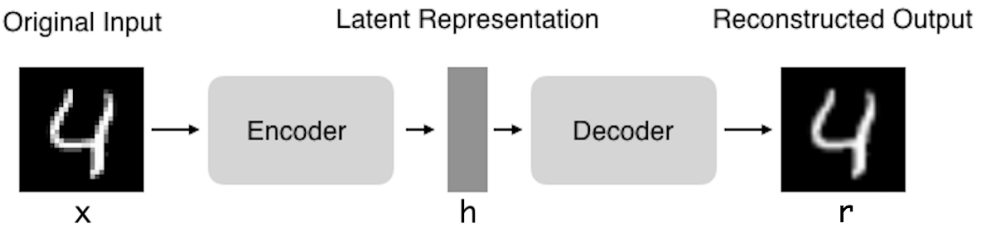
Ինքնակոդավորիչ

Հայկ Կարապետյան

Խնդիրը, երբ մուտքային տվյալը ստանալով անհրաժեշտ է ելքում ստանալ նույն մուտքային տվյալը, կոչվում է identity mapping: Ինքնակոդավորիչ (Autoencoder) ցանցերը կարողանում են լուծել յաս խնդիրը։Օրինակ՝ մուտքում ցանցին փոխանցել ենք նկար։ Ելքում ցանկանում ենք ստանալ նույն նկարը։ Այս ցանցերը կազմված են երկու մասից՝ կոդավորիչ (encoder) և դեկոդավորիչ (decoder)։ Encoder-ի նպատակն է մուտքային տվյալները տանել ուրիշ չափողականություն։ Այդ չափողականության վեկտորները կոչվում են latent-space representation։ Decoder-ի նպատակն է latent-space representation-ից վերականգնել իրական մուտքային տվյալը։ Օրինակ՝ մուտքում կարող ենք վերցնել 28×28 չափի նկար, հարթեցնել (flatten) այն, ստանալ 784 չափանի վեկտոր, ապա փոխանցել Encoder-ին և այն մուտքային վեկտորը կվերածի 20 չափողականության վեկտորի։ Decoder-ը է 20 չափողականության վեկտորը հետ է վերածելու 784 չափանի վեկտորի (Նկար 1)։



Նկար 1: Autoencoder ցանցի օրինակ

Ուսուցման ընթացքում մեր նպատակն է նվազեցնել կորուստը (հեռավորությունը) մուտքային և ելքային տվյալների միջև։ Autoencoder-ում այս կորուստը անվանում են վերակառուցման սխալ (reconstruction loss)։ Կարող ենք օգտագործել քառակուսային հեռավորության կորուստը։

$$L = (f(x) - x)^2$$

Իսկ ինչի՞ համար է անհրաժեշտ ցանց, որը ստանալով մուտքային տվյալ, output-ում կվերադարձնի նույն մուտքային տվյալը։ Ամենակարևոր հատվածը latent-space representation-ն է, որը կարողանում էր 784 չափանի վեկտորը վերածել 20 չափանի վեկտորի այնպես, որ հնարավոր լինի վերականգնել 784 չափանի վեկտորը։ Auotencoder-ները կարող են օգտագործվել խնդիրներում, ինչպիսիք են՝ 1. չափողականության նվազեցում (dimensionality reduction), 2. անոմալիաների հայտնաբերում (anomaly detection), 3. տվյալներում աղմուկի հեռացում (data denoising) և 4. ուրիշ խնդիրներում։

1. Չափողականության նվազեցումը արդեն պարզ է՝ կիրառում ենք միայն Encoder-ի հատվածը և ստանում ավելի փոքր չափողականության վեկտոր։

