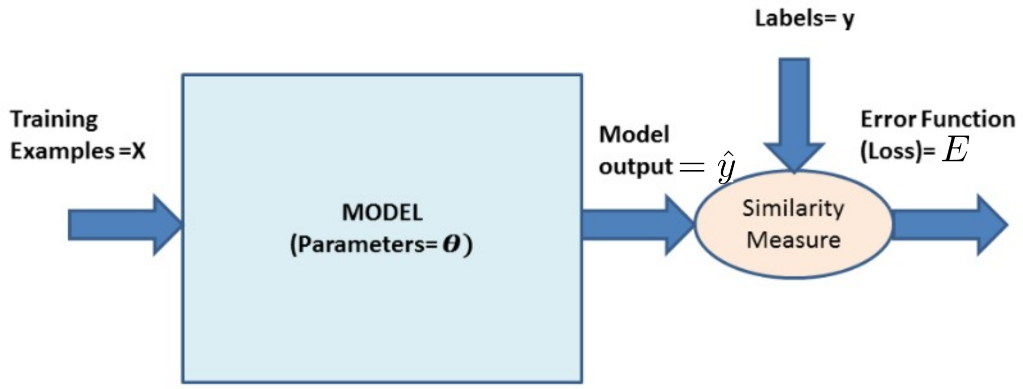
Supervised Learning & CNNs

Narada Warakagoda

Supervised Learning outline

- Given: A set of training examples (input, label) pairs { (x,y)}
- Find: Model parameters $\boldsymbol{\theta}$ such that the similarity between the model output $\hat{\mathbf{y}}$ and the labels $\mathbf{y}$ are maximized. Similarity is expressed as a loss/error/cost function.

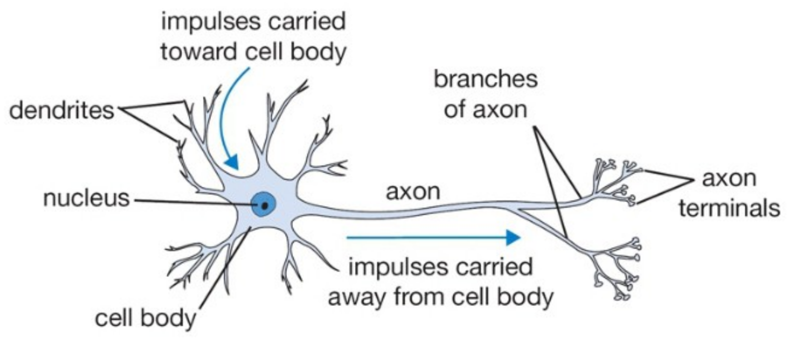


Main aspects of supervised deep learning

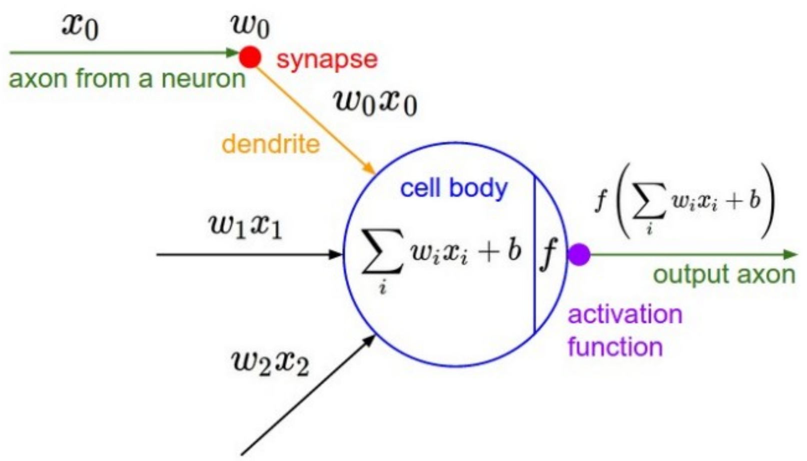
- What is the model architecture?
- What is the loss function?
- How do we update the model parameters?
- How do we maintain the generalization ability of the model?

