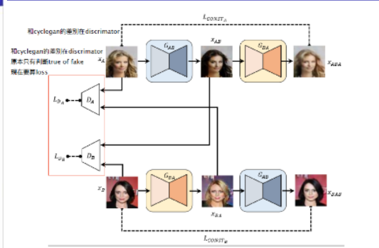
- F is another generator, which can be viewed as an inverse function of G

- Objective function

$$\mathcal{L}(G, D_X, D_Y) = \mathcal{L}_{\mathrm{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\mathrm{GAN}}(F, D_X, Y, X) + \lambda\mathcal{L}_{\mathrm{cyc}}(G,F)$$

- $G^*, F^* = \arg\min_{G,F}\max_{D_X,D_Y}\mathcal{L}(G, F, D_X, D_Y)$

Architecture of DiscoGAN



$$\mathrm{x}_{AB} = G_{AB}(\mathrm{x}_A)$$

$$\mathrm{x}_{ABA} = G_{BA}(\mathrm{x}_{AB} = G_{BA}\circ G_{AB}(\mathrm{x}_A))$$ A轉到B再轉回A

$$\mathcal{L}_{\mathrm{rec}_A} = d(G_{BA}\circ G_{AB}(\mathrm{x}_A), \mathrm{x}_A)$$

$$\mathcal{L}_{\mathrm{GAN}_B} = -\mathbb{E}_{\mathrm{x}_A\sim P_A}[\log D_B(G_{AB}(\mathrm{x}_A))]$$

DiscoGAN

- G_{AB} receives two types of losses

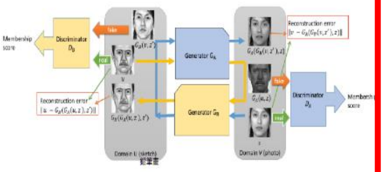
$$\mathcal{L}_{G_{AB}} = \mathcal{L}_{\mathsf{GAN_B}} + \mathcal{L}_{\mathsf{rec_A}}$$
$$\mathcal{L}_{D_B} = -\mathbb{E}_{\mathbf{x}_B \sim P_B}[\log D_B(\mathbf{x}_B)] - \mathbb{E}_{\mathbf{x}_A \sim P_A}[\log(1 - D_B(G_{AB}(\mathbf{x}_A)))]$$

  - reconstruction loss $\mathcal{L}_{\mathsf{rec_A}}$ that measures how well the original input is reconstructed
  - standard GAN loss $\mathcal{L}_{\mathsf{GAN_B}}$ that measures how realistic the generated image is in domain B
- DiscoGAN objective function

$$\mathcal{L}_G = \mathcal{L}_{G_{AB}} + \mathcal{L}_{G_{BA}} = \mathcal{L}_{\mathsf{GAN}_B} + \mathcal{L}_{\mathsf{rec_A}} + \mathcal{L}_{\mathsf{GAN_A}} + \mathcal{L}_{\mathsf{rec_B}}$$
$$\boxed{\mathcal{L}_D = \mathcal{L}_{D_A} + \mathcal{L}_{D_B}}$$

想法跟cyclegan 只是差在D的loss

Architecture of DualGAN

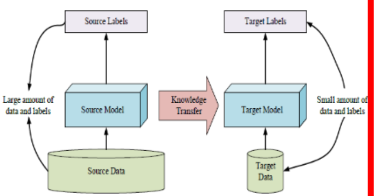


u透過sample z產生一個圖片
再透過這個圖片產生回來原本圖片
然後去算重構誤差

反之v也相同

Transfer learning

- Transferring knowledge from source domain to enhance learning capability in target domain

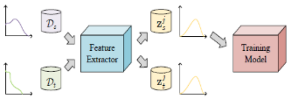


- The distribution of source domain and target domain are different, we can't directly use source domain data to train the model for target domain
- Labels of target domain data are often not observed
- We can solve these problems by applying distribution matching

Distribution matching

Domain adaptation aims to find a feature extractor where the domain of output features coming from source domain and target domain are same or similar

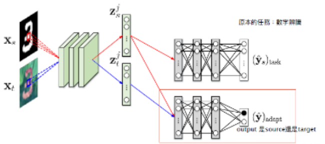
The model trained by features of source domain can work properly with the features of target domain



不同的domain再feature上要取出相似的資訊

- Maximum mean discrepancy(Gretton et al., 2012) is a kernel method that measures the discrepancy between distributions

Adversarial domain adaptation



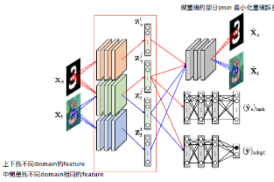
- DAN (Ganin et al., 2016) incorporates the adversarial learning into the training procedure of domain adaptation

- Task classification loss is measured by cross entropy

$$\mathcal{L}_{\mathsf{task}}(\mathsf{Enc}^j, C_{\mathsf{task}}) = -\frac{1}{N}\sum_{n=1}^{N_s}\sum_{k=1}^{K} y_{snk}\log(\hat{y}_{snk})_{\mathsf{task}}$$

Domain separation network

跟上面差不多 只是多了一些feature



- DSN (Bousmalis et al., 2016) introduces two additional private encoder to capture individual features