2. Հիմա դիտարկենք անոմալիաների հայտնաբերման խնդիրը։ Ունենք տվյալներ և դրանց մեջ կան և՛ շների նկարներ, և՛ փղերի նկարներ։ Մեր խնդիրն է հեռացնել փղերի նկարները մեր տվյալներից։ Փղերի նկարները անոմալիան են։ Համացանցից կարող ենք գտնել շատ շների նկարներ և դրանց վրա ուսուցանել Autoencoder։ Դրանից հետո Autoencoder-ը աշխատացնենք մի քանի ձեռքով ընտրված շան նկարների վրա։ Հաշվենք վերակառուցման սխալը և որոշենք threshold, որից մեծ լինելու դեպքում տյալը կհամարվի անոմալիա։ Նույն կերպ կարող ենք հաշվել ձեռքով ընտրված շան նկարների latent-space representation վեկտորների միջև հեռավորություն (euclidean) և նորից սահմանենք threshold, եթե նախապես ընտրված շան նկարի և մուտքում եկած տվյալի latent-space վեկտորների հեռավորությունը մեծ է threshold-ից, մուտքային տվյալը անոմալիա է։

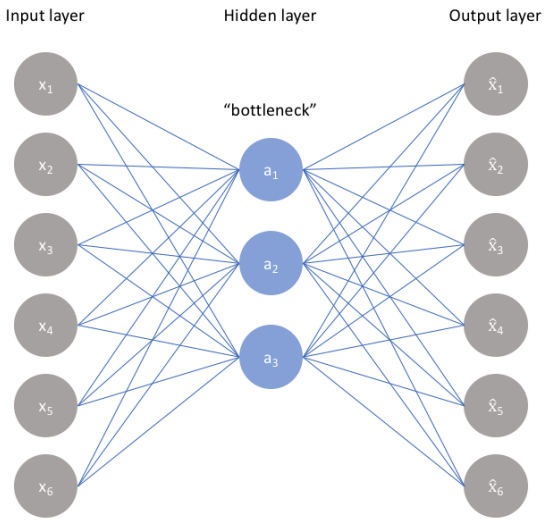
3. Տվյալներից աղմուկի հեռացման համար մեզ անհրաժեշտ են աղմկոտ տվյալներ և դրանց համապատասխան անաղմուկ տվյալներ։ Իսկ այդպիսի տվյալներ ստեղծելը հեշտ է։ Կարող ենք վերցնել մեծ քանակությամբ շների նկարներ և ամեն նկարին գումարենք գաուսյան աղմուկ (gaussian noise)։ Մուտքային տվյալը կլինի ավելացված աղմուկով նկարը,

իսկ որպես ելքային տվյալ կպահանջենք անաղմուկ նկարը: Պահանջելը իրականացվելու է կորստի ֆունկցիայի միջոցով, երբ այն սահմանենք ցանցի ելքի և իրական պիտակի համար:

4. Autoencoder-ները կարող են օգտագործվել շատ տարբեր խնդիրներում: Օրինակ՝ կարող ենք ստանալ մուտքային նկարի latent-space representation-ը և դրա վրա ավելացնելով մեկ dense շերտ կատարել classification: Կամ կարող ենք այդ նույն վեկտորների վրա աշխատացնել կլաստերավորում (clustering[1]): Նույնիսկ կարող ենք նկարի վրա աշխատացնել VGG ցանցը, ստանալ 4096 չափանի վեկտորներ, դրանց վրա ուսուցանել Autoencoder, հետո վերցնել 4096 չափանի վեկտորը և դարձնել 200 չափանի վեկտոր և նոր կատարել classification կամ clustering: Autoencoder-ի հիմնական նպատակն է մուտքային տվյալներից քաղել գլխավոր ինֆորմացիան:

Այժմ դիտարկենք Autoencoder-ների մի քանի տեսակներ:

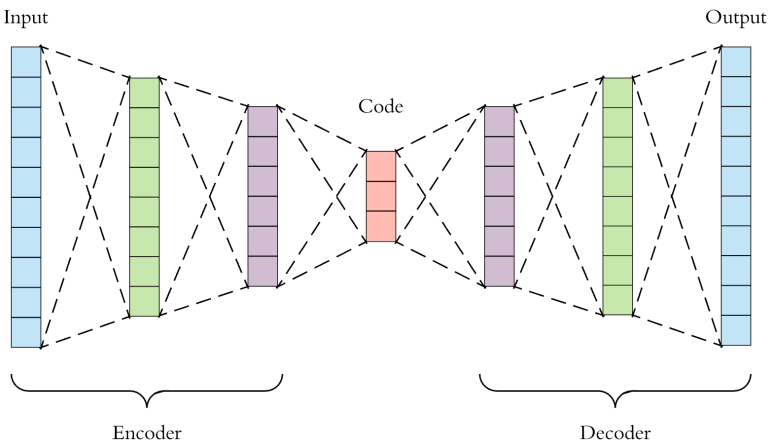
1. Vanilla Autoencoder
   Այս Autoencoder-ը ունի ամենահասարակ կառուցվածքը: Մեկ մուտքային dense շերտ, մեկ hidden dense շերտ և մեկ ելքային dense շերտ (Նկար 2):



Նկար 2: Vanilla Autoencoder

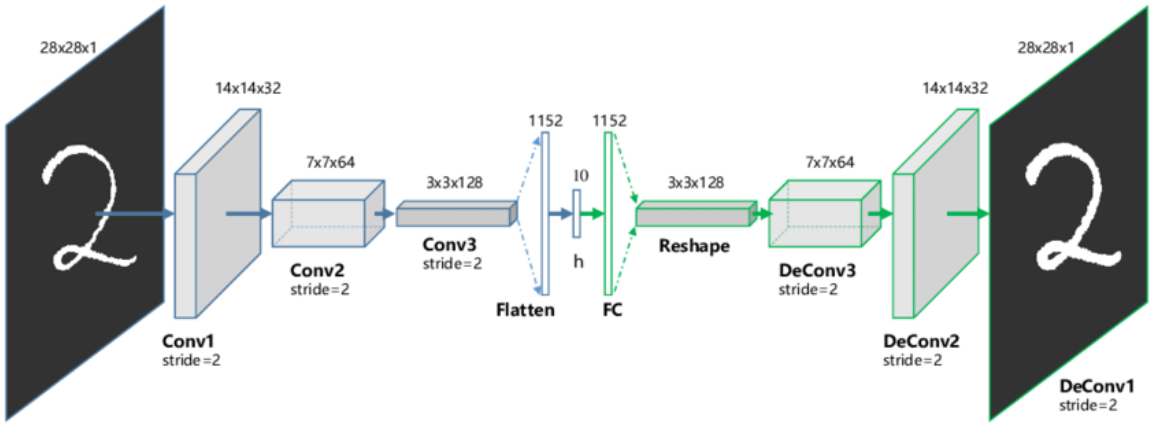
2. Multilayer/Deep Autoencoder
   Անունից արդեն կարող ենք հասկանալ, որ այս Autoencoder-ը ավելի խորն է՝ շերտերի քանակը ավելի շատ են: Եթե Vanilla Autencoder-ի դեպքում ունեինք Simple Neural Network, այս դեպքում ունենք Deep Neural Network (Նկար 3):

1. clustering - Խնդիր, երբ անհրաժեշտ է գտնել նման տվյալներ և խմբավորել իրար հետ: Այս խնդրում մենք չունենք պիտակավորված տվյալներ, թե որ տվյալը, որ խմբին է պատկանում: ԱՆհրաժեշտ է գտնել իրար ավելի նման տվյալներ և խմբավորել:



Նկար 3: Deep Autoencoder

3. Convolutional Autoencoder
   Այս Autoencoder-ը օգտագործում է convolution շերտեր՝ պահելով նկարի խորությունը (Նկար 4)։

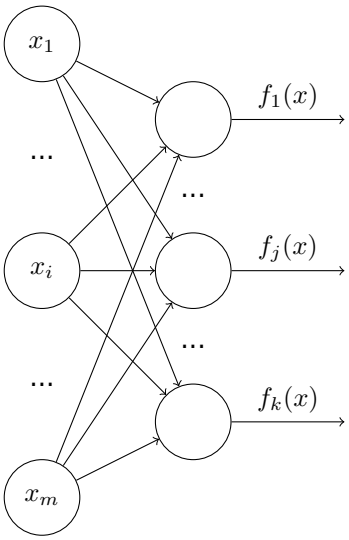


Նկար 4: Convolutional Autoencoder

   Այն պարունակում է և՛ convolution շերտեր, և՛ deconvolution շերտեր, և՛ dense շերտեր։ Convolution շերտերը օգտագործվում են նկարի չափը փոքրացնելու և խորությունը պահպանելու համար։ Deconvolution շերտեր կատարում են transposed convolution գործողությունը՝ նկարի չափերը հետ մեծացնելու համար։ Իսկ Dense շերտերը օգտագործվում են վերջնական latent-space representation-ը ստանալու համար։ Նկար 4-ում կարող ենք տեսնել, որ մուտքային $28 \times 28 \times 1$ նկարը դարձրել ենք 10 չափողականությամբ վեկտոր։ Autoencoder-ների կառուցվածքները դիտարկելիս կարող եք տեսնել, որ Encoder-ի և Decoder-ի հատվածը սիմետրիկ են։ Կիրառված են նույն քանակությամբ, նույն շերտերը։ Եվ դա պատահական չէ։ Ինչպես մեր տվյալներից ստացել ենք latent-space representation-ը, նույն կերպ անհրաժեշտ է այն հետ վերականգնել։

4. Contractive Autoencoder
   Autoencoder-ներում, մուտքային տվյալի փոքր փոփոխության դեպքում՝ latent-space representation-ը շատ է փոխվում։ Օրինակ՝ մուտքային տվյալում պատկերված է շուն։ Այդ շանը մի փոքր աջ տեղափոխելիս (shift), Encoder-ից ստացված վեկտորը շատ է փոփոխվում։ Contractive Autoencoder-ի նպատակն է այնպես ուսուցանել ցանցը, որպեսզի մուտքային տվյալի փոքր փոփոխության դեպքում՛ latent-space representation-ը նույնպես քիչ փոփոխվի։ Ունենք $y = x^2$ ֆունկցիան։ Ուզում ենք դիտարկել x-ի փոքր փոփոխության ժամանակ, ինչպես է փոփոխվում y-ը։ Դրա համար անհրաժեշտ է դիտարկել y-ի ածանցյալը

ըստ x-ի ($\frac{\partial y}{\partial x}$)։ Նույն կերպ, երբ ցանկանում ենք մուտքային տվյալի քիչ փոփոխության դեպքում, քիչ փոփոխել latent-space representation-ը, մեզ անհրաժեշտ է հաշվել դրա ածանցյալը ըստ մուտքային տվյալի։



Գծագիր 1։ Autoencoder, որի latent-space representation-ը k չափողականությամբ վեկտոր է

Հասկանանք գրաֆիկ 1-ը։ Մուտքային տվյալը m չափողականությամբ վեկտոր է, որի latent-space representation-ը k չափանի է։ Այս դեպքում $f(x) = \phi(wx + b)$, բայց ավելի խորը, այսինքն մի քանի շերտերից կազմված Encoder-ի կառուցվածքի դեպքում, $f(x)$ ֆունկցիան կփոփոխվի։ $f(x)$ ասելով հասկանում ենք մուտքային տվյալի latent-space representation-ը։ Այժմ մեզ անհրաժեշտ է մուտքային տվյալի քիչ փոփոխության դեպքում, latent-space representation-ը նույնպես քիչ փոփոխվի։ Դրա համար հաշվենք հետևյալ մասնակի ածանցյալները և գրենք մատրիցի տեսքով։

$$J_f(x) = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_j(x)}{\partial x_1} & \cdots & \frac{\partial f_k(x)}{\partial x_1} \\ \vdots & & \vdots & & \vdots \\ \frac{\partial f_1(x)}{\partial x_i} & \cdots & \frac{\partial f_j(x)}{\partial x_i} & \cdots & \frac{\partial f_k(x)}{\partial x_i} \\ \vdots & & \vdots & & \vdots \\ \frac{\partial f_1(x)}{\partial x_m} & \cdots & \frac{\partial f_j(x)}{\partial x_m} & \cdots & \frac{\partial f_k(x)}{\partial x_m} \end{bmatrix}$$

