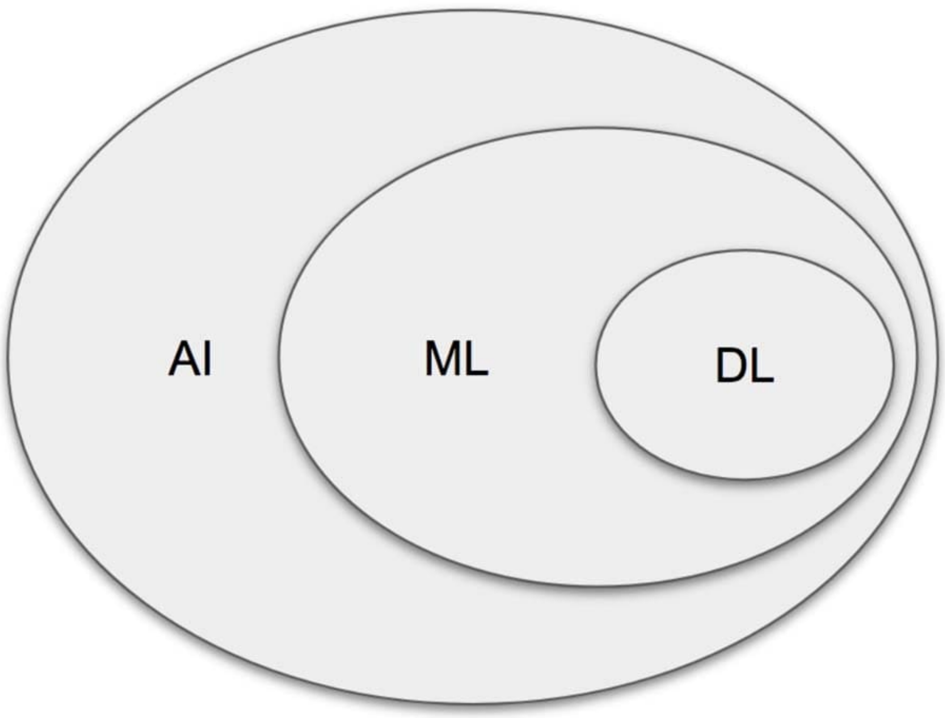
1) What’s the difference between AI, ML and DL?

In [6]:

```python
from IPython.display import Image
from IPython.core.display import HTML
Image(url= "https://sonix.ai/packs/media/images/corp/articles/classification-of-ai-ml-dl-
44838b636b886c49760da4ab7f6b6fa4.jpg")
```

Out[6]:



AI:
AI is incorporating human intelligence to machines. Whenever a machine completes tasks based on a set of rules that solve problems (algorithms), such an “intelligent” behaviour is what is called artificial intelligence.

ML:
ML is to enable machines to learn by themselves using the provided data and make accurate predictions. It is a method of training algorithms such that they can learn how to make decisions. And, a machine learning algorithm can be developed to try to identify whether the fruit is an orange or an apple. After the algorithm is fed with the training data, it will learn the differing characteristics between an orange and an apple. Therefore, if provided with data of weight and texture, it can predict accurately the type of fruit with those characteristics.

DL:
In other words, DL is the next evolution of machine learning. DL algorithms are roughly inspired by the information processing patterns found in the human brain. Just like we use our brains to identify patterns and classify various types of information, deep learning algorithms can be taught to accomplish the same tasks for machines.

2) What is supervised learning?

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.
$Y = f(X)$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

Supervised learning problems can be further grouped into regression and classification problems.

- [illegible]

- Classification: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
- Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

3) What is unsupervised learning?

Unsupervised learning is where you only have input data (X) and no corresponding output variables.

The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.

Unsupervised learning problems can be further grouped into clustering and association problems.

- Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

4) Describe the following supervised learning problems in short:

a) Regression

- A regression problem is when the output variable is a real or continuous value, such as “salary” or “weight”. Many different models can be used, the simplest is the linear regression. It tries to fit data with the best hyper-plane which goes through the points.

b) Classification

- A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”. A classification model attempts to draw some conclusion from observed values. Classification models include logistic regression, decision tree, random forest, gradient-boosted tree, multilayer perceptron, one-vs-rest, and Naive Bayes.

c, d, e, f) Localization, Object detection, Semantic segmentation, Instance segmentation

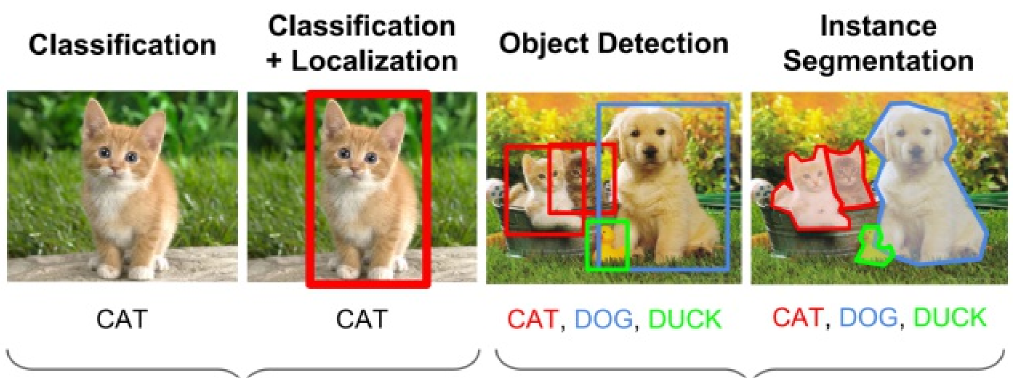
- Classification/Recognition: Given an image with an object , find out what that object is. In other words, classify it in a class from a set of predefined categories.

