



دانشگاه صنعتی شریف

دانشکده مهندسی مکانیک

## شبکه‌های عصبی پیچشی

گزارش تمرین ۲- تشریحی

رباتیک اجتماعی و شناختی

مریم کریمی جعفری، ۹۹۱۰۶۶۱۷

استاد

دکتر طاهری

بهار ۱۴۰۳

## فهرست

سوال ۱:	۱
الف)	۱
MSE (MSD)	۱
RMSE (RMSD)	۱
NRMSE (NRMSD)	۱
RRMSE	۲
ب)	۳
ReLU (Rectified Linear Unit)	۳
PReLU (Parametric Rectified Linear Unit)	۳
RReLU (Randomized Leaky Rectified Linear Unit)	۳
LeakyReLU	۳
ELU (Exponential Linear Unit)	۳
SELU (Scaled Exponential Linear Unit)	۴
Gelu (Gaussian Error Linear Unit)	۴
پ)	۶
DropOut2D	۶
DropOut3D	۶
ت)	۷
ث)	۹
سوال ۲:	۱۲
الف)	۱۲
ب)	۱۳
پ)	۱۴
سوال ۳:	۱۴
سوال ۴:	۱۶

## سوال ۱:

(الف)

### 📊 MSE (MSD)

Mean Squared Error:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

### 📊 RMSE (RMSD)

Root Mean Squared Error:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

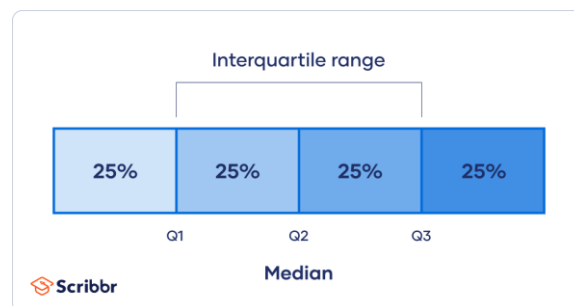
### 📊 NRMSE (NRMSD)

Normalized Root Mean Squared Error:

در این روش RMSE نرمالایز می‌شود. یعنی آن را بر یک مقداری تقسیم می‌کنند. این مقدار می‌تواند مقادیر متفاوتی داشته باشد:

- میانگین
- ماکسیمم مقدار از میان مقادیر واقعی
- تفاضل ماکسیمم و مینیمم مقادیر از میان مقادیر واقعی
- انحراف معیار standard deviation
- دامنه بین چارکی interquartile range:

$$Q_3 - Q_1$$



از این مقیاس زمانی استفاده می‌کنیم که بخواهیم مدل‌مان را بر دیتاستی که رنج وسیعی از scale‌های مختلف دارد، تست کنیم. مثل پیش‌بینی کردن قیمت کالاهای مختلف که هر کدام در یک رنج قیمتی هستند.

### RRMSE

Relative Root Mean Squared Error:

$$\text{RRMSE} = \sqrt{\frac{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (\hat{y}_i)^2}}$$

در واقع در این مقیاس MSE را نسبت به میانگین داده‌های پیش‌بینی شده محاسبه می‌کنیم. در واقع بیان می‌کنیم که مدل ما نسبت به میانگین چقدر دقیق است. می‌توان خروجی را به صورت درصد بیان کرد که اگر:

- کمتر از ۱۰٪ باشد: مدل عالی عمل می‌کند.
- بین ۱۰ و ۲۰٪ باشد: مدل عملکرد خوبی دارد.
- بین ۲۰ تا ۳۰٪ باشد: مدل عملکرد متوسطی دارد.
- بیش از ۳۰٪ باشد: مدل عملکرد ضعیفی دارد.

بگذارید یک مثال برای این مورد بزنم. برای مثال اگر شما بخواهید مدلی برای پیش‌بینی فروش ماهانه یک فروشگاه داشته باشید، با فرض اینکه این فروشگاه ماهانه به طور متوسط ده‌هزار دلار فروش داشته باشد، اگر RRMSE آن ۱۰٪ باشد، یعنی هزار دلار خطا وجود دارد. اگر برای دو محصول مختلف بخواهیم بررسی کنیم، فرض کنیم که هر دو محصول RRMSE یکسانی دارند اما میانگین میزان فروش آن‌ها متفاوت است. این به این معنی است که مدل برای پیش‌بینی فروش یکی از آن‌ها خطای بیشتری دارد.

منابع:

- Padhma, M (2023). A Comprehensive Introduction to Evaluating Regression Models. <https://www.analyticsvidhya.com/blog/2021/10/evaluation-metric-for-regression-models/#:~:text=Relative%20Root%20Mean%20Square%20Error,to%20compare%20different%20measurement%20techniques>
- WICKIPEDIA (2024). Root-mean-square deviation. [https://en.wikipedia.org/wiki/Root-mean-square\\_deviation](https://en.wikipedia.org/wiki/Root-mean-square_deviation)

(ب)

### ReLU (Rectified Linear Unit)

$$f(x) = \max(0, x)$$

*ReLU* به طور کلی اعداد منفی را صفر و اعداد مثبت را همان خودشان، نگه می‌دارد. این تابع مشکل *vanishing gradient* را تا حدودی نسبت به توابع فعالسازی دیگر، رفع کرده اما می‌تواند باعث *dying ReLU* شود که مقادیر نرون‌ها را در صفر نگه می‌دارد. پس توابع دیگری از این خانواده آمدند که در ادامه معرفی می‌شوند:

### PReLU (Parametric Rectified Linear Unit)

$$f(x) = \max(\alpha x, x)$$

تفاوت این تابع با *ReLU* این است که در قسمت منفی دیگر صفر مطلق را بیرون نمی‌دهد بلکه یک خط با شیب  $\alpha$  خواهیم داشت که  $\alpha$  یک پارامتری است که در طول *training* یافت می‌شود، به طوریکه بهترین دقت را داشته باشیم.