Հետևյալ մատրիցի համար սահմանենք նորմ։

$$||J_f(x)||_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{k}\left(\frac{\partial f_j(x)}{\partial x_i}\right)^2$$

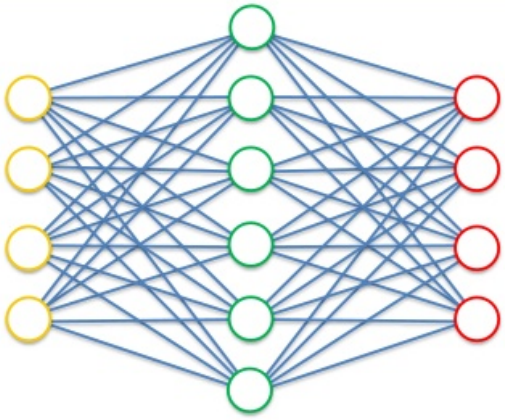
Այս նորմը կոչվում է Ֆրոբենիուս (Frobenius) նորմ։ Այն սահմանվում է մատրիցի վրա և հավասար է մատրիցի բոլոր անդամների քառակուսիների գումարին։ Մեզ անհրաժեշտ է, որպեսզի վերը նշված մատրիցի Ֆրոբենիուս նորմը փոքր լինի, այսինքն ածանցյալները փոքր լինեն, այսինքն մուտքային տվյալի քիչ փոփոխության դեպքում, latent-space representation-ը նույնպիս քիչ փոխվի։ Դրա համար գումարենք կորստի ֆունկցիան։

$$L = \sum_{x\in D}\left(L(x, g(f(x)) + \lambda||J_f(x)||_F^2)\right), \quad 0 < \lambda < 1$$

D-ն մեր տվյալների բազմությունն է, իսկ g(f(x))-ը՝ Decoder-ի output-ն է, երբ latent-space representation-ը փոխանցում ենք որպես մուտք։ Output-ի և սկզբնական x-ի միջև կորուստը

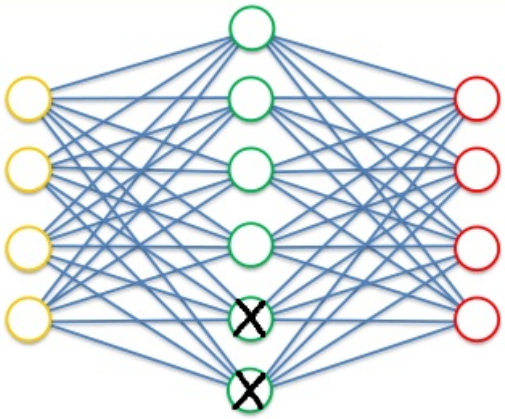
պետք է փոքր լինի: Ինչպես նաև մուտքը քիչ փոփոխելիս, latent-space representation-ը նույնպես պետք է քիչ փոփոխվի:

5. Sparse Autoencoder
   Ի՞նչ կլինի եթե latent-space representation-ը վերցնենք ավելի մեծ, քան մուտքի չափը (Նկար 5):



Նկար 5: Sparse Autoencoder

Latent-space representation-ի չափը վերցնում ենք ավելի մեծ, քան մուտքային տվյալներինը, որպեսզի կարողանանք հնարավարոնիս շատ ինֆորմացիա քաղենք մուտքային տվյալներից: Բայց եթե ոչ մի մեթոդ չկիրառենք և ուսուցանենք մեզ հայնի ձևով, այն կսովորի ուղղակի կլոնավորել մուտքային տվյալները դեպի ելքային տվյալներ: Այսինքն ոչինչ չի սովորի: Մուտքային տվյալները նշանակենք $x_1, \dots x_k$, իսկ latent-space-ի շերտի կշիռները՝ $W_1, \dots W_m \ m > k$: Մենք կատարում ենք $W_i x$ գործողությունը: Վերցնենք W_1-ի կշիռները հետևյալ տեսքով $[1, 0, \dots 0]$: W_2-ի դեպքում երկրորդ տեղում գրված կլինի 1, իսկ մնացածը 0: Նույն կերպ W_k-ի k-րդ տեղում գրված կլինի 1, իսկ մնացածը 0: W_{k+1}-ից մինչև W_m-ը մեզ չի հետաքրքրում, քանի որ Decoder-ի կշիռները հաշվի են առնելու 1-ից k նեյրոնների output-ները: Այսպիսով, եթե latent-space representation-ի չափողականությունը ավելի մեծ է, քան մուտքային տվյալներինը՝ ցանցը ոչիննչ չի սովորի ուղղակի կըկրկնօրինակի մուտքային տվյալները դեպի output (Նկար 6):