- Localization : Find where the object is and draw a bounding box around it
- Object detection: Classify and detect all objects in the image. Assign a class to each object and draw a bounding box around it.
- Semantic Segmentation: Classify every pixel in the image to a class according to its context, so that each pixel is assigned to an object
- Instance Segmentation: Classify every pixel in the image to a class so that each pixel is assigned to a different instance of an object

In [7]:

```
Image(url="https://sergioskar.github.io/assets/img/posts/cv_tasks.jpg")
```

Out[7]:



g) Clustering

- Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

h) Image captioning

- Image Captioning is the process of generating textual description of an image. It uses both Natural Language Processing and Computer Vision to generate the captions. link for paper

i) Machine translation

- https://medium.com/@ageitgey/machine-learning-is-fun-part-5-language-translation-with-deep-learning-and-the-magic-of-sequences-2ace0acca0aa

j) Density estimation

- Use statistical models to find an underlying probability distribution that gives rise to the observed variables.
  - Example: Kernel Density Estimation (Theory and Application)
  - Example: Mixture models. Normal (or Gaussian) mixture models are especially popular.

In statistics, kernel density estimation (KDE) is a non-parametric way to estimate the probability density function of a random variable

In [8]:

```
Image(url="https://wikimedia.org/api/rest_v1/media/math/render/svg/f3b09505158fb06033aabf9b0116c8c(
8bf31")
```

Out[8]:

5) Linear Regression https://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html

6) Polinomial Regression https://towardsdatascience.com/introduction-to-linear-regression-and-polynomial-regression-f8adc96f31cb

7, 8) What is Overfitting and Underfitting

- Overfitting occurs when a statistical model or machine learning algorithm captures the noise of the data. Intuitively, overfitting occurs when the model or the algorithm fits the data too well. Specifically, overfitting occurs if the model or algorithm shows low bias but high variance. Overfitting is often a result of an excessively complicated model, and it can be prevented by fitting multiple models and using validation or cross-validation to compare their predictive accuracies on test data.
- Underfitting occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data. Intuitively, underfitting occurs when the model or the algorithm does not fit the data well enough. Specifically, underfitting occurs if the model or algorithm shows low variance but high bias. Underfitting is often a result of an excessively simple model.

9) What is Logisitc Regression https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148 https://kambria.io/blog/logistic-regression-for-machine-learning/

10) What do you mean under interpretability? When is it important?

- Interpretability is closely connected with the ability of users to understand the model. Typical criteria are:
  - a small number of input features (only the necessary ones), ideally not more than 2-3 to make a simple visualization possible,
  - features fitting to user's expectation (e.g., based on a priori knowledge that can be formalized as a priori relevance),
  - a simple internal processing, e.g. by understandable rules instead of a complex processing inside an Artificial Neural Network.

11) Define the dense layer and its gradient (with formula).

- A dense layer is a Layer in which Each Input Neuron is connected to the output Neuron, like a Simple neural net, the parameters units just tells you the dimensionnality of your Output

```
model = Sequential()
model.add(Dense(, input_shape=(,)))
```

```
"""
@tf.custom_gradient
def log1pexp(x):
e = tf.exp(x)
def grad(dy):
return dy * (1 - 1 / (1 + e))
return tf.math.log(1 + e), grad
"""
```

Gradient Descent:

Given a cost function:

$f(m, b) = \frac{1}{N}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$

The Gradient can be calculated:

$$f'(m, b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N}\sum -2(y_i - (mx_i + b)) \end{bmatrix}$$

12) Why do we need activation functions

- We must use activation functions such as ReLu, sigmoid and tanh in order to add a non-linear property to the neural network. In this way, the network can model more complex relationships and patterns in the data.

13) Define the fully connected neural network (with formula): input, goal, cost function, layers

- https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html
- http://cs231n.github.io/neural-networks-1/
- https://www.researchgate.net/publication/331525817_Temporal_Convolutional_Neural_Network_for_the_Classification_of_Satelli
- https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/fc_layer.html

14) What are the partial derivatives?

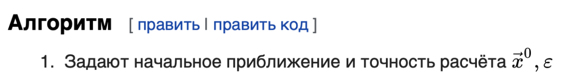
15) Define the gradient descent algorithm!

```
model = initialization(...)
n_epochs = ...
train_data = ...
for i in n_epochs:
train_data = shuffle(train_data)
X, y = split(train_data)
predictions = predict(X, model)
error = calculate_error(y, predictions)
model = update_model(model, error)
```

In [14]:

```
Image("/Users/macbook/Desktop/1.png")
```

Out[14]:



Алгоритм [править | править код]

1. Задают начальное приближение и точность расчёта $\vec{x}^0, \varepsilon$

2. Рассчитывают $\vec{x}^{[j+1]} = \vec{x}^{[j]} - \lambda^{[j]} \nabla F\left(\vec{x}^{[j]}\right)$, где $\lambda^{[j]} = \operatorname{argmin}_\lambda F\left(\vec{x}^{[j]} - \lambda \nabla F\left(\vec{x}^{[j]}\right)\right)$
3. Проверяют условие остановки:
   - Если $\left|\vec{x}^{[j+1]} - \vec{x}^{[j]}\right| > \varepsilon$, $\left|F\left(\vec{x}^{[j+1]}\right) - F\left(\vec{x}^{[j]}\right)\right| > \varepsilon$ или $\left\|\nabla F\left(\vec{x}^{[j+1]}\right)\right\| > \varepsilon$ (выбирают одно из условий), то $j = j + 1$ и переход к шагу 2.
   - Иначе $\vec{x} = \vec{x}^{[j+1]}$ и останов.

16) Define the mini-batch stochastic gradient descent algorithm!

- Batch gradient descent is a variation of the gradient descent algorithm that calculates the error for each example in the training dataset, but only updates the model after all training examples have been evaluated.
- Mini-batch gradient descent is a variation of the gradient descent algorithm that splits the training dataset into small batches that are used to calculate model error and update model coefficients.
  - Implementations may choose to sum the gradient over the mini-batch which further reduces the variance of the gradient.
  - Mini-batch gradient descent seeks to find a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It is the most common implementation of gradient descent used in the field of deep learning.

Let theta = model parameters and max_iters = number of epochs.
for itr = 1, 2, 3, …, max_iters:
for mini_batch (X_mini, y_mini):

Forward Pass on the batch X_mini:
Make predictions on the mini-batch
Compute error in predictions (J(theta)) with the current values of the parameters
Backward Pass:
Compute gradient(theta) = partial derivative of J(theta) w.r.t. theta
Update parameters:
theta = theta – learning_rate*gradient(theta)

17) What is the problem with setting the learning rate too high/low?