Neuron

$$y = f(\sum_i w_i x_i + b) = f(\mathbf{x} \cdot \mathbf{w}^T + b)$$


impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body

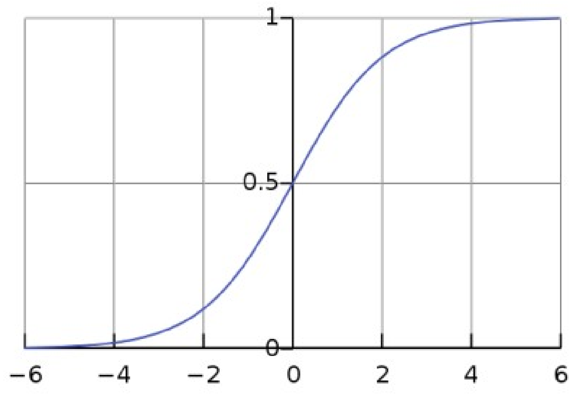
Biological Neuron



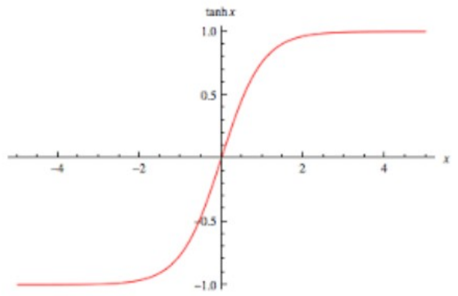
Mathematical Model

Illustrations from http://cs231n.github.io/neural-networks-1/

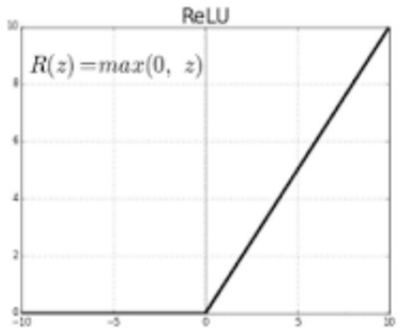
Activation Functions



Sigmoid

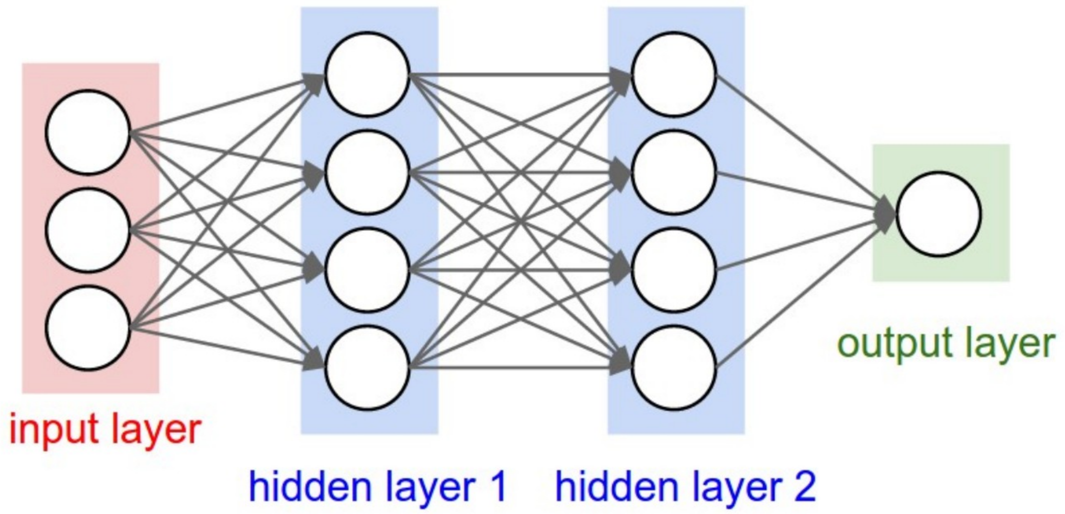


Tanh Sigmoid



Rectified Linear (ReLu)

Fully Connected (Dense) Network



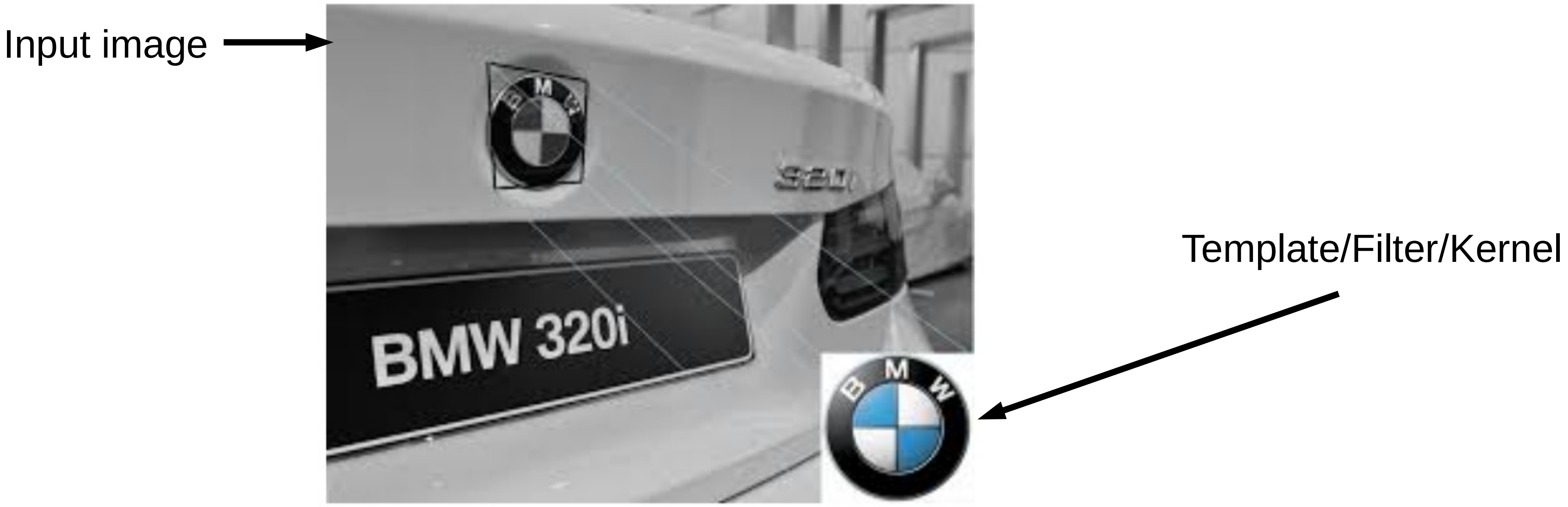
- Each layer performs $\mathbf{y} = f(\mathbf{x}\mathbf{W} + \mathbf{b})$
- Single hidden layer can approximate any function (Universal approximation theorem)
- Number of parameters can grow quickly