### RReLU (Randomized Leaky Rectified Linear Unit)

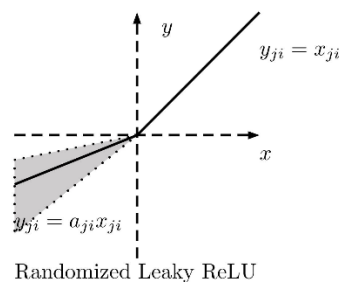
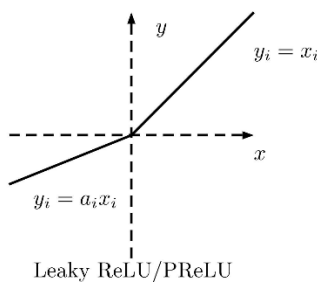
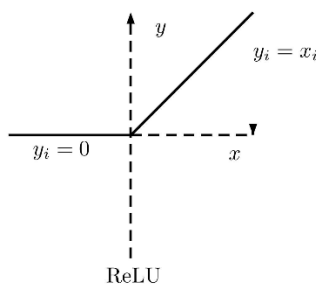
$$f(x) = \max(\alpha x, x)$$

در این تابع  $\alpha$  یک عدد تصادفی است که از یک توزیع یکنواخت یا همان یک بازه از پیش تعیین شده انتخاب می‌شود، طوری که کمترین *loss* را داشته باشیم.

### LeakyReLU

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{otherwise} \end{cases}$$

در این تابع  $\alpha = 0.01$  است.



### ELU (Exponential Linear Unit)

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{otherwise} \end{cases}$$

از آنجایی که توابع قبل در صفر شکسته می‌شدند، توابع خانواده elu به وجود آمدند که برای ورودی‌های منفی با شیب نمایی کمی، کاهش می‌یابند.  $\alpha$  معمولاً یک است.

### SELU (Scaled Exponential Linear Unit)

$$f(x) = \lambda \begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{otherwise} \end{cases}$$

از آنجایی که  $\lambda$  و  $\alpha$  هایپرپارامترهایی هستند که انتخاب می‌شوند، این تابع قابلیت scale کردن خروجی خود را دارد و می‌تواند حالتی self\_normalized داشته باشد.

### Gelu (Gaussian Error Linear Unit)

این تابع به طور کلی به صورت زیر است:

$$f(x) = x * \Phi(x_{ij})$$

$$\Phi(x_{ij}) = \text{Cumulative Distribution function}$$

$$\Phi(x_{ij}) = P(X_{ij} \leq x_{ij}) = \int_{-\infty}^{x_{ij}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2} dt$$

که  $\Phi(x_{ij})$  همان مساحت زیر نمودار توزیع گوسی با میانگین صفر و انحراف معیار واحد است. میزان آن را همچنین می‌توان برحسب  $erf$  یا همان  $Gauss Error Function$  نیز بیان کرد.

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$










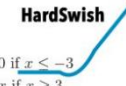
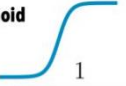
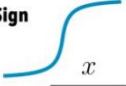

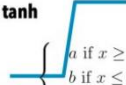
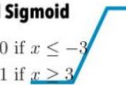
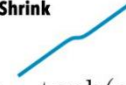
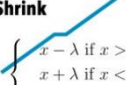
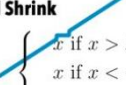
$$f(x) = \frac{1}{2} x \left( 1 + erf\left(\frac{x}{\sqrt{2}}\right) \right)$$

این تابع با ایده ادغام ReLU و عمل Dropout ایجاد شد. همچنین GeLU را می‌توان برحسب tanh و یا sigmoid نیز نوشت:













$$f(x) = 0.5x(1 + \tanh[\sqrt{\frac{2}{\pi}}(x + 0.044715x^3)])$$

$$f(x) = x\sigma(1.702x)$$

## Neural Network Activation Functions: a small subset!

<b>ReLU</b>  $\max(0, x)$	<b>GELU</b>  $\frac{x}{\sqrt{2\pi}} \left( 1 + \tanh \left( \sqrt{\frac{2}{\pi}} (x + ax^3) \right) \right)$	<b>PReLU</b>  $\max(0, x)$
<b>ELU</b>  $\begin{cases} x & \text{if } x > 0 \\ \alpha(x \exp x - 1) & \text{if } x < 0 \end{cases}$	<b>Swish</b>  $\frac{x}{1 + \exp -x}$	<b>SELU</b>  $\alpha(\max(0, x) + \min(0, \beta(\exp x - 1)))$
<b>SoftPlus</b>  $\frac{1}{\beta} \log(1 + \exp(\beta x))$	<b>Mish</b>  $x \tanh \left( \frac{1}{\beta} \log(1 + \exp(\beta x)) \right)$	<b>RReLU</b>  $\begin{cases} x & \text{if } x \geq 0 \\ ax & \text{if } x < 0 \text{ with } a \sim \mathcal{R}(l, u) \end{cases}$
<b>HardSwish</b>  $\begin{cases} 0 & \text{if } x \leq -3 \\ x & \text{if } -3 < x < 3 \\ (x+3)/6 & \text{otherwise} \end{cases}$	<b>Sigmoid</b>  $\frac{1}{1 + \exp(-x)}$	<b>SoftSign</b>  $\frac{x}{1 +  x }$
<b>Tanh</b>  $\tanh(x)$	<b>Hard tanh</b>  $\begin{cases} a & \text{if } x \geq a \\ b & \text{if } x \leq b \\ x & \text{otherwise} \end{cases}$	<b>Hard Sigmoid</b>  $\begin{cases} 0 & \text{if } x \leq -3 \\ 1 & \text{if } -3 < x < 3 \\ x/6 + 1/2 & \text{otherwise} \end{cases}$
<b>Tanh Shrink</b>  $x - \tanh(x)$	<b>Soft Shrink</b>  $\begin{cases} x - \lambda & \text{if } x > \lambda \\ x + \lambda & \text{if } x < -\lambda \\ 0 & \text{otherwise} \end{cases}$	<b>Hard Shrink</b>  $\begin{cases} x & \text{if } x > \lambda \\ 0 & \text{if } -\lambda < x < \lambda \\ x & \text{if } x < -\lambda \end{cases}$