- new_weight = existing_weight — learning_rate * gradient
- This parameter scales the magnitude of our weight updates in order to minimize the network's loss function.
  - If your learning rate is set too low, training will progress very slowly as you are making very tiny updates to the weights in your network. However, if your learning rate is set too high, it can cause undesirable divergent behavior in your loss function

18) Define the following functions and their gradients (with formula):

a) MSE

- We have a hipotise: $h_\theta = \theta_0 + \theta_1 * x$
- MSE $\frac{1}{n}\sum_{t=1}^{n}(y_i - \hat{y}_i)^2$
- $\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2$
- Gradient $\theta_j := \theta_j - \frac{1}{m}\alpha\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

b) Mean absolute error

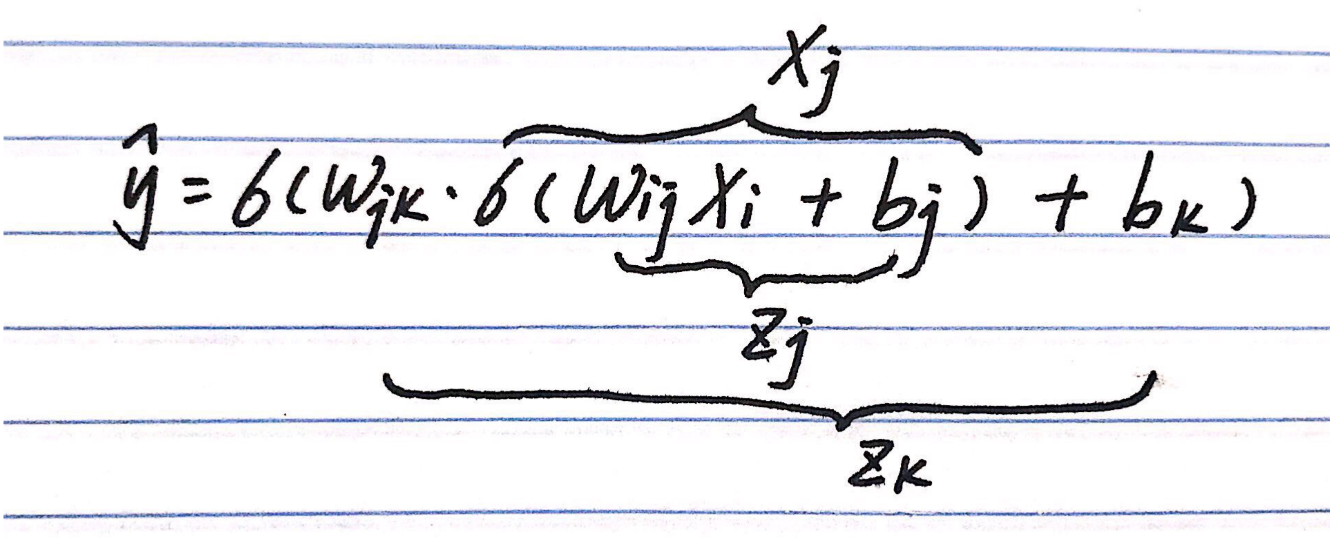
- We have a hypotise:$h_\theta = \theta_0 + \theta_1 * x$
- MAE: $\frac{1}{n}\Sigma_{i=1}^{n}\left(\frac{d_i - f_i}{\sigma_i}\right)^2$
- $\frac{d\mathrm{MAE}}{dy_{\text{pred}}} = \begin{cases} +1, & y_{\text{pred}} > y_{\text{true}} \\ -1, & y_{\text{pred}} < y_{\text{true}} \end{cases}$

c) Sigmoid function

In [17]:

```
Image(url="https://miro.medium.com/max/3840/1*OUM97X0_NvgBkCwplDtrXQ.png")
Image(url="https://miro.medium.com/max/3840/1*QHM97X0-NvgRkCwp1DtrxQ.png")
```

Out[17]:

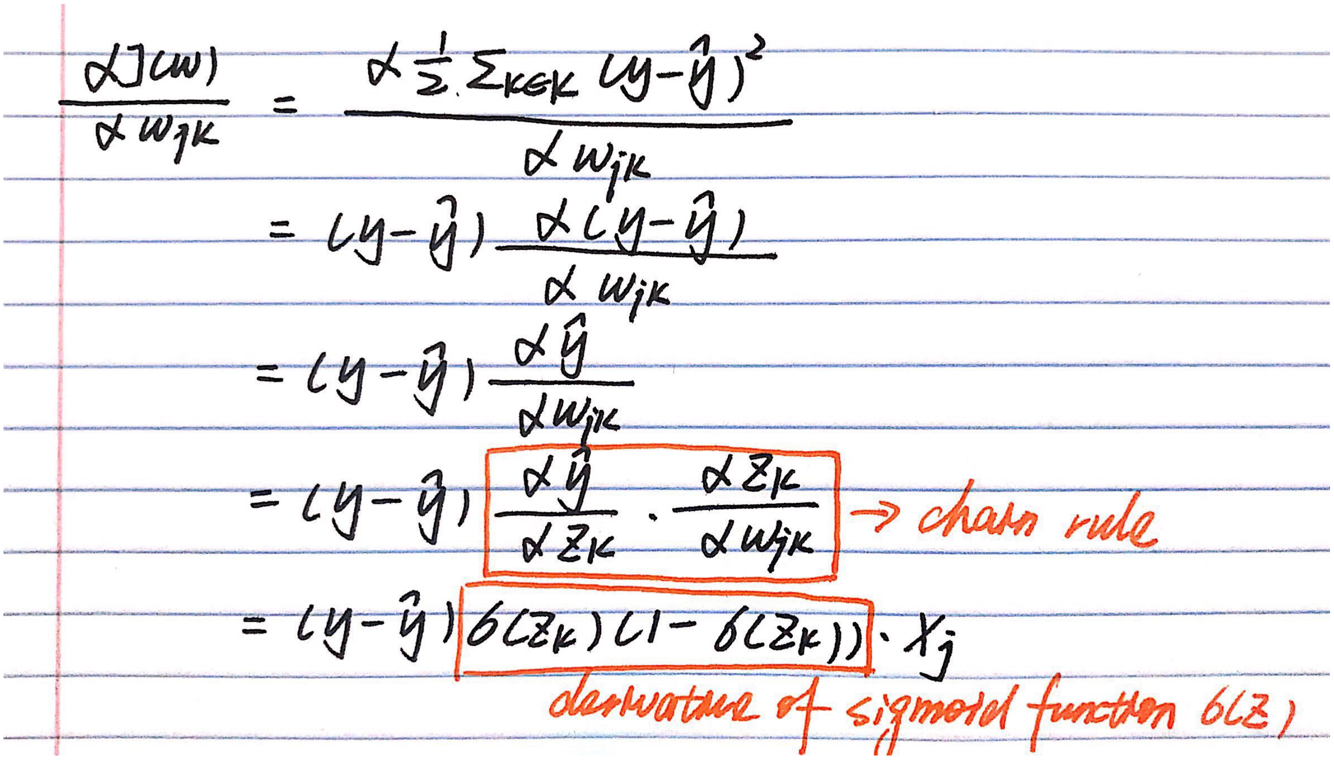


$$\hat{y} = \sigma(\underbrace{w_{jk}\cdot \overbrace{\sigma(\underbrace{w_{ij}x_i + b_j}_{z_j})}^{x_j} + b_k}_{z_k})$$

In [18]:

```
Image(url="https://miro.medium.com/max/4736/1*kCHvviwoArAoB92kawUBmg.png")
```

Out[18]:



$$\frac{\partial J(w)}{\partial w_{jk}} = \frac{\partial \frac{1}{2}\sum_{k\in K}(y-\hat{y})^2}{\partial w_{jk}}$$

$$= (y-\hat{y})\frac{\partial(y-\hat{y})}{\partial w_{jk}}$$

$$= (y-\hat{y})\frac{\partial \hat{y}}{\partial w_{jk}}$$

$$= (y-\hat{y})\boxed{\frac{\partial \hat{y}}{\partial z_k}\cdot\frac{\partial z_k}{\partial w_{jk}}} \rightarrow \text{chain rule}$$

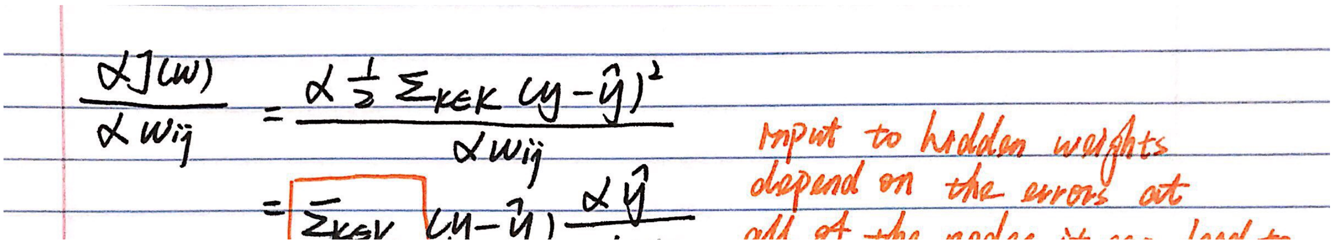
$$= (y-\hat{y})\boxed{\sigma(z_k)(1-\sigma(z_k))}\cdot x_j$$

derivative of sigmoid function $\sigma(z)$

In [19]:

```
Image(url="https://miro.medium.com/max/4800/1*NWz132H10NIQ5ZBiWripmg.png")
```

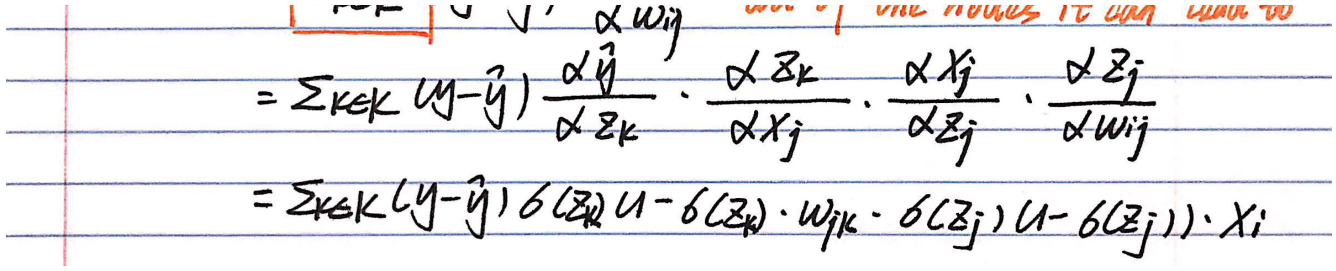
Out[19]:



$$\frac{\partial J(w)}{\partial w_{ij}} = \frac{\partial \frac{1}{2}\sum_{k\in K}(y-\hat{y})^2}{\partial w_{ij}}$$

$$= \boxed{\sum_{k\in K}}(y-\hat{y})\frac{\partial \hat{y}}{}$$

input to hidden weights depend on the errors at all of the nodes it can lead to



∂w_{ij} ... of the nodes it can lead to

$$= \sum_{k\in K}(y-\hat{y})\frac{\partial \hat{y}}{\partial z_k}\cdot\frac{\partial z_k}{\partial x_j}\cdot\frac{\partial x_j}{\partial z_j}\cdot\frac{\partial z_j}{\partial w_{ij}}$$

$$= \sum_{k\in K}(y-\hat{y})\sigma(z_k)(1-\sigma(z_k))\cdot w_{jk}\cdot\sigma(z_j)(1-\sigma(z_j))\cdot x_i$$