Convolutional Neural Network (CNN)

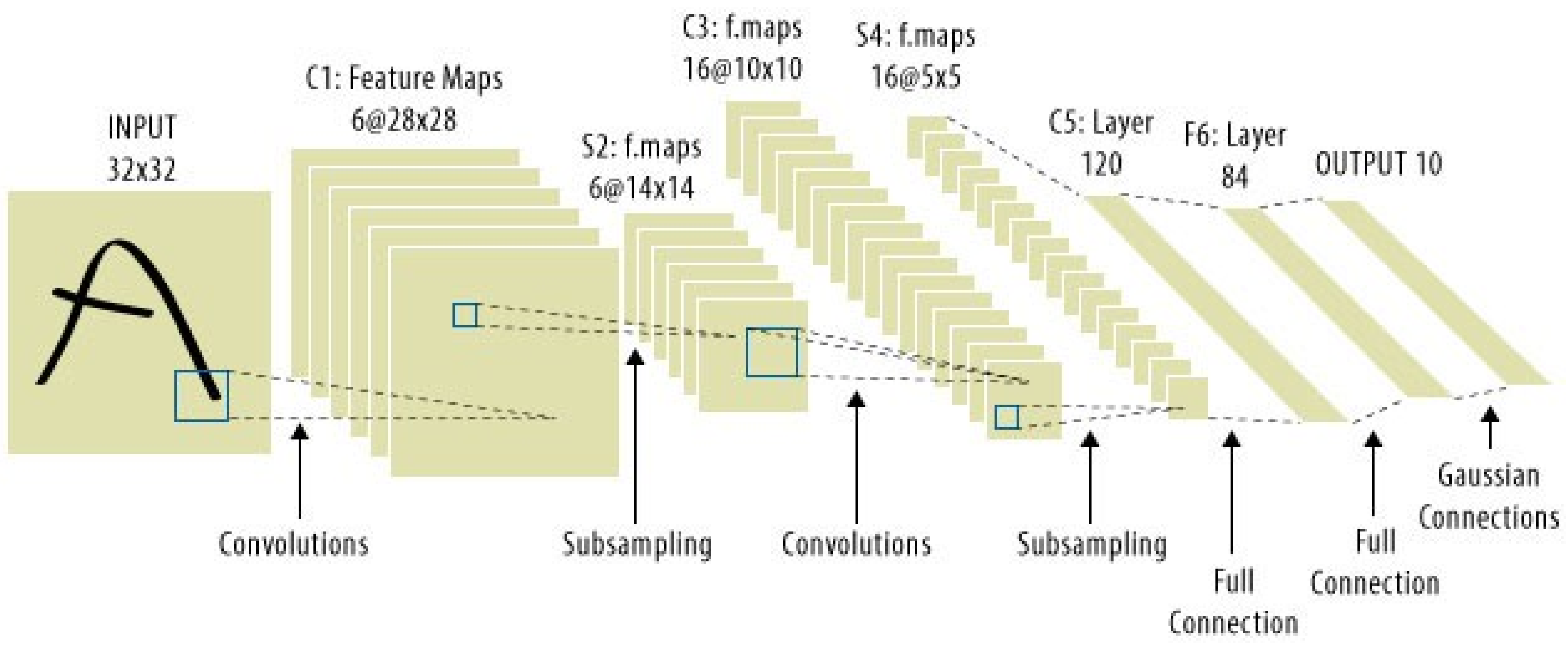
- Can be seen as:
  - Crude model of human visual cortex
  - Generalization of Gabor filters
  - Generalization of template matching
  - Way of parameter sharing and hence parameter reduction

CNN as glorified template matching

- Try to match template as each location by sliding it over the input image

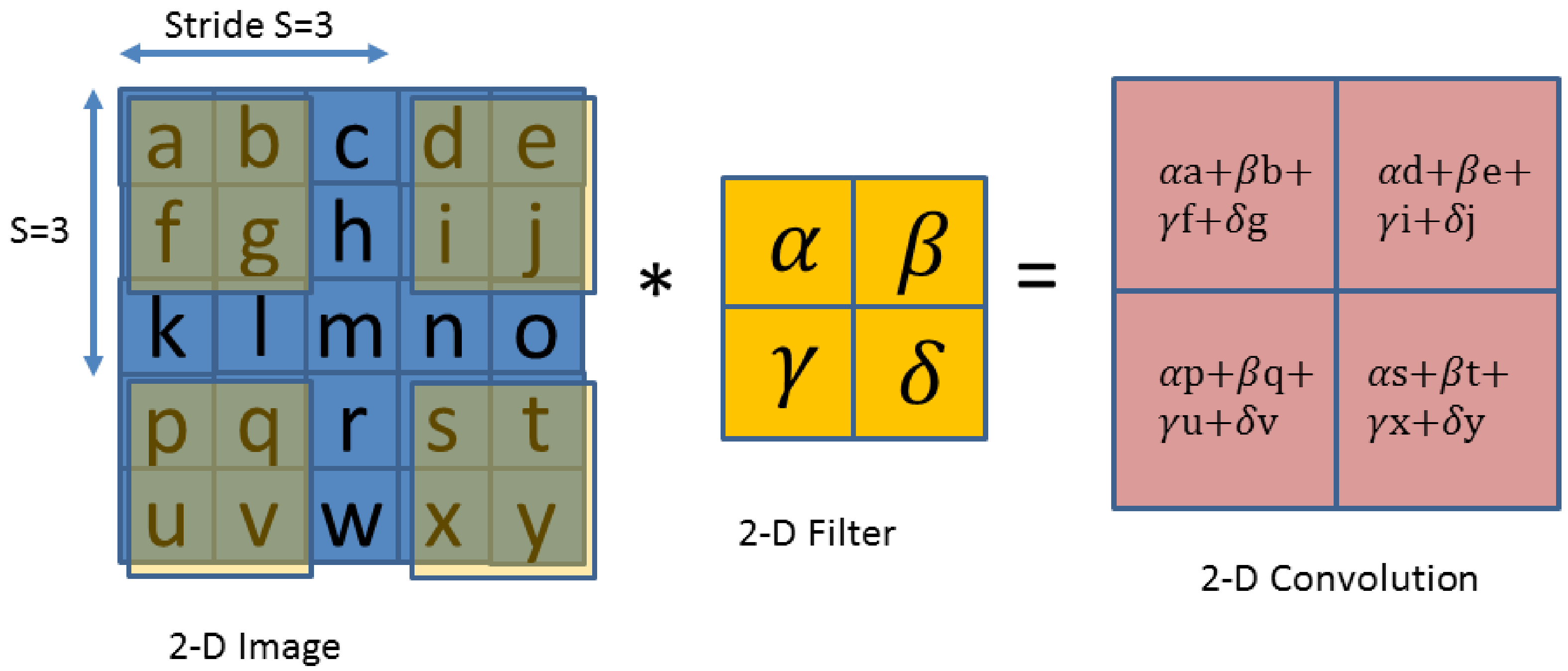


A typical CNN



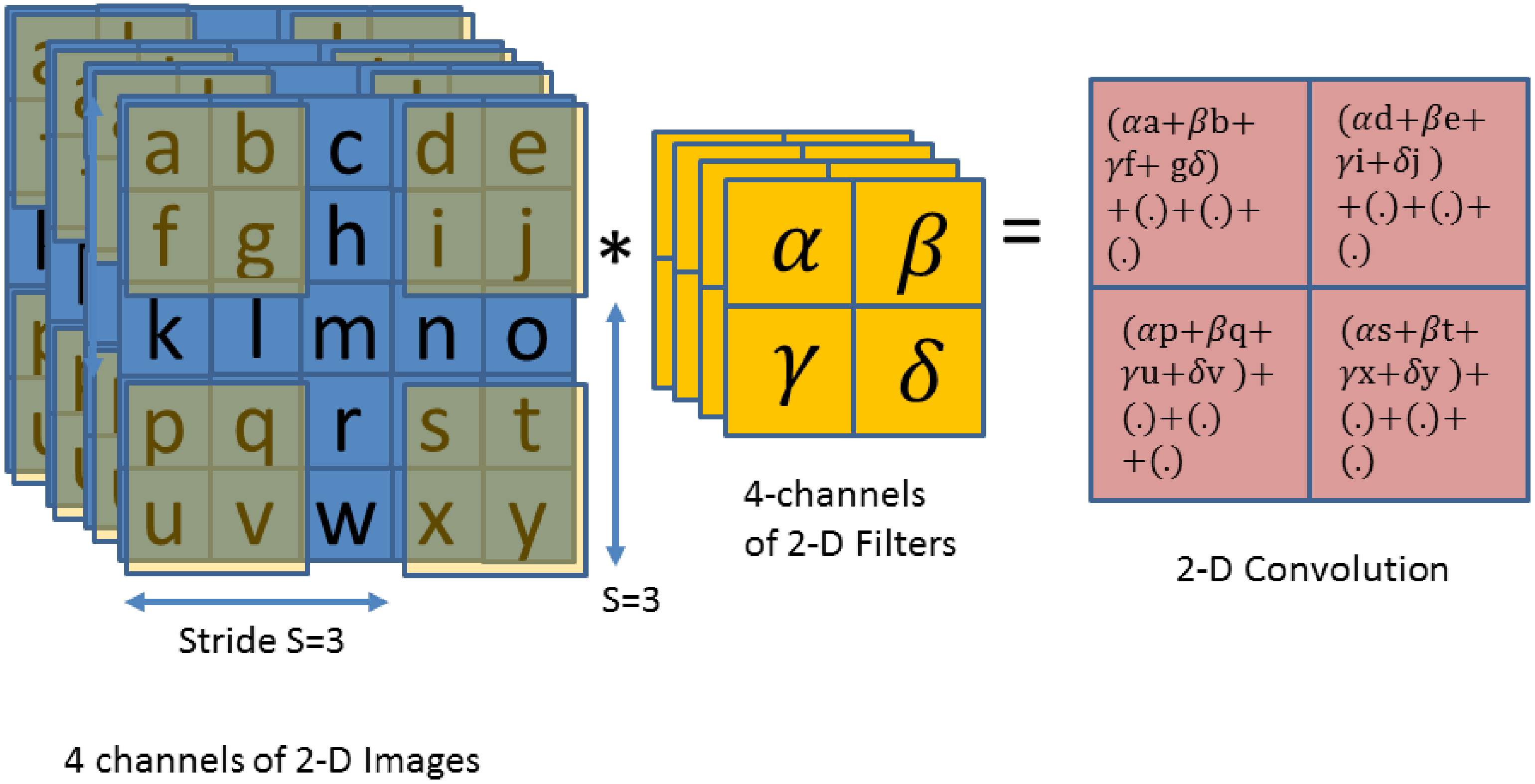
Convolutional Layer

- Convolution of 2D-inputs (eg: single channel images)



Convolutional Layer

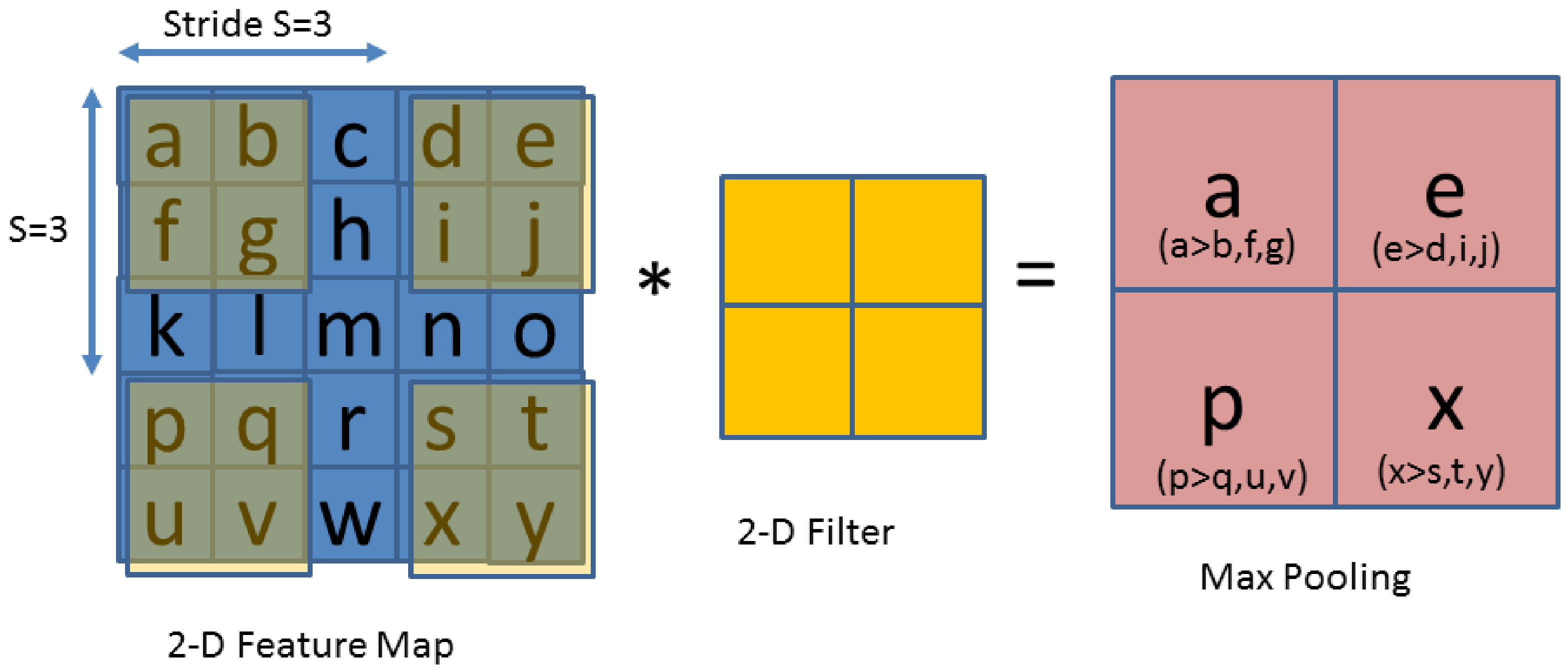
- Convolution with 3D inputs (eg: multiple channel images or feature maps)



a b c d e
f g h i j
k l m n o
p q r s t
u v w x y
*
α β
γ δ
=
(αa+βb+ γf+ gδ) +(.)+(.)+ (.)
(αd+βe+ γi+δj) +(.)+(.)+ (.)
(αp+βq+ γu+δv)+ (.)+(.) +(.)
(αs+βt+ γx+δy)+ (.)+(.)+ (.)
Stride S=3
S=3
4-channels of 2-D Filters
2-D Convolution
4 channels of 2-D Images

Sub-sampling layer

- Max-pooling operation

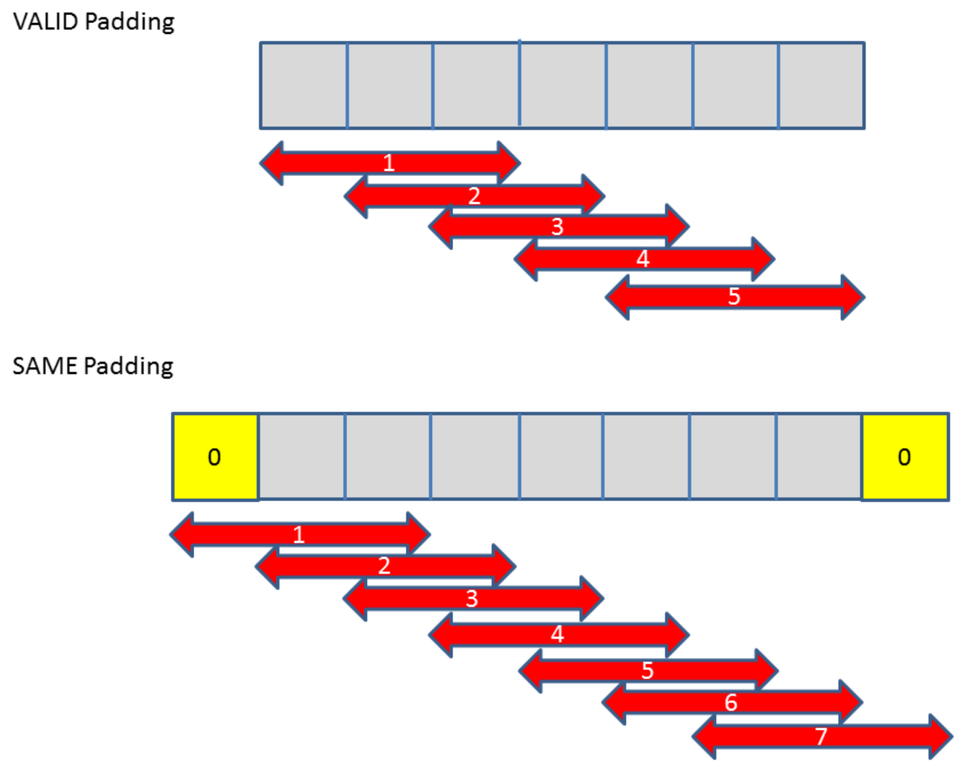


Padding

- Convolution and sub-sampling (max-pooling) lead to an output dimension $N_o = \frac{N_i - F}{s} + 1$ where N_o =input dimension, F = Filter dimension and s = stride
- Output dimension can decrease even when stride s = 1.
- Apply zero padding around the image to prevent such reduction of dimensions.

Padding

- Tensorflow padding options illustrated using 1D signals



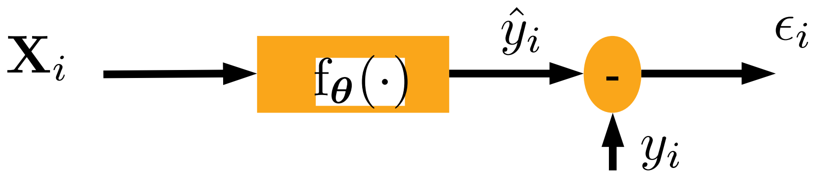
Loss functions

- Also called Cost/Error/Objective functions
- A loss function measures the similarity between model output $\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{X})$ and the corresponding label $\mathbf{y}$
- Common loss functions:
  - Mean Squared Error (MSE)
  - (Categorical) Cross Entropy (CE)

Mean Squared Error (MSE)

- $E_{MSE} = \frac{1}{N} \sum_i (y_i - \hat{y}_i)^2$
- Suitable for regression (i.e. predict y_i)
- It can be shown that MSE is equivalent to the conditional maximum likelihood of labels, when the prediction error ϵ_i has a zero mean Gaussian distribution.

$$\arg\min_{\boldsymbol{\theta}} \{E_{MSE}\} = \arg\min_{\boldsymbol{\theta}} \{-\sum_i \log p(y_i|\mathbf{X}_i)\}$$



Cross Entropy (CE)

- Suitable for classification (i.e. predicting the class probabilities of K given classes)
- Labels can be:
  - One-hot encoded or class probability label:

$$[y_i(1), y_i(2), \cdots, y_i(K)]$$

  - Sparse: index of the only correct class j is given, i.e. $y_i = j$

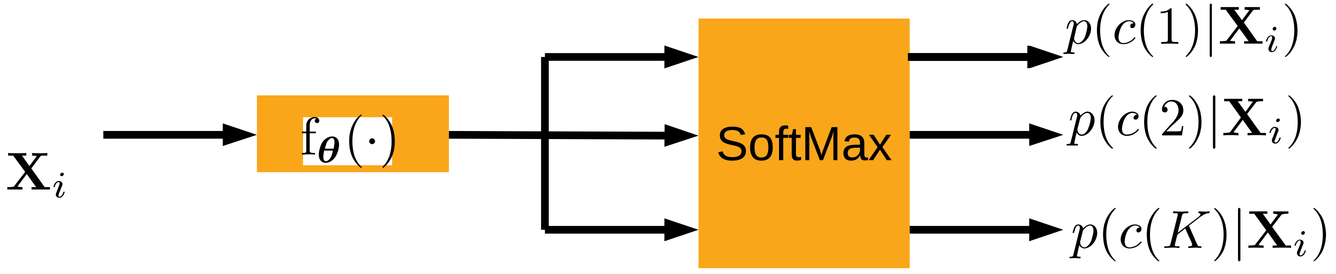
Cross Entropy (Ctd)

- Model outputs a categorical distribution i.e. probability of each class $p(c(k)|\mathbf{X}_i)$, where i is the sample index and $c(k)$ is the class of index k
- Cross Entropy:
  - One-hot encoded or class probability labels:

$$E_{CE} = -\frac{1}{N}\sum_i \sum_k y_i(k) \log p(c(k)|\mathbf{X}_i)$$

  - Sparse labels: Assume the correct class is y_i

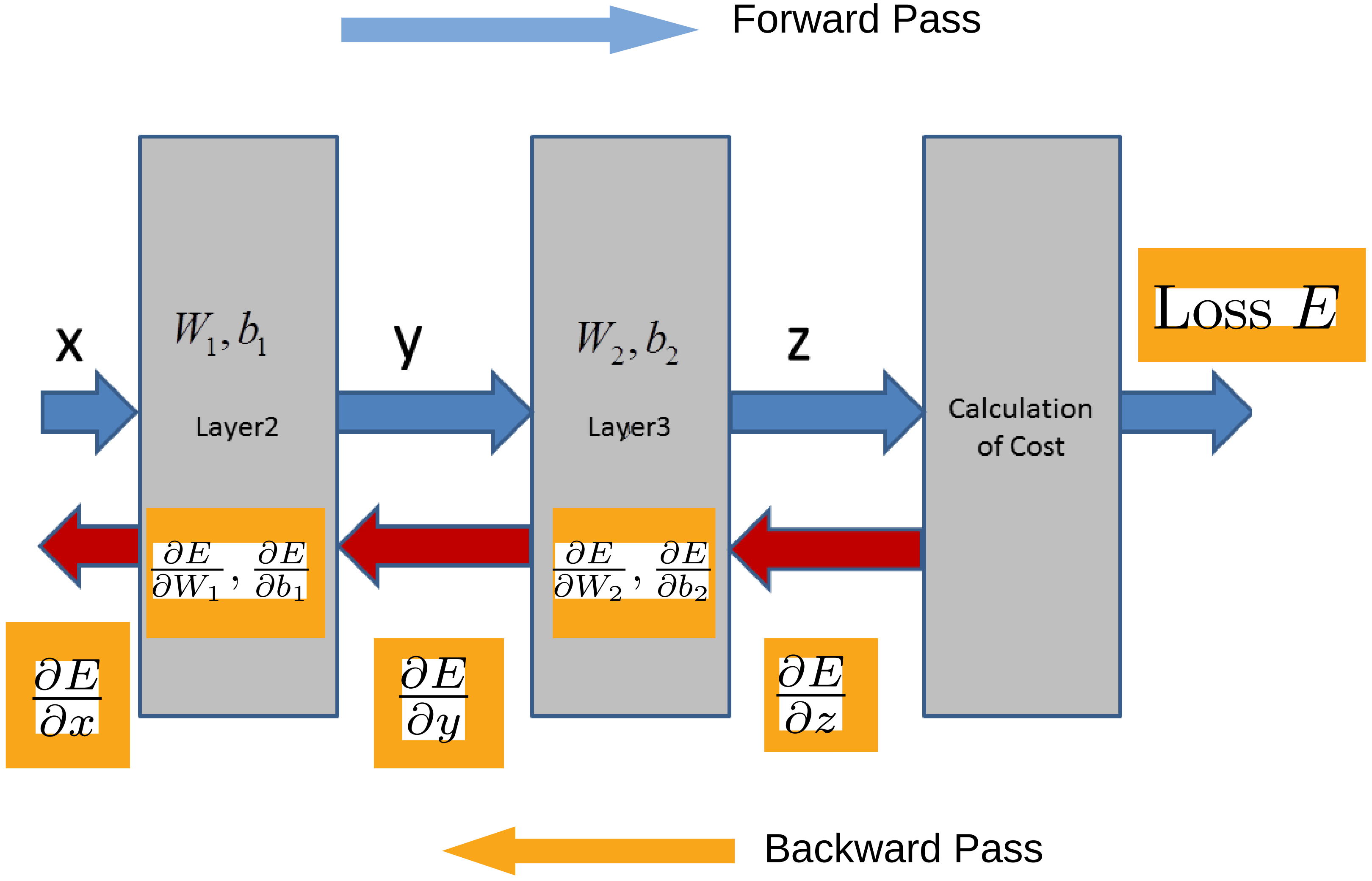
$$E_{CE} = -\frac{1}{N}\sum_i \log p(c(y_i)|\mathbf{X}_i)$$



Training

- The process of fitting the model parameters to the given training data
  - This is the same as optimizing the loss function with respect to model parameters
- Parameters are updated using gradient descent
  - Plain gradient descent: $\theta_{t+1} = \theta_t - \epsilon \frac{\partial E}{\partial \theta_t}$
  - More fanzy algorithms (ADAM, RmsPROP, and many more)
- Estimation of gradients $\frac{\partial E}{\partial \theta_t}$ is the core problem of training
  - Back-propagation algorithm

Back-Propagation

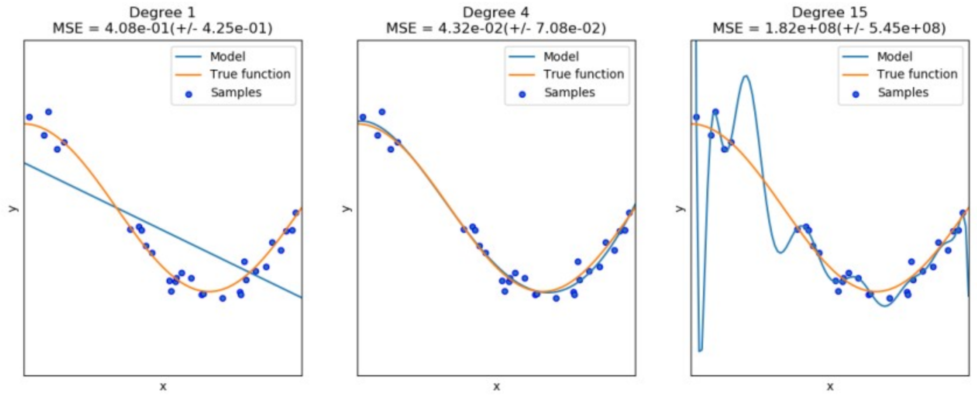

Forward Pass
x
W_1, b_1
Layer2
y
W_2, b_2
Layer3
z
Calculation of Cost
Loss E
$\frac{\partial E}{\partial W_1}, \frac{\partial E}{\partial b_1}$
$\frac{\partial E}{\partial W_2}, \frac{\partial E}{\partial b_2}$
$\frac{\partial E}{\partial x}$
$\frac{\partial E}{\partial y}$
$\frac{\partial E}{\partial z}$
Backward Pass

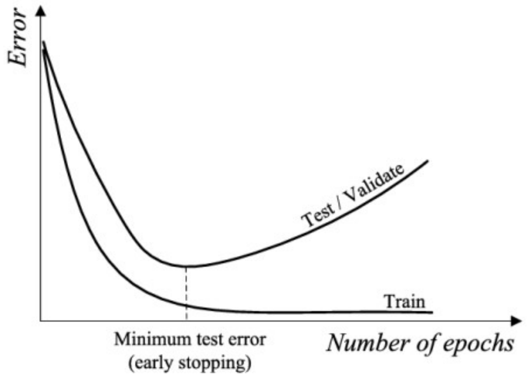
Model Parameter Update

- Batch gradient descent:
  - Gradients are accumulated for the whole training set, and do the update
- Stochastic gradient descent:
  - No gradient accumulation, update after processing each (randomly selected) sample
- Mini-batch gradient descent:
  - Divide the training set into several mini-batches, accumulate gradient and update after each (randomly selected) mini-batch

Regularization

- Network should perform well on unseen data
  - Generalization ability
- Too much adaptation to training data (over-fitting) harms generalization ability.
- Regularization prevents over-fitting





Regularization vs Optimization

- Optimization: Try to reduce training loss
  - Often test loss is reduced as a side effect
  - Test loss may not be the minimum if over-fitting occurs
- Regularization: Try to reduce the test loss
  - Regularization imposes “restrictions” on training.
  - Minimum test loss is not necessarily corresponding to the minimum training loss

Popular regularization techniques in deep learning

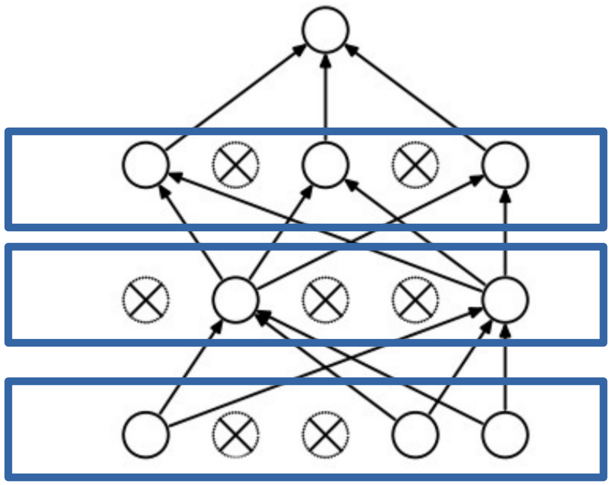
- Early stopping in training
- Data shuffling
- Data Augmentation
- Weight decay
- Dropout
- Batch Normalization

Weight Decay

- Modify the loss function
  - Add a penalty term for high parameter values
  - $E_{reg}(\boldsymbol{\theta}) = E(\boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||^2$
  - λ is a positive constant and $||\cdot||$ is Eucledian norm of the parameter vector $\boldsymbol{\theta}$

Dropout

- Randomly remove neurons from a given layer every time a sample is processed
- Removal percentage is a parameter of Dropout
- Typically done only in training (i.e. NOT in testing)



Batch Normalization (BN)

- Normalize the input of a neuron (or inputs to a set of related neurons) with the mean and standard deviation of the batch.
- This will prevent “covariate shift” (i.e. large changes of input distribution from one batch to another)
- Initially proposed as a method for improving the training speed
- BN has a regularization effect as well.

BN for image data

$$\mu_d \leftarrow \frac{1}{N*H*W}\sum_{i=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{i,h,w,d}$$

$$\sigma_d^2 \leftarrow \frac{1}{N*H*W}\sum_{i=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} (x_{i,h,w,d}-\mu_d)^2$$

$$\hat{x}_{i,h,w,d} \leftarrow \frac{x_{i,h,w,d}-\mu_d}{\sqrt{(\sigma_d^2+\epsilon)}}$$

$$y_{i,h,w,d} \leftarrow \gamma\hat{x}_{i,h,w,d}+\beta$$

- $x_{i,h,w,d}$ is an element of the input tensor of size $N \times H \times W \times D$ where $N=$ batch size, H=image height, W=image width and $D=$ image depth (number of channels)
- $y_{i,h,w,d}$ is the corresponding element of the output tensor
- γ and β are trainable parameters called *scale* and *center*
- ϵ is a small constant to prevent numerical instability (i.e. divide by zero)

BN Practise

- Full BN is performed only in training
- Global standard deviations and means of the training set σ_d^g and μ_d^g are stored during training (as the moving average of σ_d and μ_d of each batch
- In testing (inference) the stored global standard deviations and means are used.

Other Issues

- How to select hyper-parameters
  - Learning rate
  - Regularization parameters
  - Model architecture
- Initialization of model parameters