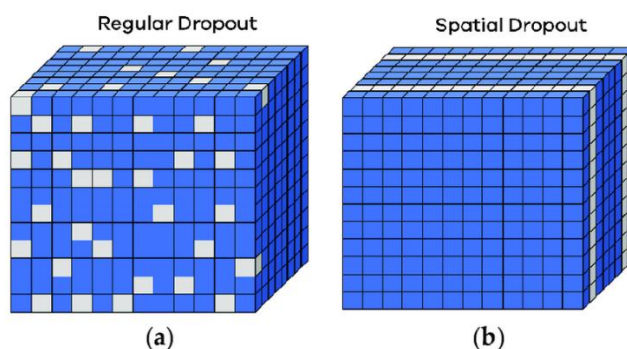
## Dance Moves of Deep Learning Activation Functions

<b>Sigmoid</b>  $y = \frac{1}{1 + e^{-x}}$	<b>Tanh</b>  $y = \tanh(x)$	<b>Step Function</b>  $y = \begin{cases} 0, & x < n \\ 1, & x \geq n \end{cases}$	<b>Softplus</b>  $y = \ln(1 + e^x)$
source: sefiks			
<b>ReLU</b>  $y = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$	<b>Softsign</b>  $y = \frac{x}{(1 +  x )}$	<b>ELU</b>  $y = \begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$	<b>Log of Sigmoid</b>  $y = \ln \left( \frac{1}{1 + e^{-x}} \right)$
<b>Swish</b>  $y = \frac{x}{1 + e^{-x}}$	<b>Sinc</b>  $y = \frac{\sin(x)}{x}$	<b>Leaky ReLU</b>  $y = \max(0, x)$	<b>Mish</b>  $y = x \cdot \tanh(\text{softplus}(x))$

(پ)

### DropOut2D

این نوع dropout در لایه‌های کانولوشنی دوبعدی و بیشتر در **image processing** استفاده می‌شود. dropout در CNNها بر روی feature map و یا همان activation map اعمال می‌شود. عادی به طور کلی از هر فیچرکمپ تعدادی خانه به طور رندوم انتخاب و صفر می‌کند. اما DropOut2D به طور رندوم تعدادی فیچرکمپ انتخاب و همه خانه‌های آنها را صفر می‌کند. از آنجایی که لایه‌های کانولوشنی می‌توانند تاثیر موقعیت مکانی در تصاویر را ببینند، بهتر است دراپ‌اوت آنها نیز همین‌طور باشد. از طرفی اگر بخواهیم فقط یک خانه را خاموش کنیم، همچنان خانه‌های اطراف، اگر رابطه مکانی با خانه مدنظر داشته باشند، تاثیر خود را می‌گذارند و احتمال overfit را افزایش می‌دهند.



### DropOut3D

این نوع dropout برای داده‌هایی نظیر ویدیوها یا تصاویر پزشکی (volumetric data) کاربرد دارد. برای مثال یک سمپل با ابعاد  $32 \times 32 \times 32 \times 3$  (width, height, depth, and channels) در نظر بگیرید. Dropout عادی به طور رندوم تعدادی voxel (volume element) را صفر می‌کند که می‌تواند از هر یک از کانال‌ها یا عمق‌ها باشند. اما DropOut3D تمام داده‌های یک کانال را صفر می‌کند. یعنی در مثال بالا که سه کانال داریم، با توجه به نرخ دراپ‌اوت، تعدادی از آنها صفر می‌شوند. به طور کلی این دو دراپ‌اوت، چنل‌ها را خاموش می‌کنند و صفر می‌گذارند.

منابع:

- Tompson, Jonathan, et al. "Efficient object localization using convolutional networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
- Cerliani, Marco (2021). Correct usage of keras SpatialDropout2D inside TimeDistributed layer - CNN LSTM network.



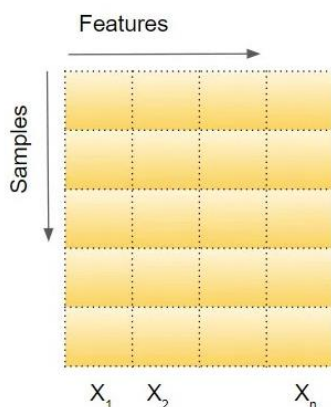
<https://stackoverflow.com/questions/66542889/correct-usage-of-keras-spatialdropout2d-inside-timedistributed-layer-cnn-lstm>

- Hadinata, P. N., Simanta, D., Eddy, L., & Nagai, K. (2023). Multiclass Segmentation of Concrete Surface Damages Using U-Net and DeepLabV3+. Applied Sciences, 13(4), 2398.

(ت)

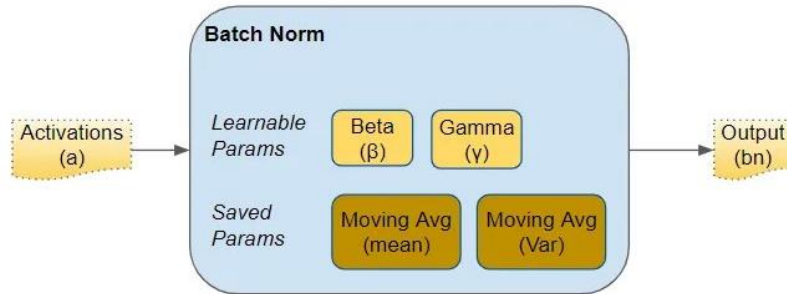
## Batch Normalization

معمولا متداول است ورودی‌های شبکه عصبی نرمالایز شده باشند؛ یعنی برای مثال همگی بین صفر و یک باشند. راه‌های متفاوتی برای نرمالایز کردن وجود دارد. یکی از آن‌ها این است که میانگین داده‌ها صفر و واریانس آن‌ها یک شود. برای این کار ابتدا میانگین و واریانس را محاسبه کرده و سپس از فرمول زیر استفاده می‌کنیم، یعنی میانگین و واریانس هر فیچر را محاسبه و داده‌ها را نرمالایز می‌کنیم:

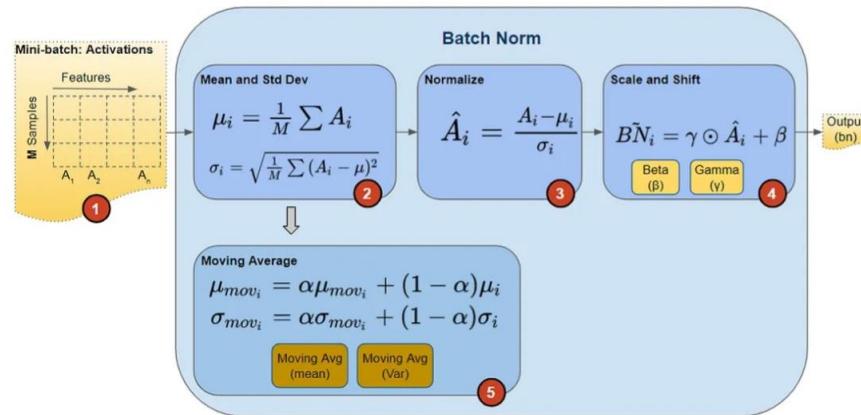


$$X_i = \frac{X_i - Mean_i}{StdDev_i}$$

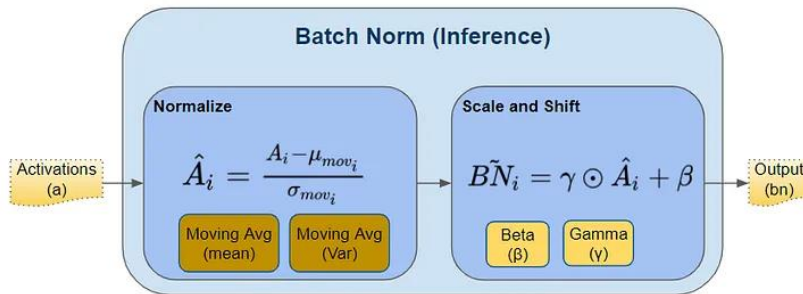
پس تا اینجا ورودی نرون‌های لایه اول یا همان فیچرها را نرمالایز کردیم. اما باید توجه کنیم که بهتر است ورودی نرون‌های لایه‌های بعدی را که همان خروجی نرون‌های لایه قبلی است، نیز نرمالایز کنیم. این کار همان batch normalization است. پس batch norm layer بین دو لایه مخفی قرار می‌گیرند تا خروجی نرون‌های لایه را قبل از ورود به نرون‌های لایه بعد نرمالایز کند. لایه‌های batch norm هم پارامترها و هایپرپارامترهایی دارد. هر batch norm layer به شکل زیر است که در آن  $\beta$  و  $\gamma$  در اثر training یافت می‌شوند:



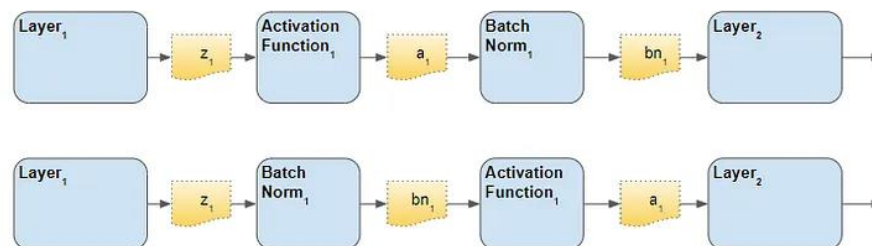
لایه‌های بچ نرم بین لایه‌های مخفی مختلف لزوماً یکی نیستند و می‌توانند پارامترهای مختلفی داشته باشند. هر mini-batch از داده‌ها در هر مرحله از training از این لایه می‌گذرد. بیایید نگاهی دقیق‌تر به فرایند محاسبات feedforward یک mini-batch از داده‌ها که از batch norm می‌گذرد، داشته باشیم:



در ابتدا برای هر mini-batch میانگین و واریانس هر فیچر (که در اینجا همان خروجی های activation function هستند) محاسبه می‌شود و سپس به همان روشی که پیش‌تر بیان شد، داده‌ها نرمالایز می‌شوند. در ادامه داده‌ها باید هم شیفت پیدا کنند و هم اسکیل شوند. با این کار میانگین و واریانس آن‌ها از صفر و یک به مقادیر دیگری تغییر پیدا می‌کنند. زیرا از آنجایی که این داده‌ها ورودی‌های نرون‌های مخفی هستند، شاید بهتر باشد نسبت به فیچرهای اولیه، میانگین و واریانس دیگری داشته باشند. اینجاست که  $\gamma$  و  $\beta$  وارد می‌شوند تا این میانگین و واریانس را در training به بهترین شکل دریاورند. در کنار این‌ها در طول فرایند training مقادیر Exponential Moving Average (EMA) برای میانگین و واریانس نیز ذخیره می‌شوند تا در مرحله predicting مورد استفاده قرار گیرند. EMA درواقع روشی است برای محاسبه میانگینی از میانگین و واریانس در طول زمان آموزش به طوری که بعد از هر mini-batch مقدار آن آپدیت می‌شود. طبیعتاً مقادیر اولیه آن همان میانگین و واریانس mini-batch است. مقدار  $\alpha$  کنترل می‌کند که مقادیر قبلی تا چه میزان تاثیر داشته باشند. پس در ادامه در مرحله تست که ما فقط یک داده و نه یک بچ از داده را داریم، از این مقادیر برای نرمالایز کردن استفاده می‌کنیم، چون دیگر فقط یک داده است و نمی‌توان میانگین یا واریانس حساب کرد.



نکته‌ای که باید به آن توجه شود این است که گاهی لایه بیج نرم قبل از activation function نیز اعمال می‌شود.



منابع:

- Doshi, Ketan (2021). Batch Norm Explained Visually — How it works, and why neural networks need it. <https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739>

(ث)

## Autoencoders

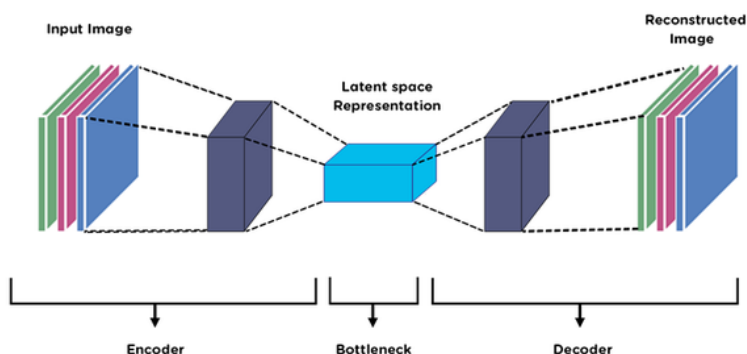
### Vanilla Autoencoders

ساده‌ترین نوع autoencoderها، وانیلی ساده‌ها هستند که از یک لایه در انکودر و دیکودرشان تشکیل شده‌اند. یعنی ابتدا یک لایه ورودی، سپس یک لایه انکودر که تعداد نرون‌های آن معمولاً از لایه ورودی کمتر است تا تصاویر را به ابعاد کوچک‌تر و فشرده‌تری برسد. در نهایت یک لایه دیکودر قرار دارد که ورودی را بازتولید می‌کند. لایه‌ها fully-connected(dense) هستند. مقادیر لایه‌های ورودی و خروجی در حالت ایده‌آل یکسان هستند.

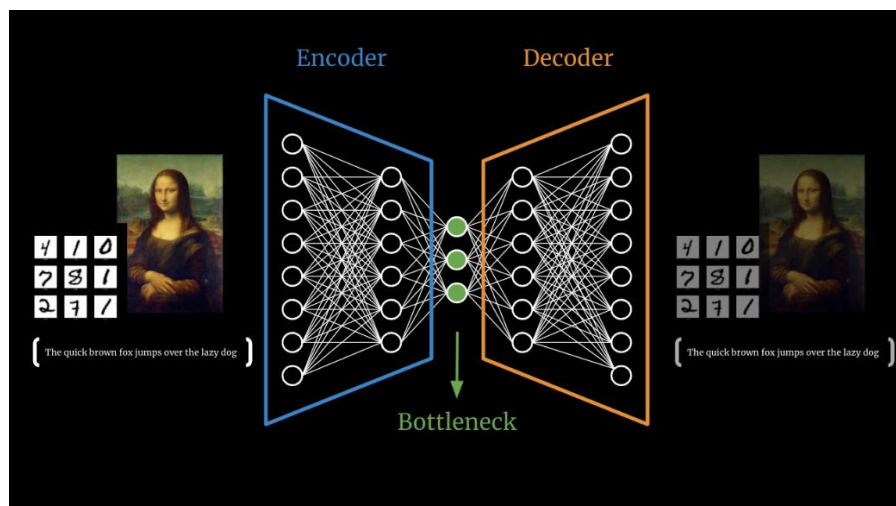
### Convolutional Autoencoders (CAE)

فکر می‌کنم ایده autoencoderها در CNN با این فکر مطرح شد که از آنجایی که می‌توان با کمک شبکه‌های کانولوشنی، تصاویر را که آرایه‌ای دو بعدی هستند، در نهایت به آرایه‌ای یک بعدی تبدیل کرد، چرا برعکس

ممکن نباشد؟! یکی از وظایف شبکه‌های کانولوشنی، feature extraction است. اما وظیفه autoencoder این است که در واقع از فیچرها تصویر تولید کنند. Convolutional Autoencoder ها به جای لایه‌های dense از لایه‌هایی با ساختار کانولوشنی تشکیل شده‌اند که در پردازش تصاویر بسیار کاربرد دارند.



autoencoder ها ساختاری از شبکه‌های عصبی هستند که شامل یک بخش encoder، یک بخش bottleneck و یک بخش decoder هستند.



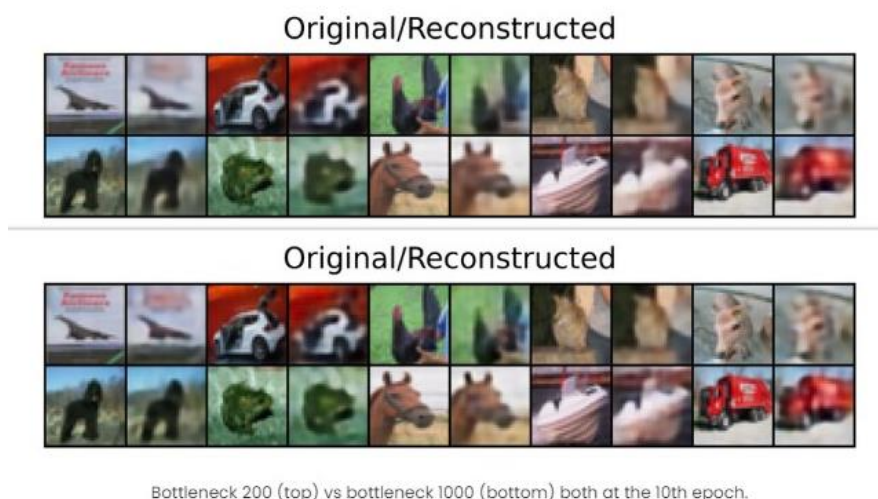
## Encoder

این بخش در واقع همان قسمت feature extraction است.

## Bottleneck

به این قسمت code نیز می‌گویند. این قسمت خروجی با ابعاد کمتر را همچنان کوچک‌تر می‌کند تا فیچرها در یک برای مثال ماتریس با ابعاد کوچک‌تر ذخیره شوند. خروجی این بخش vector representation ای است که عصاره‌ای از تصویر و فیچرها محسوب می‌شود. هر چقدر که سایز این بردار کمتر باشد، decoder در ادامه به

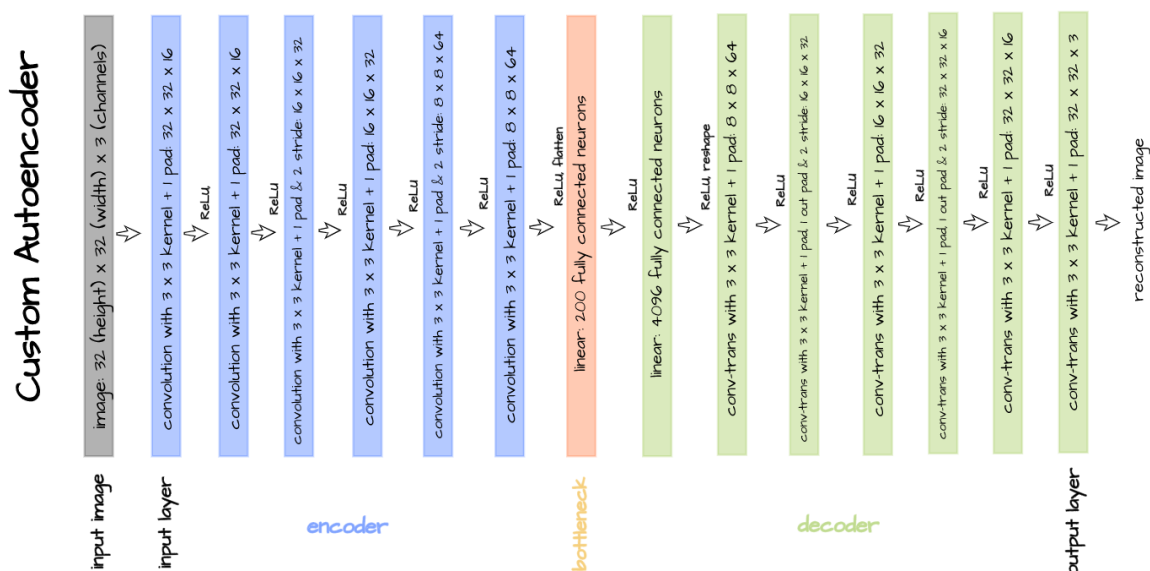
فیچرهای کمتری دسترسی دارد و تصاویری که می‌سازد با جزئیات کمتری خواهند بود. یک مثال از اینترنت بینیم:



## Decoder

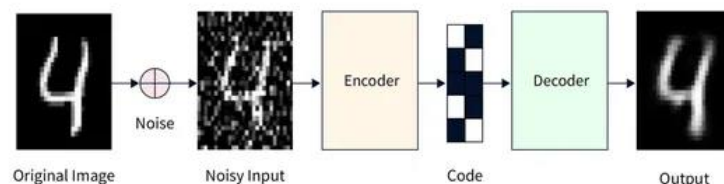
این بخش همان بخشی است که فیچرها به تصویر تبدیل می‌شوند. این بخش سعی دارد تا ویژگی‌ها را بفهمد و در نهایت تصاویر را همان‌طور که در ابتدا بودند، بسازد.

در زیر می‌توان مثالی از یک autoencoder که متشکل از شبکه‌های کانولوشنی است ببینیم:



این سوال پیش می‌آید که autoencoder ها و یا Convolutional Autoencoder ها دقیقاً چه کاربردهایی دارند؟

در قدم اول چون ابعاد تصاویر کمتر می‌شوند، فضای کمتری را برای ذخیره درگیر می‌کنند. همچنین همانطور که پیش از این بیان شد از autoencoderها برای feature extraction می‌توان استفاده کرد. یکی از کاربردهای فراوان آن‌ها این است که می‌توانند تصاویر را به صورت غیر نویزی بازسازی کنند (denoising). با ادغام این نوع ساختارهای شبکه‌ای با شبکه‌های GAN، می‌توانند تصاویر جدیدی تولید کنند. همچنین این قابلیت را دارند که به عنوان بخشی از شبکه transfer learning استفاده شوند. یک کاربرد بسیار جالب دیگر autoencoderها این است که می‌توانند تفاوت‌ها در تصاویر را پیدا کنند. به این صورت که اگر داده تست با داده‌هایی که در training set وجود دارد و دیده است متفاوت باشد، error در خروجی از یک مقدار thresholdی بیشتر می‌شود که می‌فهمیم تصاویر با هم متفاوت هستند.



منابع:

- Olu-Ipinlaye, Oreolorun (2022). Convolutional Autoencoders. <https://blog.paperspace.com/convolutional-autoencoder/>
- Dongre, Anay (2023). Overview of Autoencoders. <https://dongreanay.medium.com/overview-of-autoencoders-52c777418937>
- Bishayee, Soumallya (2023). The Basic Concept of Autoencoder — The Self-supervised Deep Learning. <https://medium.com/@soumallya160/the-basic-concept-of-autoencoder-the-self-supervised-deep-learning-454e75d93a04>

سوال ۲:

(الف)

با استفاده از dilated convolution، میدان دید افزایش می‌یابد. پس به تعداد لایه‌ها و پارامترهای کمتری نیاز است تا میدان دید موثر نهایی روی عکس ورودی، تمام ابعاد عکس را در بر بگیرد. در صورت وجود کانولوشن‌های گسترش یافته دیگر نیازی به استفاده از انواع pooling نخواهد بود. همچنین رزولوشن حفظ خواهد شد. علاوه بر آن، ویژگی‌هایی که استخراج می‌شوند، ویژگی‌هایی در مقیاس‌های مختلف خواهند بود. یعنی برای مثال شبکه‌ای را در نظر بگیرید که در لایه‌های ابتدایی شامل کانولوشن‌های عادی یا کانولوشن‌های گسترش یافته با rate پایین

باشد تا بتواند در ابتدا ویژگی‌های جزئی را بیابد. در لایه‌های بالاتر کانولوشن‌های گسترش یافته با rate بالا داشته باشد تا بتواند ویژگی‌های کلی و الگوهای بزرگتری بیابد؛ درحالی که پارامترها و تعداد لایه‌های کمتری دارد. به همین دلیل در semantic segmentation و object recognition کاربرد دارند.

اما استفاده از لایه‌های کانولوشنی گسترده در مقابل ممکن است ویژگی‌های جزئی را که به مکان وابسته هستند و برآمده از مجموعه‌ای از پیکسل‌های مجاور هم هستند، نبیند. از آنجایی که ورودی‌های این شبکه‌ها می‌توانند زرولوشن بالایی داشته باشند، ممکن است مصرف زیاد حافظه داشته باشیم. اگر از rate بالاتر و یا عمل padding استفاده کنیم نیز memory بالایی نیاز داریم. از طرفی ممکن است پیچیدگی و هزینه‌های بالای محاسباتی خود را داشته باشد.

(ب)

به طور کلی برای هر لایه میدان دید به صورت زیر خواهد بود:

$$RF_i = RF_{i-1} + (k - 1) \times d$$

برای اولین لایه ما در واقع سائز اصلی فیلتر به تعداد  $(k-1)$  تا خانه‌های  $(d-1)$  تایی اضافه می‌شود. یعنی بین هر دو خانه از فیلتر  $k$  تایی،  $(d-1)$  خانه اضافه می‌شود.

$$RF_0 = 1$$

$$RF_1 = RF_0 + (k - 1) \times d = k + (k - 1)(d - 1) = 1 + d(k - 1)$$

در ادامه برای باقی خانه‌ها خواهیم داشت:

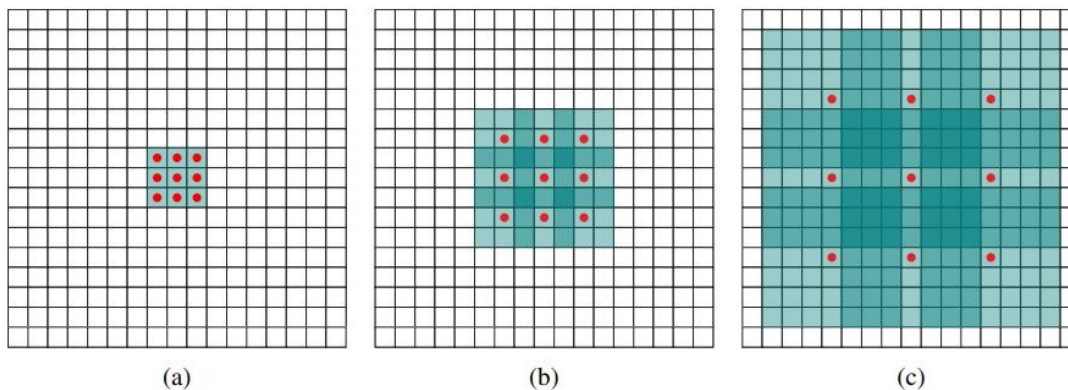
$$RF_2 = RF_1 + (k - 1) \times d = 1 + 2d(k - 1)$$

$$RF_3 = RF_2 + (k - 1) \times d = 1 + 3d(k - 1)$$

پس رابطه دیگری نیز می‌توان به طور کلی نوشت:

$$RF_i = 1 + id(k - 1)$$





(پ)

فیلتر با ابعاد  $k$  و پارامتر گسترش  $d$ ، مانند یک فیلتر کانولوشن عادی با ابعاد  $1+d(k-1)$  عمل خواهد کرد. با فرض padding برابر صفر و stride برابر یک خواهیم داشت:

$$\frac{N - (1 + d(k - 1)) + 2(0)}{1} + 1 = N - kd + d = N - d(k - 1)$$

یعنی ابعاد خروجی لایه به صورت  $m \times (N - d(k - 1)) \times (N - d(k - 1))$  خواهد بود.

منابع:

- Banerjee, Arinjoy (2022). Dilated Convolutions ( Deep Learning). <https://arinjoyemail.medium.com/dilated-convolutions-deep-learning-eb9fd3121e8e>
- geeksforgeeks (2023). Dilated Convolution. <https://www.geeksforgeeks.org/dilated-convolution/>
- Yu, Fisher, and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions." arXiv preprint arXiv:1511.07122 (2015).