d) Binary CrossEntropy

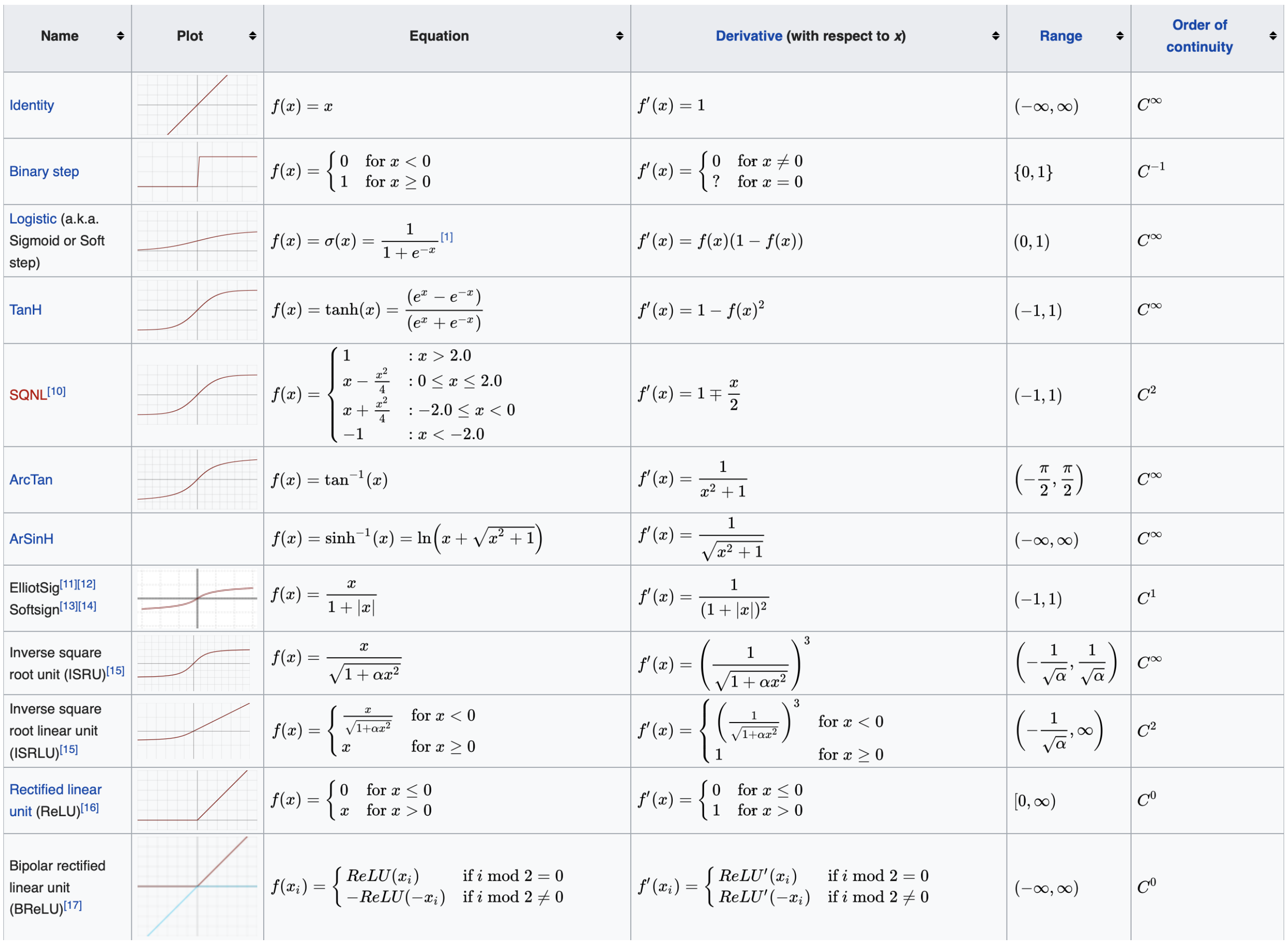
- $W(n+1) = W(n) - \eta\left(-y^{\frac{1}{z}} - (1-y)^{\frac{1}{z-1}})\sigma'(V)X\right)$

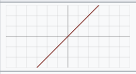
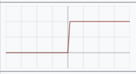
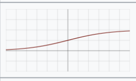
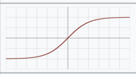
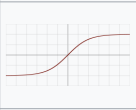
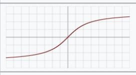
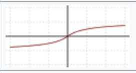
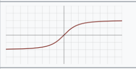
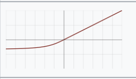
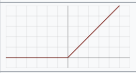
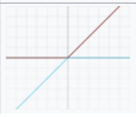
e, f, g, h, i, j)

In [20]:

```
Image("/Users/macbook/Desktop/2.png")
```

Out[20]:

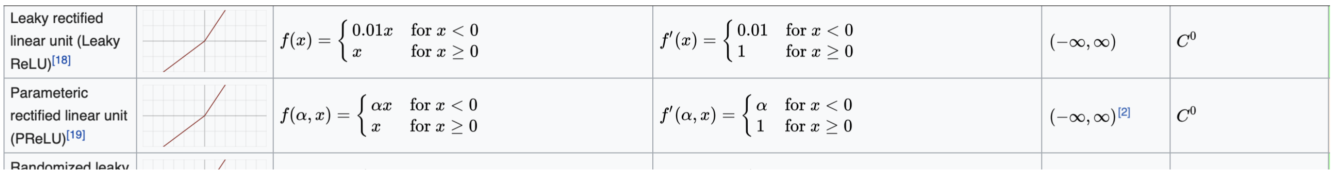


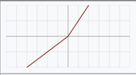
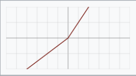
Name	Plot	Equation	Derivative (with respect to *x*)	Range	Order of continuity
Identity		$f(x)=x$	$f'(x)=1$	$(-\infty,\infty)$	C^∞
Binary step		$f(x)=\begin{cases}0 & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}0 & \text{for } x\neq 0\\ ? & \text{for } x=0\end{cases}$	$\{0,1\}$	C^{-1}
Logistic (a.k.a. Sigmoid or Soft step)		$f(x)=\sigma(x)=\frac{1}{1+e^{-x}}$[1]	$f'(x)=f(x)(1-f(x))$	$(0,1)$	C^∞
TanH		$f(x)=\tanh(x)=\frac{(e^x-e^{-x})}{(e^x+e^{-x})}$	$f'(x)=1-f(x)^2$	$(-1,1)$	C^∞
SQNL[10]		$f(x)=\begin{cases}1 & : x>2.0\\ x-\frac{x^2}{4} & : 0\leq x\leq 2.0\\ x+\frac{x^2}{4} & : -2.0\leq x<0\\ -1 & : x<-2.0\end{cases}$	$f'(x)=1\mp\frac{x}{2}$	$(-1,1)$	C^2
ArcTan		$f(x)=\tan^{-1}(x)$	$f'(x)=\frac{1}{x^2+1}$	$\left(-\frac{\pi}{2},\frac{\pi}{2}\right)$	C^∞
ArSinH		$f(x)=\sinh^{-1}(x)=\ln\left(x+\sqrt{x^2+1}\right)$	$f'(x)=\frac{1}{\sqrt{x^2+1}}$	$(-\infty,\infty)$	C^∞
ElliotSig[11][12] Softsign[13][14]		$f(x)=\frac{x}{1+\|x\|}$	$f'(x)=\frac{1}{(1+\|x\|)^2}$	$(-1,1)$	C^1
Inverse square root unit (ISRU)[15]		$f(x)=\frac{x}{\sqrt{1+\alpha x^2}}$	$f'(x)=\left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3$	$\left(-\frac{1}{\sqrt{\alpha}},\frac{1}{\sqrt{\alpha}}\right)$	C^∞
Inverse square root linear unit (ISRLU)[15]		$f(x)=\begin{cases}\frac{x}{\sqrt{1+\alpha x^2}} & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}\left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3 & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$	$\left(-\frac{1}{\sqrt{\alpha}},\infty\right)$	C^2
Rectified linear unit (ReLU)[16]		$f(x)=\begin{cases}0 & \text{for } x\leq 0\\ x & \text{for } x>0\end{cases}$	$f'(x)=\begin{cases}0 & \text{for } x\leq 0\\ 1 & \text{for } x>0\end{cases}$	$[0,\infty)$	C^0
Bipolar rectified linear unit (BReLU)[17]		$f(x_i)=\begin{cases}ReLU(x_i) & \text{if } i \bmod 2=0\\ -ReLU(-x_i) & \text{if } i \bmod 2\neq 0\end{cases}$	$f'(x_i)=\begin{cases}ReLU'(x_i) & \text{if } i \bmod 2=0\\ ReLU'(-x_i) & \text{if } i \bmod 2\neq 0\end{cases}$	$(-\infty,\infty)$	C^0

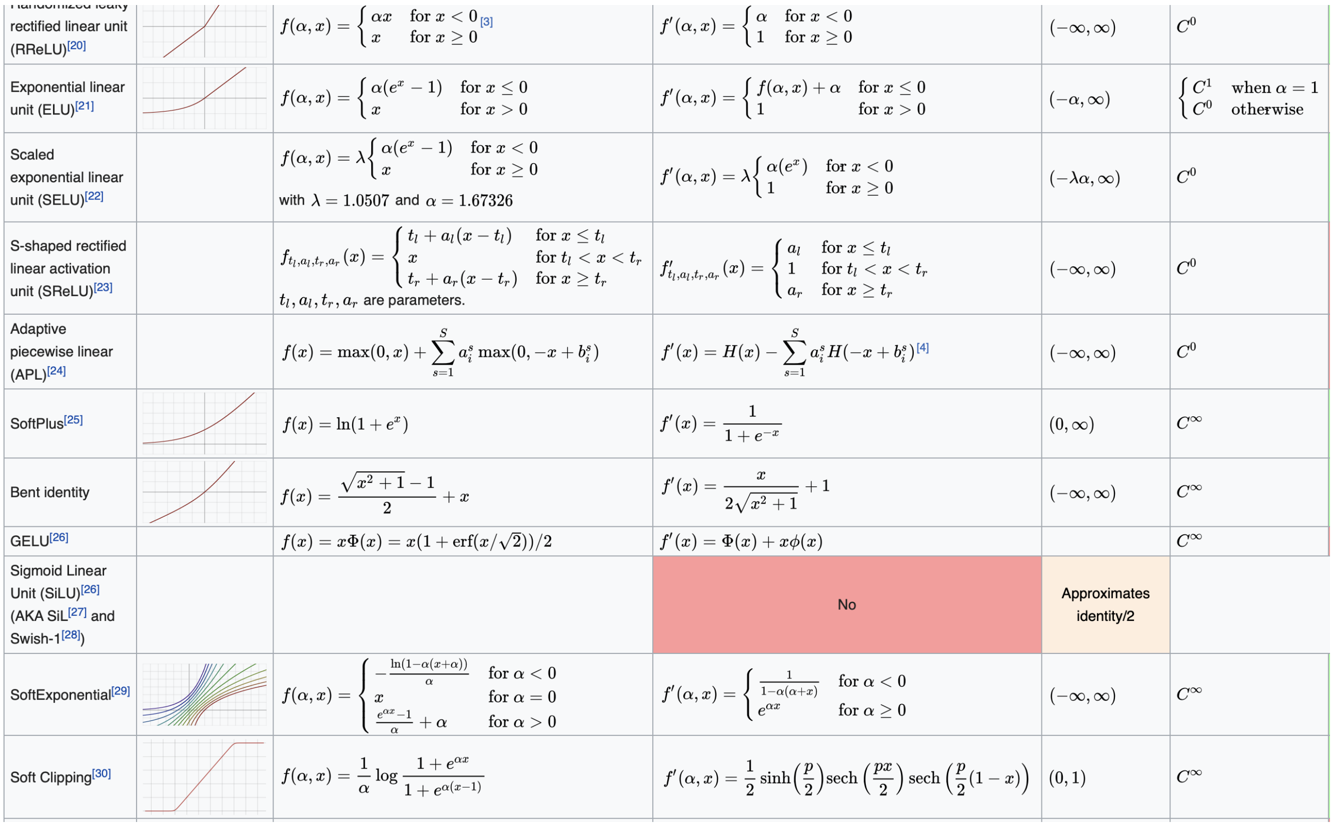
In [21]:

```
Image("/Users/macbook/Desktop/3.png")
```

Out[21]:



Leaky rectified linear unit (Leaky ReLU)[18]		$f(x)=\begin{cases}0.01x & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}0.01 & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$	$(-\infty,\infty)$	C^0
Parameteric rectified linear unit (PReLU)[19]		$f(\alpha,x)=\begin{cases}\alpha x & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(\alpha,x)=\begin{cases}\alpha & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$	$(-\infty,\infty)$[2]	C^0
Randomized leaky					

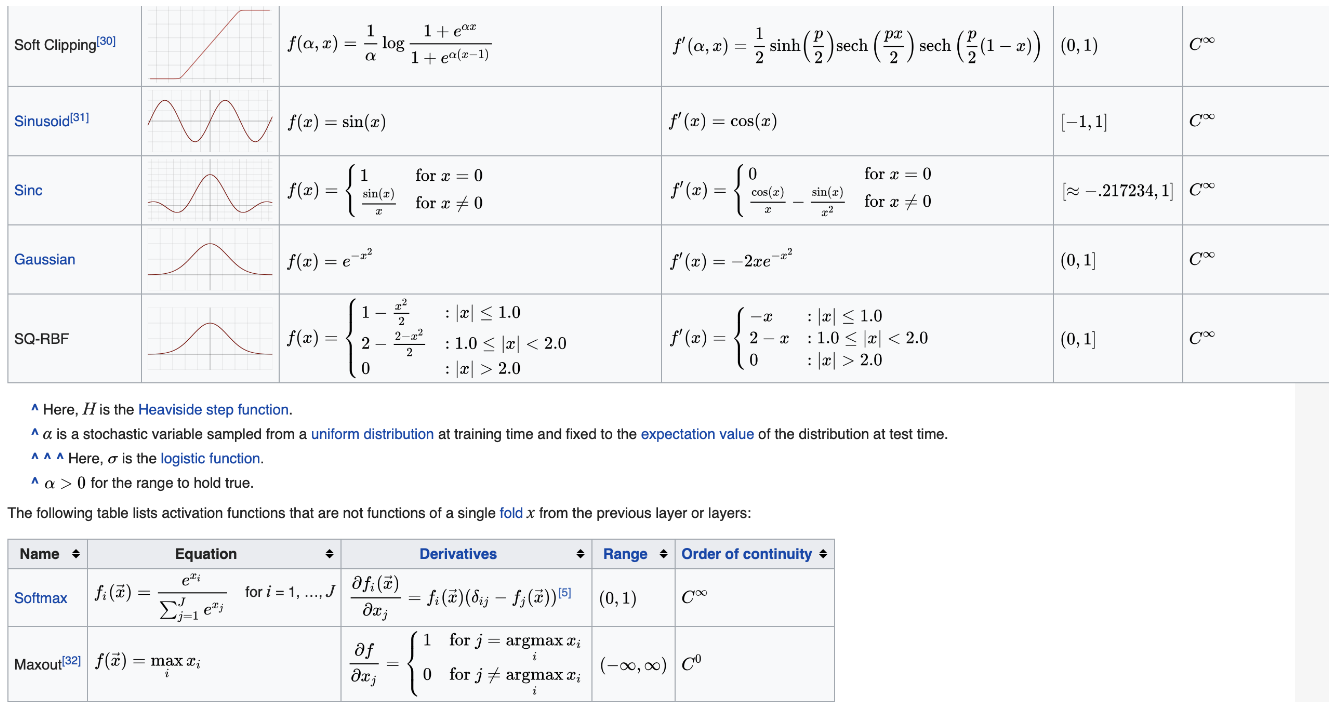


rectified linear unit (RReLU)[20]		$f(\alpha,x)=\begin{cases}\alpha x & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$[3]	$f'(\alpha,x)=\begin{cases}\alpha & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$	$(-\infty,\infty)$	C^0
Exponential linear unit (ELU)[21]		$f(\alpha,x)=\begin{cases}\alpha(e^x-1) & \text{for } x\leq 0\\ x & \text{for } x>0\end{cases}$	$f'(\alpha,x)=\begin{cases}f(\alpha,x)+\alpha & \text{for } x\leq 0\\ 1 & \text{for } x>0\end{cases}$	$(-\alpha,\infty)$	$\begin{cases}C^1 & \text{when } \alpha=1\\ C^0 & \text{otherwise}\end{cases}$
Scaled exponential linear unit (SELU)[22]		$f(\alpha,x)=\lambda\begin{cases}\alpha(e^x-1) & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$ with $\lambda=1.0507$ and $\alpha=1.67326$	$f'(\alpha,x)=\lambda\begin{cases}\alpha(e^x) & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$	$(-\lambda\alpha,\infty)$	C^0
S-shaped rectified linear activation unit (SReLU)[23]		$f_{t_l,a_l,t_r,a_r}(x)=\begin{cases}t_l+a_l(x-t_l) & \text{for } x\leq t_l\\ x & \text{for } t_l<x<t_r\\ t_r+a_r(x-t_r) & \text{for } x\geq t_r\end{cases}$ t_l, a_l, t_r, a_r are parameters.	$f'_{t_l,a_l,t_r,a_r}(x)=\begin{cases}a_l & \text{for } x\leq t_l\\ 1 & \text{for } t_l<x<t_r\\ a_r & \text{for } x\geq t_r\end{cases}$	$(-\infty,\infty)$	C^0
Adaptive piecewise linear (APL)[24]		$f(x)=\max(0,x)+\sum_{s=1}^{S}a_i^s\max(0,-x+b_i^s)$	$f'(x)=H(x)-\sum_{s=1}^{S}a_i^sH(-x+b_i^s)$[4]	$(-\infty,\infty)$	C^0
SoftPlus[25]		$f(x)=\ln(1+e^x)$	$f'(x)=\frac{1}{1+e^{-x}}$	$(0,\infty)$	C^∞
Bent identity		$f(x)=\frac{\sqrt{x^2+1}-1}{2}+x$	$f'(x)=\frac{x}{2\sqrt{x^2+1}}+1$	$(-\infty,\infty)$	C^∞
GELU[26]		$f(x)=x\Phi(x)=x(1+\mathrm{erf}(x/\sqrt{2}))/2$	$f'(x)=\Phi(x)+x\phi(x)$		C^∞
Sigmoid Linear Unit (SiLU)[26] (AKA SiL[27] and Swish-1[28])			No	Approximates identity/2	
SoftExponential[29]		$f(\alpha,x)=\begin{cases}-\frac{\ln(1-\alpha(x+\alpha))}{\alpha} & \text{for } \alpha<0\\ x & \text{for } \alpha=0\\ \frac{e^{\alpha x}-1}{\alpha}+\alpha & \text{for } \alpha>0\end{cases}$	$f'(\alpha,x)=\begin{cases}\frac{1}{1-\alpha(\alpha+x)} & \text{for } \alpha<0\\ e^{\alpha x} & \text{for } \alpha\geq 0\end{cases}$	$(-\infty,\infty)$	C^∞
Soft Clipping[30]		$f(\alpha,x)=\frac{1}{\alpha}\log\frac{1+e^{\alpha x}}{1+e^{\alpha(x-1)}}$	$f'(\alpha,x)=\frac{1}{2}\sinh\left(\frac{p}{2}\right)\mathrm{sech}\left(\frac{px}{2}\right)\mathrm{sech}\left(\frac{p}{2}(1-x)\right)$	$(0,1)$	C^∞

In [22]:

```
Image("/Users/macbook/Desktop/4.png")
```

Out[22]:



Soft Clipping[30]		$f(\alpha,x)=\frac{1}{\alpha}\log\frac{1+e^{\alpha x}}{1+e^{\alpha(x-1)}}$	$f'(\alpha,x)=\frac{1}{2}\sinh\left(\frac{p}{2}\right)\mathrm{sech}\left(\frac{px}{2}\right)\mathrm{sech}\left(\frac{p}{2}(1-x)\right)$	$(0,1)$	C^∞
Sinusoid[31]		$f(x)=\sin(x)$	$f'(x)=\cos(x)$	$[-1,1]$	C^∞
Sinc		$f(x)=\begin{cases}1 & \text{for } x=0\\ \frac{\sin(x)}{x} & \text{for } x\neq 0\end{cases}$	$f'(x)=\begin{cases}0 & \text{for } x=0\\ \frac{\cos(x)}{x}-\frac{\sin(x)}{x^2} & \text{for } x\neq 0\end{cases}$	$[\approx -.217234, 1]$	C^∞
Gaussian		$f(x)=e^{-x^2}$	$f'(x)=-2xe^{-x^2}$	$(0,1]$	C^∞
SQ-RBF		$f(x)=\begin{cases}1-\frac{x^2}{2} & :\lvert x\rvert\leq 1.0\\ 2-\frac{2-x^2}{2} & :1.0\leq\lvert x\rvert<2.0\\ 0 & :\lvert x\rvert>2.0\end{cases}$	$f'(x)=\begin{cases}-x & :\lvert x\rvert\leq 1.0\\ 2-x & :1.0\leq\lvert x\rvert<2.0\\ 0 & :\lvert x\rvert>2.0\end{cases}$	$(0,1]$	C^∞

^ Here, H is the Heaviside step function.
^ α is a stochastic variable sampled from a uniform distribution at training time and fixed to the expectation value of the distribution at test time.
^ ^ ^ Here, σ is the logistic function.
^ $\alpha>0$ for the range to hold true.

The following table lists activation functions that are not functions of a single fold x from the previous layer or layers:

Name	Equation	Derivatives	Range	Order of continuity
Softmax	$f_i(\vec{x})=\frac{e^{x_i}}{\sum_{j=1}^{J}e^{x_j}}$ for i = 1, …, J	$\frac{\partial f_i(\vec{x})}{\partial x_j}=f_i(\vec{x})(\delta_{ij}-f_j(\vec{x}))$[5]	$(0,1)$	C^∞
Maxout[32]	$f(\vec{x})=\max_i x_i$	$\frac{\partial f}{\partial x_j}=\begin{cases}1 & \text{for } j=\underset{i}{\operatorname{argmax}}\, x_i\\ 0 & \text{for } j\neq\underset{i}{\operatorname{argmax}}\, x_i\end{cases}$	$(-\infty,\infty)$	C^0

19) When and why should you normalize the data before training?

- The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

20) How do the following techniques prevent overfitting?

Data augmentation

Data augmentation

- Data augmentation is the technique of increasing the size of data used for training a model. For reliable predictions, the deep learning models often require a lot of training data, which is not always available. Therefore, the existing data is augmented in order to make a better generalized model
  - Position augmentation
  - Scaling
  - Cropping
  - Flipping
  - Padding
  - Rotation
  - Translation
  - Affine transformation

  Early stopping
- Early stopping is a technique for controlling overfitting in machine learning models, especially neural networks, by stopping training before the weights have converged. Often we stop when the performance has stopped improving on a held-out validation set.
  - backpropagation (Early stopping is commonly applied to the backpropagation algorithm.)
  - generalization (Early stopping is meant to improve generalization performance.)

  L1/L2
- Ridge regression adds “squared magnitude” of coefficient as penalty term to the loss function. Here the highlighted part represents L2 regularization element.
- Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds “absolute value of magnitude” of coefficient as penalty term to the loss function.

  Denoising(Dropout)
- Dropout is a simple and efficient way to prevent overfitting. We combine stacked denoising autoencoder and dropout together, then it has achieved better performance than singular dropout method, and has reduced time complexity during fine-tune phase. We pre-train the data with stacked denoising autoencoder, and to prevent units from co-adapting too much dropout is applied in the period of training.

  Batchnorm
- It basically force your activations (Conv,FC ouputs) to be unit standard deviation and zero mean. To each learning batch of data we apply the following normalization.
  - $\hat{x}(k) = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{VAR[x^{(k)}]}}$

21) Learning rate decay

- Learning rate schedules seek to adjust the learning rate during training by reducing the learning rate according to a pre-defined schedule. Common learning rate schedules include time-based decay, step decay and exponential decay.

- Let's use the time decay function. This updates the learning rate by the expression below:

  $new_{LR} = \frac{current_{LR}}{(1 + decay - rate * epoch_n um)}$

22) What is fully convolutional neural network?

- Fully convolutional indicates that the neural network is composed of convolutional layers without any fully-connected layers or MLP usually found at the end of the network. A CNN with fully connected layers is just as end-to-end learnable as a fully convolutional one. The main difference is that the fully convolutional net is learning filters every where. Even the decision-making layers at the end of the network are filters.
  - A fully convolutional net tries to learn representations and make decisions based on local spatial input. Appending a fully connected layer enables the network to learn something using global information where the spatial arrangement of the input falls away and need not apply.

23) List the output shape for each layer in the following convolutional neural network:

- Input layer: shape 28x28x3
- Conv2D layer: 64 filters, size 5x5 ((None, 64, 32, 32))
- MaxPooling2D layer: 2x2
- Conv2D layer: 128 filters, size 3x3, stride 2
- Dense layer: 512 units