Նկար 6: Latent-space representation-ը ոչինչ չի սովորում

Այդ պատճառով կիրառում ենք ռեգուլարիզացիա: Նպատակն է latent-space representation-ի բոլոր արժեքները պահել ինչ որ արժեքից փոքր և այդ դեպքում այն ուղղակի չի կարողանա մուտքային արժեքները բազմապատկել 1-ով և փոխանցել, որպես output: Ինչպես Contractive Autoencoder-ում, այստեղ նույնպես վերցնենք f_j-ով նշանակենք

Encoder-ի output-ները։

$$p_j = \frac{1}{n} \sum_{x \in D} f_j(x)$$

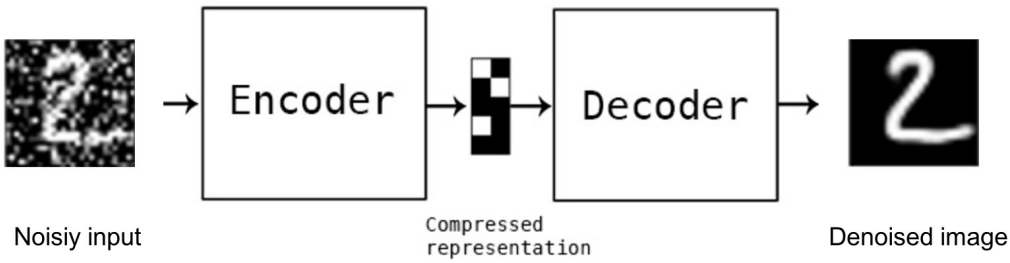
p_j ցույց կտա միջինում j-րդ նեյրոնը ինչ արժեք է ընդունում։ Վերցնենք $p = 0.05$։

$$L_{total} = \sum_{x \in D} L(x, g(f(x))) + \lambda \sum_j KL(p||p_j)$$

$$KL = (p||p_j) = -p \log \frac{p_j}{p} - (1-p) \log \frac{1-p_j}{1-p}$$

KL divergence-ը չափանիշ է, որը ցույց է տալիս երկու արժեքների կամ երկու բաշխումների նմանությունը (հեռավորությունը)։ Ինչքան փոքր է KL-ը, այնքան ավելի նման (մոտ) են մուտքային արժեքները։ Եվ կորստի ֆունկցիայի մեջ այն ներառելիս ցանկանում ենք, որ Encoder-ի output-ները միջինում մոտիկ լինեն 0.05-ի։ Այս ռեգուլարիզացիոն մեթոդը կիրառելով, կարող ենք latent-space representation-ի չափողականությունը վերցնել ավելի մեծ, քան մուտքային տվյալներինն է։