سوال ۳:



Layer	output dimension	Number of parameters
Input	$32 \times 32 \times 3$	
CONV3-8	$32 \times 32 \times 8$	$[(3 \times (3 \times 3)) + 1] \times 8 = 224$
ReLU	$32 \times 32 \times 8$	تعداد پارامترها له عمق
Pool-2	$16 \times 16 \times 8$	
BATCHNORM	$16 \times 16 \times 8$	$\beta, \gamma \Rightarrow 2$
CONV5-16(3,2)	$6 \times 6 \times 16$	$[(8 \times (5 \times 5)) + 1] \times 16 = 3276$
ReLU	$6 \times 6 \times 16$	
Pool-2	$3 \times 3 \times 16$	
FLATTEN	$1 \times 1 \times 144$	
FC-10	$1 \times 1 \times 10$	1440

محاسبات:

$$\frac{(32-3)+2}{1} + 1 = 32 \quad \frac{32-2}{2} + 1 = 16$$

$$\frac{(16-5)+2(2)}{3} + 1 = 6 \quad \frac{6-2}{2} + 1 = 3$$

$$224 + 2 + 3276 + 1440 = 4882$$

تعداد کل پارامترها برابر 4882 می باشد:

سوال 4 :

الف) پارامترهای شبکه  $w_1, w_2, w_3, a, b, k_1, k_2, k_3$

ب) این شبکه جان MLP است:

$$\frac{\partial L}{\partial w_1} = [-(\hat{y} - y)] [v_1] = (\hat{y} - y) v_1$$

$$\frac{\partial L}{\partial w_2} = (\hat{y} - y) v_2$$

$$\frac{\partial L}{\partial a} = (\hat{y} - y)$$

$$\frac{\partial L}{\partial z_1} = \frac{\partial}{\partial v_1} \frac{\partial v_1}{\partial z_1}$$

$$\max(0, z_1, z_2) = \max(0, \max(z_1, z_2)) = \max(0, z)$$

$$\frac{\partial v_1}{\partial z_1} = \begin{cases} 0 & z < 0 \Rightarrow z_1, z_2 < 0 \\ 1 & z > 0 \end{cases}$$

$$\Rightarrow \frac{\partial L}{\partial z_1} = \begin{cases} 0 & \text{if } z_1 < 0, z_2 < 0 \\ \delta_1 & \end{cases}$$

$$\Rightarrow \frac{\partial L}{\partial z_3} = \begin{cases} 0 & \text{if } z_2 < 0, z_3 < 0 \\ \delta_2 & \end{cases}$$

به طور مشابه:

$$\frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial v_1} \frac{\partial v_1}{\partial z_2} + \frac{\partial L}{\partial v_2} \frac{\partial v_2}{\partial z_2} = \delta_1 \frac{\partial v_1}{\partial z_2} + \delta_2 \frac{\partial v_2}{\partial z_2}$$

اما برای  $z_2$ ، 2 مسیر داریم:

فلسفه عبارت با چندین حالت خواهد داشت که فقط کافیست به روابط زیر دقت کرد:

$$\frac{\partial V_1}{\partial z_2} = \begin{cases} 0 & \text{if } z_1 < 0, z_2 < 0 \\ 1 & \end{cases}$$

$$\frac{\partial V_2}{\partial z_2} = \begin{cases} 0 & \text{if } z_2 < 0, z_3 < 0 \\ 1 & \end{cases}$$

(۲)

$$z_1 = k_1 u_1 + k_2 u_2 + k_3 u_3 + b$$

$$z_2 = k_1 u_2 + k_2 u_3 + k_3 u_4 + b$$

$$z_3 = k_1 u_3 + k_2 u_4 + k_3 u_5 + b$$

$$\begin{aligned} \frac{\partial L}{\partial k_1} &= \frac{\partial L}{\partial z_1} \frac{\partial z_1}{\partial k_1} + \frac{\partial L}{\partial z_2} \frac{\partial z_2}{\partial k_1} + \frac{\partial L}{\partial z_3} \frac{\partial z_3}{\partial k_1} \\ &= \alpha_1 u_1 + \alpha_2 u_2 + \alpha_3 u_3 \end{aligned}$$

$$\begin{aligned} \frac{\partial L}{\partial k_2} &= \frac{\partial L}{\partial z_1} \frac{\partial z_1}{\partial k_2} + \frac{\partial L}{\partial z_2} \frac{\partial z_2}{\partial k_2} + \frac{\partial L}{\partial z_3} \frac{\partial z_3}{\partial k_2} \\ &= \alpha_1 u_2 + \alpha_2 u_3 + \alpha_3 u_4 \end{aligned}$$

$$\begin{aligned} \frac{\partial L}{\partial k_3} &= \frac{\partial L}{\partial z_1} \frac{\partial z_1}{\partial k_3} + \frac{\partial L}{\partial z_2} \frac{\partial z_2}{\partial k_3} + \frac{\partial L}{\partial z_3} \frac{\partial z_3}{\partial k_3} \\ &= \alpha_1 u_3 + \alpha_2 u_4 + \alpha_3 u_5 \end{aligned}$$

$$\begin{aligned} \frac{\partial L}{\partial b} &= \frac{\partial L}{\partial z_1} \frac{\partial z_1}{\partial b} + \frac{\partial L}{\partial z_2} \frac{\partial z_2}{\partial b} + \frac{\partial L}{\partial z_3} \frac{\partial z_3}{\partial b} \\ &= \alpha_1 + \alpha_2 + \alpha_3 \end{aligned}$$

(ع)

$$\frac{\partial}{\partial k_i} = \alpha_1 u_i + \alpha_2 u_{i+1} + \alpha_3 u_{i+2} + \dots + \alpha_m u_{i+m-1}$$

تعداد  $m$  عبارت جمع می شوند

$$\frac{\partial}{\partial b} = \alpha_1 + \alpha_2 + \dots + \alpha_m$$