Introduction to Deep Learning Concepts & Framework

CS 529 - Security Analytics

Typically Used Loss Functions :

(1) Mean Squared Loss / Quadratic Loss / L2 Loss
(2) Mean Absolute Error
(3) Hinge Loss / MultiClass SVM Loss
(4) Cross Entropy Loss / Negative Log Likelihood Loss
(5) Huber Loss (Fancy Loss)

[to name a few!]

Different Types of Activation Functions

(1) Binary Step
(2) Sigmoid
(3) Tanh (tan hyperbolic)
(4) ReLU (Rectified Linear Units)
(5) Leaky ReLU
(6) Softmax

How do you choose the right activation function ?

Good or bad – there is no rule of thumb.

Depending upon the properties of the problem we might be able to make a better choice for easy and quicker convergence of the network.

- Sigmoid functions and their combinations generally work better in the case of classifiers.
- Sigmoids and tanh functions are sometimes avoided due to the vanishing gradient problem.
- ReLU function is a general activation function and is used in most cases these days.
- If we encounter a case of dead neurons in our networks the leaky ReLU function is the best choice.
- Always keep in mind that ReLU function should only be used in the hidden layers.
- As a rule of thumb, you can begin with using ReLU function and then move over to other activation functions in case ReLU doesn’t provide with optimum results.

Commonly used Optimizers ?

(1) Momentum
(2) Nesterov Accelerated Gradient
(3) AdaGrad - Adaptive Gradient Algorithm
(4) AdaDelta
(5) RMSProp - Root Mean Squared Propagation
(6) Adam - Adaptive Moment Estimation

Why do we use Optimizers ?

- Optimizers update the weight parameters to minimize the loss function.

- Loss function acts as guides to the terrain telling optimizer if it is moving in the right direction to reach the bottom of the valley, the global minimum.

Keras Models

Difference between trainable vs non-trainable parameters ?

Trainable Parameters : Neuron Weights, biases.

Non-Trainable Parameters : Number of hidden layers, hidden neurons per layer, etc.

Sklearn vs Keras framework for deep learning?

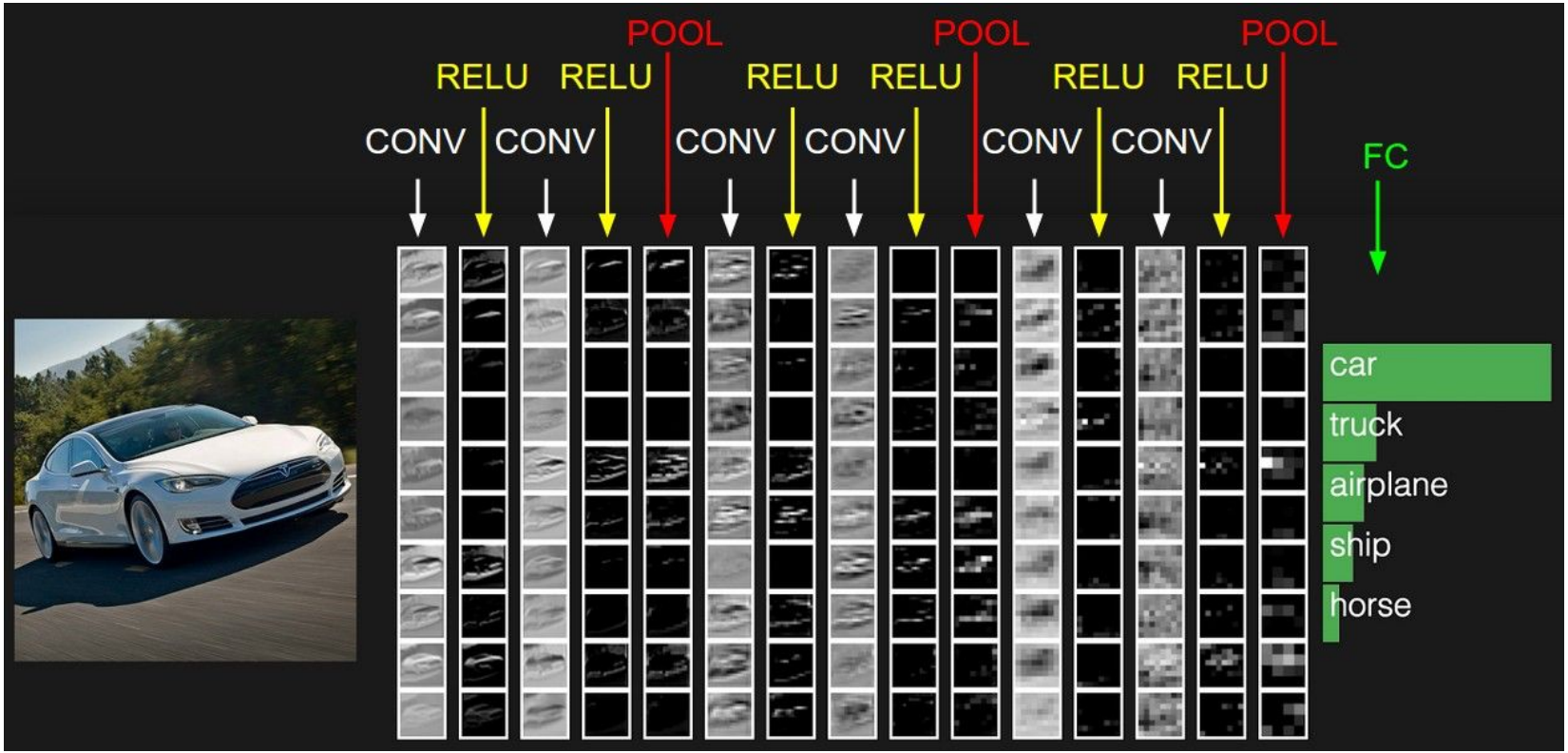
- SciKit Learn is a general machine learning library, built on top of NumPy.
- It features a lot of machine learning algorithms such as support vector machines, random forests, as well as a lot of utilities for general pre- and postprocessing of data.
- It is not a neural network framework.

- Keras is a higher-level deep learning framework.
- Abstracts many details away, making code simpler and more concise than in PyTorch or TensorFlow, at the cost of limited hackability.
- It abstracts away the computation backend, which can be TensorFlow, Theano or CNTK.

Important Terms related to a Convolutional Layer

1. Spatial Extent (F)
2. Stride (S)
3. No. of Filters (K)
4. Pooling (e.g. Max Pooling)

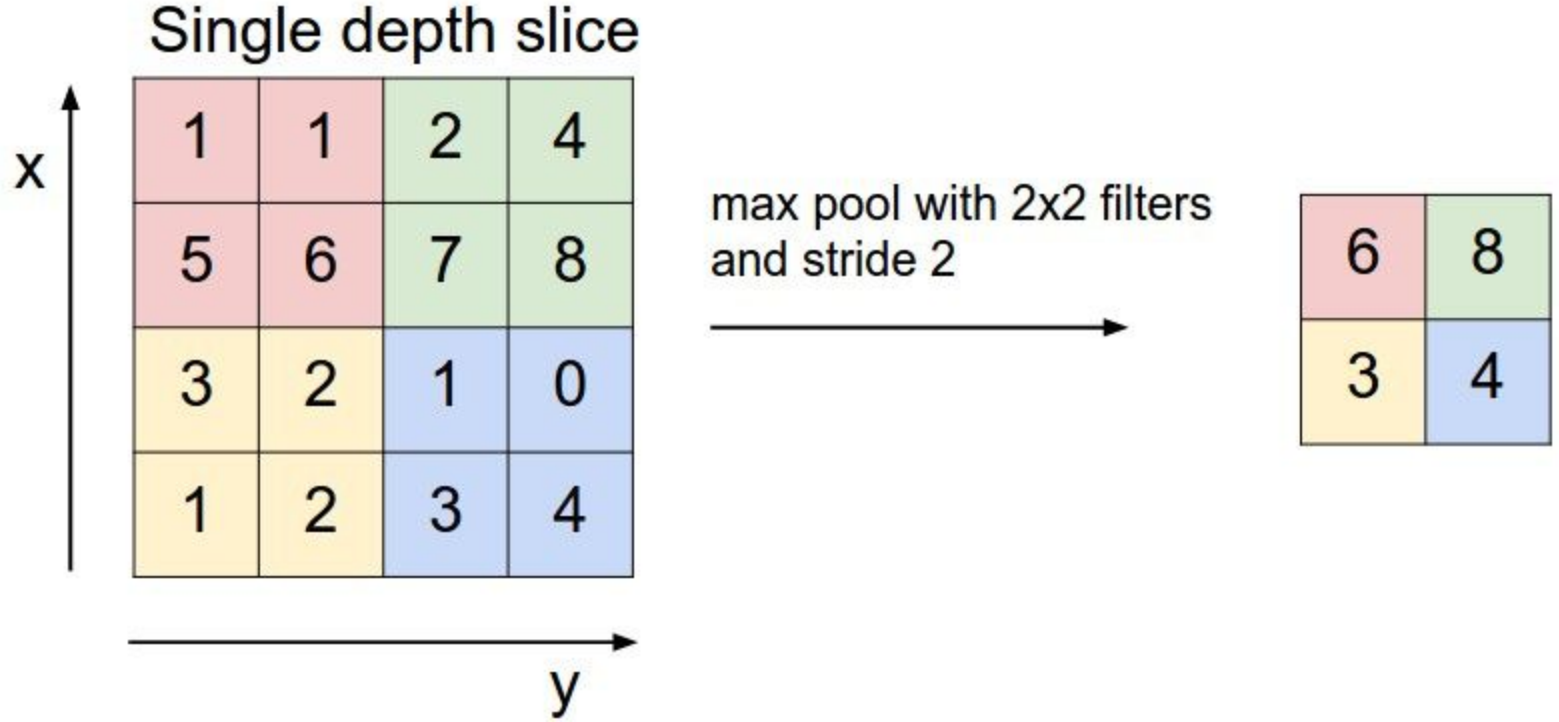
Sample Convolutional Architecture



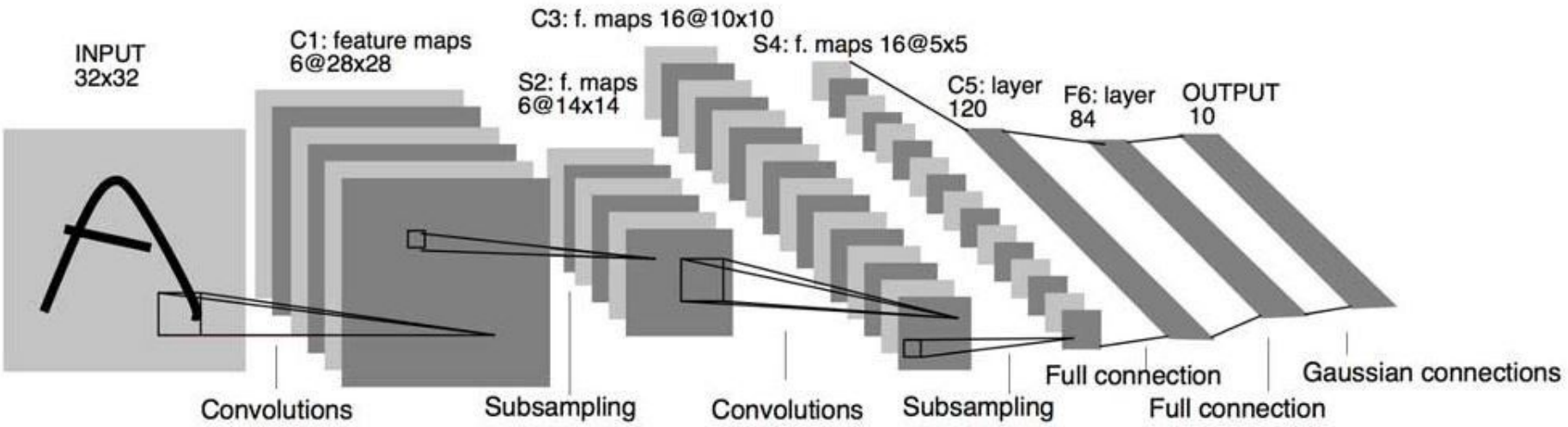
Pooling Layer

- It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture.
- Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting.
- The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation.
- The depth dimension remains unchanged.

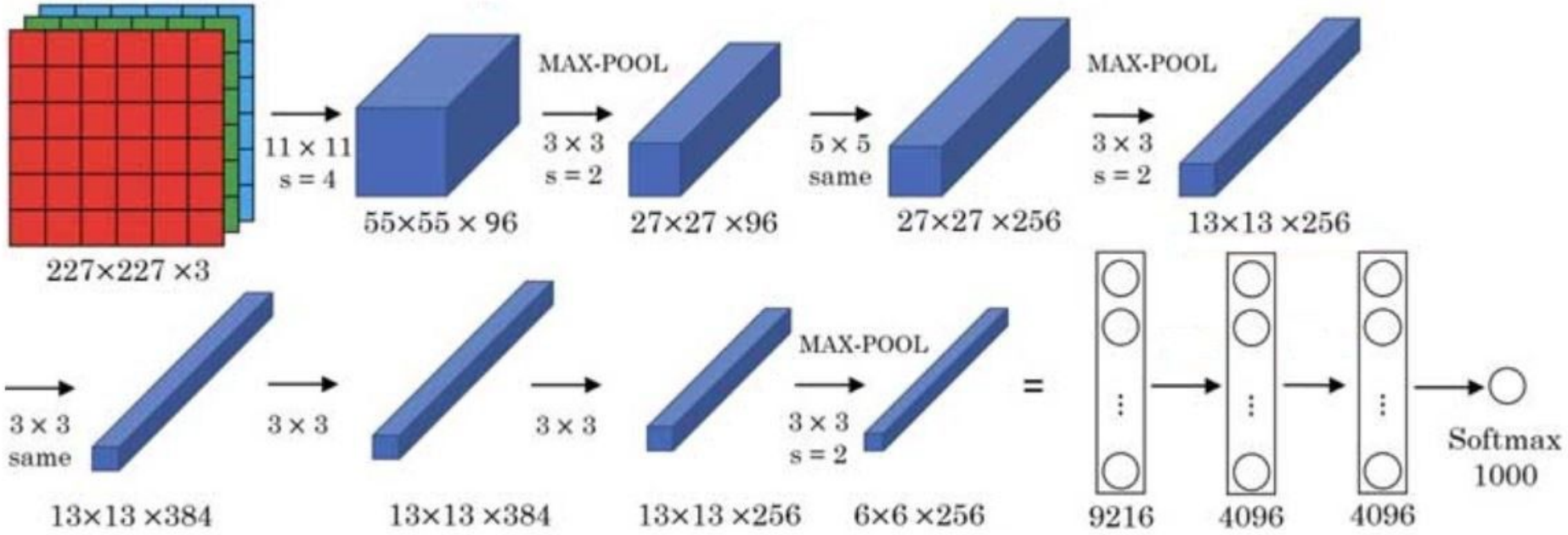
Understanding how Max Pool works ?



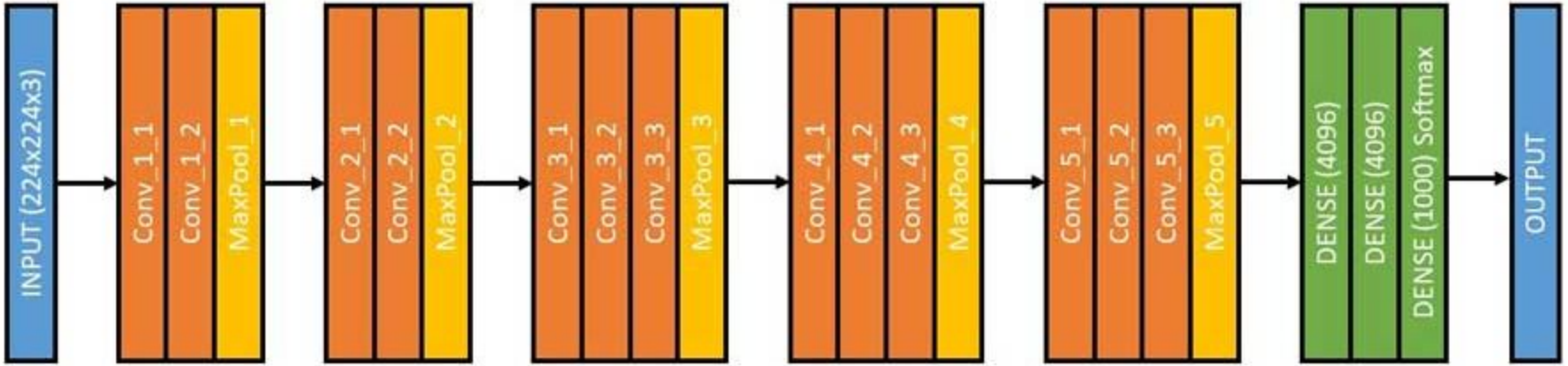
LeNET 5 - LeCun et al



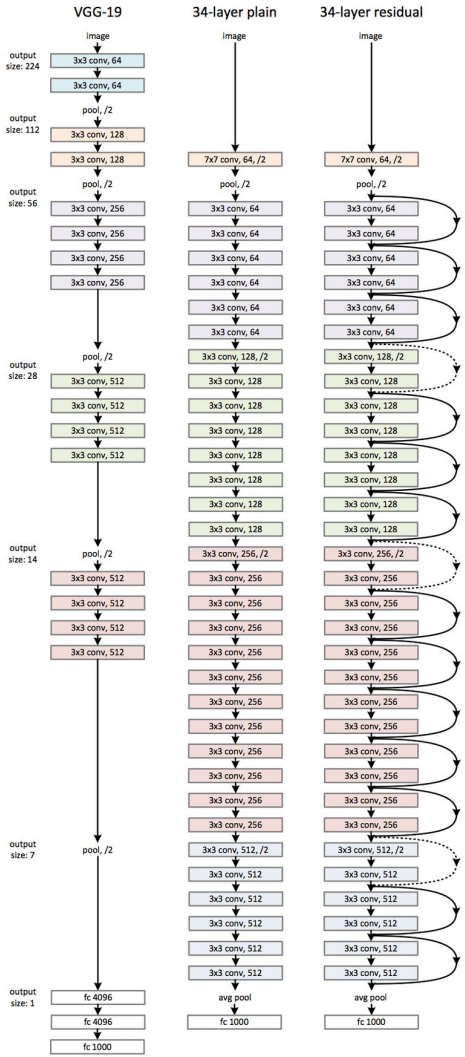
AlexNET - Krizhevsky et al



VGGNet - Simonyan et al



ResNET : Kaiming He et al



Questions to ponder about ?

(1) Why don’t we use multi-layer perceptrons, over neural networks?
(2) Why don’t we prefer to use scikit-learn and why do we stick with Keras?
(3) How do we define the number of layers in a neural network or even the number of neurons?
(4) If I make my network very deep and increase the number of neurons, shouldn’t the performance of the network obviously increase?
(5) How do I go about tuning the parameters?
(6) What is the difference between accuracy and f1-score?
(7) What is the intuition behind choosing what algorithms to begin with?

(1) When do we go in for Precision and when do we prefer Recall ?
(2) Why do we go in for F1 - Score when we have precision and recall as metrics?
(3) When do we prefer F1 - Score over accuracy ?

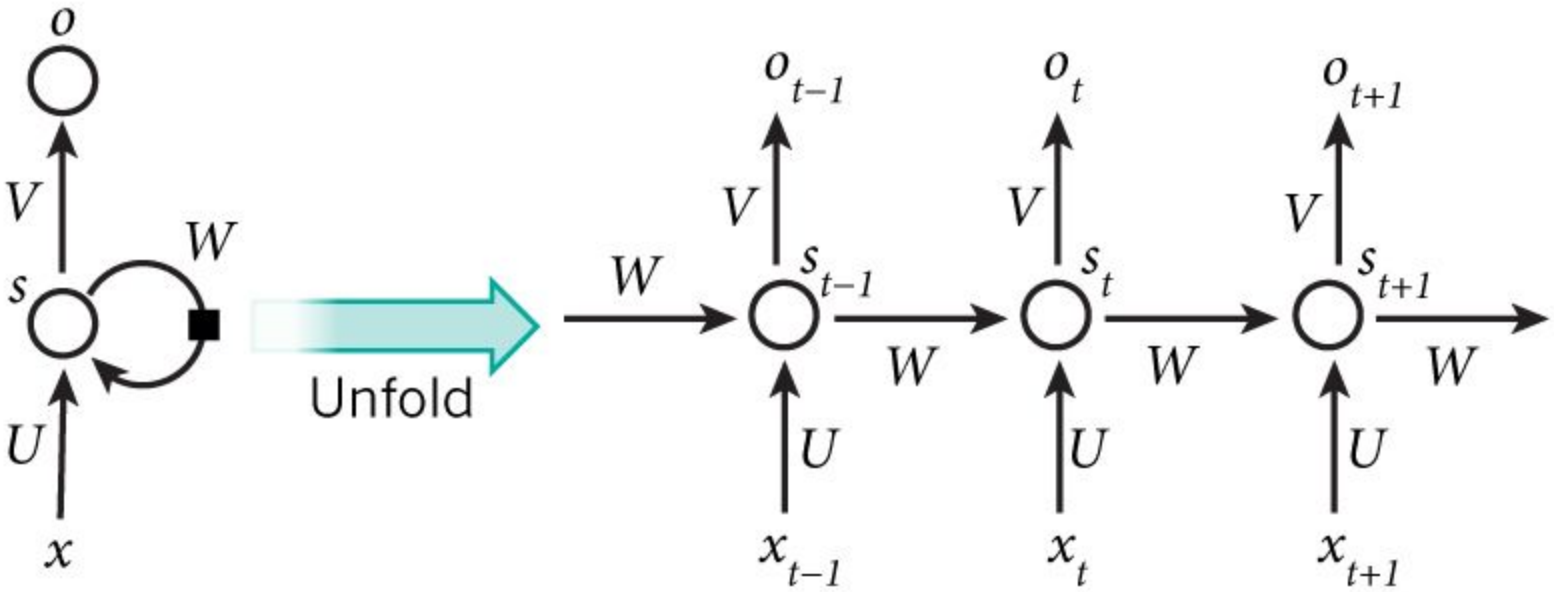
Additional DL related questions?

(1) When do we go in for a CNN over a DNN?
(2) How do we decide the number of filters and padding and stride and other parameters ?

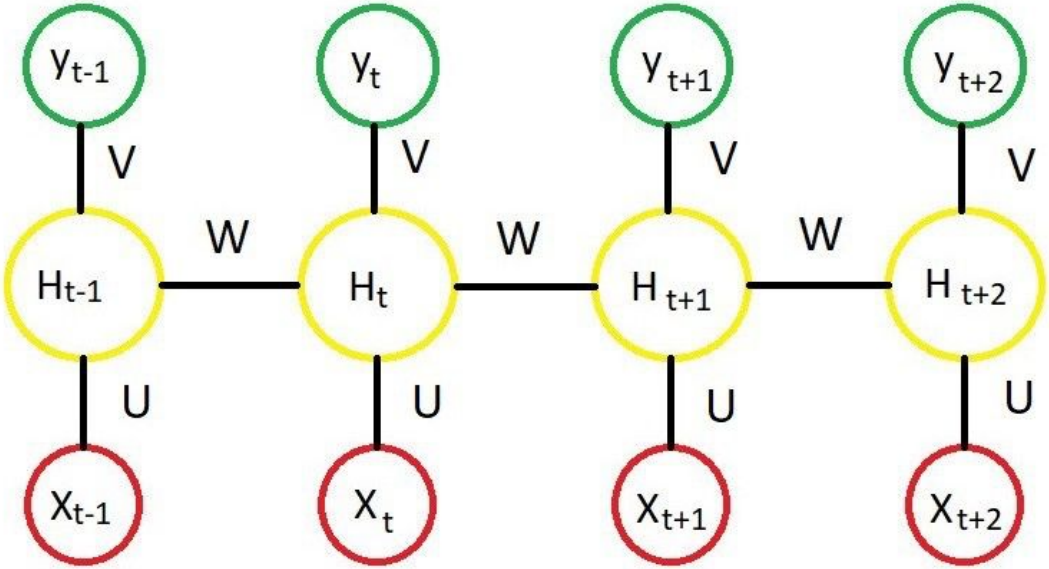
State of the Art Networks in Neural Networks

(1) Convolutional Neural Networks (CNNs)
(2) Recurrent Networks: (comprised of RNNs along with the following (2-5))
(3) Long Short Term Memory Networks (LSTMs)
(4) Gated Recurrent Networks
(5) Bidirectional - LSTMs (powerful concept in NLP, used in time series too!)
(6) Generative Adversarial Networks (GANs) [State of the art networks in modelling the underlying probability distribution of the data.]

A typical unrolled view of a recurrent network



Recurrent neural networks



U = Weight vector for Hidden layer
V = Weight vector for Output layer
W = Same weight vector for different Timesteps
X = Word vector for Input word
y = Word vector for Output word

At Timestep (t)

$Ht = \sigma (U * Xt + W * Ht-1)$

$yt = Softmax (V * Ht)$

$$J^t(\theta) = - \sum_{j=1}^{|M|} y_{t,j} \log \bar{y}_{t,j}$$

$$J(\theta) = - \frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{|M|} y_{t,j} \log \bar{y}_{t,j}$$

M = vocabulary , J(θ) = Cost function

Cross Entropy Loss

Useful Resources

http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (Recurrent Networks)

http://cs231n.stanford.edu (Convolutional Neural Networks for Visual Recognition)

https://www.coursera.org/specializations/deep-learning (Deep Learning Specialization)

https://www.coursera.org/learn/machine-learning (Machine Learning)

Useful Resources

https://projector.tensorflow.org (t-SNE Plots)

https://playground.tensorflow.org (Intuition behind how an NN trains?)

http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/ (DL Blog)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (Guide to understanding RNNs)

https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b (Understanding the backpropagation algorithm in detail)

Useful Resources

https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3 (Types of Optimizers)

http://cs231n.github.io/convolutional-networks/ (Understanding CNNs)

https://github.com/mbadry1/DeepLearning.ai-Summary/tree/master/4-%20Convolutional%20Neural%20Networks#why-resnets-work (Notes from the Deep Learning Specialization on Coursera.)