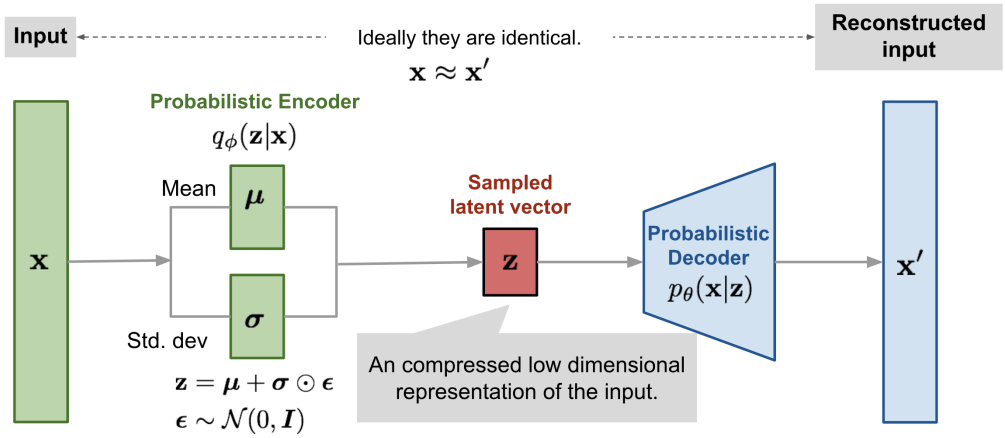
6. Denoising Autoencoder
   Autoencoder-ները կարող են լուծել նաև նկարներից աղմուկի հեռացման խնդիրը։ Դրա համար անհրաժեշտ է պատրաստել ուսուցման տվյալներ։ Մուտքային տվյալը լինելու է աղմուկով նկարը, իսկ ելքայինը՝ նույն նկարը առանց աղմուկի։ Այդպես պիտակավորված տվյալներ անհրաժեշտ չէ գտնել։ Կարող ենք ուղղակի վերցնել շատ նկարներ և դրանց ավելացնելով գաուսյան աղմուկ կամ պատահական աղմուկ՝ ստանալ անհրաժեշտ ուսուցման տվյալները (Նկար 7)։



Նկար 7: Denoising Autoencoder

Այս դեպքում latent-space representation-ը կարող է լինել ավելի մեծ քան մուտքային տվյալներինը, քանի որ ուղղակի կրկնօրինակելով ցանցը չի կարողանա ստանալ անհրժեշտ output-ը։

7. Variational Autoencoder
   Հնարավո՞ր է արդյոք Autoencoder-ների միջոցով գեներացնել նոր ինֆորմացիա։ Այս հարցին պատասխանում են Variational Autoencoder-ները։ Նախորդ Autoencoder-ներում ամեն մուտքային տվյալին համապատասխանեցնում էինք վեկտոր։ Իսկ ինչ կլինի, եթե ամեն մուտքային տվյալի համապատասխանեցնենք բաշխում։ Encoder-ի վերադարձած վեկտորը բաժանենք երկու հավասար մասերի։ Առաջին հատվախը կներկայացնի մեր բաշխման միջինը, իսկ երկրորդ հատվածը ստանդարտ բաշխումը։ Ունենալով այդ երկու արժեքները կարող ենք ստանալ գաուսյան բաշխում տրված միջինով և ստանդարտ բաշխումով ($\mu + \sigma \cdot N(0, I)$)։ Դրանից հետո ստացված բաշխումից պատահական կերպով կընտրենք մի վեկտոր և փոխանցելով Decoder-ին՝ կպահանջենք վերականգնված (reconstructed) մուտքը (Նկար 8)։



Նկար 8: Variational Autoencoder

Իսկ ինչպե՞ս ենք գեներացնելու նոր նկարներ: Մեզ ընդամենը անհրաժեշտ է վերցնել գուսյան բաշխում, ընտրել պատահական վեկտոր այդ բաշխումից, փոխանցել Decoder-ին և վերջ: Decoder-ը մեզ համար կգեներացնի նոր նկար:

Ամփոփենք: Autoencoder-ները հիմնականում նախատեսված են չափողականության նվազեցման համար, բայց դրանց միջոցով նաև կարող ենք գեներացնել նոր տվյալներ և կատարել տվյալներից աղմուկի հեռացում: Շատ տվյալների վրա մոդելով ուսուցանելով, կարող ենք ստանալ ավելի որակով սեղմման (compressing) միջոց, քան jpeg-ն է: Բայց այն կաշխատի ավելի դանդաղ, քանի որ պետք է կատարի բազմաթիվ բազմապատկումներ: Autoencoder-ների Encoder և Decoder հատվածները կարող ենք վերցնել Transformer-ի Encoder, Decoder architecture-ները: